IBM DB2 10.1
for Linux, UNIX and Windows

Spatial Extender
Benutzer- und Referenzhandbuch

SC12-4688-00

IBM DB2 10.1
for Linux, UNIX and Windows

Spatial Extender
Benutzer- und Referenzhandbuch

SC12-4688-00

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen in Anhang B, „Bemerkungen", auf Seite 475 gelesen werden.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM DB2 10.1 for Linux, UNIX, and Windows, Spatial Extender User's Guide and Reference,
IBM Form SC27-3894-00,
herausgegeben von International Business Machines Corporation, USA

Informationen, die nur für bestimmte Länder Gültigkeit haben und für Deutschland, Österreich und die Schweiz nicht zutreffen, wurden in dieser Veröffentlichung im Originaltext übernommen.

Möglicherweise sind nicht alle in dieser Übersetzung aufgeführten Produkte in Deutschland angekündigt und verfügbar; vor Entscheidungen empfiehlt sich der Kontakt mit der zuständigen IBM Geschäftsstelle.

Änderung des Textes bleibt vorbehalten.

Herausgegeben von:
TSC Germany
Kst. 2877
Mai 2012

Inhaltsverzeichnis

Kapitel 1. Verwendungszweck von DB2 Spatial Extender 1
Darstellungsweise von geografischen Objekten durch Daten 2
Charakter räumlicher Daten 3
Quellen für räumliche Daten 4
Funktionen, räumliche Informationen, räumliche Daten und Geometrien - Zusammenhänge 5

Kapitel 2. Geometrien 7
Eigenschaften von Geometrien 9
Geometrietypen 9
Koordinaten der Geometrie 10
X- und Y-Koordinaten 10
Z-Koordinaten 10
M-Koordinaten 10
Innenbereich, Begrenzung und Außenbereich . . 10
Einfache oder nicht einfache Geometrien . . . 10
Geschlossene Geometrien 11
Leere oder nicht leere Geometrien 11
Minimal einschließendes Rechteck (MBR) . . . 11
Dimension von Geometrien 11
Kennung des räumlichen Bezugssystems . . . 12

Kapitel 3. DB2 Spatial Extender verwenden 13
Schnittstellen zu DB2 Spatial Extender und ihre Funktionalität 13
DB2 Spatial Extender einrichten 13
Projekte erstellen, die räumliche Daten verwenden 15

Kapitel 4. Erste Schritte mit DB2 Spatial Extender 19
DB2 Spatial Extender einrichten und installieren . . 19
Systemvoraussetzungen für die Installation von DB2 Spatial Extender 20
Installieren von DB2 Spatial Extender mit dem **DB2-Installationsassistenten** (Windows) 21
Installieren von DB2 Spatial Extender mit dem **DB2-Installationsassistenten** (Linux und UNIX) 22
DB2 Spatial Extender-Installation prüfen. 24

Kapitel 5. Upgrade auf DB2 Spatial Extender Version 10.1 27
Upgrade für DB2 Spatial Extender. 27
Aktualisierung von DB2 Spatial Extender (32-Bit) auf DB2 Spatial Extender (64-Bit) 28

Kapitel 6. Räumliche Ressourcen für eine Datenbank konfigurieren 31
Für die Datenbank bereitgestellte Ressourcen . . . 31
Datenbank für räumliche Operationen aktivieren . . 32
Geocoder registrieren 33

Kapitel 7. Räumliche Ressourcen für ein Projekt konfigurieren 35
Verwendungshinweise für Koordinatensysteme . . 35
Koordinatensysteme 35
Geografisches Koordinatensystem 36
Projizierte Koordinatensysteme 41
Zu verwendendes Koordinatensystem festlegen 42
Hinweise zur Konfiguration räumlicher Bezugssysteme. 43
Räumliche Bezugssysteme 44
Über Verwendung eines vorhandenen oder Erstellung eines neuen Bezugssystems entscheiden . 45
Räumliche Bezugssysteme, die mit DB2 Spatial Extender geliefert werden 46
Umwandlungsfaktoren, die Koordinatendaten in Integer umsetzen 49
Räumliches Bezugssystem erstellen 50
Maßstabsfaktoren berechnen. 51
Umwandlungsfaktoren, die Koordinatendaten in Integer umsetzen 52
Minimal- und Maximalkoordinaten und Bemaßungen ermitteln 52
Offsetwerte berechnen. 53
Räumliches Bezugssystem erstellen 54

Kapitel 8. Räumliche Spalten konfigurieren 57
Darstellung räumlicher Spalten. 57
Räumliche Datentypen 57
Datentypen für einteilige Objekte 58
Datentypen für mehrteilige Objekte 58
Universaldatentyp für alle Objekte. 59
Räumliche Spalten erstellen 59
Räumliche Spalten registrieren 60

Kapitel 9. Räumliche Spalten ausfüllen 63
Informationen zum Importieren und Exportieren von räumlichen Daten. 63
Formdaten in eine neue oder eine vorhandene Tabelle importieren 64
Daten in eine Formdatei exportieren 65
Verwendungshinweise für einen Geocoder 66
Geocoder und Geocodierung 66
Geocodierungsoperationen definieren. 68
Geocoder für automatische Ausführung definieren 70
Geocoder im Stapelmodus ausführen 71

Kapitel 10. DB2 Spatial Extender in Umgebung mit partitionierten Datenbanken 73
Räumliche Daten in einer Umgebung mit partitionierten Datenbanken erstellen und laden 73

Abfrageleistung für räumliche Daten in einer partitionierten Umgebung verbessern 74

Kapitel 11. Indizes und Sichten für den Zugriff auf räumliche Daten verwenden. 77

Räumliche Rasterindizes 77
Räumliche Rasterindizes generieren 77
Verwendung räumlicher Funktionen in einer Abfrage 78
Verwendung des räumlichen Rasterindexes durch eine Abfrage 78
Überlegungen zur Anzahl der Indexstufen und Rastergrößen 79
Anzahl der Rasterebenen 79
Größe von Rasterzellen 80
Räumliche Rasterindizes erstellen 84
Anweisung CREATE INDEX für den räumlichen Rasterindex 85
Räumliche Rasterindizes mit dem Indexadvisor optimieren 87
Rastergrößen für räumliche Rasterindizes festlegen 87
Statistikdaten für räumliche Rasterindizes analysieren 88
Befehl gseidx 93
Über Sichten auf räumliche Spalten zugreifen . . . 96

Kapitel 12. Räumliche Informationen analysieren und generieren 97

Umgebungen zur Ausführung einer räumlichen Analyse 97
Beispiele für die Operationen von räumlichen Funktionen 97
Funktionen, die zur Abfrageoptimierung Indizes verwenden 98

Kapitel 13. Anwendungen schreiben und Beispielprogramm verwenden . . 101

Kopfdatendatei von DB2 Spatial Extender in räumliche Anwendungen integrieren 101
Gespeicherte Prozeduren von DB2 Spatial Extender über eine Anwendung aufrufen 101
Beispielprogramm von DB2 Spatial Extender . . . 103

Kapitel 14. Fehler bei DB2 Spatial Extender erkennen 109

Hinweise zum Interpretieren der Nachrichten von DB2 Spatial Extender 109
Ausgabeparameter von gespeicherten Prozeduren von DB2 Spatial Extender 111
Nachrichten von DB2 Spatial Extender-Funktionen 113
Nachrichten des Befehlszeilenprozessors von DB2 Spatial Extender 114
Trace für Fehler bei DB2 Spatial Extender mit dem Befehl 'db2trc' durchführen 116
Benachrichtigungsdatei für Systemverwaltung . . 118

Kapitel 15. Katalogsichten 119

Katalogsicht DB2GSE.ST_COORDINATE_SYSTEMS 119
Katalogsicht DB2GSE.ST_GEOMETRY_COLUMNS 120
Katalogsicht DB2GSE.ST_GEOCODER_PARAMETERS 121
Katalogsicht DB2GSE.ST_GEOCODERS. 123
Katalogsicht DB2GSE.ST_GEOCODING 124
Katalogsicht DB2GSE.ST_GEOCODING_PARAMETERS . . . 125
Katalogsicht DB2GSE.ST_SIZINGS 127
Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS . . 128
Katalogsicht DB2GSE.ST_UNITS_OF_MEASURE 131
Katalogsicht DB2GSE.SPATIAL_REF_SYS 132

Kapitel 16. Befehle von DB2 Spatial Extender 135

Befehle zur Konfiguration von DB2 Spatial Extender und Entwicklung von Projekten aufrufen. . . 135
Befehl 'db2se alter_cs' 136
Befehl 'db2se alter_srs' 138
Befehl 'db2se create_cs' 140
Befehl 'db2se create_srs' 142
Befehl 'db2se disable_autogc' 146
Befehl 'db2se disable_db' 147
Befehl 'db2se drop_cs' 149
Befehl 'db2se drop_srs' 150
Befehl 'db2se enable_autogc' 151
Befehl 'db2se enable_db' 152
Befehl 'db2se export_shape' 153
Befehl 'db2se import_shape' 157
Befehl 'db2se register_gc' 164
Befehl 'db2se register_spatial_column' 167
Befehl 'db2se remove_gc_setup' 169
Befehl 'db2se restore_indexes' 170
Befehl 'db2se save_indexes' 171
Befehl 'db2se run_gc' 172
Befehl 'db2se setup_gc' 174
Befehl 'db2se shape_info' 177
Befehl 'db2se unregister_gc' 178
Befehl 'db2se unregister_spatial_column' 179
Befehl 'db2se upgrade' 180
Befehl 'db2se migrate' 182

Kapitel 17. Gespeicherte Prozeduren 185

Prozedur ST_ALTER_COORDSYS 186
Prozedur ST_ALTER_SRS 188
Prozedur ST_CREATE_COORDSYS 192
Prozedur ST_CREATE_SRS 194
Prozedur ST_DISABLE_AUTOGEOCODING . . . 202
Prozedur ST_DISABLE_DB 203
Prozedur ST_DROP_COORDSYS 205
Prozedur ST_DROP_SRS 206
Prozedur ST_ENABLE_AUTOGEOCODING . . . 207
Prozedur ST_ENABLE_DB 210
Prozedur ST_EXPORT_SHAPE 211
Prozedur ST_IMPORT_SHAPE 215
Prozedur ST_REGISTER_GEOCODER 224
Prozedur ST_REGISTER_SPATIAL_COLUMN . . 228
Prozedur ST_REMOVE_GEOCODING_SETUP . . 230
Prozedur ST_RUN_GEOCODING 232
Prozedur ST_SETUP_GEOCODING 235

Prozedur ST_UNREGISTER_GEOCODER 239
Prozedur ST_UNREGISTER_SPATIAL_COLUMN 240

Kapitel 18. Räumliche Funktionen . . 243

Überlegungen und zugehörige Datentypen zu räumlichen Funktionen 243
Werte vom Typ ST_Geometry als Werte eines untergeordneten Typs behandeln 244
Räumliche Funktionen nach Eingabetyp . . . 245
Kategorien und Verwendungsmöglichkeiten für räumliche Funktionen 247
Konstruktorfunktionen für die Umwandlung in und aus Datenaustauschformate(n) 247
Vergleichsfunktionen für geografische Objekte 253
Funktionen für das Abrufen von Informationen zu Geometrien und Indizes. 258
Funktionen für das Generieren neuer Geometrien aus vorhandenen Geometrien 261
Funktion EnvelopesIntersect 263
MBR Aggregate-Funktionen 265
Funktion ST_AppendPoint 267
Funktion ST_Area 268
Funktion ST_AsBinary 270
Funktion ST_AsGML 272
Funktion ST_AsShape 273
Funktion ST_AsText 274
Funktion ST_Boundary 275
Funktion ST_Buffer 277
MBR Aggregate-Funktionen 280
Union-Gesamtverknüpfungen 281
Funktion ST_Centroid 283
Funktion ST_ChangePoint 284
Funktion ST_Contains 285
Funktion ST_ConvexHull 288
Funktion ST_CoordDim 290
Funktion ST_Crosses 291
Funktion ST_Difference 292
Funktion ST_Dimension 294
Funktion ST_Disjoint 295
Funktion ST_Distance 297
Funktion ST_DistanceToPoint 300
Funktion ST_Endpoint 301
Funktion ST_Envelope 301
Funktion ST_EnvIntersects 303
Funktion ST_EqualCoordsys 304
Funktion ST_Equals 305
Funktion ST_EqualSRS 307
Funktion ST_ExteriorRing 308
Funktion ST_FindMeasure oder ST_LocateAlong 309
Funktion ST_Generalize 310
Funktion ST_GeomCollection 312
Funktion ST_GeomCollFromTxt 314
Funktion ST_GeomCollFromWKB 316
Funktion ST_Geometry 317
Funktion ST_GeometryN 319
Funktion ST_GeometryType 320
Funktion ST_GeomFromText 321
Funktion ST_GeomFromWKB 322
MBR Aggregate-Funktionen 324
Union-Gesamtverknüpfungen 325
Funktion ST_GetIndexParms 327
N 329
Funktion ST_Intersection 330
Funktion ST_Intersects 332
Funktion ST_Is3d 335
Funktion ST_IsClosed 336
Funktion ST_IsEmpty. 337
Funktion ST_IsMeasured 338
Funktion ST_IsRing 339
Funktion ST_IsSimple 340
Funktion ST_IsValid 342
Funktion ST_Length 343
Funktion ST_LineFromText 344
Funktion ST_LineFromWKB 346
Funktion ST_LineString 347
Funktion ST_LineStringN 348
Funktion ST_M. 350
Funktion ST_MaxM 351
Funktion ST_MaxX 352
Funktion ST_MaxY 354
Funktion ST_MaxZ 355
Funktion ST_MBR. 357
Funktion ST_MBRIntersects 358
Funktion ST_LocateBetween oder ST_MeasureBetween 359
Funktion ST_LocateBetween oder ST_MeasureBetween 361
Funktion ST_MidPoint 362
Funktion ST_MinM 363
Funktion ST_MinX 365
Funktion ST_MinY 366
Funktion ST_MinZ 367
Funktion ST_MLineFromText 369
Funktion ST_MLineFromWKB. 370
Funktion ST_MPointFromText 372
Funktion ST_MPointFromWKB 373
Funktion ST_MPolyFromText 374
Funktion ST_MPolyFromWKB. 376
Funktion ST_MultiLineString 378
Funktion ST_MultiPoint 380
Funktion ST_MultiPolygon 381
Funktion ST_NumGeometries 383
Funktion ST_NumInteriorRing 384
Funktion ST_NumLineStrings 385
Funktion ST_NumPoints 386
Funktion ST_NumPolygons 387
Funktion ST_Overlaps 388
Funktion ST_Perimeter 390
Funktion ST_PerpPoints 391
Funktion ST_Point. 394
Funktion ST_PointAtDistance 396
Funktion ST_PointFromText 397
Funktion ST_PointFromWKB 399
Funktion ST_PointN 400
Funktion ST_PointOnSurface 401
Funktion ST_PolyFromText 402
Funktion ST_PolyFromWKB 403
Funktion ST_Polygon. 405
Funktion ST_PolygonN 407
Funktion ST_Relate 408
Funktion ST_RemovePoint 409
Funktion ST_SrsId oder ST_SRID 411

Funktion ST_SrsId oder ST_SRID 412
Funktion ST_SrsName 413
Funktion ST_StartPoint 414
Funktion ST_SymDifference 415
Funktion ST_ToGeomColl 418
Funktion ST_ToLineString 419
Funktion ST_ToMultiLine 420
Funktion ST_ToMultiPoint 421
Funktion ST_ToMultiPolygon 422
Funktion ST_ToPoint 423
Funktion ST_ToPolygon 424
Funktion ST_Touches 425
Funktion ST_Transform 427
Funktion ST_Union 429
Funktion ST_Within 431
Funktion ST_WKBToSQL 434
Funktion ST_WKTToSQL 435
Funktion ST_X 436
Funktion ST_Y 437
Funktion ST_Z 439
Union-Gesamtverknüpfungen 440

Kapitel 19. Umsetzungsgruppen . . . 443
Umsetzungsgruppe ST_WellKnownText 443
Umsetzungsgruppe ST_WellKnownBinary 444
Umsetzungsgruppe ST_Shape 446
Umsetzungsgruppe ST_GML 447

Kapitel 20. Unterstützte Datenformate 449
WKT-Darstellung (WKT = Well-Known Text) . . . 449
WKB-Darstellung (WKB = Well-Known Binary) . . 454
Formdarstellung 456
GML-Darstellung (GML = Geography Markup Language) 456

Kapitel 21. Unterstützte Koordinatensysteme. 457
Syntax von Koordinatensystemen. 457
Unterstützte lineare Einheiten 459
Unterstützte Winkeleinheiten 459
Unterstützte Sphäroide 460
Unterstützte Nullmeridiane. 462
Unterstützte Kartenprojektionen 462

Anhang A. Übersicht über technische Informationen zu DB2. 465
Bibliothek mit technischen Informationen zu DB2 im Hardcopy- oder PDF-Format 466
Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor 468
Zugriff auf verschiedene Versionen des DB2 Information Center 468
Aktualisieren des auf Ihrem Computer oder Intranet-Server installierten DB2 Information Center . . 469
Manuelles Aktualisieren des auf Ihrem Computer oder Intranet-Server installierten DB2 Information Center. 470
DB2-Lernprogramme 472
Informationen zur Fehlerbehebung in DB2 . . . 473
Bedingungen 473

Anhang B. Bemerkungen 475

Index 479

Kapitel 1. Verwendungszweck von DB2 Spatial Extender

Mit DB2 Spatial Extender können Sie räumliche Informationen zu geografischen Objekten generieren und analysieren sowie die Daten speichern und verwalten, auf denen diese Informationen basieren. Ein geografisches Objekt (im Rahmen der vorliegenden Informationen wird auch die Kurzform Objekt verwendet) ist jedes Objekt in der realen Welt, dessen Position identifiziert werden kann, bzw. jedes Objekt, das an einer identifizierbaren Position vorhanden sein könnte. Bei einem Objekt kann es sich um Folgendes handeln:

- Ein Objekt (also ein konkretes Element beliebigen Typs), beispielsweise ein Fluss, ein Wald oder ein Gebirgszug.
- Ein Bereich, beispielsweise die Sicherheitszone um einen gefährlichen Standort herum oder aber der Marketingbereich, der durch ein bestimmtes Unternehmen abgedeckt wird.
- Ein Ereignis, das an einem genau bestimmbaren Standort eintritt, beispielsweise ein Verkehrsunfall, der an einer bestimmten Kreuzung stattfindet, oder aber eine Verkaufstransaktion in einer bestimmten Verkaufsstelle.

Objekte gibt es in unterschiedlichen Umgebungen. Die Objekte, die in der vorstehenden Liste angegeben wurden (Fluss, Wald, Gebirgszug), gehören beispielsweise zur natürlichen Umgebung. Andere Objekte wie Städte, Gebäude und Büros gehören zur Infrastrukturumgebung. Wiederum andere Objekte (z. B. Parks, zoologische Gärten und Ackerland) stellen eine Kombination aus natürlicher und Infrastrukturumgebung dar.

In den folgenden Erläuterungen bezieht sich der Terminus räumliche Informationen auf die Art von Informationen, die DB2 Spatial Extender den Benutzern zur Verfügung stellt. Dies sind im Wesentlichen Fakten und grafische Darstellungen zu den Positionen geografischer Objekte. Beispiele für räumliche Informationen:

- Positionen geografischer Objekte auf der Landkarte (beispielsweise Längen- und Breitengradwerte, die die Lage von Städten definieren)
- Positionen geografischer Objekte relativ zu anderen Objekten (z. B. Standorte von Krankenhäusern in Städten oder die Nähe von Wohngebieten zu Erdbebengebieten)
- Beziehungen zwischen verschiedenen geografischen Objekten (z. B. Informationen darüber, dass sich ein Fluss-System in einer bestimmten Region befindet oder dass bestimmte Brücken in dieser Region über die Zuflüsse dieses Fluss-Systems führen)
- Maße, die für ein oder mehrere geografische Objekte (z. B. die Entfernung zwischen einem Bürogebäude und dem Rand des Grundstücks oder den Umkreis eines Vogelschutzgebiets) gelten

Räumliche Informationen können - für sich selbst genommen oder in Verbindung mit herkömmlichen relationalen Daten - als Unterstützung für Aktivitäten dienen, beispielsweise bei der Definition von Bereichen, in denen Dienstleistungen angeboten werden, oder bei der Ermittlung von potenziellen Absatzbereichen. Der Leiter einer Sozialbehörde muss beispielsweise überprüfen, welche Antragsteller und Empfänger von Unterstützungsleistungen tatsächlich im Zuständigkeitsbereich seiner Behörde wohnen. DB2 Spatial Extender kann diese Informationen aus der Angabe des Zuständigkeitsbereichs und den Adressen der Personen ableiten.

Denkbar wäre aber auch der Fall, in dem der Besitzer einer Restaurantkette in anderen Orten der Umgebung weitere Filialen eröffnen möchte. Um geeignete Standorte für neue Restaurants zu ermitteln, muss er Fragen wie die folgenden beantworten: Wo in diesen Städten sind die typischen Gäste für meine Restaurants konzentriert? Wo liegen die wichtigen Zufahrtsstraßen? Wo ist die Kriminalität am niedrigsten? Wo befinden sich die Restaurants der Konkurrenz? DB2 Spatial Extender und DB2 können Informationen erzeugen, mit denen diese Fragen beantwortet werden können. Außerdem können Front-End-Tools (wenngleich nicht zwingend erforderlich) eine Rolle spielen. Zur Veranschaulichung das folgende Beispiel: Ein Darstellungstool kann durch DB2 Spatial Extender erzeugte Informationen (z. B. konzentriertes Vorkommen von Kunden und Nähe wichtiger Zufahrtsstraßen zu vorgeschlagenen Restaurants) grafisch in Form einer Landkarte darstellen. Business-Intelligence-Tools können zugehörige Informationen - beispielsweise Namen und Beschreibungen von Restaurants der Konkurrenz - in Berichtsform erstellen.

Darstellungsweise von geografischen Objekten durch Daten

In DB2 Spatial Extender kann ein geografisches Objekt durch ein oder mehrere Datenelemente dargestellt werden, z. B. durch die Datenelemente in der Zeile einer Tabelle. (Als Datenelement werden die Werte bezeichnet, die in einer Zelle einer relationalen Tabelle gespeichert sind.) Am Beispiel von Gewerbegebieten und Wohngebieten lässt sich Folgendes feststellen: In Abb. 1 steht jede Zeile der Tabelle BRANCHES für eine Zweigstelle einer Bank. Analog steht jede Zeile der Tabelle CUSTOMERS in Abb. 1 als Ganzes für einen Kunden der Bank. Eine Untergruppe jeder Zeile (insbesondere die Datenelemente für die Kundenadresse) stellt den Wohnort des Kunden dar.

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

Abbildung 1. Zur Darstellung von geografischen Objekten verwendete Daten. Die Datenzeile in der Tabelle BRANCHES steht für eine Zweigstelle einer Bank. Die Adressdaten in der Tabelle CUSTOMERS stehen für den Wohnort des Kunden. Namen und Adressen in beiden Tabellen sind frei erfunden.

Die Tabellen in Abb. 1 enthalten Daten, die die Zweigstellen und Kunden der Bank kennzeichnen und beschreiben. In den vorliegenden Erläuterungen werden solche Daten als Geschäftsdaten bezeichnet.

Eine Untermenge der Geschäftsdaten (die Werte, die die Adresse der Zweigstelle und des Kunden angeben) können in Werte umgesetzt werden, aus denen räumliche Daten generiert werden. In dem Beispiel in Abb. 1 lautet die Adresse einer Filiale 92467 Airzone Blvd., San Jose, CA 95141, USA. Die Adresse eines Kunden lautet 9 Concourt Circle, San Jose, CA 95141, USA. DB2 Spatial Extender kann diese Adressen in Werte umsetzen, die angeben, wo sich die Zweigstelle und der Wohnort des Kunden in Bezug aufeinander befinden. Abb. 2 auf Seite 3 zeigt die Tabellen BRANCHES und CUSTOMERS mit den neuen Spalten, die solche Werte aufnehmen können.

BRANCHES

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

CUSTOMERS

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

Abbildung 2. Tabellen mit hinzugefügten Spalten für räumliche Daten. In jeder Tabelle wird die Spalte LOCATION die Koordinaten zu der jeweiligen Adresse enthalten.

Da räumliche Informationen aus den Datenelementen abgeleitet werden, die in der Spalte LOCATION gespeichert sind, werden diese Datenelemente in den vorliegenden Erläuterungen als räumliche Daten bezeichnet.

Charakter räumlicher Daten

Räumliche Daten werden aus Koordinaten gebildet, die eine bestimmte Position definieren. DB2 Spatial Extender arbeitet mit zweidimensionalen Koordinaten, die mithilfe von X- und Y- oder Längen- und Breitengradwerten angegeben werden.

Als Koordinate wird ein numerischer Wert bezeichnet, der eine der folgenden Angaben definiert:

- Eine Position auf einer Achse relativ zu einem Ursprungspunkt in einer angegebenen Längeneinheit.
- Eine Richtung relativ zu einer Basislinie oder -ebene in einem angegebenen Winkelmaß.

Beim Breitengrad handelt es sich beispielsweise um eine Koordinate, die einen Winkel relativ zur Äquatorialebene (normalerweise in Grad) angibt. Beim Längengrad handelt es sich hingegen um eine Koordinate, die einen Winkel relativ zum Greenwich-Meridian angibt, der normalerweise ebenfalls in Grad definiert ist. Auf einer Karte wird die Position des Yellowstone-Nationalparks deswegen mit 44,45 Grad nördlicher Breite (relativ zum Äquator) und 110,40 Grad westlicher Länge (relativ zum Greenwich-Meridian) angegeben. Genauer gesagt geben diese Koordinaten den Mittelpunkt des Yellowstone-Nationalparks in den USA an.

Die Definitionen von Breiten- und Längengraden, deren Punkte, Linien und Bezugsebenen, Maßeinheiten und andere zugehörige Parameter werden zusammen als Koordinatensystem bezeichnet. Koordinatensysteme können auch auf anderen Werten als den Breiten- und Längengradwerten basieren. Diese Koordinatensysteme verfügen über eigene Punkte, Linien und Bezugsebenen, Maßeinheiten und zusätzlichen Parametern (z. B. zur Projektionstransformation).

Das einfachste räumliche Datenelement besteht aus einem einzelnen Koordinatenpaar, das die Position einer einzelnen geografischen Position definiert. Ein umfangreicheres räumliches Datenelement besteht aus mehreren Koordinaten, die einen linearen Verlauf wie etwa eine Straße oder einen Fluss definieren. Eine dritte Art besteht aus Koordinaten, die die Begrenzung eines Gebiets definieren, beispielsweise die Grenze eines Grundstücks oder eines Überschwemmungsgebiets.

Jedes räumliche Datenelement ist eine Instanz eines räumlichen Datentyps. Der Datentyp für zwei Koordinaten, die einen Einzelstandort kennzeichnen, ist ST-

_Point. Als Datentyp für Koordinaten, die einen linearen Verlauf definieren, wird ST-LineString verwendet, und als Datentyp für Koordinaten, die die Begrenzung eines Bereichs definieren, ST_Polygon. Diese Typen sowie andere räumliche Datentypen sind strukturierte Typen, die zu einer gemeinsamen Hierarchie gehören.

Quellen für räumliche Daten

Räumliche Daten können mithilfe verschiedener Methoden abgerufen werden.

Sie können räumliche Daten auf folgende Arten erhalten:

- Ableitung aus Geschäftsdaten
- Generierung mit räumlichen Funktionen
- Import aus externen Quellen

Geschäftsdaten als Quellendaten verwenden

DB2 Spatial Extender kann räumliche Daten aus Geschäftsdaten ableiten, beispielsweise aus Adressen. Dieser Prozess wird als Geocodierung bezeichnet.

Zur Veranschaulichung der betreffenden Sequenz betrachten Sie Abb. 2 auf Seite 3 als Darstellung des „Vorher-Status" und Abb. 3 als Darstellung des „Nachher-Status". Abb. 2 auf Seite 3 zeigt, dass die beiden Tabellen BRANCHES und CUSTOMERS jeweils eine Spalte enthalten, die räumliche Daten aufnehmen kann. Angenommen, die Adressen in diesen Tabellen werden mit DB2 Spatial Extender geocodiert, um Koordinaten dieser Adressen zu erhalten. Anschließend werden die Koordinaten in den Spalten platziert. Abb. 3 zeigt das Ergebnis dieser Operation.

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

Abbildung 3. Tabellen mit räumlichen Daten, die aus Quellendaten abgeleitet wurden. Die Spalte LOCATION in der Tabelle CUSTOMERS enthält Koordinaten, die aus der Adresse in den Spalten ADDRESS, CITY, POSTAL CODE, STATE_PROV und COUNTRY abgeleitet wurden. Ebenso enthält die Spalte LOCATION in der Tabelle BRANCHES Koordinaten, die aus der Adresse in den Spalten ADDRESS, CITY, POSTAL CODE, STATE_PROV und COUNTRY dieser Tabelle abgeleitet wurden.

DB2 Spatial Extender verwendet eine als geocoder bezeichnete Funktion, mit der Geschäftsdaten in Koordinaten umgesetzt werden können, um die Verarbeitung der Daten mithilfe räumlicher Funktionen zu ermöglichen.

Funktionen zum Generieren von räumlichen Daten verwenden

Mit bestimmten Funktionen können Sie aus Eingabedaten räumliche Daten generieren.

Räumliche Daten können nicht nur durch Geocoder, sondern auch durch andere Funktionen generiert werden. Die Bank, deren Zweigstellen in der Tabelle BRANCHES definiert sind, möchte beispielsweise wissen, wie viele Kunden im Umkreis von zehn Kilometern der einzelnen Zweigstellen wohnen. Bevor die Bank diese In-

formationen aus der Datenbank abrufen kann, muss die Zone definiert werden, die in einem angegebenen Umkreis um die einzelnen Zweigstellen liegt. Die Funktion ST_Buffer von DB2 Spatial Extender kann eine solche Definition erstellen. Mit den Koordinaten der einzelnen Zweigstellen als Eingabe kann ST_Buffer die Koordinaten generieren, die den Umriss der Zonen festlegen. Abb. 4 zeigt die Tabelle BRANCHES mit den von ST_Buffer gelieferten Informationen.

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

Abbildung 4. Tabelle, die neue räumliche Daten enthält, die aus vorhandenen räumlichen Daten abgeleitet wurden. Die Koordinaten in der Spalte SALES_AREA wurden mit der Funktion ST_Buffer aus den Koordinaten in der Spalte LOCATION abgeleitet. Wie die Koordinaten in der Spalte LOCATION sind auch diejenigen in der Spalte SALES_AREA nur simuliert und stellen keine realen Koordinaten dar.

Neben ST_Buffer bietet DB2 Spatial Extender verschiedene andere Funktionen, mit denen neue räumliche Daten aus bereits vorhandenen räumlichen Daten abgeleitet werden können.

Räumliche Daten importieren

Spatial Extender stellt Services zum Importieren räumlicher Daten im Formdateiformat zur Verfügung.

Räumliche Daten im Formdateiformat können aus zahlreichen Quellen über das Internet abgerufen werden. Sie können Daten und Karten für in den USA und weltweit verwendete Objekte wie z. B. Länder, Staaten, Städte, Flüsse etc. herunterladen, indem Sie das Angebot für Beispielkartendaten zu DB2 Spatial Extender auf der Webseite 'Trials and Demos' auswählen, das unter der folgenden Adresse zur Verfügung steht: http://www.ibm.com/software/data/spatial/db2spatial.

Sie können räumliche Daten aus Dateien importieren, die in externen Datenquellen zur Verfügung gestellt werden. Diese Dateien enthalten in der Regel Daten, die Karten zugeordnet werden: Straßennetze, Überschwemmungsgebiete, Erdbebenzonen usw. Durch die Verwendung solcher Daten in Verbindung mit den generierten räumlichen Daten können Sie die verfügbaren räumlichen Informationen erweitern. Wenn eine Katastrophenschutzbehörde beispielsweise ermitteln will, welchen Risiken ein Wohngebiet ausgesetzt ist, könnte mit ST_Buffer eine Zone um das Gebiet herum definiert werden. Anschließend könnten dann Daten zu Überschwemmungsgebieten und Erdbebenzonen importiert werden, um festzustellen, welche dieser Risiken in dieser Zone vorliegen.

Funktionen, räumliche Informationen, räumliche Daten und Geometrien - Zusammenhänge

Es wird ein zusammenfassender Überblick über die unterschiedlichen Basiskonzepte, die den Operationen von DB2 Spatial Extender zugrunde liegen, vermittelt. Hierzu gehören geografische Objekte, räumliche Informationen und Daten sowie Geometrien.

Mit DB2 Spatial Extender können Sie Fakten und Formen von realen Objekten erhalten, die geografisch - also in Bezug auf ihre Position auf dem Globus bzw. in einer Region der Erde - definiert werden können. In der DB2-Dokumentation wer-

den solche Fakten und Formen als *räumliche Informationen* und die entsprechenden realen Objekte als *geografische Objekte* (in der vorliegenden Dokumentation auch kurz *Objekte* genannt) bezeichnet.

Beispielsweise könnten Sie mit DB2 Spatial Extender ermitteln, ob sich ein bewohntes Gebiet mit dem Standortvorschlag für eine Müllhalde überlappt. Die bewohnten Gebiete und der Standortvorschlag sind Objekte. Die räumliche Information dieses Beispiels wäre die Feststellung, ob sich diese beiden Objekte überlappen. Wird eine Überlappung festgestellt, wäre auch das Ausmaß dieser Überlappung eine räumliche Information.

Zur Erzeugung räumlicher Informationen muss DB2 Spatial Extender Daten verarbeiten, die die Standorte von Objekten definieren. Solche Daten, die als *räumliche Daten* bezeichnet werden, bestehen aus Koordinaten, die die Standorte auf einer Karte oder einer ähnlichen Projektion angeben. Wenn DB2 Spatial Extender beispielsweise ermitteln soll, ob sich ein Objekt mit einem anderen überlappt, müssen die Koordinaten der entsprechenden Objekte miteinander verglichen werden.

In der Technologie der räumlichen Informationen werden Objekte allgemein durch Symbole dargestellt, die als *Geometrien* bezeichnet werden. Geometrien bestehen aus einem optischen und einem mathematischen Teil. Zunächst soll der optische Aspekt betrachtet werden. Das Symbol für ein Objekt, das eine Breitenausdehnung aufweist (z. B. ein Park oder eine Stadt), ist eine mehrseitige Form. Eine solche Geometrie wird als *Polygon* bezeichnet. Das Symbol für ein lineares Objekt, beispielsweise einen Fluss oder eine Straße, ist eine Linie. Solche Geometrien werden als *Linienfolge* bezeichnet.

Eine Geometrie verfügt über Eigenschaften, die mit den Fakten zu dem dargestellten Objekt übereinstimmen. Viele dieser Eigenschaften können mathematisch ausgedrückt werden. Beispielsweise bilden die Koordinaten eines Objekts zusammen eine der Eigenschaften für die Geometrie des Objekts. Eine weitere Eigenschaft, die *Dimension*, ist ein numerischer Wert, mit dem angegeben wird, ob das Objekt eine Längen- oder Breitenausdehnung aufweist.

Räumliche Daten und bestimmte räumliche Informationen können als Geometrien betrachtet werden. Zur Verdeutlichung soll erneut das zuvor beschriebene Beispiel der bewohnten Gebiete und des Standortvorschlags für eine Müllhalde aufgegriffen werden. Die räumlichen Daten für die bewohnten Gebiete enthalten Koordinaten, die in der Spalte einer DB2-Datenbanktabelle gespeichert sind. Konventionsgemäß werden gespeicherte Angaben nicht einfach nur als Daten, sondern als echte Geometrien betrachtet. Da bewohnte Gebiete Breitenausdehnungen aufweisen, werden diese Geometrien als Polygone dargestellt.

Wie räumliche Daten werden auch bestimmte räumliche Informationen als Geometrien betrachtet. Wenn DB2 Spatial Extender beispielsweise ermitteln soll, ob sich ein bewohntes Gebiet mit einem Standortvorschlag für eine Müllhalde überlappt, müssen die Koordinaten des Polygons, das den Standort symbolisiert, mit den Koordinaten der Polygone für die bewohnten Gebiete verglichen werden. Die resultierenden Informationen (in diesem Fall die sich überlappenden Bereiche) werden ebenfalls als Polygone betrachtet, also als Geometrien mit Koordinaten, Dimensionen und weiteren Eigenschaften.

Kapitel 2. Geometrien

Die in DB2 Spatial Extender verwendete praxisorientierte Definition des Begriffs "Geometrie" lautet: „Das Modell eines geografischen Objekts".

Im Lexikon wird der Begriff *Geometrie* sinngemäß als „Teilgebiet der Mathematik definiert, das die Beziehungen, Eigenschaften und Abmessungen von Körpern, Flächen, Linien und Winkeln untersucht", ferner als „Wissenschaft, die sich mit den Eigenschaften und Beziehungen von Ausmaßen beschäftigt", sowie als „Wissenschaft der räumlichen Beziehungen". Das Wort Geometrie wird außerdem verwendet, um die geometrischen Objekte zu bezeichnen, mit deren Hilfe Kartografen seit mehr als einem Jahrtausend die Erde darstellen. Eine abstrakte Definition dieser neuen Bedeutung für den Begriff "Geometrie" lautet wie folgt: Ein Punkt oder eine Gruppe von Punkten, die ein Objekt auf der Erdoberfläche darstellen.

In DB2 Spatial Extender kann das Modell als Koordinaten des Objekts ausgedrückt werden. Das Modell umfasst Informationen; die Koordinaten kennzeichnen beispielsweise die Position des Objekts in Bezug auf feste Referenzpunkte. Darüber hinaus kann das Modell verwendet werden, um Informationen zu erstellen; die Funktion ST_Overlaps kann beispielsweise die Koordinaten von zwei benachbarten Regionen als Eingabe verwenden und als Ausgabe Informationen dazu zurückgeben, ob sich die Regionen überlappen oder nicht.

Die Koordinaten eines Objekts, das von einer Geometrie dargestellt wird, werden als Eigenschaften der Geometrie betrachtet. Unterschiedliche Arten von Geometrien haben auch weitere Eigenschaften, z. B. Bereich, Länge und Begrenzung.

Die von DB2 Spatial Extender unterstützten Geometrien bilden eine Hierarchie, die in der folgenden Abbildung dargestellt ist. Die Geometrienhierarchie wird im Dokument "OpenGIS Simple Features Specification for SQL" des OpenGIS Consortium, Inc. (OGC) definiert. Sieben Elemente der Hierarchie sind instanziierbar. Dies bedeutet, dass sie mit speziellen Koordinatenwerten definiert und optisch wie in der Abbildung dargestellt werden können.

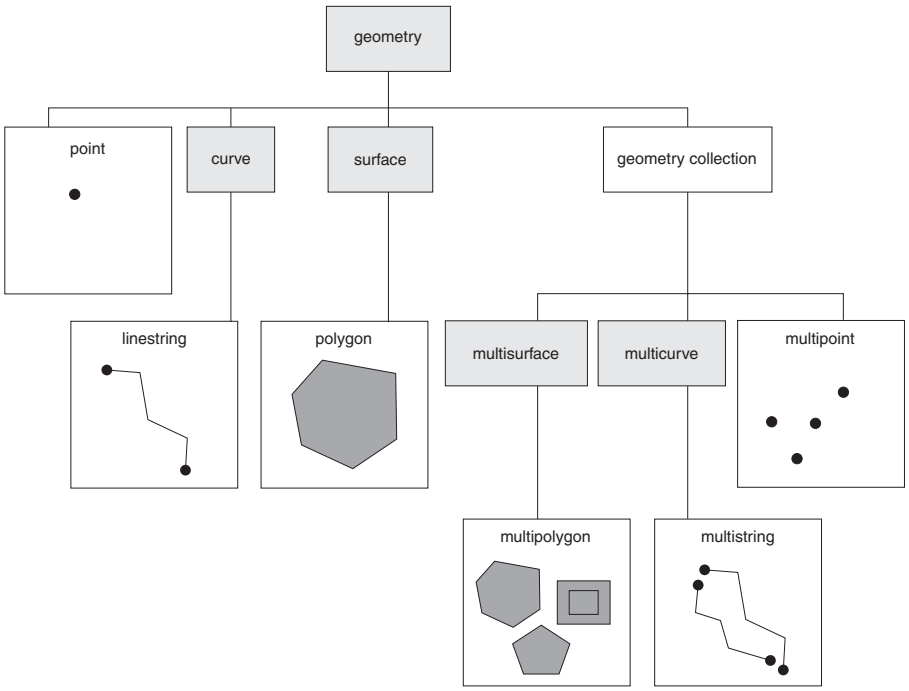


Abbildung 5. Von DB2 Spatial Extender unterstützte Geometriehierarchie. Die instanziierbaren Geometrien in dieser Abbildung sind mit Beispielen für ihre optische Wiedergabe dargestellt.

Die von DB2 Spatial Extender unterstützten räumlichen Datentypen sind Implementierungen der in der Abbildung dargestellten Geometrien.

Die Abbildung zeigt, dass eine Superklasse namens Geometrie den Ausgangspunkt der Hierarchie (Stammelement) bildet. Der Stammelementtyp und andere eigene untergeordnete Typen in der Hierarchie sind nicht instanziierbar. Außerdem können Benutzer eigene untergeordnete Typen definieren, die instanziierbar oder nicht instanziierbar sein können.

Die untergeordneten Typen sind in zwei Kategorien unterteilt: die untergeordneten Typen der Basisgeometrien und die untergeordneten Typen der homogenen Gruppen.

Die Basisgeometrien umfassen folgende Elemente:

Punkte (Points)
Ein einzelner Punkt. Punkte stellen eindeutige Objekte dar, die den Ort belegen, an dem sich eine Ost-West-Koordinatenlinie (z. B. ein Breitenkreis) mit einer Nord-Süd-Koordinatenlinie (z. B. einem Meridian) schneidet. Angenommen, die Notation einer Weltkarte zeigt, dass sich jede Stadt auf der Karte am Schnittpunkt eines Breitenkreises und eines Meridians befindet. Auf dieser Karte könnte jede Stadt durch einen Punkt dargestellt sein.

Linienfolgen (Linestrings)
Eine Linie zwischen zwei oder mehr Punkten. Hierbei muss es sich nicht um eine gerade Linie handeln. Linienfolgen stellen lineare geografische Objekte wie z. B. Straßen, Kanäle und Rohrleitungen dar.

Polygone (Polygons)
Ein Vieleck oder eine Fläche innerhalb eines Vielecks. Polygone stellen mehrseitige geografische Objekte dar, z. B. Erholungsgebiete, Wälder und Naturschutzgebiete.

Die homogenen Gruppen umfassen folgende Elemente:

Mehrpunktangaben (Multipoints)
Eine Geometriengruppe mit mehreren Punkten. Mehrpunktangaben stellen mehrteilige Objekte dar, deren Komponenten sich jeweils am Schnittpunkt einer Ost-West-Koordinatenlinie und einer Nord-Süd-Koordinatenlinie befinden (beispielsweise eine Inselkette, deren einzelne Inseln sich jeweils am Schnittpunkt eines Breitenkreises und eines Meridians befinden).

Mehrlinienfolgen (Multilinestrings)
Eine Geometriegruppe mit mehreren Kurven und mehreren Linienfolgen. Mehrlinienfolgen stellen mehrteilige Objekte dar, die aus mehreren Linienfolgen bestehen, z. B. Fluss-Systeme und Autobahnnetze.

Multipolygone (Multipolygons)
Eine aus mehreren Flächen bestehende Geometriengruppe mit mehreren Polygonen. Multipolygone (Mehrpunktflächen) dienen zur Darstellung mehrteiliger Objekte, die aus mehrseitigen Einheiten oder Komponenten bestehen. Ein Beispiel für ein Multipolygon ist das Ackerland einer bestimmten Region oder eine aus mehreren Seen bestehende Seenplatte.

Homogene Gruppen sind, wie der Name schon andeutet, Gruppen von Basisgeometrien. Neben den gemeinsamen Eigenschaften der Basisgeometrie haben homogene Gruppen auch eigene Eigenschaften.

Eigenschaften von Geometrien

Der folgende Abschnitt beschreibt Eigenschaften, die für Geometrien verfügbar sind.

Die Eigenschaften lauten wie folgt:
- Typ der Geometrie
- Koordinaten der Geometrie
- Innenbereich, Begrenzung und Außenbereich einer Geometrie
- Qualität (einfach oder nicht einfach)
- Qualität (leer oder nicht leer)
- Minimal einschließendes Rechteck oder Hülle einer Geometrie
- Dimension
- Kennung des räumlichen Bezugsystems, dem eine Geometrie zugeordnet ist

Geometrietypen

Jede Geometrie gehört zu einem Typ in der Hierarchie der Geometrien, die von DB2 Spatial Extender unterstützt werden.

Die Typen in der Hierarchie, die mit bestimmten Koordinatenwerten definiert werden müssen, sind im Einzelnen:
- Punkte (Points)
- Linienfolgen (Linestrings)
- Polygone (Polygons)
- Geometriegruppen (Geometry Collections)
- Mehrpunktangaben (Multipoints)
- Mehrlinienfolgen (Multilinestrings)
- Multipolygone (Multipolygons)

Koordinaten der Geometrie

Alle Geometrien enthalten mindestens eine X-Koordinate und eine Y-Koordinate, sofern es sich nicht um leere Geometrien handelt, die überhaupt keine Koordinaten enthalten.

Darüber hinaus kann eine Geometrie eine oder mehrere Z-Koordinaten und M-Koordinaten beinhalten. X-, Y-, Z- und M-Koordinaten werden als Zahlen mit doppelter Genauigkeit dargestellt. Die nachfolgenden Unterabschnitte enthalten folgende Erläuterungen:

- X- und Y-Koordinaten
- Z-Koordinaten
- M-Koordinaten

X- und Y-Koordinaten

Ein X-Koordinatenwert kennzeichnet eine Position relativ zu einem Bezugspunkt im Osten oder Westen. Ein Y-Koordinatenwert gibt eine Position bezogen auf einen Bezugspunkt im Norden oder Süden an.

Z-Koordinaten

Bei Geometrien mit einer zugeordneten Höhe oder Tiefe kann jeder Punkt der Geometrie eines Objekts eine optionale Z-Koordinate enthalten, die eine Höhe oder Tiefe gegenüber der Erdoberfläche angibt.

M-Koordinaten

Eine M-Koordinate (Bemaßung) ist ein Wert, der Informationen über ein geografisches Objekt enthält und der zusammen mit den Koordinaten für die Position dieses Objekts gespeichert wird.

Angenommen, in einer Anwendung sollen Autobahnen dargestellt werden. Wenn Ihre Anwendung Werte verarbeiten soll, die lineare Entfernungen (Luftlinie) angeben, können Sie diese Werte zusammen mit den Koordinaten speichern, die Standorte entlang der Autobahn angeben. M-Koordinaten werden als Zahlen mit doppelter Genauigkeit dargestellt.

Innenbereich, Begrenzung und Außenbereich

Alle Geometrien belegen eine Position im Raum, die durch ihre Innenbereiche, ihre Begrenzungen und ihre Außenbereiche definiert ist.

Der Außenbereich einer Geometrie ist der gesamte Raum, der nicht von der Geometrie belegt wird. Die Begrenzung einer Geometrie dient als Schnittstelle zwischen ihrem Innen- und ihrem Außenbereich. Der Innenbereich ist der von der Geometrie eingenommene Raum.

Einfache oder nicht einfache Geometrien

Die Werte einiger untergeordneter Typen von Geometrien (Linienfolgen, Mehrpunktangaben und Mehrlinienfolgen) sind entweder einfache Werte oder nicht einfache Werte. Eine Geometrie ist einfach, wenn sie alle topologischen Regeln einhält, die für ihren untergeordneten Typ gelten. Werden nicht alle diese Regeln eingehalten, ist die Geometrie nicht einfach.

Eine Linienfolge ist einfach, wenn sie ihren eigenen Innenbereich nicht schneidet. Eine Mehrpunktangabe ist einfach, wenn keines ihrer Elemente den gleichen Koordinatenraum belegt. Punkte, Oberflächen, Mehrfachoberflächen und leere Geometrien sind immer einfach.

Geschlossene Geometrien

Eine Kurve ist geschlossen, wenn ihr Startpunkt mit dem Endpunkt identisch ist. Eine Mehrfachkurve ist geschlossen, wenn jedes ihrer Elemente geschlossen ist. Ein Ring ist eine einfache geschlossene Kurve.

Leere oder nicht leere Geometrien

Eine Geometrie ist leer, wenn sie keine Punkte enthält. Die Hülle, die Begrenzung, der Innenbereich und der Außenbereich einer leeren Geometrie sind nicht definiert und werden als Nullwert dargestellt.

Eine leere Geometrie ist immer einfach. Leere Polygone und Multipolygone haben die Fläche 0.

Minimal einschließendes Rechteck (MBR)

Das minimal einschließende Rechteck bzw. der minimale Begrenzungsrahmen (MBR = Minimal Bounding Rectangle) einer Geometrie ist die Begrenzungsgeometrie, die durch die minimalen und maximalen Koordinaten (X,Y) gebildet wird.

Mit Ausnahme der folgenden Sonderfälle bilden die MBRs einer Geometrie immer ein einschließendes Rechteck:

- Das MBR eines Punkts ist der Punkt selbst, da seine minimalen und maximalen X-Koordinaten identisch sind und seine minimalen und maximalen Y-Koordinaten identisch sind.
- Das MBR einer horizontalen oder vertikalen Linienfolge ist eine Linienfolge, die durch die Begrenzung (die Endpunkte) der Quellenlinienfolge gebildet wird.

Dimension von Geometrien

Eine Geometrie kann einen Dimensionswert haben, der angibt, ob die Geometrie leer ist oder eine Länge bzw. Fläche hat.

Die Dimensionswerte sind im Einzelnen:

-1 Gibt an, dass eine Geometrie leer ist.

0 Gibt an, dass eine Geometrie keine Länge und die Fläche 0 (null) hat.

1 Gibt an, dass eine Geometrie eine Länge größer 0 (null) und die Fläche 0 (null) hat.

2 Gibt an, dass eine Geometrie eine Fläche größer 0 (null) hat.

Die untergeordneten Typen Punkt und Mehrpunktangabe haben die Dimension 0. Punkte stellen dimensionale Objekte dar, die mit einem einzigen Koordinatentupel modelliert werden können. Mehrpunktsubtypen hingegen stellen Daten dar, die mit einer Gruppe von Punkten modelliert werden müssen.

Die untergeordneten Typen Linien und Mehrlinienfolge haben die Dimension 1. In diesen untergeordneten Typen werden Straßenabschnitte, verzweigte Fluss-Systeme und andere lineare Objekte gespeichert.

Die untergeordneten Typen Polygon und Multipolygon haben die Dimension 2. Objekte, deren Umfang einen definierbaren Bereich umschließt, wie beispielsweise Wälder, Flächen oder Seen, können mit dem Datentyp Polygon oder Multipolygon dargestellt werden.

Kennung des räumlichen Bezugssystems

Die numerische Kennung für ein räumliches Bezugssystem bestimmt, welches räumliche Bezugssystem zur Darstellung der Geometrie verwendet wird.

Alle in der Datenbank bekannten räumlichen Bezugssysteme können über die Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgerufen werden.

Kapitel 3. DB2 Spatial Extender verwenden

Die Unterstützung und Verwendung von DB2 Spatial Extender umfasst zwei Hauptaktivitäten: Das Konfigurieren von DB2 Spatial Extender und das Arbeiten mit Projekten, die räumliche Daten verwenden. Dieser Abschnitt enthält eine Einführung in die Schnittstellen, die Sie zur Ausführung von Aufgaben mit räumlichen Daten verwenden können.

Schnittstellen zu DB2 Spatial Extender und ihre Funktionalität

Es gibt eine Vielzahl von Schnittstellen, die zum Einrichten von DB2 Spatial Extender verwendet werden können.

Sie können die folgenden Schnittstellen verwenden, um DB2 Spatial Extender einzurichten und Projekte zu erstellen, die mit räumlichen Daten arbeiten:

- Anwendungsprogramme, die gespeicherte Prozeduren von DB2 Spatial Extender aufrufen.
- Ein Befehlszeilenprozessor (Command Line Processor - CLP), der von DB2 Spatial Extender zur Verfügung gestellt wird. Der Befehlszeilenprozessor von DB2 Spatial Extender wird auch als Befehlszeilenprozessor db2se bezeichnet.
- SQL-Abfragen, die Sie über den DB2-Befehlszeilenprozessor oder über ein Anwendungsprogramm übergeben.
- Open-Source-Projekte mit Unterstützung für DB2 Spatial Extender. Zu den Open-Source-Projekten, die diese Unterstützungsfunktion bieten, gehören folgende Komponenten:
  - IBM Data Management Geobrowser for DB2 and Informix ist ein Geobrowser, mit dem räumliche Tabellen und die Ergebnisse der räumlichen Analyse in grafischer Form dargestellt werden können. Weitere Informationen finden Sie im Abschnitt http://www.ibm.com/developerworks/data/tutorials/dm-1103db2geobrowser/index.html.
  - GeoTools () ist eine Java-Bibliothek zum Erstellen räumlicher Anwendungen. Weitere Informationen finden Sie im Abschnitt http://www.geotools.org/.
  - GeoServer ist ein Web-Karten- und Web-Objektserver. Weitere Informationen finden Sie im Abschnitt http://geoserver.org/display/GEOS/Welcome.
  - uDIG ist eine desktopbasierte Darstellungs- und Analyseanwendung für räumliche Daten. Weitere Informationen finden Sie im Abschnitt http://udig.refractions.net/.
  - GIS-Produkte wie z. B. Esri ArcGIS.

DB2 Spatial Extender einrichten

In dieser Prozedur werden die Schritte beschrieben, die Sie ausführen müssen, um DB2 Spatial Extender einzurichten.

Vorgehensweise

Gehen Sie wie folgt vor, um DB2 Spatial Extender einzurichten:

1. Erstellen Sie Pläne, und treffen Sie die nötigen Vorbereitungen (legen Sie fest, welche Projekte erstellt werden sollen und welche Schnittstelle(n) zu verwenden sind, wählen Sie Mitarbeiter für die Verwaltung von DB2 Spatial Extender aus, erstellen Sie Projekte usw.).
2. Installieren Sie DB2 Spatial Extender.
3. Überprüfen Sie die Einstellungen für die Datenbankkonfigurationsparameter und passen Sie sie ggf. an. Sie müssen sicherstellen, dass für Ihre Datenbank ausreichend Hauptspeicher und Speicherbereich für räumliche Funktionen, Protokolldateien und Anwendungen von DB2 Spatial Extender verfügbar ist.
4. Definieren Sie räumliche Ressourcen für die Datenbank. Zu diesen Ressourcen gehören ein Systemkatalog, räumliche Datentypen, räumliche Funktionen, ein Geocoder und andere Objekte. Weitere Informationen finden Sie in Datenbank für räumliche Operationen aktivierenDatenbank für räumliche Operationen aktivieren.

Beispiel

Im folgenden Beispiel werden die Schritte dargestellt, die von einem fiktiven Unternehmen, der Versicherungsgesellschaft "Safe Harbor Real Estate" ausgeführt werden, um DB2 Spatial Extender einzurichten:

1. Die Informationssystemumgebung der Versicherungsgesellschaft "Safe Harbor Real Estate" umfasst ein DB2-Datenbanksystem und ein separates Dateisystem, das für räumliche Daten reserviert ist. Abfrageergebnisse können bis zu einem gewissen Maß Kombinationen von Daten enthalten, die aus beiden Systemen stammen. In einer DB2-Tabelle sind beispielsweise Informationen zum Umsatz gespeichert, und eine Datei im Dateisystem enthält die Standorte der Niederlassungen des Unternehmens. Es ist daher möglich, die Niederlassungen zu ermitteln, die einen angegebenen Umsatz erzielen. Anschließend kann festgestellt werden, wo sich diese Niederlassungen befinden. Die Daten aus den beiden Systemen können jedoch nicht integriert werden (beispielsweise können die Benutzer nicht DB2-Spalten mit den Datensätzen aus dem Dateisystem verknüpfen, und DB2-Dienste wie die Optimierung von Abfragen stehen für das Dateisystem nicht zur Verfügung). Zur Überwindung dieser Nachteile kauft Safe Harbor DB2 Spatial Extender und richtet eine neue Abteilung für die räumliche Entwicklung (kurz "Raumentwicklung" genannt) ein.

   Die erste Aufgabe der Abteilung "Raumentwicklung" besteht darin, DB2 Spatial Extender in die DB2-Umgebung von Safe Harbor zu integrieren:

   - Das Managementteam der Abteilung benennt ein Team für die räumliche Verwaltung, das DB2 Spatial Extender installieren und implementieren soll. Ein anderes Team für die räumliche Analyse soll räumliche Daten generieren und analysieren.
   - Da das Verwaltungsteam umfassende Erfahrungen mit UNIX® besitzt, wird beschlossen DB2 Spatial Extender über den Befehlszeilenprozessor db2se zu verwalten.
   - Da die unternehmerischen Entscheidungen von Safe Harbor hauptsächlich durch die Kundenanforderungen bestimmt sind, beschließt das Managementteam, DB2 Spatial Extender in der Datenbank zu installieren, die Informationen zu den Kunden enthält. Die meisten dieser Informationen sind in der Tabelle CUSTOMERS gespeichert. Die Tabelle CUSTOMERS verfügt über die Spalten LATITUDE und LONGITUDE, die im Rahmen des Adressbereinigungsprozesses mit Daten gefüllt wurden.
2. Das für die räumliche Verwaltung zuständige Team installiert DB2 Spatial Extender auf einer UNIX-Maschine in einer DB2-Umgebung.

3. Ein Mitglied des Teams für die räumliche Verwaltung passt die Kenndaten des Transaktionsprotokolls, die Größe des Zwischenspeichers für Anwendungen sowie die Größe des Zwischenspeichers für die Anwendungssteuerung an und verwendet Werte, die den Anforderungen von DB2 Spatial Extender entsprechen.
4. Das Team für die räumliche Verwaltung definiert die Ressourcen, die für die geplanten Projekte benötigt werden.
   - Ein Mitglied des Teams setzt den Befehl **db2se enable_db** ab, um die Ressourcen anzufordern, die zum Aktivieren der Datenbank für räumliche Operationen benötigt werden. Hierzu gehören der Katalog von DB2 Spatial Extender, räumliche Datentypen, räumliche Funktionen usw.

Nächste Schritte

Nach der Einrichtung von DB2 Spatial Extender können Sie Projekte beginnen, die räumliche Daten verwenden.

Projekte erstellen, die räumliche Daten verwenden

Nach dem Einrichten von DB2 Spatial Extender können Sie Projekte beginnen, die räumliche Daten verwenden. In dieser Prozedur werden die Schritte beschrieben, die zur Erstellung solcher Projekte ausgeführt werden müssen.

Vorbereitende Schritte

- Richten Sie DB2 Spatial Extender ein.

Vorgehensweise

So erstellen Sie ein Projekt, das räumliche Daten verwendet:

1. Erstellen Sie Pläne, und treffen Sie Vorbereitungen (definieren Sie die Projektziele, entscheiden Sie, welche Tabellen und Daten erforderlich sind, legen Sie die zu verwendenden Koordinatensysteme fest usw.).
2. Ermitteln Sie, ob es bereits ein räumliches Bezugssystem gibt, das Ihren Anforderungen gerecht wird. Erstellen Sie ein solches System, wenn dies nicht der Fall ist.

   Ein räumliches Bezugssystem ist eine Gruppe von Parameterwerten, die Folgendes umfasst:
   - Koordinaten, die die größtmögliche Ausdehnung eines Bereichs definieren, der durch einen angegebenen Bereich von Koordinaten angegeben ist. Sie müssen den größtmöglichen Bereich von Koordinaten ermitteln, die über das verwendete Koordinatensystem ermittelt werden können, und ein räumliches Bezugssystem auswählen bzw. erstellen, das diesen Bereich angibt.
   - Der Name des Koordinatensystems, aus dem die Koordinaten abgeleitet werden.
   - Die in mathematischen Operationen verwendeten Faktoren für die Umwandlung von Koordinaten, die als Eingabewerte empfangen wurden. Diese Umwandlung hat das Ziel, die Werte mit der größtmöglichen Effizienz zu verarbeiten. Die Koordinaten werden in der umgewandelten Form gespeichert und an den Benutzer in ihrer ursprünglichen Form zurückgegeben.
3. Erstellen Sie nach Bedarf räumliche Spalten. Wenn Daten in einer räumlichen Spalte durch ein Darstellungstool gelesen werden sollen, müssen Sie in vielen Fällen beachten, dass die Spalte die einzige räumliche Spalte in der Tabelle oder Sicht sein muss, zu der sie gehört. Handelt es sich bei der Spalte um eine von

mehreren räumlichen Spalten in einer Tabelle, besteht alternativ die Möglichkeit, sie in eine Sicht aufzunehmen, die keine weiteren räumlichen Spalten enthält, und die Daten durch das Darstellungstool über diese Sicht lesen zu lassen.

4. Definieren Sie bei Bedarf räumliche Spalten, auf die durch Darstellungstools zugegriffen wird. Hierzu registrieren Sie die Spalten im Katalog von DB2 Spatial Extender. Wenn Sie eine räumliche Spalte registrieren, legt DB2 Spatial Extender die Integritätsbedingung fest, dass alle Daten in der Spalte zum gleichen räumlichen Bezugssystem gehören müssen. Diese Integritätsbedingung gewährleistet die Integrität der Daten und erfüllt somit eine Anforderung der meisten Darstellungstools.
5. Füllen Sie die räumlichen Spalten, indem Sie eine der folgenden Aktionen ausführen:
   a. Importieren Sie die räumlichen Daten, falls dies für das Projekt erforderlich ist.
   b. Führen Sie für ein Projekt, für das eine räumliche Spalte anhand der Spalten LATITUDE und LONGITUDE definiert werden muss, die Anweisung SQL UPDATE mit dem Konstruktor db2gse.ST_Point aus.
6. Vereinfachen Sie bei Bedarf den Zugriff auf räumliche Spalten. Dies umfasst das Definieren von Indizes, die DB2 einen schnellen Zugriff auf die räumlichen Daten ermöglichen, sowie das Definieren von Sichten, über die die Benutzer die in Wechselbeziehung zueinander stehenden Daten effizient abrufen können. Wenn Darstellungstools auf die räumlichen Spalten der Sichten zugreifen sollen, müssen Sie diese Spalten unter Umständen auch für DB2 Spatial Extender registrieren.
7. Analysieren Sie die generierten räumlichen Informationen und zugehörigen Geschäftsinformationen. Diese Aufgabe erfordert ein Abfragen räumlicher Spalten und der dazugehörigen Spalten mit nicht-räumlichen Daten. Bei solchen Abfragen können Sie Funktionen von DB2 Spatial Extender verwenden, die eine Vielzahl unterschiedlicher Informationen zurückgeben. Hierzu gehören beispielsweise Koordinaten, die eine empfohlene Sicherheitszone um einen gefährlichen Standort herum definieren, oder auch den Mindestabstand zwischen diesem Standort und dem nächstgelegenen öffentlichen Gebäude.

Beispiel

Das folgende Beispiel stellt eine Fortsetzung des Beispiels im Abschnitt zum Einrichten von DB2 Spatial Extender dar. Es zeigt die Schritte, die von der Versicherungsgesellschaft "Safe Harbor Real Estate" ausgeführt werden, um ein Projekt für die Integration von Geschäftsdaten und räumlichen Daten zu erstellen.

1. Die Abteilung "Raumentwicklung" bereitet die Entwicklung eines Projektes vor. Beispiel:
   - Das Managementteam legt die Ziele des Projekts fest:
     - Standorte für die Einrichtung neuer Niederlassungen ermitteln
     - Prämien entsprechend der Nähe des Kundenwohnorts zu Gefahrengebieten (Gebieten mit hoher Verkehrsunfallquote, hoher Kriminalitätsrate, Überschwemmungsgebiet, Erdbebengebiet usw.) anpassen
   - Dieses spezielle Projekt betrifft Kunden und Niederlassungen in den USA. Aus diesem Grund entscheidet das Team für die räumliche Verwaltung sich für die Verwendung des Koordinatensystems GCS_NORTH_AMERICAN_1983 für die Vereinigten Staten, das von DB2 Spatial Extender bereitgestellt wird.

- Das Team für die räumliche Verwaltung legt fest, welche Daten zum Erreichen der Projektziele erforderlich sind und welche Tabellen diese Daten enthalten sollen.

2. DB2 Spatial Extender stellt ein räumliches Bezugssystem namens NAD83_SRS_1 zur Verfügung, das zur Verwendung mit dem Koordinatensystem GCS_NORTH_AMERICAN_1983 vorgesehen ist. Das Team für die räumliche Verwaltung beschließt, das System NAD83_SRS_1 zu verwenden.
3. Das Team für die räumliche Verwaltung definiert Spalten, die räumliche Daten enthalten sollen.
   - Das Team überprüft, ob die Tabelle bereits die Spalten LATITUDE und LONGITUDE des Kunden enthält. Die Werte in diesen Spalten werden nacheinander verwendet, um die Umwandlung in die räumlichen Punktwerte durchzuführen.
   - Das Team erstellt die Tabellen OFFICE_LOCATIONS und OFFICE_SALES für die Daten, die gegenwärtig in einem separaten Dateisystem und unter Verwendung eines dem Branchenstandard entsprechenden Formdateiformat gespeichert sind. Diese Daten enthalten die Adressen der Zweigstellen von Safe Harbor, räumliche Daten, die über einen Geocoder aus diesen Adressen abgeleitet wurden, und räumliche Daten, die eine Zone mit einem Umkreis von fünf Meilen um jede Niederlassung definieren. Die durch den Geocoder abgeleiteten Daten werden in der Tabelle OFFICE_LOCATIONS in der Spalte LOCATION abgelegt. Die Daten, die die Zonen definieren, werden in der Spalte SALES_AREA der Tabelle OFFICE_SALES gespeichert.
4. Das Team für die räumliche Verwaltung plant den Einsatz von Darstellungstools, um den Inhalt der Spalten LOCATION und der Spalte SALES_AREA grafisch in Form einer Landkarte wiederzugeben. Daher registriert das Team alle drei Spalten.
5. Das Team für die räumliche Verwaltung füllt die Spalte LOCATION in der Tabelle CUSTOMER, die Tabelle OFFICE_LOCATIONS, die Tabelle OFFICE_SALES und eine neue Tabelle HAZARD_ZONES:
   - Das Team verwendet die Anweisung UPDATE CUSTOMERS SET LOCATION = db2gse.ST_Point(LONGITUDE, LATITUDE,1), um den Wert für LOCATION aus LATITUDE und LONGITUDE einzugeben.
   - Das Team importiert Daten in die Tabellen OFFICE_LOCATIONS und OFFICE_SALES und verwendet dazu den Befehl **db2se import_shape**.
   - Das Team erstellt eine Tabelle HAZARD_ ZONES, registriert ihre räumlichen Spalten und importiert Daten in diese Spalten. Die Daten stammen aus einer Datei, die der Anbieter der Karte zur Verfügung gestellt hat.
6. Das Team für die räumliche Verwaltung erstellt Indizes für die registrierten Spalten. Anschließend erstellt das Team eine Sicht, die Spalten aus den Tabellen CUSTOMERS und HAZARD ZONES verknüpft und die räumlichen Spalten in dieser Sicht registriert.
7. Das Team für die räumliche Analyse führt Abfragen aus, um Informationen abzurufen, mit denen der Standort der neuen Zweigstelle festgelegt und das Anpassen der Prämien entsprechend der Nähe der Kunden zu Gefahrengebieten durchgeführt werden kann.

Kapitel 4. Erste Schritte mit DB2 Spatial Extender

Machen Sie sich mit den Anweisungen für die Installation und Konfiguration von Spatial Extender unter unterstützten Betriebssystemen vertraut. Erfahren Sie außerdem, wie bestimmte Installations- und Konfigurationsfehler behoben werden können, die möglicherweise bei der Arbeit mit DB2 Spatial Extender auftreten.

DB2 Spatial Extender einrichten und installieren

DB2 Spatial Extender muss konfiguriert und installiert werden, um eine Umgebung einzurichten, die räumliche Operationen unterstützt.

Vorbereitende Schritte

Stellen Sie sicher, dass die Installationsvoraussetzungen für DB2 Spatial Extender und DB2-Datenbankprodukte erfüllt sind. Wenn Ihr System eine oder mehrere der Softwarevoraussetzungen nicht erfüllt, schlägt die Installation fehl.

Informationen zu diesem Vorgang

Eine DB2 Spatial Extender-Umgebung besteht aus einem Server und einem Client.

Ein DB2 Spatial Extender-Server besteht aus einer DB2-Datenserverinstallation mit DB2 Spatial Extender-Software.

Ein DB2 Spatial Extender-Client besteht aus einer IBM® Data Server-Client-Installation mit DB2 Spatial Extender-Software.

Datenbanken, die für räumliche Operationen aktiviert sind, befinden sich auf dem Server. Auf sie kann von einem Client aus zugegriffen werden. Auf räumliche Daten in den Datenbanken kann mit gespeicherten DB2 Spatial Extender-Prozeduren und räumlichen Abfragen vom Server oder Client aus zugegriffen werden.

Ebenfalls Teil einer typischen Komponente einer DB2 Spatial Extender-Umgebung ist ein Geobrowser. Geobrowser sind zwar nicht erforderlich, jedoch bei der visuellen Wiedergabe von Ergebnissen räumlicher Abfragen - in der Regel in Form von Karten - nützlich. Geobrowser gehören nicht zum Umfang einer DB2 Spatial Extender-Installation. Sie können jedoch den IBM Data Management Geobrowser for DB2 & Informix herunterladen, um räumliche Daten anzuzeigen.

Vorgehensweise

Gehen Sie wie folgt vor, um DB2 Spatial Extender auf einem DB2-Server oder IBM Data Server-Client zu konfigurieren und zu installieren:

1. Stellen Sie sicher, dass Ihr System alle geltenden Softwarevoraussetzungen erfüllt. Lesen Sie hierzu die Informationen in „Systemvoraussetzungen für die Installation von DB2 Spatial Extender" auf Seite 20.
2. Installieren Sie DB2 Spatial Extender, indem Sie eine der folgenden Tasks ausführen:
   - „Installieren von DB2 Spatial Extender mit dem **DB2-Installationsassistenten** (Windows)" auf Seite 21

- „Installieren von DB2 Spatial Extender mit dem **DB2-Installationsassistenten** (Linux und UNIX)" auf Seite 22
- Installieren eines DB2-Produkts mithilfe einer Antwortdatei (Windows)
- Installieren eines DB2-Produkts mithilfe einer Antwortdatei (Linux und UNIX)

Bei Installationen mithilfe einer Antwortdatei müssen Sie die folgenden Schlüsselwörter für die DB2-Serverinstallation angeben:

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

Sie müssen die folgenden Schlüsselwörter für eine IBM Data Server-Client-Installation angeben:

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

3. Erstellen Sie eine DB2-Instanz, falls nicht bereits eine vorhanden ist. Verwenden Sie hierzu den DB2-Befehl **db2icrt** in einem DB2-Befehlsfenster.
4. Überprüfen Sie, ob die DB2 Spatial Extender-Installation erfolgreich war, indem Sie die DB2 Spatial Extender-Umgebung testen. Lesen Sie hierzu die Informationen in „DB2 Spatial Extender-Installation prüfen" auf Seite 24.
5. Optional: Laden Sie IBM Data Management Geobrowser for DB2 and Informix herunter und installieren Sie das Produkt. Sie können eine kostenlose Kopie unter https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=swg-dm-geobrowser herunterladen.

Zugehörige Informationen:

Räumliche Daten von DB2 mit einem kostenlosen Geobrowser analysieren

Systemvoraussetzungen für die Installation von DB2 Spatial Extender

Vergewissern Sie sich, dass Ihr System alle Voraussetzungen hinsichtlich des Speicherbedarfs der Software erfüllt, bevor Sie DB2 Spatial Extender installieren.

Betriebssysteme

DB2 Spatial Extender kann unter 32-Bit-Betriebssystemen auf Intel-basierten Computern installiert werden:

- Windows
- Linux

DB2 Spatial Extender kann auch unter 64-Bit-Betriebssystemen installiert werden:

- AIX
- HP-UX
- Solaris SPARC
- Linux x86
- Linux for System z
- Windows

Softwarevoraussetzungen

Wenn Sie DB2 Spatial Extender installieren möchten, müssen die Softwarevoraussetzungen für DB2-Datenbankprodukte erfüllt sein. Weitere Details finden Sie im Abschnitt „Installationsvoraussetzungen für DB2-Datenbankprodukte" bei der *DB2-Server - Installation*.

Spatial Extender auf DB2-Servern

Sie können den Spatial Extender als Teil eines DB2-Datenbankprodukts oder auf einer vorhandenen DB2-Kopie installieren. Die Softwarevoraussetzungen für die DB2-Datenbankserverprodukte müssen erfüllt sein.

Spatial Extender auf IBM Data Server-Clients

Wenn Sie den IBM Data Server-Client unter Windows-Betriebssystemen installieren, ist Spatial Extender Teil der angepassten Installation.

Bei der Installation eines DB2-Datenbankprodukts unter AIX-, HP-UX-, Solaris-, Linux for Intel- oder Linux on System z-Betriebssystemen kann Spatial Extender optional installiert werden, wenn die angepasste Installation ausgewählt wird.

Die Softwarevoraussetzungen für IBM Data Server-Client müssen erfüllt sein.

Installieren von DB2 Spatial Extender mit dem DB2-Installationsassistenten (Windows)

Installieren Sie DB2 Spatial Extender mit dem **DB2-Installationsassistenten** auf Windows-Betriebssystemen als Teil einer DB2-Installation.

Vorbereitende Schritte

Vor dem Start des **DB2-Installationsassistenten**:

- Stellen Sie sicher, dass Ihr System die Anforderungen im Hinblick auf die Installation, den Hauptspeicher und die Plattenspeicherkapazität erfüllt.
- Sie benötigen das lokale Konto für Benutzer mit Administratorberechtigung mit den empfohlenen Benutzerberechtigungen zum Ausführen der Installation. Bei DB2-Datenbankservern, bei denen die Benutzer-ID LocalSystem (lokales System) als DAS und DB2-Instanzbenutzer verwendet werden kann und bei denen das Feature für die Datenbankpartitionierung nicht verwendet wird, kann ein Nicht-Administrator mit erweiterten Zugriffsrechten die Installation durchführen.

  Anmerkung: Wenn ein Benutzer mit einem Benutzerkonto ohne Administratorberechtigung die Produktinstallation durchführen soll, muss die VS2010-Laufzeitbibliothek installiert werden, bevor dieser Benutzer versucht, ein DB2-Datenbankprodukt zu installieren. Die VS2010-Laufzeitbibliothek wird im Betriebssystem benötigt, bevor das DB2-Datenbankprodukt installiert werden kann. Die VS2010-Laufzeitbibliothek ist auf der Download-Website für Microsoft-Laufzeitbibliotheken verfügbar. Sie haben zwei Auswahlmöglichkeiten: `vcredist_x86.exe` für 32-Bit-Systeme oder `vcredist_x64.exe` für 64-Bit-Systeme.
- Sie sollten alle Programme schließen, damit das Installationsprogramm alle Dateien auf dem Computer aktualisieren kann, ohne dass dazu ein Warmstart erforderlich ist.
- Bei Installationen von einem virtuellen Laufwerk müssen Sie das Netzlaufwerk einem Windows-Laufwerksbuchstaben zuordnen. Der **DB2-Installationsassistent** unterstützt keine Installation von einem virtuellen Laufwerk oder einem nicht zugeordneten Netzlaufwerk (z. B. *hostname**sharename* im Windows-Explorer).

Informationen zu diesem Vorgang

Diese Task wird im Rahmen der übergeordneten Task '„DB2 Spatial Extender einrichten und installieren” auf Seite 19' ausgeführt.

Vorgehensweise

So installieren Sie DB2 Spatial Extender als Teil eines DB2-Servers oder IBM Data Server-Clients:

1. Melden Sie sich mit dem für die Installation von DB2 definierten lokalen Administratorkonto am System an.
2. Wenn Sie über die DB2-Datenbankprodukt-DVD verfügen, legen Sie sie in das DVD-Laufwerk ein. Das **DB2 Setup-Launchpad** wird von der Funktion für automatische Ausführung automatisch gestartet, sofern diese Funktion aktiviert ist. Wenn die Funktion für automatische Ausführung nicht funktionieren sollte, durchsuchen Sie im Windows Explorer die DB2-Datenbankprodukt-DVD und klicken das Installationssymbol (**setup**) doppelt an, um das **DB2 Setup-Launchpad** zu starten.
3. Wenn Sie das DB2-Datenbankprodukt von Passport Advantage heruntergeladen haben, führen Sie die ausführbare Datei aus, um die Installationsdateien des DB2-Datenbankprodukts zu extrahieren. Durchsuchen Sie im Windows Explorer die DB2-Installationsdateien und klicken das Installationssymbol (**setup**) doppelt an, um das **DB2 Setup-Launchpad** zu starten.
4. Im **DB2 Setup-Launchpad** können Sie die Installationsvoraussetzungen und Release-Informationen lesen, um die neuesten Informationen zu erhalten, oder direkt mit der Installation fortfahren.
5. Klicken Sie **Produkt installieren** an. Im Fenster **Produkt installieren** werden die Produkte angezeigt, die zur Installation zur Verfügung stehen.
6. Installieren Sie Spatial Extender, indem Sie eine der folgenden Optionen wählen:
   - Wenn Sie eine neue DB2-Kopie mit Spatial Extender erstellen möchten, klicken Sie auf **Neue Installation**, um die Installation zu starten. Wählen Sie die angepasste Installation aus, um Spatial Extender bei der Installation zu berücksichtigen. Führen Sie die Installation aus, indem Sie den Eingabeaufforderungen des **DB2-Installationsassistenten** folgen.
   - Wenn Sie Spatial Extender auf einer vorhandenen DB2-Kopie installieren möchten, klicken Sie auf **Mit vorhandener Installation arbeiten**. Wählen Sie die angepasste Installation aus, um Spatial Extender bei der Installation zu berücksichtigen. Führen Sie die Installation aus, indem Sie den Eingabeaufforderungen des **DB2-Installationsassistenten** folgen.

   Prüfen Sie nach Abschluss der Installation, ob in der angegebenen Protokolldatei Warnungen oder Fehlernachrichten dokumentiert sind.

Nächste Schritte

Erstellen Sie nach der Installation von DB2 Spatial Extender eine DB2-Instanz auf neuen DB2-Kopien, und prüfen Sie dann, ob die Installation erfolgreich war.

Installieren von DB2 Spatial Extender mit dem DB2-Installationsassistenten (Linux und UNIX)

Um DB2 Spatial Extender unter dem Betriebssystem Linux oder UNIX zu installieren, müssen Sie den Assistenten **DB2 Setup** oder eine Antwortdatei verwenden.

Vorbereitende Schritte

Vor dem Start des **DB2-Installationsassistenten**:

- Stellen Sie sicher, dass Ihr System die Anforderungen im Hinblick auf die Installation, den Hauptspeicher und die Plattenspeicherkapazität erfüllt.
- Stellen Sie sicher, dass ein unterstützter Browser installiert ist.
- Stellen Sie sicher, dass das Produktimage der DB2-Datenbank auf dem Computer verfügbar ist. DB2-Installationsimages sind entweder durch den Erwerb einer physischen DB2-Datenbankprodukt-DVD oder durch Herunterladen eines Installationsimages von Passport Advantage erhältlich.
- Wenn Sie landessprachliche Versionen eines DB2-Datenbankprodukts installieren, benötigen Sie die entsprechenden Landessprachenpakete.
- Stellen Sie sicher, dass die X Linux-Software zur Wiedergabe einer grafischen Benutzerschnittstelle installiert ist, dass der X Linux-Server ausgeführt wird und dass die Variable *DISPLAY* gesetzt ist. Der **DB2-Installationsassistent** ist ein grafisch orientiertes Installationsprogramm.
- Wird in Ihrer Umgebung Sicherheitssoftware verwendet, müssen Sie die erforderlichen DB2-Benutzer manuell erstellen, bevor Sie den **DB2-Installationsassistenten** starten.

Informationen zu diesem Vorgang

Diese Task wird im Rahmen der übergeordneten Task '„DB2 Spatial Extender einrichten und installieren” auf Seite 19' ausgeführt.

Sie können ein DB2-Datenbankprodukt entweder mit oder ohne Rootberechtigung installieren.

Vorgehensweise

So installieren Sie DB2 Spatial Extender als Teil eines DB2-Servers oder IBM Data Server-Clients:

1. Wenn Sie über eine physische DB2-Datenbankprodukt-DVD verfügen, wechseln Sie in das Verzeichnis, in dem die DB2-Datenbankprodukt-DVD angehängt ist. Geben Sie dazu den folgenden Befehl ein:

   ```
   cd /dvdrom
   ```

   Dabei steht */dvdrom* für den Mountpunkt der DB2-Datenbankprodukt-DVD.
2. Wenn Sie das DB2-Datenbankproduktimage heruntergeladen haben, müssen Sie die Produktdatei extrahieren und entpacken.
   a. Extrahieren Sie die Produktdatei:

      ```
      gzip -d produkt.tar.gz
      ```

      Dabei steht *produkt* für den Namen des Produkts, das Sie heruntergeladen haben.
   b. Entpacken Sie die Produktdatei:

      Unter Linux-Betriebssystemen

      ```
      tar -xvf produkt.tar
      ```

      Unter AIX-, HP-UX- und Solaris-Betriebssystemen

      ```
      gnutar -xvf produkt.tar
      ```

Dabei steht *produkt* für den Namen des Produkts, das Sie heruntergeladen haben.

c. Wechseln Sie das Verzeichnis:

```
cd ./produkt
```

Dabei steht *produkt* für den Namen des Produkts, das Sie heruntergeladen haben.

Anmerkung: Wenn Sie das Landessprachenpaket heruntergeladen haben, entpacken Sie es in demselben Verzeichnis. So werden die Unterverzeichnisse (z. B. `./nlpack`) in demselben Verzeichnis erstellt und das Installationsprogramm kann die Installationsimages automatisch und ohne Aufforderung an den Benutzer finden.

3. Geben Sie den Befehl **./db2setup** von dem Verzeichnis aus ein, in dem sich das Datenbankproduktimage befindet, um den **DB2-Installationsassistenten** zu starten.
4. Das Fenster **IBM DB2 Setup - Launchpad** wird geöffnet. In diesem Fenster können Sie die Installationsvoraussetzungen und die Release-Informationen anzeigen oder direkt mit der Installation fortfahren. Sie können auch die Installationsvoraussetzungen und die Release-Informationen aufrufen, um die neuesten Informationen abzurufen.
5. Klicken Sie **Produkt installieren** an. Im Fenster **Produkt installieren** werden die Produkte angezeigt, die zur Installation zur Verfügung stehen.
6. Installieren Sie Spatial Extender, indem Sie eine der folgenden Optionen wählen:
   - Wenn Sie eine neue DB2-Kopie mit Spatial Extender erstellen möchten, klicken Sie auf **Neue Installation**, um die Installation zu starten. Wählen Sie die angepasste Installation aus, um Spatial Extender bei der Installation zu berücksichtigen. Führen Sie die Installation aus, indem Sie den Eingabeaufforderungen des **DB2-Installationsassistenten** folgen.
   - Wenn Sie Spatial Extender auf einer vorhandenen DB2-Kopie installieren möchten, klicken Sie auf **Mit vorhandener Installation arbeiten**. Wählen Sie die angepasste Installation aus, um Spatial Extender bei der Installation zu berücksichtigen. Führen Sie die Installation aus, indem Sie den Eingabeaufforderungen des **DB2-Installationsassistenten** folgen.

   Prüfen Sie nach Abschluss der Installation, ob in der angegebenen Protokolldatei Warnungen oder Fehlernachrichten dokumentiert sind.

Nächste Schritte

Erstellen Sie nach der Installation von DB2 Spatial Extender eine DB2-Instanz auf neuen DB2-Kopien, und prüfen Sie dann, ob die Installation erfolgreich war.

DB2 Spatial Extender-Installation prüfen

Nach der Installation von DB2 Spatial Extender sollten Sie die Installation überprüfen.

Vorbereitende Schritte

Die folgenden Voraussetzungen müssen erfüllt sein, bevor eine DB2 Spatial Extender-Installation geprüft werden kann:

- DB2 Spatial Extender muss auf einem Computer installiert sein.

- Auf dem Computer, auf dem der DB2-Datenserver installiert ist, muss eine DB2-Instanz erstellt worden sein.

Informationen zu diesem Vorgang

Diese Task wird normalerweise im Rahmen der übergeordneten Task zur Einrichtung und Konfiguration von DB2 Spatial Extender ausgeführt. Weitere Informationen finden Sie im Abschnitt „DB2 Spatial Extender einrichten und installieren" auf Seite 19. Die Task umfasst das Erstellen einer DB2-Datenbank und das Ausführen einer DB2 Spatial Extender-Musteranwendung, die mit DB2 bereitgestellt wird und dazu verwendet werden kann, zu prüfen, ob eine Reihe zentraler Funktionen von DB2 Spatial Extender ordnungsgemäß ausgeführt werden kann. Dieser Test reicht aus, um die ordnungsgemäße Installation und Konfiguration von DB2 Spatial Extender zu prüfen.

Vorgehensweise

Gehen Sie wie folgt vor, um die Installation von DB2 Spatial Extender zu überprüfen:

1. Linux und UNIX: Melden Sie sich mit der Benutzer-ID am System an, die der DB2-Instanzeignerrolle entspricht.
2. Erstellen Sie eine DB2-Datenbank. Öffnen Sie hierzu ein DB2-Befehlsfenster, und geben Sie Folgendes ein:

   ```
   db2 create database meinedb
   ```

   Hierbei steht *meinedb* für den Datenbanknamen.
3. Stellen Sie sicher, dass ein Tabellenbereich für temporäre Systemtabellen mit einer Seitengröße von mindestens 8 KB und einer Mindestgröße von 500 Seiten vorhanden ist. Ist ein solcher Tabellenbereich nicht vorhanden, lesen Sie die Informationen im Abschnitt „Erstellen temporärer Tabellenbereiche" in der Veröffentlichung *Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen*, der Einzelheiten hierzu enthält. Dies ist eine Voraussetzung für die Ausführung des Programms 'runGSEdemo'.
4. Setzen Sie bei Windows-Betriebssystemen für den Konfigurationsparameter des Datenbankmanagers **agent_stack_sz** einen Wert von mindestens 100. Im folgenden Beispiel wird der Befehlszeilenprozessorbefehl zum Setzen des Parameters auf den Wert 110 dargestellt:

   ```
   UPDATE DBM CFG USING AGENT_STACK_SZ 110
   ```
5. Wechseln Sie in das Verzeichnis, in dem sich das Programm 'runGSEdemo' befindet:
   - Geben Sie bei DB2 Spatial Extender-Installationen unter Linux- und UNIX-Betriebssystemen Folgendes ein:

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

     Hierbei steht *$HOME* für das Ausgangsverzeichnis des Instanzeigners.
   - Geben Sie bei DB2 Spatial Extender-Installationen unter Windows-Betriebssystemen Folgendes ein:

     ```
     cd c:\Programme\IBM\sqllib\samples\extenders\spatial
     ```

     Hierbei steht *c:\Programme\IBM\sqllib* für das Verzeichnis, in dem Sie DB2 Spatial Extender installiert haben.
6. Führen Sie das Installationsprüfprogramm aus. Geben Sie in der DB2-Befehlszeile den Befehl **runGseDemo** ein:

```
runGseDemo meinedb benutzerID kennwort
```

Hierbei steht *meinedb* für den Datenbanknamen.

Kapitel 5. Upgrade auf DB2 Spatial Extender Version 10.1

Zum Durchführen eines Upgrades auf DB2 Spatial Extender Version 10.1 für Systeme, auf denen Sie DB2 Spatial Extender Version 9.5 oder Version 9.7 installiert haben, ist mehr erforderlich als nur die Installation von DB2 Spatial Extender Version 10.1. Sie müssen die entsprechenden Upgrade-Tasks für diese Systeme ausführen.

Die folgenden Tasks enthalten eine Beschreibung aller Schritte zur Durchführung eines Upgrades von DB2 Spatial Extender von Version 9.5 oder Version 9.7 auf Version 10.1:

- „Upgrade für DB2 Spatial Extender"
- „Aktualisierung von DB2 Spatial Extender (32-Bit) auf DB2 Spatial Extender (64-Bit)" auf Seite 28

Wenn Ihre DB2-Umgebung weitere Komponenten, z. B. DB2-Server, -Clients und -Datenbankanwendungen enthält, können Sie unter „Upgrade auf DB2 Version 10.1" in der Veröffentlichung *Upgrade auf DB2 Version 10.1* Details zur Durchführung eines Upgrades für diese Komponenten nachlesen.

Upgrade für DB2 Spatial Extender

Für das Upgrade von DB2 Spatial Extender müssen Sie zunächst ein Upgrade für Ihren DB2-Server und anschließend für bestimmte Datenbankobjekte und Daten in für räumliche Operationen aktivierten Datenbanken durchführen.

Vorbereitende Schritte

Vor dem Starten des Upgradeprozesses:

- Vergewissern Sie sich, dass Ihr System die Installationsvoraussetzungen für DB2 Spatial Extender Version 10.1 erfüllt.
- Stellen Sie sicher, dass Sie über die Berechtigungen DBADM und DATAACCESS für die für räumliche Operationen aktivierte Datenbank verfügen.
- Stellen Sie sicher, dass ein Tabellenbereich für temporäre Systemtabellen mit einer Seitengröße von mindestens 8 KB und einer Mindestgröße von 500 Seiten vorhanden ist.

Informationen zu diesem Vorgang

Wenn Sie DB2 Spatial Extender Version 9.5 oder Version 9.7 installiert haben, müssen Sie die folgenden Schritte ausführen, bevor Sie eine vorhandene, für räumliche Operationen aktivierte Datenbank mit DB2 Spatial Extender Version 10.1 verwenden können. In diesem Abschnitt werden die Schritte beschrieben, die zur Durchführung eines Upgrades für Datenbanken für räumliche Operationen von einer Vorgängerversion von DB2 Spatial Extender erforderlich sind.

Vorgehensweise

Gehen Sie wie folgt vor, um ein Upgrade von DB2 Spatial Extender auf Version 10.1 durchzuführen:

1. Führen Sie ein Upgrade für Ihren DB2-Server von Version 9.5 oder Version 9.7 auf Version 10.1 durch und führen Sie dazu eine der folgenden Tasks aus:

- „Upgrade für einen DB2-Server (Windows)" in der Veröffentlichung *Upgrade auf DB2 Version 10.1*
- „Upgrade für einen DB2-Server (Linux und UNIX)" in der Veröffentlichung *Upgrade auf DB2 Version 10.1*.

Im Rahmen der Upgrade-Task müssen Sie DB2 Spatial Extender Version 10.1 installieren, nachdem Sie DB2 Version 10.1 installiert haben, um das Upgrade für Ihre Instanzen erfolgreich durchzuführen.

2. Beenden Sie alle Verbindungen zur Datenbank.
3. Führen Sie mithilfe des Befehls **db2se upgrade** ein Upgrade für Ihre für räumliche Operationen aktivierten Datenbanken von Version 9.5 oder Version 9.7 auf Version 10.1 durch.

Ergebnisse

Überprüfen Sie, ob die Nachrichtendatei Details zu einem der ausgegebenen Fehler enthält. Die Nachrichtendatei enthält darüber hinaus hilfreiche Informationen zu den vom Upgrade betroffenen Indizes, Sichten und zur Geocodierungskonfiguration.

Aktualisierung von DB2 Spatial Extender (32-Bit) auf DB2 Spatial Extender (64-Bit)

Für eine Aktualisierung von DB2 Spatial Extender Version 10.1 (32-Bit) auf DB2 Spatial Extender Version 10.1 (64-Bit) müssen Sie ein Backup der räumlichen Indizes durchführen, den DB2-Server auf DB2 Version 10.1 (64-Bit) aktualisieren, DB2 Spatial Extender Version 10.1 (64-Bit) installieren und anschließend einen Restore der räumlichen Indizes, für die Sie ein Backup erstellt haben, durchführen.

Vorbereitende Schritte

- Stellen Sie sicher, dass ein Tabellenbereich für temporäre Systemtabellen mit einer Seitengröße von mindestens 8 KB und einer Mindestgröße von 500 Seiten vorhanden ist.

Informationen zu diesem Vorgang

Wenn Sie eine Aktualisierung von DB2 Spatial Extender Version 9.5 oder Version 9.7 (32-Bit) auf DB2 Spatial Extender Version 9.7 (64-Bit) durchführen, müssen Sie stattdessen die Task „Upgrade für DB2 Spatial Extender" auf Seite 27 ausführen.

Vorgehensweise

Gehen Sie wie folgt vor, um einen DB2 Spatial Extender Version 9.7 (32-Bit)-Server auf DB2 Spatial Extender Version 9.7 (64-Bit) zu aktualisieren:

1. Erstellen Sie ein Backup Ihrer Datenbank.
2. Sichern Sie die vorhandenen räumlichen Indizes, indem Sie den Befehl **db2se save_indexes** in einer Eingabeaufforderung des Betriebssystems eingeben.
3. Führen Sie eine der folgenden Tasks aus, um Ihren DB2-Server (32-Bit) von Version 9.5 oder Version 9.7 auf DB2 Version 10.1 (64-Bit) zu aktualisieren:
   - „Durchführen einer Aktualisierung von 32-Bit-DB2-Instanzen auf 64-Bit-DB2-Instanzen (Windows)" in *DB2-Server - Installation* .
   - „Aktualisieren von DB2-Kopien (Linux und UNIX)" in der Veröffentlichung *Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen*.
4. Installieren Sie DB2 Spatial Extender Version 10.1.

5. Führen Sie einen Restore der räumlichen Indizes durch, indem Sie den Befehl **db2se restore_indexes** in einer Eingabeaufforderung des Betriebssystems eingeben.

Kapitel 6. Räumliche Ressourcen für eine Datenbank konfigurieren

Nach der Konfiguration der Datenbank zur Aufnahme räumlicher Daten können Sie der Datenbank Ressourcen zur Verfügung stellen, die Sie beim Erstellen und Verwalten räumlicher Spalten und bei der Analyse räumlicher Daten benötigen.

Die räumlichen Ressourcen sind im Einzelnen:

- Objekte, die von Spatial Extender zur Unterstützung räumlicher Operationen zur Verfügung gestellt wurden. Zum Beispiel gespeicherte Prozeduren zur Verwaltung einer Datenbank, räumliche Datentypen und räumliche Dienstprogramme zur Geocodierung und für den Import oder Export räumlicher Daten.
- Alle Geocoder, die von Benutzern oder Lieferanten zur Verfügung gestellt werden.

Um diese Ressourcen zur Verfügung stellen können, müssen Sie Ihre Datenbank für räumliche Operationen aktivieren, den Zugriff auf Bezugsdaten konfigurieren und Geocoder anderer Anbieter registrieren.

Für die Datenbank bereitgestellte Ressourcen

DB2 Spatial Extender stellt der Datenbank mehrere Ressourcen bereit, damit räumliche Operationen unterstützt werden können.

Diese Ressourcen sind im Einzelnen:

- Gespeicherte Prozeduren: Sobald Sie eine räumliche Operation anfordern (beispielsweise durch Absetzen eines Befehls zum Importieren von räumlichen Daten), ruft DB2 Spatial Extender eine dieser gespeicherten Prozeduren auf, um die Operation auszuführen.
- Räumliche Datentypen: Jeder Tabellen- oder Sichtspalte, die räumliche Daten aufnehmen soll, muss ein räumlicher Datentyp zugeordnet werden.
- Katalog von DB2 Spatial Extender. Bestimmte Operationen sind von diesem Katalog abhängig. Bevor Sie beispielsweise über die Darstellungstools auf eine räumliche Spalte zugreifen können, ist es unter Umständen für das Tool erforderlich, dass die räumliche Spalte im Katalog registriert wird.
- Ein räumlicher Rasterindex. Mit diesem Typ können Sie Rasterindizes für räumliche Spalten definieren.
- Räumliche Funktionen. Sie verwenden diese Funktionen zum Arbeiten mit räumlichen Daten auf verschiedene Arten, beispielsweise zum Ermitteln der Beziehung zwischen Geometrien und zum Generieren weiterer räumlicher Daten.
- Definitionen von Koordinatensystemen.
- Räumliche Standardbezugssysteme.
- Zwei Schemata: DB2GSE und ST_INFORMTN_SCHEMA. Das Schema DB2GSE enthält die zuvor aufgelisteten Objekte, also gespeicherte Prozeduren, räumliche Datentypen, den Katalog von DB2 Spatial Extender usw. Die Sichten im Katalog sind auch im Schema ST_INFORMTN_SCHEMA verfügbar. Auf diese Weise wird dem SQL/MM-Standard entsprochen.

Datenbank für räumliche Operationen aktivieren

Zum Aktivieren der Datenbank für räumliche Operationen muss DB2 Spatial Extender einer Datenbank Ressourcen für die Erstellung räumlicher Spalten und für die Bearbeitung räumlicher Daten zur Verfügung stellen.

Vorbereitende Schritte

Vor der Aktivierung einer Datenbank für räumliche Operationen:

- Stellen Sie sicher, dass Ihre Benutzer-ID über die Berechtigung DBADM für die Datenbank verfügt.
- Stellen Sie sicher, dass ein Tabellenbereich für temporäre Systemtabellen mit einer Seitengröße von mindestens 8 KB und einer Mindestgröße von 500 Seiten vorhanden ist.

Vorgehensweise

Zur Aktivierung einer Datenbank für räumliche Operationen gibt es die folgenden Methoden:

- Geben Sie den Befehl **db2se enable_db** ein.
- Führen Sie eine Anwendung aus, die die Prozedur DB2GSE.ST_ENABLE_DB aufruft.

Sie können den Tabellenbereich, in dem der Katalog von DB2 Spatial Extender angesiedelt werden soll, explizit auswählen. Wenn Sie dies nicht tun, verwendet die DB2-Datenbank den Standardtabellenbereich.
Wenn Sie räumliche Daten in einer Umgebung mit partitionierten Datenbanken verwenden möchten, dann dürfen Sie den Standardtabellenbereich nicht benutzen. Die besten Ergebnisse erzielen Sie, wenn Sie die Datenbank in einem Tabellenbereich aktivieren, der für einen Einzelknoten definiert ist. Normalerweise sind Einzelknotentabellenbereiche für kleine Tabellen definiert, bei denen eine Partitionierung keine Vorteile bringt.
Sie können einen Einzelknotentabellenbereich zum Beispiel mithilfe des db2se-Befehlszeilenprozessors definieren:

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

Zahlreiche Abfragen können effizienter durchgeführt werden, wenn die Tabelle DB2GSE.GSE_SPATIAL_REFERENCE_SYSTEMS für räumliche Bezugssysteme auf allen Knoten repliziert wird, die für Geschäftstabellen verwendet werden, für die Abfragen ausgeführt werden. Die Tabelle DB2GSE.GSE_SRS_REPLICATED_AST kann mit einer Anweisung wie der folgenden erneut erstellt werden:

```
drop table db2gse.gse_srs_replicated_ast;
-- MQT zur Replikation der Daten des räumlichen Bezugssystems auf allen Knoten
-- in einer Umgebung mit partitionierten Datenbanken
CREATE TABLE db2gse.gse_srs_replicated_ast AS
   ( SELECT srs_name, srs_id, x_offset, x_scale, y_offset, z_offset, z_scale,
       m_offset, m_scale, definition
     FROM db2gse.gse_spatial_reference_systems )
   DATA INITIALLY DEFERRED
   REFRESH IMMEDIATE
   ENABLE QUERY OPTIMIZATION
   REPLICATED
   IN ts_data  partitionierter_tabellenbereich
;

REFRESH TABLE db2gse.gse_srs_replicated_ast;

CREATE  INDEX db2gse.gse_srs_id_ast
```

```
   ON db2gse.gse_srs_replicated_ast ( srs_id )
;

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

Geocoder registrieren

Bevor Sie Geocoder verwenden können, müssen Sie sie registrieren.

Vorbereitende Schritte

Damit Sie einen Geocoder registrieren können, muss Ihre Benutzer-ID die Berechtigung DBADM für die Datenbank besitzen, in der sich der Geocoder befindet.

Vorgehensweise

Zur Registrierung eines Geocoders gibt es die folgenden Methoden:

- Setzen Sie den Befehl **db2se register_gc** ab.
- Führen Sie eine Anwendung aus, die die Prozedur DB2GSE.ST_REGISTER_GEOCODER aufruft.

Kapitel 7. Räumliche Ressourcen für ein Projekt konfigurieren

Nachdem die Datenbank für räumliche Operationen aktiviert ist, können Sie Projekte erstellen, die räumliche Daten verwenden.

Zu den Ressourcen, die für jedes Projekt erforderlich sind, gehören ein Koordinatensystem, dem die räumlichen Daten entsprechen, und ein räumliches Bezugssystem, das das Ausmaß des geografischen Bereichs definiert, auf den sich die Daten beziehen.

Es ist wichtig, dass Sie sich mit den Eigenschaften von Koordinatensystemen auseinandersetzen und wissen, was räumliche Bezugssysteme sind.

Verwendungshinweise für Koordinatensysteme

Wenn Sie ein Projekt planen, das räumliche Daten verwendet, müssen Sie festlegen, ob die Daten auf einem der Koordinatensysteme basieren sollen, die im Katalog von Spatial Extender registriert sind.

Wenn keines dieser Koordinatensysteme Ihren Anforderungen entspricht, können Sie ein Koordinatensystem erstellen, das Ihre Anforderungen erfüllt. An dieser Stelle wird das Konzept des Koordinatensystems erklärt und eine Einführung in die Tasks der Auswahl eines zu verwendenden Koordinatensystems sowie des Erstellens eines neuen Koordinatensystems gegeben.

Koordinatensysteme

Ein Koordinatensystem stellt ein Gerüst bereit, mit dem die relativen Positionen von Objekten in einem angegebenen Bereich definiert werden können, beispielsweise in einem Bereich der Erdoberfläche oder aber für die gesamte Erdoberfläche.

DB2 Spatial Extender unterstützt die folgenden Koordinatensystemtypen, mit denen die Position eines geografischen Objekts bestimmt werden kann:

Geografisches Koordinatensystem
Bei einem *geografischen Koordinatensystem* handelt es sich um ein Bezugssystem, das eine dreidimensionale kugelförmige Oberfläche verwendet, um die Position von Punkten auf dem Globus zu ermitteln. Jede Position auf dem Globus kann mithilfe eines Punktes referenziert werden, der seinerseits durch eine Längen- und eine Breitengradkoordinate angegeben ist, die auf der Basis von Winkelmaßeinheiten definiert werden.

Projiziertes Koordinatensystem
Ein *projiziertes Koordinatensystem* ist eine plane, zweidimensionale Darstellung der Erde. In diesem Koordinatensystem werden rechtwinklig-lineare (kartesische) Koordinaten verwendet, die auf der Basis linearer Maßeinheiten ermittelt werden. Es basiert auf einem kugelförmigen (sphärischen) Modell der Erde, und seine Koordinaten werden über eine Projektionstransformation mit den entsprechenden geografischen Koordinaten in Beziehung gesetzt.

Geografisches Koordinatensystem

Bei einem *geografischen Koordinatensystem* handelt es sich um ein Koordinatensystem, das eine dreidimensionale kugelförmige Oberfläche verwendet, um die Position von Punkten auf dem Globus zu ermitteln. Alle Positionen auf dem Globus können mithilfe eines Punktes referenziert werden, der seinerseits durch eine Längen- und eine Breitenkoordinate angegeben ist.

In Abb. 6 ist ein geografisches Koordinatensystem dargestellt, in dem eine Position durch die Angabe 55 Grad nördlicher Breite und 80 Grad östlicher Länge (55 Grad Nord 80 Grad Ost) dargestellt ist.

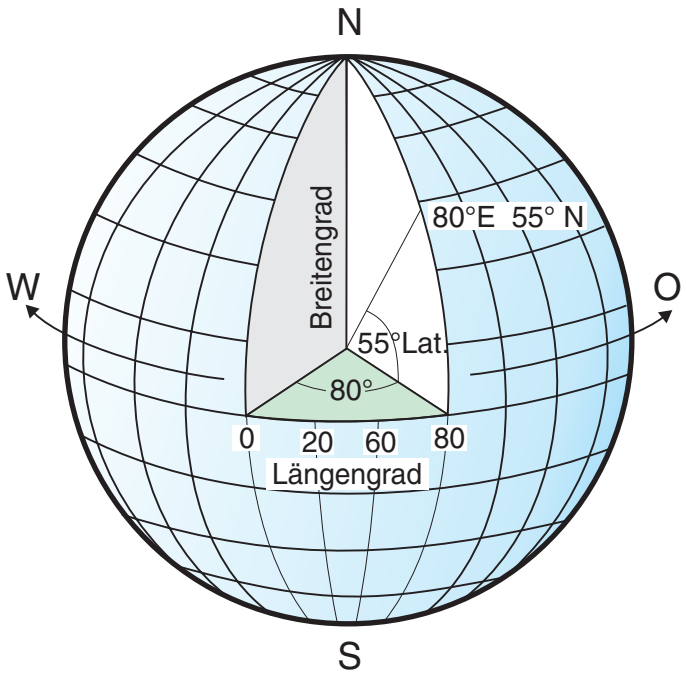


Abbildung 6. Geografisches Koordinatensystem

Die in waagrechter Richtung nach Ost bzw. West verlaufenden Linien weisen alle einen konstanten Breitengradwert auf und werden als *breitenparallel* bezeichnet. Sie verlaufen mit identischem Abstand parallel zueinander und bilden konzentrische Kreise um die Erde herum. Der *Äquator* stellt hierbei den größten dieser Kreise dar und teilt den Globus in zwei Hälften. Seine Entfernung zu beiden Polen ist identisch und der Wert dieser Breitengradlinie beträgt 0. Die nördlich des Äquators gelegenen Positionen weisen positive Breitengradwerte zwischen 0 und + 90 Grad auf, während die Positionen südlich des Äquators über negative Breitengradwerte im Bereich von 0 bis - 90 Grad verfügen.

Abb. 7 zeigt die Breitengradlinien.

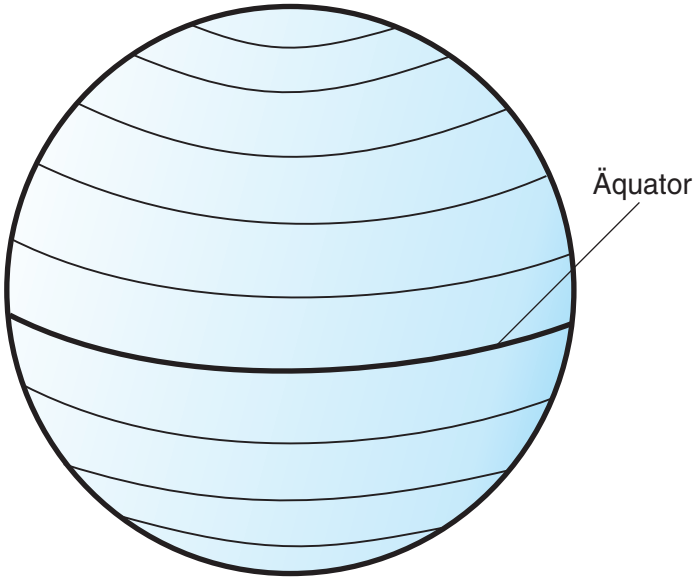


Abbildung 7. Breitengradlinien

Die Linien, die in senkrechter Richtung nach Norden bzw. Süden verlaufen, weisen alle einen konstanten Längengradwert auf und werden als *Meridiane* bezeichnet. Sie bilden Kreise von identischer Größe um die Erde herum, die sich an den Polen schneiden. Der *Nullmeridian* ist der Längengrad, der den Ursprung (also 0 Grad) für Längengradkoordinaten definiert. Eine der am häufigsten verwendeten Positionen für den Nullmeridian ist die Linie, die durch Greenwich in England verläuft. Es gibt jedoch noch weitere Längengradlinien, die z. B. durch Bern, Bogota und Paris verlaufen und ebenfalls als Nullmeridian verwendet werden können. Die Positionen, die östlich des Nullmeridians bis zum *antipodischen* Meridian (der Fortsetzung des Nullmeridians auf der anderen Seite des Globus) liegen, weisen positive Längengradwerte zwischen 0 und + 180 Grad auf. Die Positionen westlich des Nullmeridians verfügen über negative Längengradwerte zwischen 0 und –180 Grad.

Abb. 8 auf Seite 38 zeigt die Längengradlinien.

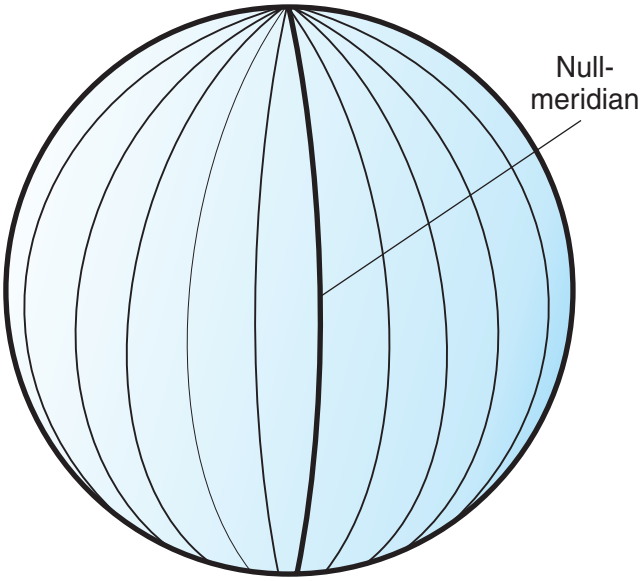


Abbildung 8. Längengradlinien

Die Längen- und die Breitengradlinien bilden ein Rasternetz um den Globus herum, das als *geografisches Netz* bezeichnet wird. Der Ursprungspunkt des geografischen Netzes ist der Punkt (0,0), an dem sich Äquator und Nullmeridian schneiden. Der Äquator ist die einzige Position im geografischen Netz, an der die lineare Distanz von 1 Grad Breite ungefähr identisch mit der Distanz für 1 Grand Länge ist. Da die Längengradlinien an den Polen konvergieren, ist der Abstand zwischen den einzelnen Meridianen an jedem Breitenkreis unterschiedlich. Je näher die Pole rücken, desto größer ist die einem Breitengrad entsprechende Strecke im Vergleich zu der Strecke, die einem Längengrad entspricht.

Außerdem ist es kompliziert, die Längen der Breitengradlinien mithilfe des geografischen Netzes zu ermitteln. Die Breitengradlinien sind konzentrische Kreise, die mit abnehmendem Abstand zu den Polen immer kleiner werden. An den Polen bestehen sie aus einem einzigen Punkt, an dem die Meridiane beginnen. Am Äquator beträgt der Wert für 1 Grad Länge ca. 111,321 km. Am 60. Breitengrad beträgt der Wert für 1 Grad Länge hingegen nur noch 55,802 km. (Diese Näherung basiert auf dem von Clarke im Jahr 1866 definierten Sphäroid.) Da es keine einheitliche Länge der Breiten- und Längengrade gibt, können die Abstände zwischen Punkten mithilfe von Winkelmaßeinheiten nicht exakt ermittelt werden.

Abb. 9 auf Seite 39 zeigt die unterschiedlichen Dimensionen zwischen Positionen im geografischen Netz.

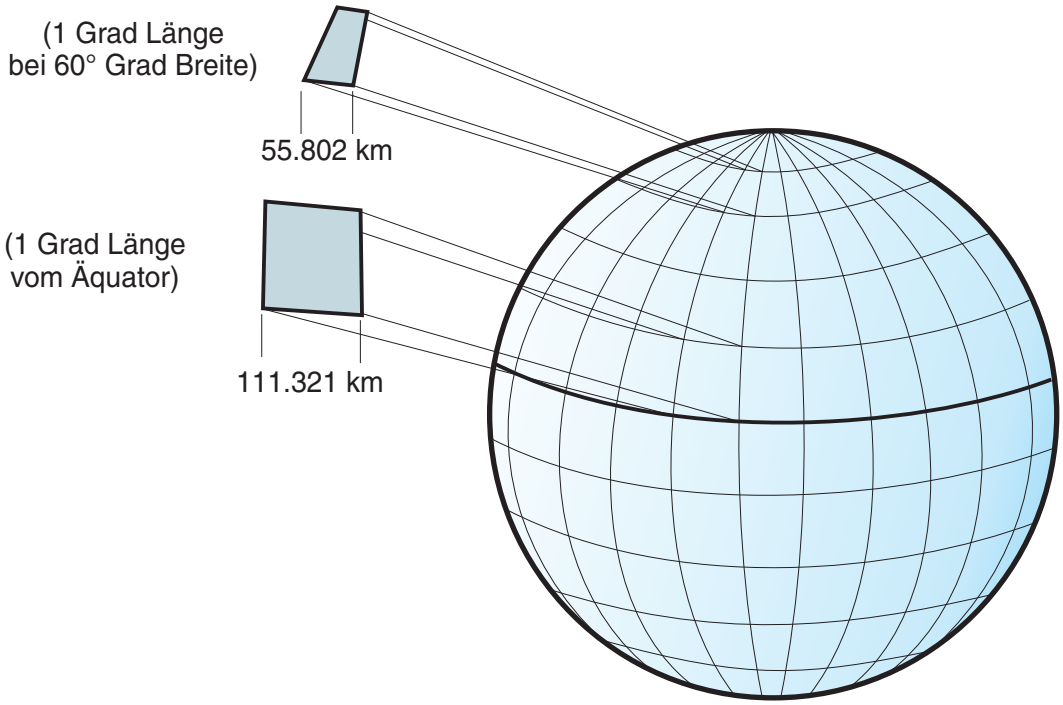


Abbildung 9. Unterschiedliche Dimensionen zwischen Positionen im geografischen Netz

Ein Koordinatensystem kann entweder durch eine Kugel (= Sphäre) oder eine sphäroide Approximation der Erdform definiert werden. Da die Erde keine perfekte Kugelform besitzt, kann ein Sphäroid hilfreich sein, um die Genauigkeit von Landkarten in Abhängigkeit von der Position auf dem Globus zu gewährleisten. Ein *Sphäroid* ist ein Ellipsoid, der auf einer Ellipse basiert. Eine Kugel (= Sphäre) basiert hingegen auf einem Kreis.

Die Form der Ellipse wird durch zwei Radien bestimmt. Der längere Radius wird als große Halbachse, der kürzere Radius als kleine Halbachse bezeichnet. Bei einem Ellipsoid handelt es sich um eine dreidimensionale Form, die entsteht, indem eine Ellipse um eine ihrer Achsen gedreht wird.

Abb. 10 auf Seite 40 zeigt die kugelförmige und die sphäroide Approximation des Globus sowie die große und die kleine Achse einer Ellipse.

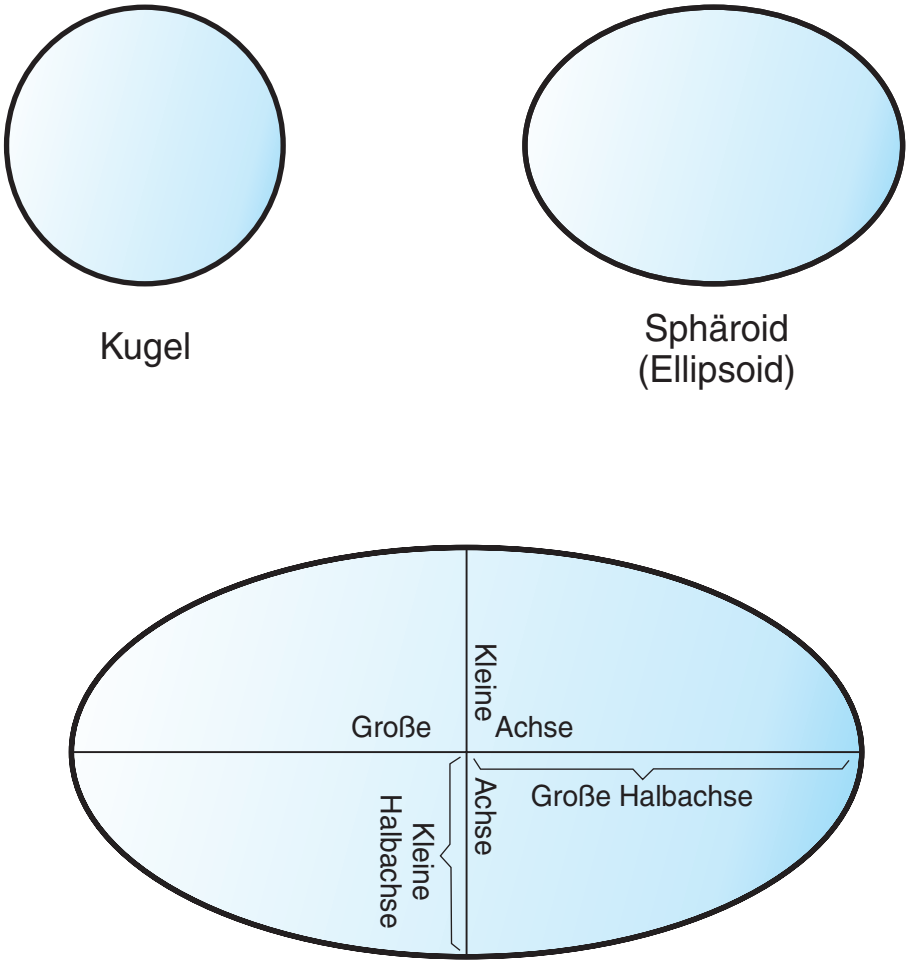


Abbildung 10. Kugelförmige und sphäroide Approximation

Ein *geodätisches Datum* (kurz "Datum" genannt) ist eine Gruppe von Werten, die die Position des Sphäroids bezogen auf den Erdmittelpunkt definiert. Das Datum bietet einen Bezugsrahmen für Standortmessungen und definiert Ursprung und Ausrichtung der Breiten- und Längengrade. Bestimmte Datumsangaben sind global und stellen weltweit eine verlässliche Genauigkeit zur Verfügung. Ein lokales Datum richtet seinen Sphäroiden so genau wie möglich an der Erdoberfläche eines bestimmten Bereichs aus. Daher sind die Maße des Koordinatensystems nicht genau, wenn sie mit einem anderen Bereich als dem Bereich verwendet werden, für den sie erstellt wurden.

Abb. 11 auf Seite 41 zeigt, wie unterschiedliche Datumsangaben an die Erdoberfläche angeglichen werden. Das lokale geodätische Datum (NAD27) weist eine genauere Angleichung an die Erdoberfläche auf als das geozentrische Datum (WGS84) für diese spezielle Position.

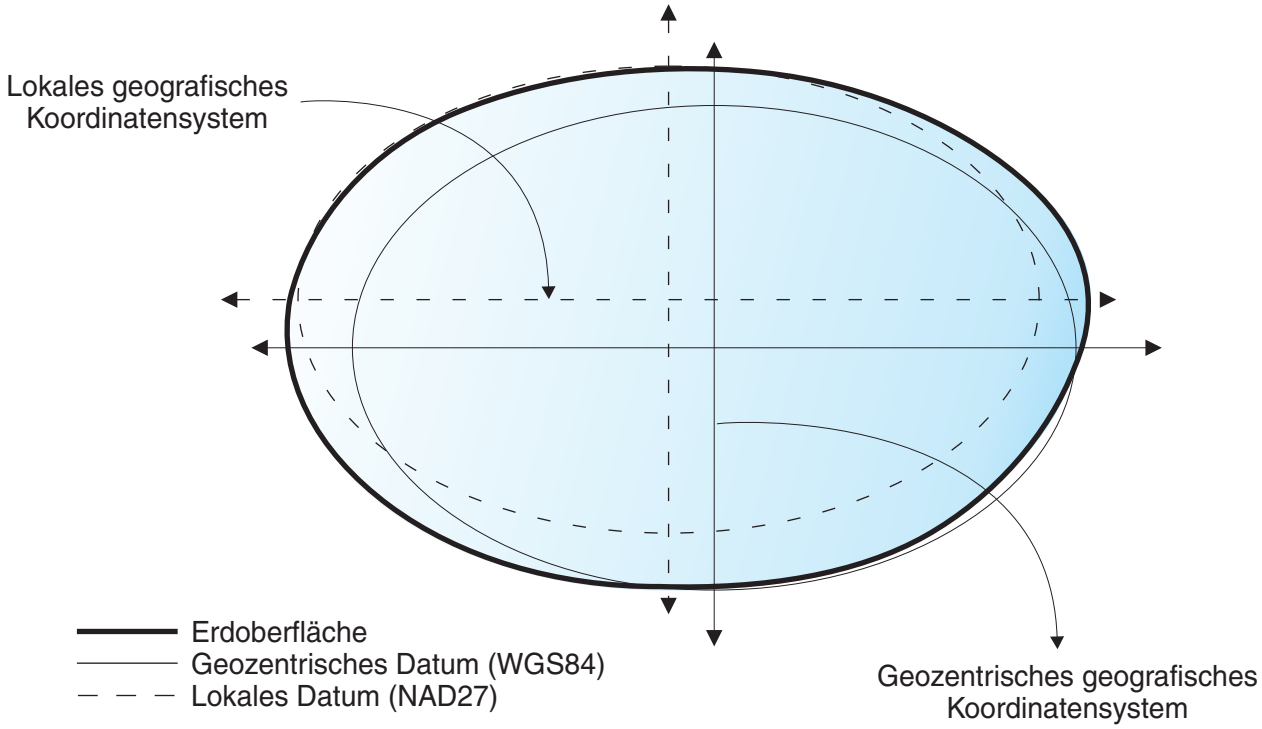


Abbildung 11. Datumsangleichung

Bei jeder Änderung des Datums wird das geografische Koordinatensystem verändert, und die Koordinatenwerte ändern sich. Die DMS-Koordinaten eines Steuerpunkts in Redlands (Kalifornien) lauten beispielsweise bei Verwendung des nordamerikanischen Datums von 1983 (NAD 1983): "-117 12 57.75961 34 01 43.77884". Bei Verwendung des nordamerikanischen Datums von 1927 (NAD 1927) lauten die Koordinaten desselben Punkts hingegen: "-117 12 54.61539 34 01 43.72995".

Projizierte Koordinatensysteme

Ein *projiziertes Koordinatensystem* ist eine plane, zweidimensionale Darstellung der Erde. Es basiert auf einem geografischen Koordinatensystem einer Kugel oder eines Sphäroiden, verwendet jedoch lineare Maßeinheiten für Koordinaten, sodass Abstands- und Bereichsberechnungen in diesen Einheiten auf einfache Weise durchgeführt werden können.

Die Breitengrad- und Längengradkoordinaten werden in X- bzw. Y-Koordinaten der Flachprojektion umgewandelt. Die X-Koordinate stellt normalerweise die in östlicher Richtung verlaufende Linie durch einen bestimmten Punkt, und die Y-Koordinate die in nördlicher Richtung verlaufende Linie durch einen Punkt dar. Die Mittellinie, die in Ost-West-Richtung verläuft, wird als X-Achse und die in Nord-Süd-Richtung verlaufende Mittellinie wird als Y-Achse bezeichnet.

Der Schnittpunkt der X- und der Y-Achse markiert den Ursprungspunkt und weist normalerweise die Koordinate (0,0) auf. Die über der X-Achse gelegenen Werte sind positive Werte. Werte unterhalb der X-Achse sind negative Werte. Die parallel zur X-Achse verlaufenden Linien sind abstandstreu zueinander. Die rechts neben der Y-Achse gelegenen Werte sind positive Werte. Die Werte links der Y-Achse sind negative Werte. Die Linien parallel zur Y-Achse sind abstandstreu.

Ein dreidimensionales geografisches Koordinatensystem kann mithilfe mathematischer Formeln in ein zweidimensionales Koordinatensystem mit Flachprojektion umwandelt werden. Die Umwandlung wird als *kartografische Projektion* bezeichnet. Kartografische Projektionen werden gewöhnlich durch die verwendete Projektionsoberfläche (z. B. kegelförmige, zylindrische und planare Oberflächen) klassifiziert. Je nach verwendeter Projektion werden unterschiedliche räumliche Eigenschaften verzerrt dargestellt. Projektionen sollen die Verzerrung von einem oder zwei Datenmerkmalen verringern. Abstand, Bereich, Form, Richtung oder eine Kombination dieser Eigenschaften sind daher möglicherweise keine genauen Darstellungen der abgebildeten Daten. Es sind mehrere Arten von Projektionen verfügbar. Die meisten kartografischen Projektionen versuchen, die Genauigkeit der räumlichen Eigenschaften bis zu einem gewissen Grad zu erhalten. Es gibt allerdings auch andere Projektionen, die versuchen, stattdessen die Gesamtverzerrung zu minimieren. Ein Beispiel hierfür ist die *Robinson-Projektion*. Zu den gängigsten Typen von kartografischen Projektionen gehören:

Flächentreue Projektionen
: Bei diesen Projektionen bleibt der Bereich, den spezifische Objekte einnehmen, erhalten. Form, Winkel und Maßstab sind jedoch verzerrt. Ein Beispiel für eine flächentreue Projektion ist die *flächentreue Kegelprojektion nach Albers*.

Winkeltreue Projektionen
: Bei solchen Projektionen bleibt die lokale Form kleiner Bereiche erhalten. Diese Projektionen behalten einzelne Winkel bei, um die räumlichen Beziehungen zu beschreiben, indem senkrechte Linien des geografischen Netzes dargestellt werden, die sich in 90-Grad-Winkeln auf der Karte schneiden. Alle Winkel bleiben erhalten. Der Bereich der Karte ist jedoch verzerrt. Beispiele für winkeltreue Projektionen sind die *Mercator-Projektion* und die *winkeltreue Kegelprojektion nach Lambert*.

Abstandstreue Projektionen
: Bei diesen Projektionen bleiben die Abstände zwischen bestimmten Punkten erhalten, indem der Maßstab eines bestimmten Datensatzes beibehalten wird. Manche Abstände sind reale Abstände, die dieselben Abstände im gleichen Maßstab wie der Globus sind. Außerhalb des Datensatzes ist der Maßstab zunehmend verzerrt. Beispiele für abstandstreue Projektionen sind die *sinusoidale Projektion* und die *abstandstreue Kegelprojektion*.

Richtungstreue Projektionen oder Azimutalprojektionen
: Bei diesen Projektionen bleibt die Richtung von einem Punkt zu allen anderen Punkten erhalten, indem einige der großen Kreisbögen beibehalten werden. Sie geben die Richtungen (oder Azimuten) aller Punkte auf der Karte bezogen auf den Mittelpunkt korrekt wieder. Azimutalkarten können mit flächentreuen Projektionen, winkeltreuen Projektionen und abstandstreuen Projektionen kombiniert werden. Beispiele für Azimutalprojektionen sind die *flächentreue Azimutalprojektion nach Lambert* und die *mittabstandstreue Azimutalprojektion*.

Zu verwendendes Koordinatensystem festlegen

Der erste Schritt bei der Planung eines Projekts besteht darin, das zu verwendende Koordinatensystem zu bestimmen.

Vorbereitende Schritte

Damit Sie ein Koordinatensystem erstellen können, muss Ihre Benutzer-ID die Berechtigung DBADM für die Datenbank besitzen, die für räumliche Operationen ak-

tiviert wurde. Für die Verwendung eines vorhandenen Koordinatensystems ist keine Berechtigung erforderlich.

Informationen zu diesem Vorgang

Nachdem Sie eine Datenbank für räumliche Operationen aktiviert haben, können Sie damit beginnen, Projekte zu planen, die räumliche Daten verwenden. Sie können ein mit DB2 Spatial Extender geliefertes oder auch ein anderes Koordinatensystem verwenden. Mit DB2 Spatial Extender werden über 2000 Koordinatensysteme ausgeliefert.

Weitere Informationen zu diesen Koordinatensystemen können Sie in der Katalogsicht DB2SSE. ST_COORDINATE_SYSTEMS ermitteln. Dort erfahren Sie außerdem, welche anderen Koordinatensysteme mit DB2 Spatial Extender geliefert wurden und ob gegebenenfalls durch andere Benutzer Koordinatensysteme erstellt wurden.

Vorgehensweise

Gehen Sie wie folgt vor, um das zu verwendende Koordinatensystem festzulegen:

1. Überprüfen Sie die vorhandenen Koordinatensysteme, die zum Lieferumfang von DB2 Spatial Extender gehören, und verwenden Sie das entsprechende räumliche Bezugssystem, das auf dem Koordinatensystem Ihrer Wahl basiert.

   Am häufigsten werden die folgenden Koordinatensysteme verwendet:

   - GCS_NORTH_AMERICAN_1983. Dieses Koordinatensystem verwenden Sie, wenn Sie Standorte in den USA definieren müssen. Beispiele:
     - Sie importieren räumliche Daten für die USA von den für den Download verfügbaren räumlichen Kartendaten (Spatial Map Data).
   - Ein Koordinatensystem, das von DB2 Spatial Extender als "UNSPECIFIED" (nicht spezifiziert) bezeichnet wird. Dieses Koordinatensystem verwenden Sie in den folgenden Situationen:
     - Sie müssen Standorte definieren, die keine direkte Beziehung zur Erdoberfläche aufweisen, beispielsweise Standorte von Büros in einem Bürogebäude oder Standorte von Regalen in einem Lagerraum.
     - Sie können diese Standorte in Form von positiven Koordinaten definieren, die keine oder wenige Dezimalwerte enthalten.
2. Wenn Sie kein vorhandenes Koordinatensystem in DB2 Spatial Extender finden können, dann können Sie mit einer der folgenden Methoden selbst ein Koordinatensystem erstellen:

   - Setzen Sie den Befehl db2se create_cs über den Befehlszeilenprozessor db2se ab.
   - Führen Sie eine Anwendung aus, die die Prozedur DB2SE.ST_CREATE_COORDSYS aufruft.

   Das Erstellen eines Koordinatensystems ist nur selten erforderlich. Sie müssen auch ein räumliches Bezugssystem erstellen, das auf dem neuen Koordinatensystem basiert.

Hinweise zur Konfiguration räumlicher Bezugssysteme

Wenn Sie ein Projekt planen, das räumliche Daten verwendet, müssen Sie festlegen, ob eines der verfügbaren räumlichen Bezugssysteme für diese Daten verwendet werden kann.

Wenn keines der verfügbaren Systeme für die Daten geeignet ist, können Sie ein geeignetes System erstellen. Dieser Abschnitt erläutert das Konzept der räumlichen Bezugssysteme und beschreibt die Tasks der Auswahl eines geeigneten Systems sowie des Erstellens eines neuen Systems.

Räumliche Bezugssysteme

Ein *räumliches Bezugssystem* besteht aus einer Gruppe von Parametern, die zur Darstellung einer Geometrie verwendet werden:

Diese Parameter sind im Einzelnen:

- Der Name des Koordinatensystems, aus dem die Koordinaten abgeleitet werden.
- Die numerische Kennung, die das räumliche Bezugssystem eindeutig kennzeichnet.
- Koordinaten, die die größtmögliche Ausdehnung eines Bereichs definieren, der durch einen angegebenen Bereich von Koordinaten angegeben ist.
- Zahlen, die in mathematischen Operationen angewendet werden, um die als Eingabewerte empfangenen Koordinaten umzuwandeln. Diese Umwandlung hat das Ziel, die Werte mit der größtmöglichen Effizienz zu verarbeiten.

In den folgenden Abschnitten werden die Parameterwerte erläutert, die eine Kennung, die maximale räumliche Ausdehnung und die Umwandlungsfaktoren definieren.

Kennung des räumlichen Bezugssystems

Die Kennung des räumlichen Bezugssystems (SRID = Spatial Reference System Identifier) wird als Eingabeparameter für verschiedene räumliche Funktionen verwendet.

Bereich definieren, der die in einer räumlichen Spalte gespeicherten Koordinaten umgibt

Die Koordinaten in einer räumlichen Spalte definieren normalerweise Positionen, die sich über einen Teil der Erde erstrecken. Der auf diese Weise abgedeckte Bereich (von Ost nach West und von Nord nach Süd) wird als *räumlicher Bereich* bezeichnet. Zur Erläuterung wird das Beispiel eines Überschwemmungsgebiets verwendet, dessen Koordinaten in einer räumlichen Spalte gespeichert sind. Angenommen, die westlichste und die östlichste Koordinate haben die Breitengradwerte - 24,556 bzw. - 19,338, und die nördlichste und südlichste Koordinate haben die Längengradwerte 18,819 bzw. 15,809. Der räumliche Bereich des Überschwemmungsgebiets erstreckt sich über eine West-Ost-Fläche zwischen den beiden Breitengraden und über eine Nord-Süd-Fläche zwischen den beiden Längengraden. Diese Werte können in ein räumliches Bezugssystem aufgenommen werden, indem sie zu bestimmten Parametern zugeordnet werden. Wenn die räumliche Spalte Z-Koordinaten und -Bemaßungen enthält, müssten Sie außerdem auch die höchsten und niedrigsten Z-Koordinaten und -Bemaßungen in das räumliche Bezugssystem aufnehmen.

Der Begriff räumlicher Bereich kann nicht nur auf die tatsächliche Ausdehnung von Standorten (wie im vorherigen Abschnitt) bezogen werden, sondern auch auf eine potenzielle Ausdehnung. Angenommen, das im vorherigen Beispiel beschriebene Überschwemmungsgebiet wird sich in den nächsten fünf Jahren voraussichtlich vergrößern. Sie könnten eine Schätzung bezüglich der westlichsten, östlichsten, nördlichsten und südlichsten Koordinaten des Gebiets am Ende des fünf-

ten Jahres vornehmen. Anschließend könnten Sie diese Schätzwerte (anstelle der aktuellen Koordinaten) zu den Parametern für einen räumlichen Bereich zuordnen. Auf diese Weise könnten Sie das räumliche Bezugssystem auch bei Ausdehnung des Gebiets beibehalten, und dessen größere Breiten- und Längengradwerte werden zur räumlichen Spalte hinzugefügt. Bei einer Begrenzung des räumlichen Bezugssystems auf die ursprünglichen Breiten- und Längengradwerte müssten Sie andernfalls das System im Zuge der Ausweitung des Überschwemmungsgebiets ändern oder ersetzen.

Umwandlung in Werte vornehmen, die die Leistung verbessern

Normalerweise handelt es sich bei den meisten Koordinaten in einem Koordinatensystem um Dezimalzahlen. Manche Werte sind ganze Zahlen. Zusätzlich sind Koordinaten östlich vom Ursprung positive Werte. Koordinaten, die westlich des Ursprungs liegen, sind negative Werte. Negative Koordinaten werden in positive Werte umgewandelt, bevor sie durch DB2 Spatial Extender gespeichert werden. Dezimalkoordinaten werden in ganze Zahlen umgewandelt. Infolgedessen werden alle Koordinaten durch DB2 Spatial Extender in Form von positiven ganzen Zahlen gespeichert. Dies verfolgt den Zweck, die Leistung bei der Verarbeitung der Koordinaten zu verbessern.

Bestimmte Parameter in einem räumlichen Bezugssystem werden verwendet, um die im vorherigen Abschnitt beschriebenen Umwandlungen durchzuführen. Einer der Parameter, das sog. *Offset*, wird von jeder negativen Koordinate subtrahiert und ergibt als Rest einen positiven Wert. Jede Dezimalkoordinate wird mit einem anderen Parameter, dem sog. *Maßstabsfaktor*, multipliziert. Das Ergebnis ist eine ganze Zahl, deren Genauigkeit mit der Genauigkeit der Dezimalkoordinate identisch ist. (Der Offset wird von positiven wie von negativen Koordinaten subtrahiert. Die Multiplikation mit dem Maßstabsfaktor wird nicht nur bei Dezimalkoordinaten vorgenommen, sondern auch bei Koordinaten, die keine Dezimalzahlen sind. Auf diese Weise behalten alle positiven und nicht dezimalen Koordinaten das entsprechende Verhältnis zu negativen und Dezimalkoordinaten bei.)

Die oben beschriebenen Umwandlungen finden intern statt. Sie bleiben nur bis zum Abruf von Koordinaten wirksam. Eingabedaten und Abfrageergebnisse enthalten die Koordinaten immer in ihrer ursprünglichen, also nicht umgewandelten Form.

Über Verwendung eines vorhandenen oder Erstellung eines neuen Bezugssystems entscheiden

Die Beantwortung der folgenden Fragen kann Sie bei der Entscheidung unterstützen, ob ein vorhandenes räumliches Bezugsystem verwendet oder ein neues räumliches Bezugssystem erstellt werden soll.

Informationen zu diesem Vorgang

Nachdem Sie festgelegt haben, welches Koordinatensystem verwendet werden soll, können Sie ein räumliches Bezugssystem für das Koordinatensystem angeben, mit dem Sie arbeiten. DB2 Spatial Extender stellt fünf räumliche Bezugssysteme für räumliche Daten bereit.

Vorgehensweise

Gehen Sie wie folgt vor, um zu entscheiden, ob ein vorhandenes Bezugssystem verwendet oder ein neues Bezugssystem erstellt werden soll:

1. Um festzustellen, ob Sie eines der vorhandenen räumlichen Bezugssysteme verwenden können, müssen Sie die folgenden Fragen beantworten:
   a. Deckt das Koordinatensystem, auf dem das vorhandene räumliche Bezugssystem basiert, den geografischen Bereich ab, mit dem Sie arbeiten? Diese Koordinatensysteme werden in „Räumliche Bezugssysteme, die mit DB2 Spatial Extender geliefert werden" dargestellt.
   b. Können die Umwandlungsfaktoren eines der vorhandenen räumlichen Bezugssysteme für Ihre Koordinatendaten eingesetzt werden?

      DB2 Spatial Extender verwendet Offsetwerte und Maßstabsfaktoren, um die von Ihnen bereitgestellten Koordinatendaten in positive ganze Zahlen (Integerwerte) umzusetzen. Um festzustellen, ob Ihre Koordinatendaten mit den angegebenen Offsetwerten und Maßstabsfaktoren für eines der vorhandenen räumlichen Bezugssysteme arbeiten, müssen Sie wie folgt vorgehen:
      1) Überprüfen Sie die Informationen in „Umwandlungsfaktoren, die Koordinatendaten in Integer umsetzen" auf Seite 49.
      2) Überprüfen Sie, wie diese Faktoren für die vorhandenen räumlichen Bezugssysteme definiert sind. Wenn nach dem Anwenden des Offsetwerts bei den minimalen X- und Y-Koordinaten diese Koordinaten nicht beide größer als 0 sind, müssen sie ein neues räumliches Bezugssystem erstellen und Sie müssen die Abstände selber definieren. Weitere Informationen zur Erstellung eines neuen räumlichen Bezugssystems finden Sie in „Räumliches Bezugssystem erstellen" auf Seite 50.
   c. Enthalten die Daten, mit denen Sie arbeiten, Höhen- und Tiefenkoordinaten (Z-Koordinaten) bzw. Bemaßungen (M-Koordinaten)? Wenn Sie mit Z- und M-Koordinaten arbeiten, müssen Sie möglicherweise ein neues räumliches Bezugssystem mit Z- oder M-Offsetwerten und Maßstabsfaktoren erstellen, die für Ihre Daten geeignet sind.
2. Wenn die vorhandenen räumlichen Bezugssysteme für Ihre Daten nicht eingesetzt werden können, müssen Sie ein eigenes räumliches Bezugssystem (siehe „Räumliches Bezugssystem erstellen" auf Seite 50) erstellen.

Nächste Schritte

Nachdem Sie entschieden haben, welches räumliche Bezugssystem Ihren Anforderungen entspricht, geben Sie diese Auswahl bei DB2 Spatial Extender in Funktionen oder Prozeduraufrufen an.

Räumliche Bezugssysteme, die mit DB2 Spatial Extender geliefert werden

Das räumliche Bezugssystem wandelt die Koordinatendaten in positive ganze Zahlen um.

DB2 Spatial Extender stellt die räumlichen Bezugssysteme zur Verfügung, die in der folgenden Tabelle aufgeführt sind. Außerdem stellt das Produkt die Koordinatensysteme, auf denen die einzelnen räumlichen Bezugssysteme basieren, und die Offsetwerte sowie die Maßstabsfaktoren bereit, die von DB2 Spatial Extender zum Umwandeln der Koordinatendaten in positive ganze Zahlen benutzt werden. Informationen zu diesen räumlichen Bezugssystemen finden Sie in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS.

Wenn Sie mit Dezimalgradwerten arbeiten, unterstützen die Offsetwerte und Maßstabsfaktoren für die standardmäßigen räumlichen Bezugssysteme den gesamten Bereich der Längen- und Breitengradkoordinaten und behalten 6 Dezimalstellen

bei, was ca. 10 cm entspricht. Die räumlichen Beispielkartendaten und die Geocoder-Bezugsdaten benutzen Dezimalgradwerte.

Wenn Sie den Geocoder verwenden wollen, der nur mit US-Adressen arbeitet, sollten Sie unbedingt ein räumliches Bezugssystem auswählen bzw. erstellen, das die Verarbeitung von US-Koordinaten unterstützt (z. B. das Koordinatensystem GCS_NORTH_AMERICAN_1983). Wenn Sie nicht angeben, welches Koordinatensystem für die Ableitung der räumlichen Daten verwendet werden soll, benutzt DB2 Spatial Extender das räumliche Bezugssystem DEFAULT_SRS.

Wenn keines der standardmäßigen räumlichen Bezugssysteme Ihren Bedürfnissen entspricht, können Sie ein neues räumliches Bezugssystem erstellen.

Tabelle 1. Räumliche Bezugssysteme, die mit DB2 Spatial Extender geliefert werden

Räumliches Bezugssystem (SRS)	**SRS-ID**	**Koordinatensystem**	**Offsetwerte**	**Maßstabsfaktoren**	**Verwendung**
DEFAULT_SRS	0	Keines	xOffset = 0 yOffset = 0 zOffset = 0 mOffset = 0	xScale = 1 yScale = 1 zScale = 1 mScale = 1	Sie können dieses System auswählen, wenn Ihre Daten unabhängig von einem Koordinatensystem sind oder wenn Sie keines angeben können oder müssen.
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	Sie können dieses räumliche Bezugssystem auswählen, wenn Sie die US-Beispieldaten verwenden wollen, die mit DB2 Spatial Extender geliefert werden. Wenn die Koordinatendaten, mit denen Sie arbeiten, nach 1983 erfasst wurden, verwenden Sie dieses System statt NAD27_SRS_1002.

Tabelle 1. Räumliche Bezugssysteme, die mit DB2 Spatial Extender geliefert werden (Forts.)

Räumliches Bezugssystem (SRS)	SRS-ID	Koordinatensystem	Offsetwerte	Maßstabsfaktoren	Verwendung
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	Sie können dieses räumliche Bezugssystem auswählen, wenn Sie die US-Beispieldaten verwenden wollen, die mit DB2 Spatial Extender geliefert werden. Wenn die Koordinatendaten, mit denen Sie arbeiten, vor 1983 erfasst wurden, verwenden Sie dieses System statt NAD83_SRS_1. Diese System bietet einen größeren Präzisionsgrad als die anderen räumlichen Bezugssysteme.
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	Sie können dieses räumliche Bezugssystem auswählen, wenn Sie die Daten verwenden wollen, die außerhalb der USA liegen (dieses System verarbeitet weltweite Daten). Verwenden Sie dieses System nicht, wenn Sie den standardmäßigen Geocoder verwenden wollen, der mit DB2 Spatial Extender geliefert wird, da der Geocoder nur für US-Adressen vorgesehen ist.
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	Dieses räumliche Bezugssystem basiert auf einem Koordinatensystem für deutsche Adressen.

Umwandlungsfaktoren, die Koordinatendaten in Integer umsetzen

DB2 Spatial Extender verwendet Offsetwerte und Maßstabsfaktoren, um die von Ihnen bereitgestellten Koordinatendaten in positive ganze Zahlen umzusetzen. Die standardmäßigen räumlichen Bezugssysteme verfügen bereits über zugeordnete Offsetwerte und Maßstabsfaktoren. Wenn Sie ein neues räumliches Bezugssystem erstellen, müssen Sie die Maßstabsfaktoren und (optional) die Offsetwerte festlegen, die am besten für Ihre Daten geeignet sind.

Offsetwerte

Ein Offsetwert ist eine Zahl, die von allen Koordinaten subtrahiert wird, wobei nur positive Werte als Rest bleiben.

DB2 Spatial Extender wandelt Ihre Koordinatendaten mit den folgenden Formeln um, um sicherzustellen, dass alle angepassten Koordinatenwerte größer als 0 sind.

Formelnotation: In diesen Formeln steht die Notation „min" für „das jeweilige Minimum aller Elemente". Beispielsweise steht „min(x)" für das „Minimum aller X-Koordinaten". Der Offsetwert für jede geografische Richtung wird als Dimensionsoffset dargestellt. Beispielsweise ist xOffset der Offsetwert, der auf alle X-Koordinaten angewendet wird.

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

Maßstabsfaktoren

Maßstabsfaktoren sind Werte, deren Multiplikation mit dezimalen Koordinaten und Bemaßungen, ganze Zahlen (Integerwerte) ergibt, wobei mindestens dieselbe Anzahl signifikanter Ziffern ermittelt wird wie bei den ursprünglichen Koordinaten und Bemaßungen.

DB2 Spatial Extender wandelt Ihre Koordinatendaten mit den folgenden Formeln um, um sicherzustellen, dass alle angepassten Koordinatenwerte positive Integerwerte sind. Die umgewandelten Werte dürfen nicht größer als 2^{53} (ca. $9 * 10^{15}$) sein.

Formelnotation: In diesen Formeln steht die Notation „max" für „das Maximum aller Elemente". Der Offset für jede geografische Dimension wird als Dimensionsoffset dargestellt (z. B. ist xOffset der Offsetwert, der auf alle X-Koordinaten angewendet wird). Der Maßstabsfaktor für jede geografische Dimension wird als Dimensionsmaßstab dargestellt (z. B. ist xScale der Maßstabsfaktor, der auf X-Koordinaten angewendet wird).

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

Wenn Sie auswählen, welche Maßstabsfaktoren am besten mit Ihren Koordinatendaten arbeiten, stellen Sie Folgendes sicher:

- Sie verwenden den gleichen Maßstabsfaktor für X- und Y-Koordinaten.
- Wenn er mit einer dezimalen X- oder Y-Koordinate multipliziert wird, ergibt der Maßstabsfaktor einen Wert, der kleiner ist als 2^{53}. Ein häufig verwendetes Verfahren besteht darin, den Maßstabsfaktor als Potenz von 10 darzustellen. Dies bedeutet, dass für den Maßstabsfaktor 10 hoch eins (10), 10 hoch zwei (100), 10 hoch drei (1000) oder ggf. eine höhere Zehnerpotenz verwendet werden sollte.

- Der Maßstabsfaktor ist groß genug, um sicherzustellen, dass die Anzahl der signifikanten Ziffern in dem neuen Integerwert mit der Anzahl der signifikanten Ziffern in der ursprünglichen dezimalen Koordinate übereinstimmt.

Beispiel

Gehen Sie z. B. davon aus, dass eine Eingabe für die Funktion ST_Point die X-Koordinate 10,01, die Y-Koordinate 20,03 und die Kennung eines räumlichen Bezugssystems umfasst. Wenn die Funktion ST_Point aufgerufen wird, multipliziert sie den Wert 10,01 und den Wert 20,03 mit dem Maßstabsfaktor des räumlichen Bezugssystems für X- und Y-Koordinaten. Wenn dieser Maßstabsfaktor 10 ist, lauten die resultierenden Integerwerte, die DB2 Spatial Extender speichert, 100 bzw. 200. Da die Anzahl der signifikanten Ziffern dieser Integerwerte (3) niedriger ist als die der signifikanten Ziffern in den Koordinaten (4), kann DB2 Spatial Extender diese Integerwerte nicht wieder in die ursprünglichen Koordinaten umwandeln und keine Werte von ihnen ableiten, die mit dem Koordinatensystem konsistent sind, zu dem diese Koordinaten gehören. Ist der Maßstabsfaktor jedoch 100, speichert DB2 Spatial Extender die ganzen Zahlen 1001 und 2003. Diese Werte können wieder in die ursprünglichen Koordinaten umgewandelt werden und es können kompatible Koordinaten von diesen Werten abgeleitet werden.

Einheiten für Offsetwerte und Maßstabsfaktoren

Ob Sie ein bestehendes räumliches Bezugssystem verwenden oder ein neues erstellen - die Einheiten für die Offsetwerte und Maßstabsfaktoren unterscheiden sich jeweils abhängig vom Typ des von Ihnen verwendeten Koordinatensystems.

Wenn Sie beispielsweise ein geografisches Koordinatensystem verwenden, werden die Werte in Winkeleinheiten (z. B. in Dezimalgraden) angegeben. Wenn Sie ein projiziertes Koordinatensystem verwenden, werden die Werte in linearen Einheiten (z. B. in Meter oder Fuß) angegeben.

Räumliches Bezugssystem erstellen

Wenn sich keines der zum Lieferumfang von DB2 Spatial Extender gehörenden räumlichen Bezugssysteme für die Verarbeitung Ihrer Daten eignet, können Sie auch ein neues räumliches Bezugssystem erstellen.

Vorgehensweise

Gehen Sie wie folgt vor, um ein räumliches Bezugssystem zu erstellen:

1. Wählen Sie die ID eines räumlichen Bezugssystems (SRID) aus, die noch nicht belegt ist.
2. Geben Sie den Grad der Genauigkeit für das räumliche Bezugssystem an, indem Sie eine der folgenden Methoden einsetzen:
   - Angabe der Ausdehnung des geografischen Bereichs, mit dem Sie arbeiten, und der Maßstabsfaktoren, die Sie mit Ihren Koordinatendaten verwenden wollen. Spatial Extender berechnet die Offsetwerte.
   - Angabe der Offsetwerte (für DB2 Spatial Extender zur Umwandlung negativer Werte in positive Werte) und Maßstabsfaktoren (für DB2 Spatial Extender zur Umwandlung dezimaler Werte in ganze Zahlen erforderlich). Verwenden Sie diese Methode, wenn in Bezug auf Richtigkeit und Genauigkeit strikte Regeln eingehalten werden müssen.

3. Berechnen Sie die Umwandlungsinformationen, die DB2 Spatial Extender benötigt, um die Koordinatendaten in positive Integerwerte umzusetzen. Die zu berechnenden Informationen variieren abhängig von der Methode, die Sie unter 2 auf Seite 50 ausgewählt haben.
   - Für die Methode Bereiche müssen Sie die folgenden Informationen berechnen:
     - *Maßstabsfaktoren*. Wenn Koordinaten, mit denen Sie arbeiten, Dezimalwerte enthalten, müssen Sie Maßstabsfaktoren berechnen. Bei Maßstabsfaktoren handelt es sich um numerische Werte, deren Multiplikation mit dezimalen Koordinaten und Bemaßungen Integerwerte ergibt, wobei mindestens dieselbe Anzahl signifikanter Ziffern ermittelt wird wie bei den ursprünglichen Koordinaten und Bemaßungen. Wenn die Koordinaten ganze Zahlen sind, können die Maßstabsfaktoren auf 1 festgelegt werden. Handelt es sich bei den Koordinaten um Dezimalwerte, sollte für den Maßstabsfaktor eine Zahl angegeben werden, die den Dezimalteil in einen ganzen Wert umwandelt. Wenn beispielsweise die Koordinateneinheiten Meter sind und die Genauigkeit der Daten 1 cm beträgt, würden Sie den Maßstabsfaktor 100 benötigen.
     - *Minimal- und Maximalwerte* für Ihre Koordinaten und Bemaßungen.
   - Für die Methode Relative Position müssen Sie die folgenden Informationen berechnen:
     - *Offsetwerte*. Wenn Ihre Koordinatendaten negative Zahlen oder Bemaßungen enthalten, müssen Sie die Offsetwerte angeben, die Sie verwenden wollen. Ein Offset (relative Position) ist eine Zahl, die von allen Koordinaten subtrahiert wird, sodass nur positive Werte übrig bleiben. Wenn Sie mit positiven Koordinaten arbeiten, setzen Sie alle Offsetwerte auf 0. Wenn Sie nicht mit positiven Koordinaten arbeiten, wählen Sie einen Offset aus, dessen Anwendung auf die Koordinatendaten ganzzahlige Ergebnisse (Integerwerte) erzeugt, die kleiner als der Wert des größten positiven Integerwerts (9.007.199.254.740.992) sind.
     - *Maßstabsfaktoren*. Falls die Koordinaten für die darzustellenden Positionen Dezimalzahlen enthalten, ermitteln Sie, welche Maßstabsfaktoren zu verwenden sind, und geben Sie diese Maßstabsfaktoren im Fenster "Räumliches Bezugssystem erstellen" ein.
4. Geben Sie den Befehl **db2se create_srs** aus, oder rufen Sie die Prozedur DB2SE.ST_CREATE_SRS auf, um das räumliche Bezugssystem zu erstellen.

   Im folgenden Beispiel wird dargestellt, wie ein räumliches Bezugssystem namens 'mysrs' erstellt wird:

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

Maßstabsfaktoren berechnen

Wenn Sie ein räumliches Bezugssystem erstellen und hierbei Koordinaten verwenden, die Dezimalwerte aufweisen, müssen Sie die entsprechenden Maßstabsfaktoren für Ihre Koordinaten und Bemaßungen berechnen. Bei Maßstabsfaktoren handelt es sich um numerische Werte, deren Multiplikation mit dezimalen Koordinaten und Bemaßungen Integerwerte ergibt, wobei mindestens dieselbe Anzahl signifikanter Ziffern ermittelt wird wie bei den ursprünglichen Koordinaten und Bemaßungen.

Informationen zu diesem Vorgang

Nach der Berechnung von Maßstabsfaktoren müssen Sie die Bereichswerte festlegen. Anschließend setzen Sie den Befehl **db2se create_srs** ab oder rufen die Prozedur DB2SE.ST_CREATE_SRS auf.

Vorgehensweise

Gehen Sie wie folgt vor, um Maßstabsfaktoren zu berechnen:

1. Ermitteln Sie, welche X- und Y-Koordinaten Dezimalzahlen sind bzw. sein werden. Angenommen, Sie stellen fest, dass drei der verschiedenen X- und Y-Koordinaten, mit denen Sie arbeiten, Dezimalzahlen sind: 1,23, 5,1235 und 6,789.
2. Suchen Sie die Dezimalkoordinate mit der längsten Dezimalgenauigkeit. Stellen Sie anschließend fest, mit welcher Zehnerpotenz diese Koordinate multipliziert werden kann, um eine ganze Zahl mit gleichwertiger Genauigkeit zu erhalten. In unserem Beispiel hat 5,1235 von den drei Dezimalkoordinaten die längste Dezimalgenauigkeit. Durch eine Multiplikation mit zehn hoch vier (10000) ergibt sich die ganze Zahl 51235.
3. Stellen Sie fest, ob die durch die Multiplikation ermittelte ganze Zahl kleiner als 2^{53} ist. 51235 ist nicht zu lang. Nehmen Sie nun aber an, dass Ihr Bereich von X- und Y-Koordinaten zusätzlich zu 1,23, 5,11235 und 6,789 noch den vierten Dezimalwert 10000000006,789876 umfasst. Da die Dezimalgenauigkeit dieser Koordinate länger als die der anderen drei Koordinaten ist, müssen Sie diese Koordinate (und nicht 5,1235) mit einer Zehnerpotenz multiplizieren. Zur Umwandlung in eine ganze Zahl könnten Sie sie mit zehn hoch sechs (1000000) multiplizieren. Der resultierende Wert von 10000000006789876 ist jedoch größer als 2^{53}. Wenn DB2 Spatial Extender versucht, diesen Wert zu speichern, sind die Ergebnisse nicht vorhersehbar.

   Vermeiden Sie dieses Problem, indem Sie eine Potenz von zehn auswählen, die bei Multiplikation mit der ursprünglichen Koordinate eine Dezimalzahl ergibt, die DB2 Spatial Extender so abschneiden kann, dass sich bei minimalem Genauigkeitsverlust eine ganze Zahl ergibt, die gespeichert werden kann. In diesem Fall können Sie zehn hoch fünf (100000) auswählen. Durch die Multiplikation von 100000 mit 10000000006,789876 erhalten Sie den Wert 1000000000678987,6. DB2 Spatial Extender rundet diese Zahl auf den Wert 1000000000678988, wodurch die Genauigkeit geringfügig reduziert wird.

Umwandlungsfaktoren, die Koordinatendaten in Integer umsetzen

DB2 Spatial Extender verwendet Offsetwerte und Maßstabsfaktoren, um die von Ihnen bereitgestellten Koordinatendaten in positive ganze Zahlen umzusetzen. Die standardmäßigen räumlichen Bezugssysteme verfügen bereits über zugeordnete Offsetwerte und Maßstabsfaktoren. Wenn Sie ein neues räumliches Bezugssystem erstellen, müssen Sie die Maßstabsfaktoren und (optional) die Offsetwerte festlegen, die am besten für Ihre Daten geeignet sind.

Minimal- und Maximalkoordinaten und Bemaßungen ermitteln

Minimal- und Maximalkoordinaten sowie Bemaßungen sollten ermittelt werden, wenn Sie bei der Erstellung eines räumlichen Bezugssystems Bereichstransformationen angeben.

Informationen zu diesem Vorgang

Nach der Festlegung der Bereichswerte müssen Sie, wenn die Koordinaten Dezimalwerte umfassen, die Maßstabsfaktoren berechnen. Oder Sie geben den Befehl **db2se create_srs** ein bzw. rufen die Prozedur DB2SE.ST_CREATE_SRS auf.

Vorgehensweise

Gehen Sie wie folgt vor, um die Maximal- und Minimalkoordinaten sowie die Bemaßungen der Positionen zu ermitteln, die Sie darstellen wollen:

1. Ermitteln Sie die minimalen und maximalen X-Koordinaten. Geben Sie die X-Koordinate in Ihrem Bereich an, die sich am weitesten im Westen befindet, um die minimale X-Koordinate zu finden. (Falls der Standort westlich vom Ursprungspunkt liegt, ist diese Koordinate ein negativer Wert.) Geben Sie die X-Koordinate in Ihrem Bereich an, die sich am weitesten im Osten befindet, um die maximale X-Koordinate zu finden. Wenn Sie beispielsweise Ölquellen darstellen wollen und jede Quelle durch ein Paar aus X- und Y-Koordinate definiert ist, ist die westlichste X-Koordinate, die die Position der Ölquelle angibt, die minimale X-Koordinate und die östlichste X-Koordinate, die die Position der Ölquelle angibt, die maximale X-Koordinate.

   Tipp: Für Typen mit mehreren Funktionen wie Multipolygone stellen Sie sicher, dass Sie den weitesten Punkt auf dem weitesten Polygon in der Richtung auswählen, den Sie berechnen. Wenn Sie beispielsweise versuchen, die minimale X-Koordinate anzugeben, geben Sie die westlichste X-Koordinate des Polygons an, die sich am weitesten im Westen des Multipolygons befindet.
2. Ermitteln Sie die minimalen und maximalen Y-Koordinaten. Geben Sie die Y-Koordinate in Ihrem Bereich an, die sich am weitesten im Süden befindet, um die minimale X-Koordinate zu finden. (Falls der Standort südlich vom Ursprungspunkt liegt, ist diese Koordinate ein negativer Wert.) Suchen Sie die Y-Koordinate in Ihrem Bereich, die sich am weitesten im Norden befindet, um die maximale Y-Koordinate zu ermitteln.
3. Ermitteln Sie die minimalen und maximalen Z-Koordinaten. Die minimale Z-Koordinate ist die größte der Tiefenkoordinaten und die maximale Z-Koordinate ist die größte der Höhenkoordinaten.
4. Ermitteln Sie die minimalen und maximalen Bemaßungen. Falls Sie Bemaßungen in die räumlichen Daten einbeziehen wollen, ermitteln Sie, welche Bemaßung den höchsten numerischen Wert hat.

Offsetwerte berechnen

Geben Sie Offsetwerte an, wenn Ihre Koordinatendaten negative Zahlen oder Bemaßungen umfassen. Ein Offset (relative Position) ist eine Zahl, die von allen Koordinaten subtrahiert wird, sodass nur positive Werte übrig bleiben.

Informationen zu diesem Vorgang

Wenn Ihre Koordinatendaten bei der Erstellung eines räumlichen Bezugssystems negative Zahlen oder Bemaßungen enthalten, müssen Sie die Offsetwerte angeben, die Sie verwenden wollen. Sie können die Leistung räumlicher Operationen verbessern, wenn als Koordinaten positive ganze Zahlen anstelle negativer Zahlen oder Bemaßungen verwendet werden.

Vorgehensweise

Gehen Sie wie folgt vor, um Offsetwerte für die Koordinaten zu berechnen, mit denen Sie arbeiten:

1. Bestimmen Sie die niedrigsten negativen X-, Y- und Z-Koordinaten unter den Koordinaten der Standorte, die Sie darstellen wollen. Wenn die Daten negative Bemaßungen enthalten, ermitteln Sie die niedrigste Bemaßung.
2. Optionaler, aber empfohlener Schritt: Geben Sie für DB2 Spatial Extender ein größeres Gebiet an als das Gebiet, das die betreffenden Standorte tatsächlich enthält. Auf diese Weise können Sie nach der Aufnahme der Daten zu diesen Standorten in eine räumliche Spalte Daten zu den Standorten neuer Objekte hinzufügen, die an den äußeren Rändern Ihres Bereichs hinzugefügt werden, ohne das räumliche Bezugssystem durch ein anderes räumliches Bezugssystem ersetzen zu müssen.

   Fügen Sie für alle Koordinaten und Bemaßungen, die in Schritt 1 ermittelt wurden, einen Betrag von fünf bis zehn Prozent der jeweiligen Koordinate oder Bemaßung hinzu. Das Ergebnis wird als *erweiterter Wert* bezeichnet. Wenn die niedrigste negative X-Koordinate beispielsweise -100 ist, könnten Sie -5 addieren und so einen erweiterten Wert von -105 erhalten. Später geben Sie bei der Erstellung des räumlichen Bezugssystems an, dass die niedrigste X-Koordinate -105 ist, statt den realen Wert von -100 zu verwenden. DB2 Spatial Extender interpretiert dann die Koordinate -105 als die westlichste Grenze des Gebiets.
3. Suchen Sie einen Wert, der nach dem Subtrahieren von Ihrem erweiterten X-Wert den Wert null liefert. Dies ist der Offsetwert für X-Koordinaten. DB2 Spatial Extender subtrahiert diese Zahl von allen X-Koordinaten, um ausschließlich positive Werte hervorzubringen.

   Wenn der erweiterte X-Wert beispielsweise -105 lautet, müssen Sie von diesem Wert -105 subtrahieren, um 0 zu erhalten. DB2 subtrahiert dann -105 von allen X-Koordinaten, die den dargestellten Objekten zugeordnet sind. Da keine dieser Koordinaten größer als -100 ist, sind alle durch diese Subtraktion erhaltenen Werte positiv.
4. Wiederholen Sie Schritt 3 für den erweiterten Y-Wert, den erweiterten Z-Wert und die erweiterte Bemaßung.

Räumliches Bezugssystem erstellen

Wenn sich keines der zum Lieferumfang von DB2 Spatial Extender gehörenden räumlichen Bezugssysteme für die Verarbeitung Ihrer Daten eignet, können Sie auch ein neues räumliches Bezugssystem erstellen.

Vorgehensweise

Gehen Sie wie folgt vor, um ein räumliches Bezugssystem zu erstellen:

1. Wählen Sie die ID eines räumlichen Bezugssystems (SRID) aus, die noch nicht belegt ist.
2. Geben Sie den Grad der Genauigkeit für das räumliche Bezugssystem an, indem Sie eine der folgenden Methoden einsetzen:
   - Angabe der Ausdehnung des geografischen Bereichs, mit dem Sie arbeiten, und der Maßstabsfaktoren, die Sie mit Ihren Koordinatendaten verwenden wollen. Spatial Extender berechnet die Offsetwerte.
   - Angabe der Offsetwerte (für DB2 Spatial Extender zur Umwandlung negativer Werte in positive Werte) und Maßstabsfaktoren (für DB2 Spatial Extender

zur Umwandlung dezimaler Werte in ganze Zahlen erforderlich). Verwenden Sie diese Methode, wenn in Bezug auf Richtigkeit und Genauigkeit strikte Regeln eingehalten werden müssen.

3. Berechnen Sie die Umwandlungsinformationen, die DB2 Spatial Extender benötigt, um die Koordinatendaten in positive Integerwerte umzusetzen. Die zu berechnenden Informationen variieren abhängig von der Methode, die Sie unter 2 auf Seite 50 ausgewählt haben.
   - Für die Methode Bereiche müssen Sie die folgenden Informationen berechnen:
     - *Maßstabsfaktoren*. Wenn Koordinaten, mit denen Sie arbeiten, Dezimalwerte enthalten, müssen Sie Maßstabsfaktoren berechnen. Bei Maßstabsfaktoren handelt es sich um numerische Werte, deren Multiplikation mit dezimalen Koordinaten und Bemaßungen Integerwerte ergibt, wobei mindestens dieselbe Anzahl signifikanter Ziffern ermittelt wird wie bei den ursprünglichen Koordinaten und Bemaßungen. Wenn die Koordinaten ganze Zahlen sind, können die Maßstabsfaktoren auf 1 festgelegt werden. Handelt es sich bei den Koordinaten um Dezimalwerte, sollte für den Maßstabsfaktor eine Zahl angegeben werden, die den Dezimalteil in einen ganzen Wert umwandelt. Wenn beispielsweise die Koordinateneinheiten Meter sind und die Genauigkeit der Daten 1 cm beträgt, würden Sie den Maßstabsfaktor 100 benötigen.
     - *Minimal- und Maximalwerte* für Ihre Koordinaten und Bemaßungen.
   - Für die Methode Relative Position müssen Sie die folgenden Informationen berechnen:
     - *Offsetwerte*. Wenn Ihre Koordinatendaten negative Zahlen oder Bemaßungen enthalten, müssen Sie die Offsetwerte angeben, die Sie verwenden wollen. Ein Offset (relative Position) ist eine Zahl, die von allen Koordinaten subtrahiert wird, sodass nur positive Werte übrig bleiben. Wenn Sie mit positiven Koordinaten arbeiten, setzen Sie alle Offsetwerte auf 0. Wenn Sie nicht mit positiven Koordinaten arbeiten, wählen Sie einen Offset aus, dessen Anwendung auf die Koordinatendaten ganzzahlige Ergebnisse (Integerwerte) erzeugt, die kleiner als der Wert des größten positiven Integerwerts (9.007.199.254.740.992) sind.
     - *Maßstabsfaktoren*. Falls die Koordinaten für die darzustellenden Positionen Dezimalzahlen enthalten, ermitteln Sie, welche Maßstabsfaktoren zu verwenden sind, und geben Sie diese Maßstabsfaktoren im Fenster "Räumliches Bezugssystem erstellen" ein.
4. Geben Sie den Befehl **db2se create_srs** aus, oder rufen Sie die Prozedur DB2SE.ST_CREATE_SRS auf, um das räumliche Bezugssystem zu erstellen.

   Im folgenden Beispiel wird dargestellt, wie ein räumliches Bezugssystem namens 'mysrs' erstellt wird:

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

Kapitel 8. Räumliche Spalten konfigurieren

Bei der Vorbereitung zum Abrufen räumlicher Daten für ein Projekt müssen Sie ein Koordinatensystem und ein räumliches Bezugssystem auswählen und dann mindestens eine Tabellenspalte für die Daten zur Verfügung stellen.

Darstellung räumlicher Spalten

Bei Verwendung eines Darstellungstools, z. B. ArcExplorer for DB2, gibt das Tool bei der Abfrage einer räumlichen Spalte die Ergebnisse in Form einer grafischen Anzeige zurück, beispielsweise als Karte mit Parzellengrenzen oder als Layout eines Straßensystems.

Bei manchen Darstellungstools ist es erforderlich, dass alle Zeilen der Spalte dasselbe räumliche Bezugssystem verwenden. Diese Integritätsbedingung können Sie durchsetzen, indem Sie die Spalte für ein räumliches Bezugssystem registrieren.

Räumliche Datentypen

Wenn Sie eine Datenbank für räumliche Operationen aktivieren, stellt DB2 Spatial Extender der Datenbank eine Hierarchie strukturierter Datentypen zur Verfügung.

Abb. 12 zeigt diese Hierarchie. In dieser Abbildung haben die instanziierbaren Typen einen weißen Hintergrund; die nicht-instanziierbaren Typen sind vor einem grauen Hintergrund dargestellt.

Instanziierbare Datentypen sind ST_Point, ST_LineString, ST_Polygon, ST_GeomCollection, ST_MultiPoint, ST_MultiPolygon und ST_MultiLineString.

Nicht instanziierbare Datentypen sind ST_Geometry, ST_Curve, ST_Surface, ST_MultiSurface und ST_MultiCurve.

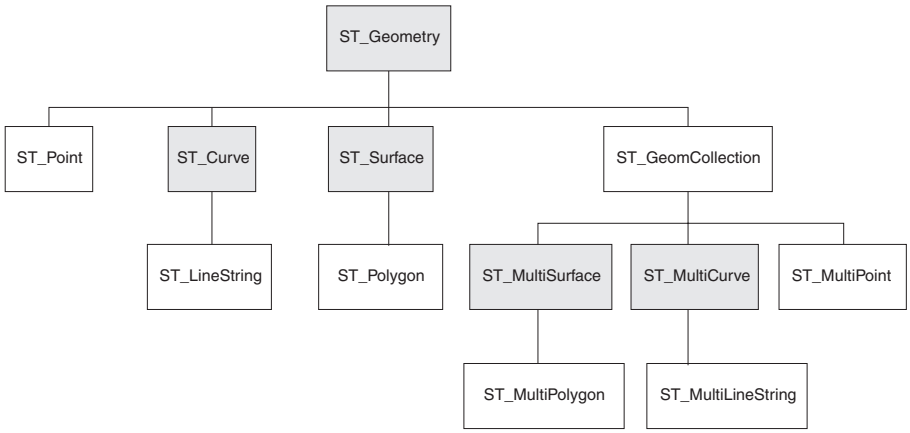


Abbildung 12. Hierarchie der räumlichen Datentypen. Die Datentypen in weißen Kästchen sind instanziierbar. Die Datentypen in grauen Kästchen sind nicht instanziierbar.

Die Hierarchie in Abb. 12 umfasst:

- Datentypen für geografische Objekte, die als Einheit betrachtet werden können, beispielsweise einzelne Wohngebiete und isolierte Seen.

- Datentypen für geografische Objekte, die aus mehreren Einheiten oder Komponenten bestehen, beispielsweise Kanalsysteme und Inselgruppen in einem See.
- Ein Datentyp für geografische Objekte aller Art.

Datentypen für einteilige Objekte

Verwenden Sie ST_Point, ST_LineString und ST_Polygon zum Speichern von Koordinaten zur Definition des Raumes, der von Objekten belegt wird, die aus nur einer einer einzigen Komponente bestehen.

- Verwenden Sie ST_Point, wenn Sie einen Punkt im Raum kennzeichnen wollen, der von einem diskreten geografischen Objekt belegt wird. Das Objekt kann sehr klein (z. B. eine Wasserquelle), sehr groß (z. B. eine Stadt) oder von mittlerer Größe (z. B. ein Gebäudekomplex oder ein Park) sein. In jedem Fall befindet sich der Punkt im Raum am Schnittpunkt der Ost-West-Koordinatenlinie (z. B. einem Breitengrad) mit einer Nord-Süd-Koordinatenlinie (z. B. einem Längengrad). Ein Datenelement des Typs ST_Point enthält eine X-Koordinate und eine Y-Koordinate, die einen solchen Schnittpunkt definieren. Die X-Koordinate gibt an, wo der Punkt auf der Ost-West-Linie liegt. Die Y-Koordinate gibt an, wo er auf der Nord-Süd-Linie liegt.
- Verwenden Sie ST_LineString für Koordinaten, die den Raum definieren, der von linearen Objekten (z. B. Straßen, Kanälen oder Rohrleitungen) belegt wird.
- Verwenden Sie ST_Polygon, wenn Sie einen Bereich kennzeichnen wollen, der von einem mehrseitigen Objekt abgedeckt wird, beispielsweise von einem Wahlbezirk, Wald oder Naturschutzgebiet. Ein Datenelement des Typs ST_Polygon besteht aus den Koordinaten, die die Begrenzung eines solchen Objekts definieren.

In manchen Fällen können ST_Polygon und ST_Point für dasselbe Objekt verwendet werden. Angenommen, Sie benötigen räumliche Informationen über einen Apartmentkomplex. Wenn Sie einen Punkt im Raum darstellen wollen, an dem sich die einzelnen Gebäude in dem Komplex befinden, verwenden Sie ST_Point zum Speichern der X- und Y-Koordinaten, die einen solchen Punkt definieren. Wenn Sie andererseits den Bereich darstellen wollen, der insgesamt durch den Komplex belegt ist, verwenden Sie ST_Polygon zum Speichern der Koordinaten, die die Begrenzung des Bereichs angeben.

Datentypen für mehrteilige Objekte

Verwenden Sie ST_MultiPoint, ST_MultiLineString und ST_MultiPolygon zum Speichern von Koordinaten für den Raum, der von Objekten belegt wird, die aus mehreren Teilen (Einheiten) bestehen.

- Verwenden Sie ST_MultiPoint für Objekte, die aus Einheiten bestehen, deren Standorte durch eine X-Koordinate und eine Y-Koordinate angegeben werden können. Ein Beispiel hierfür wäre eine Tabelle, deren Zeilen Inselketten darstellen. Für jede Insel wurde die X-Koordinate und die Y-Koordinate angegeben. Wenn die Tabelle diese Koordinaten und die Koordinaten jeder Kette als Ganzes enthalten soll, definieren Sie eine Spalte des Typs ST_MultiPoint für diese Koordinaten.
- Verwenden Sie ST_MultiLineString, wenn Objekte aus linearen Einheiten bestehen und Sie die Koordinaten für die Standorte dieser Einheiten sowie den Standort eines Objekts als Ganzes speichern wollen. Ein Beispiel hierfür wäre eine Tabelle, deren Zeilen Fluss-Systeme darstellen. Wenn die Tabelle Koordinaten für die Standorte dieser Systeme und deren Komponenten enthalten soll, definieren Sie für diese Koordinaten eine Spalte des Typs ST_MultiLineString.

- Verwenden Sie ST_MultiPolygon, wenn Objekte aus mehrseitigen Einheiten bestehen und Sie die Koordinaten für die Standorte dieser Einheiten sowie den Standort des Objekts als Ganzes speichern wollen. Ein Beispiel wäre eine Tabelle, deren Zeilen Landkreise und die Landwirtschaftsbetriebe in jedem Landkreis darstellen. Wenn die Tabelle Koordinaten für die Standorte der Landkreise und der Landwirtschaftsbetriebe enthalten soll, definieren Sie für diese Koordinaten eine Spalte des Typs ST_MultiPolygon.

Ein aus mehreren Einheiten bestehendes Objekt (mehrteiliges Objekt) darf nicht als Gruppe einzelner Einheiten verstanden werden. Dieser Terminus bezeichnet vielmehr einen Verbund der einelnen Teile, aus denen die Gesamteinheit besteht.

Universaldatentyp für alle Objekte

Wenn Sie sich nicht sicher sind, welcher der anderen Datentypen verwendet werden muss, können Sie den Datentyp ST_Geometry verwenden.

Da der Typ ST_Geometry den Stamm der Hierarchie bildet, zu der die anderen Datentypen gehören, kann eine Spalte des Typs ST_Geometry dieselben Datenelemente wie Spalten der anderen Datentypen enthalten.

Achtung: Manche Darstellungstools unterstützen keine Spalten des Typs ST_Geometry, sondern nur Spalten, denen ein geeigneter untergeordneter Typ von ST_Geometry zugeordnet wurde.

Räumliche Spalten erstellen

Räumliche Spalten müssen erstellt werden, um räumliche Daten speichern und abzurufen zu können.

Vorbereitende Schritte

Damit Sie eine räumliche Spalte erstellen können, muss Ihre Benutzer-ID die Berechtigungen besitzen, die für die DB2-SQL-Anweisung CREATE TABLE oder ALTER TABLE erforderlich sind. Die Benutzer-ID muss mindestens über eine der folgenden Berechtigungen verfügen:

- Berechtigung DBADM für die Datenbank, die die Tabelle enthält, in der sich die Spalte befindet
- Berechtigung CREATETAB für die Datenbank und das Zugriffsrecht USE für den Tabellenbereich sowie eine der folgenden Berechtigungen bzw. Zugriffsrechte:
  - Berechtigung IMPLICIT_SCHEMA für die Datenbank, falls das Schema des Index nicht vorhanden ist
  - Zugriffsrecht CREATEIN für das Schema, wenn der Schemaname des Indexes auf ein vorhandenes Schema verweist
- Zugriffsrecht ALTER für die zu ändernde Tabelle
- Zugriffsrecht CONTROL für die zu ändernde Tabelle
- Zugriffsrecht ALTERIN für das Schema der zu ändernden Tabellen

Informationen zu diesem Vorgang

Diese Task wird im Rahmen der übergeordneten Task zur Einrichtung räumlicher Ressourcen für ein Projekt ausgeführt. Nach der Auswahl des gewünschten Koordinatensystems und der Festlegung des für die Daten zu verwendenden räumlichen Bezugssystems müssen Sie eine räumliche Spalte in einer bereits vorhandenen Tabelle erstellen bzw. räumliche Daten in eine neue Tabelle importieren.

Vorgehensweise

Gehen Sie wie folgt vor, um räumliche Spalten zu erstellen:

Es gibt die folgenden Methoden, mit denen Sie räumliche Spalten für Ihre Datenbank bereitstellen können:

- Verwenden Sie die SQL-Anweisung CREATE TABLE, um eine Tabelle zu erstellen und eine räumliche Spalte in diese Tabelle aufzunehmen.
- Mit der SQL-Anweisung ALTER TABLE können Sie einer vorhandenen Tabelle eine räumliche Spalte hinzufügen.
- Falls Sie räumliche Daten aus einer Formdatei importieren, verwenden Sie DB2 Spatial Extender, um eine Tabelle zu erstellen und in diese Tabelle eine Spalte für die Daten aufzunehmen. Weitere Informationen finden Sie unter "Formdaten in eine neue oder eine vorhandene Tabelle importieren".

Nächste Schritte

Die nächste Task zum Definieren räumlicher Ressourcen besteht im „Registrieren räumlicher Spalten”.

Räumliche Spalten registrieren

Beim Registrieren einer räumlichen Spalte wird (sofern möglich) eine Integritätsbedingung für die Tabelle erstellt, die sicherstellen soll, dass alle Geometrien das angegebene räumliche Bezugssystem verwenden.

Vorbereitende Schritte

Wenn Sie den Befehlszeilenprozessor db2se oder ein Anwendungsprogramm benutzen, dann muss Ihre Benutzer-ID für die Datenbank über die Berechtigung SYSADM oder DBADM verfügen.

Informationen zu diesem Vorgang

In den folgenden Fällen kann es sinnvoll sein, eine räumliche Spalte zu registrieren:

- Zugriff durch Darstellungstools

  Wenn Sie durch bestimmte Darstellungstools (z. B. ArcExplorer for DB2) grafische Anzeigen der Daten in einer räumlichen Spalte generieren lassen möchten, müssen Sie die Integrität der Daten in der Spalte sicherstellen. Dazu definieren Sie die Integritätsbedingung, dass alle Zeilen der Spalte dasselbe räumliche Bezugssystem verwenden müssen. Zur Inkraftsetzung dieser Integritätsbedingung registrieren Sie die Spalte, wobei Sie ihren Namen und das räumliche Bezugssystem angeben, das für die Spalte gilt.
- Zugriff durch räumliche Indizes

  Verwenden Sie dasselbe Koordinatensystem für alle Daten einer räumlichen Spalte, für die ein Index erstellt werden soll. Hierdurch wird sichergestellt, dass der räumliche Index die korrekten Ergebnisse zurückgibt. Räumliche Spalten werden registriert, um festzulegen, dass alle Daten dasselbe räumliche Bezugssystem und außerdem dasselbe Koordinatensystem verwenden müssen.

Vorgehensweise

Registrieren Sie eine räumliche Spalte, indem Sie eine der folgenden Methoden anwenden:

- Setzen Sie den Befehl 'db2se register_spatial_column' ab.
- Führen Sie eine Anwendung aus, die die Prozedur DB2GSE.ST_REGISTER_SPATIAL_COLUMN aufruft.

Überprüfen Sie anhand der Spalte SRS_NAME in der Sicht DB2GSE.ST_GEOMETRY_COLUMNS das räumliche Bezugssystem, das Sie für eine bestimmte Spalte nach der Registrierung der Spalte auswählen.

Kapitel 9. Räumliche Spalten ausfüllen

Nachdem Sie räumliche Spalten erstellt und diejenigen registriert haben, auf die von diesen Darstellungstools zugegriffen werden soll, können Sie die Spalten mit räumlichen Daten füllen, indem Sie diese importieren oder mithilfe eines Geocoders aus Unternehmensdaten ableiten. Sie können auch räumliche Funktionen verwenden, um die Daten zu erstellen oder aus Unternehmensdaten bzw. anderen räumlichen Daten abzuleiten.

Informationen zum Importieren und Exportieren von räumlichen Daten

Mit DB2 Spatial Extender können Sie räumliche Daten zwischen der Datenbank und externen Datenquellen austauschen. Genauer gesagt können Sie die räumlichen Daten aus externen Quellen importieren, indem Sie diese in Form von Dateien an die Datenbank übertragen. Diese Dateien werden als Datenaustauschdateien bezeichnet. Sie können räumliche Daten auch aus Ihrer Datenbank in Datenaustauschdateien exportieren, aus denen sie durch externe Quellen entnommen werden können. Der folgende Abschnitt stellt einige Gründe für das Importieren und Exportieren von räumlichen Daten vor und beschreibt wesentliche Merkmale der Datenaustauschdateien, die von DB2 Spatial Extender unterstützt werden.

Gründe für das Importieren und Exportieren von räumlichen Daten

Durch das Importieren von räumlichen Daten können Sie große Mengen von räumlichen Informationen nutzen, die bereits in der Industrie vorhanden sind. Durch einen Export können Sie diese Daten für vorhandene Anwendungen in einem Standarddateiformat verfügbar machen. Dies könnte beispielsweise für folgende Szenarios sinnvoll sein:

- Ihre Datenbank enthält räumliche Daten für Ihre Verkaufsniederlassungen, Kunden und andere Unternehmensaspekte. Sie wollen diese Daten durch räumliche Daten zur Infrastrukturumgebung Ihres Unternehmens ergänzen (z. B. Städte, Straßen, Verkehrsknotenpunkte usw.). Die gewünschten Daten sind bei einem Kartenhersteller erhältlich. Mit DB2 Spatial Extender können Sie die Daten aus einer Datenaustauschdatei importieren, die vom Hersteller geliefert wird.
- Sie wollen räumliche Daten von einem Oracle-System auf Ihre DB2-Umgebung migrieren. Zunächst schreiben Sie die Daten mit einem Oracle-Dienstprogramm in eine Datenaustauschdatei. Anschließend importieren Sie die Daten mit DB2 Spatial Extender aus dieser Datei in die Datenbank, die Sie für räumliche Operationen eingerichtet haben.
- Sie sind nicht mit DB2 verbunden und möchten Kunden mit einem Geobrowser visuelle Darstellungen von räumlichen Informationen zeigen. Der Browser benötigt lediglich Daten, mit denen er arbeiten kann. Eine Verbindung zu einer Datenbank ist nicht erforderlich. In diesem Fall können Sie die Daten mit DB2 Spatial Extender in eine Datenaustauschdatei exportieren und anschließend durch den Browser in visueller Form wiedergeben lassen.

Formdateien

DB2 Spatial Extender unterstützt Formdateien für den Datenaustausch. Der Begriff Formdatei bezeichnet eigentlich eine Gruppe von Dateien mit identischen Dateina-

men, aber unterschiedlichen Dateierweiterungen. Eine solche Gruppe von Dateien kann bis zu vier Dateien umfassen. Dies sind folgende Dateien:

- Eine Datei, die räumliche Daten im Formformat enthält. Hierbei handelt es sich um ein De-facto-Standardformat, das von ESRI entwickelt wurde. Solche Daten werden häufig als Formdaten bezeichnet. Die Erweiterung einer Datei mit Formdaten lautet ".shp".
- Eine Datei, die Geschäftsdaten für Standorte enthält, die durch Formdaten definiert sind. Die Erweiterung einer solchen Datei lautet ".dbf".
- Eine Datei, die einen Index für Formdaten enthält. Eine solche Datei hat die Erweiterung ".shx".
- Eine Datei, die eine Spezifikation des Koordinatensystems enthält, auf dem die Daten in einer SHP-Datei basieren. Die Erweiterung einer solchen Datei lautet ".prj".

Formdateien werden häufig zum Importieren von Daten verwendet, die aus Dateisystemen stammen, sowie zum Exportieren von Daten in Dateien, die sich in einem Dateisystem befinden.

Wenn Sie Formdaten mit DB2 Spatial Extender importieren, empfangen Sie mindestens eine .shp-Datei. In den meisten Fällen empfangen Sie außerdem eine oder mehrere Dateien der anderen Formdateitypen.

Formdaten in eine neue oder eine vorhandene Tabelle importieren

Sie können Formdaten in eine vorhandene Tabelle bzw. Sicht importieren. Sie können aber auch in einer einzigen Operation eine Tabelle erstellen und Formdaten in diese Tabelle importieren.

Vorbereitende Schritte

Damit Sie Formdaten in eine vorhandene Tabelle oder Sicht importieren können, muss Ihre Benutzer-ID die folgenden Berechtigungen bzw. Zugriffsrechte besitzen:

- Berechtigung DATAACCESS für die Datenbank, die die Tabelle bzw. Sicht enthält
- Zugriffsrecht CONTROL für die Tabelle bzw. Sicht
- Zugriffsrecht INSERT für die Tabelle bzw. Sicht
- Zugriffsrecht SELECT für die Tabelle bzw. Sicht (dieses Zugriffsrecht ist nur dann erforderlich, wenn die Tabelle eine ID-Spalte enthält, die keine Spalte IDENTITY ist)

Darüber hinaus benötigen Sie eines der folgenden Zugriffsrechte:

- Zugriffsrechte für die Verzeichnisse, in denen sich die Eingabedateien und die Fehlerdateien befinden
- Lesezugriffsrechte für die Eingabedateien und Schreibzugriffsrechte für die Fehlerdateien

Damit Sie automatisch eine Tabelle erstellen und Formdaten in die Tabelle importieren können, muss Ihre Benutzer-ID die folgenden Berechtigungen besitzen:

- Berechtigung DBADM für die Datenbank, die die Tabelle enthalten soll
- Berechtigung CREATETAB für die Datenbank, die die Tabelle enthalten soll
- Zugriffsrecht CREATEIN für das Schema, zu dem die Tabelle gehört (dieses Zugriffsrecht ist erforderlich, wenn das Schema bereits vorhanden ist)

- Berechtigung IMPLICIT_SCHEMA für die Datenbank, die die Tabelle enthält (dieses Zugriffsrecht ist erforderlich, wenn das für die Tabelle angegebene Schema nicht vorhanden ist)

Darüber hinaus benötigen Sie eines der folgenden Zugriffsrechte:

- Zugriffsrechte für die Verzeichnisse, in denen sich die Eingabedateien und die Fehlerdateien befinden
- Lesezugriffsrechte für die Eingabedateien und Schreibzugriffsrechte für die Fehlerdateien

Informationen zu diesem Vorgang

Wählen Sie eine der folgenden Methoden aus, um Formdaten zu importieren:

- Formdaten in eine räumliche Spalte einer vorhandenen Tabelle, eine vorhandene aktualisierbare Sicht oder eine vorhandene Sicht importieren, für die ein Trigger INSTEAD OF für INSERT-Operationen definiert ist.
- Eine Tabelle mit einer räumlichen Spalte automatisch erstellen und die Formdaten in diese Spalte importieren.

Vorgehensweise

Gehen Sie wie folgt vor, um Formdaten in eine neue oder vorhandene Tabelle zu importieren:

Wählen Sie die gewünschte Methode zum Importieren der Formdaten aus:

- Setzen Sie den Befehl db2se import_shape ab.
- Führen Sie eine Anwendung aus, die die Prozedur DB2GSE.ST_IMPORT_SHAPE aufruft.

Daten in eine Formdatei exportieren

Sie können räumliche Daten, die in Abfrageergebnissen zurückgegeben wurden, in eine Formdatei exportieren.

Vorbereitende Schritte

Damit Sie Daten in eine Formdatei exportieren können, muss Ihre Benutzer-ID die folgenden Zugriffsrechte besitzen:

- Zugriffsrecht für die Ausführung eines Subselects, der die zu exportierenden Ergebnisse zurückgibt
- Zugriffsrecht zum Schreiben in das Verzeichnis, in dem sich die Datei befindet, in die die Daten exportiert werden
- Zugriffsrecht zum Erstellen einer Datei für die exportierten Daten (dieses Zugriffsrecht ist erforderlich, wenn eine solche Datei noch nicht vorhanden ist)

Informationen dazu, wie diese Zugriffsrechte definiert sind und wie Sie diese Zugriffsrechte erhalten, können Sie bei Ihrem Datenbankadministrator erfragen.

Informationen zu diesem Vorgang

Die räumlichen Daten, die Sie in eine Formdatei exportieren, können aus Quellen wie z. B. einer Basistabelle, einem Join oder einer Union-Verknüpfung von mehreren Tabellen, Ergebnismengen, die beim Abfragen von Sichten zurückgegeben wurden, oder der Ausgabe einer räumlichen Funktion stammen.

Falls eine Datei, in die Daten exportiert werden sollen, vorhanden ist, kann DB2 Spatial Extender die Daten an diese Datei anhängen. Ist keine solche Datei vorhanden, können Sie von DB2 Spatial Extender eine Datei erstellen lassen.

Vorgehensweise

Gehen Sie wie folgt vor, um Daten in eine Formdatei zu exportieren:

Legen Sie eine Methode zum Exportieren von Daten in eine Formdatei fest:

- Setzen Sie den Befehl db2se export_shape über den Befehlszeilenprozessor db2se ab.
- Führen Sie eine Anwendung aus, die die Prozedur DB2GSE.ST_EXPORT_SHAPE aufruft.

Verwendungshinweise für einen Geocoder

Die Verwendung eines Geocoders umfasst das Definieren der Arbeit, die ein Geocoder ausführen soll, das Konfigurieren des Geocoders und das Ausführen des Geocoders im Stapelmodus.

Geocoder und Geocodierung

Die Termini Geocoder und Geocodierung werden in einem unterschiedlichen Zusammenhang verwendet. Diese unterschiedlichen Verwendungskontexte werden in den folgenden Erläuterungen gegeneinander abgegrenzt, damit die Bedeutung der Termini bei ihrem Auftreten jederzeit eindeutig erkennbar ist. Im Folgenden werden die Termini Geocoder und Geocodierung definiert und die Modi beschrieben, in denen ein Geocoder arbeitet. Außerdem wird die übergeordnete Aktivität beschrieben, zu der die Geocodierung gehört, und anschließend erhalten Sie einen Überblick über die Benutzertasks, die im Rahmen der Geocodierung ausgeführt werden müssen.

In DB2 Spatial Extender wird als Geocoder eine Skalarfunktion bezeichnet, die vorhandene Daten (Eingabe der Funktion) in Daten umsetzt, die zur Ausführung räumlicher Operationen verwendet werden können (Ausgabe der Funktion). In der Regel handelt es sich bei den vorhandenen Daten um relationale Daten, die einen Standort beschreiben oder benennen. DB2 Spatial Extender kann von Lieferanten bereitgestellte und benutzerdefinierte Geocoder unterstützen.

Ein von einem Lieferanten bereitgestellter Geocoder könnte Adressen in Koordinaten umsetzen, die DB2 nicht speichert, sondern in eine Datei schreibt. Ein anderer Geocoder könnte beispielsweise in der Lage sein, die Nummer eines Büros in einem Bürogebäude in Koordinaten umzusetzen, die die Position des Büros im Gebäude definieren. Denkbar wäre auch die Umsetzung der Kennung eines Regals in einem Kaufhaus in Koordinaten, die den Standort des Regals im Kaufhaus definieren.

In anderen Fällen können die von einem Geocoder umgesetzten vorhandenen Daten auch räumliche Daten sein. Ein Beispiel hierfür wäre ein benutzerdefinierter Geocoder, der X- und Y-Koordinaten in Daten umsetzt, die einem der Datentypen von DB2 Spatial Extender entsprechen.

In DB2 Spatial Extender wird als Geocodierung die Operation bezeichnet, mit der ein Geocoder seine Eingabe in Ausgabedaten (z. B. Adressen in Koordinaten) umsetzt.

Modi

Ein Geocoder kann in zwei verschiedenen Modi arbeiten:

- Im Stapelmodus versucht ein Geocoder, alle Eingabedaten aus einer Tabelle (oder alternativ alle Adressen in einer angegebenen Untermenge von Zeilen in der Tabelle) in einer einzigen Operation umzusetzen.
- Im automatischen Modus setzt ein Geocoder Daten um, sobald diese in eine Tabelle eingefügt oder dort aktualisiert werden. Der Geocoder wird durch INSERT- und UPDATE-Trigger definiert, die für die Tabelle definiert sind.

Geocodierungsprozesse

Die Geocodierung ist eine von mehreren Operationen, mit denen der Inhalt einer räumlichen Spalte einer DB2-Tabelle aus anderen Daten abgeleitet wird. In den vorliegenden Erläuterungen werden diese Operationen zusammenfassend als Geocodierungsprozess bezeichnet. Die Geocodierungsprozesse können je nach Geocoder variieren. Ein Geocoder kann nach Dateien mit bekannten Adressen suchen, um zu bestimmen, ob eine als Eingabe empfangene Adresse zu einem bestimmten Grad mit einer bekannten Adresse übereinstimmt. Da die bekannten Adressen Ähnlichkeit mit dem Material haben, das von Menschen bei Nachforschungen hinzugezogen wird, werden diese Adressen zusammenfassend als Bezugsdaten bezeichnet. Andere Geocoder benötigen unter Umständen keine Bezugsdaten und können ihre Eingabe auf anderem Weg prüfen.

Benutzertasks

In DB2 Spatial Extender ergeben sich für den Benutzer bei der Geocodierung die folgenden Tasks:

- Der Benutzer muss beschreiben, wie bestimmte Teile des Geocodierungsprozesses für eine angegebene räumliche Spalte ausgeführt werden sollen. Er muss beispielsweise den Mindestgrad für die Übereinstimmung von Straßennamen in Eingabesätzen mit Straßennamen in Bezugsdaten festlegen. Außerdem muss er den Mindestübereinstimmungsgrad für Adressen in Eingabesätzen und Adressen in Bezugsdaten definieren sowie festlegen, wie viele Datensätze vor jedem Commit verarbeitet werden sollen. Diese Tasks kann auch als Konfiguration der Geocodierung oder Konfiguration von Geocodierungsoperationen bezeichnet werden.
- Der Benutzer muss angeben, ob die Daten immer dann automatisch geocodiert werden sollen, wenn sie zu einer Tabelle hinzugefügt bzw. dort aktualisiert werden. Bei einer automatischen Geocodierung werden die Anweisungen wirksam, die der Benutzer bei der Konfiguration von Geocodierungsoperationen angegeben hat. Hiervon ausgenommen sind jedoch die Anweisungen, die Commits beinhalten - sie werden nur bei der Geocodierung im Stapelmodus angewendet. Diese Task wird als Konfiguration eines Geocoders für die automatische Ausführung bezeichnet.
- Der Geocoder wird im Stapelmodus ausgeführt. Wenn der Benutzer bereits Geocodierungsoperationen definiert hat, bleiben seine Anweisungen während allen Stapelsitzungen wirksam, sofern er sie nicht außer Kraft setzt. Falls der Benutzer vor einer bestimmten Sitzung noch keine Geocodierungsoperationen definiert hat, kann er angeben, dass sie für diese bestimmte Sitzung gelten sollen, und die Operationen definieren. Diese Task wird als Ausführung eines Geocoders im Stapelmodus und Ausführung der Geocodierung im Stapelmodus bezeichnet.

Geocodierungsoperationen definieren

Mit DB2 Spatial Extender können Sie vorab die Aufgaben definieren, die beim Aufrufen eines Geocoders ausgeführt werden müssen.

Vorbereitende Schritte

Damit Sie Geocodierungsoperationen für einen bestimmten Geocoder definieren können, muss Ihre Benutzer-ID die folgenden Berechtigungen bzw. Zugriffsrechte besitzen:

- Berechtigung DATAACCESS für die Datenbank, die die Tabelle enthält, für die der angegebene Geocoder ausgeführt werden soll
- Zugriffsrecht CONTROL für jede Tabelle, für die der Geocoder ausgeführt werden soll
- Zugriffsrecht SELECT und Zugriffsrecht UPDATE für jede Tabelle, für die der Geocoder ausgeführt werden soll

Informationen zu diesem Vorgang

Beim Aufruf eines Geocoders können Sie die folgenden Parameter angeben:

- Die Spalte, für die der Geocoder Daten bereitstellen soll.
- Die Angabe, ob die durch den Geocoder (aus einer Tabelle oder Sicht) gelesene Eingabe auf eine Untermenge von Zeilen in der Tabelle oder Sicht begrenzt sein soll.
- Der Bereich oder die Anzahl der Datensätze, die der Geocoder in Stapelsitzungen in einer UOW (Unit of Work, Arbeitseinheit) geocodieren soll.
- Voraussetzungen für geocoderspezifische Operationen. Beispiel: Ein Geocoder kann nur solche Datensätze geocodieren, die mit ihren Gegenstücken in den Bezugsdaten zu einem angegebenen (oder einem höheren) Grad übereinstimmen. Dieser Grad wird als Mindestübereinstimmungsquote bezeichnet.

Sie müssen die in obiger Liste aufgeführten Parameter angeben, bevor Sie den Geocoder für eine Ausführung im automatischen Modus konfigurieren. Anschließend werden die Geocodierungsoperationen bei jedem Aufruf des Geocoders (nicht nur bei automatischen Verarbeitungen, sondern auch bei Ausführungen im Stapelbetrieb) gemäß Ihren Spezifikationen ausgeführt. Wenn Sie beispielsweise angeben, dass im Stapelmodus in jeder UOW 45 Datensätze geocodiert werden sollen, wird nach jedem 45. geocodierten Datensatz ein Commit durchgeführt. (Ausnahme: Sie können Ihre Angaben für einzelne Sitzungen der Geocodierung im Stapelbetrieb außer Kraft setzen.)

Sie müssen keine Standardwerte für Geocodierungsoperationen angeben, bevor Sie den Geocoder im Stapelbetrieb ausführen. Stattdessen können Sie beim Starten einer Stapelsitzung angeben, wie die Operationen für die Dauer der Ausführung ausgeführt werden sollen. Wenn Sie Standardwerte für Stapelsitzungen angeben, können Sie sie je nach Bedarf bei einzelnen Sitzungen außer Kraft setzen.

Vorgehensweise

Gehen Sie wie folgt vor, um Geocodierungsoperationen zu definieren:

Legen Sie die gewünschte Vorgehensweise für das Definieren der Geocodierungsoperationen fest:

- Setzen Sie den Befehl db2se setup_gc ab.

- Führen Sie eine Anwendung aus, die die Prozedur DB2GSE.ST_SETUP_GEOCODING aufruft.

Nächste Schritte

Empfehlungen: Wenn ein Geocoder einen Datensatz mit Adressendaten liest, versucht er, diesen Datensatz mit einem entsprechenden Element in den Bezugsdaten abzugleichen. Grob skizziert geht er hierbei folgendermaßen vor: Zunächst werden die Bezugsdaten nach Straßen durchsucht, deren Postleitzahl mit der Postleitzahl im Datensatz identisch ist. Sobald ein Straßenname gefunden wird, der zu einem gewissen Mindestprozentsatz (oder zu einem höheren Grad) Ähnlichkeit mit dem Namen im Datensatz hat, wird nach einer vollständigen Adresse gesucht. Falls eine vollständige Adresse gefunden wird, die zu einem gewissen Mindestprozentsatz (oder zu einem höheren Grad) Ähnlichkeit mit der Adresse im Datensatz hat, wird der Datensatz geocodiert. Wird keine solche Adresse gefunden, gibt der Geocoder einen Nullwert zurück.

Der Mindestprozentsatz, zu dem Straßennamen übereinstimmen müssen, wird als Schreibweisengenauigkeit bezeichnet. Der Mindestprozentsatz für die Übereinstimmung der gesamten Adresse wird Mindestübereinstimmungsquote genannt. Wenn die Schreibweisengenauigkeit beispielsweise 80 beträgt, muss die Übereinstimmung der Straßennamen mindestens 80 Prozent betragen, damit der Geocoder nach der vollständigen Adresse sucht. Liegt die Mindestübereinstimmungsquote bei 60, muss die Übereinstimmung zwischen den Adressen bei mindestens 60 Prozent liegen, damit der Geocoder den Datensatz geocodiert.

Sie können angeben, wie hoch die Schreibweisengenauigkeit und die Mindestübereinstimmungsquote sein sollen. Bitte beachten Sie, dass Sie diese Werte unter Umständen anpassen müssen. Angenommen, Sie haben für Schreibweisengenauigkeit und die Mindestübereinstimmungsquote jeweils einen Wert von 95 definiert. Wenn die Adressen, die geocodiert werden sollen, nicht sorgfältig ausgewertet wurden, sind Übereinstimmungen von 95 Prozent ziemlich unwahrscheinlich. Infolgedessen gibt der Geocoder bei der Verarbeitung dieser Datensätze wahrscheinlich einen Nullwert zurück. In einem solchen Fall ist es ratsam, die Schreibweisengenauigkeit und die Mindestübereinstimmungsquote herabzusetzen und den Geocoder erneut auszuführen. Empfohlene Prozentsätze für die Schreibweisengenauigkeit und die Mindestübereinstimmungsquote sind 70 bzw. 60 Prozent.

Wie bereits am Anfang dieses Abschnitts erwähnt, können Sie angeben, ob die Eingabe, die der Geocoder aus einer Tabelle oder Sicht liest, auf eine Untermenge von Zeilen in der Tabelle oder Sicht begrenzt sein soll. Denkbar wären beispielsweise die folgenden Szenarios:

- Sie rufen den Geocoder auf, um Adressen in einer Tabelle im Stapelmodus zu geocodieren. Die Mindestübereinstimmungsquote wurde jedoch zu hoch festgelegt, sodass der Geocoder bei der Verarbeitung der meisten Adressen einen Nullwert zurückgibt. Bei der erneuten Ausführung des Geocoders reduzieren Sie die Mindestübereinstimmungsquote. Um die Eingabe auf solche Adressen zu beschränken, die nicht geocodiert wurden, können Sie angeben, dass nur die Zeilen ausgewählt werden sollen, die den zuvor zurückgegebenen Nullwert enthalten.
- Der Geocoder wählt nur Zeilen aus, die nach einem bestimmten Datum hinzugefügt wurden.
- Der Geocoder wählt nur Zeilen aus, die Adressen in einem bestimmten Gebiet (beispielsweise einer Gruppe von Landkreisen oder einem Bundesland) enthalten.

Am Anfang dieses Abschnitts wurde bereits angegeben, dass Sie die Anzahl der Datensätze definieren können, die der Geocoder in Stapelsitzungen in einer UOW geocodieren soll. Sie können vom Geocoder in jeder UOW die gleiche Anzahl von Datensätzen verarbeiten lassen oder in einer einzigen UOW alle Datensätze einer Tabelle verarbeiten lassen. Wenn Sie sich für die zweite Alternative entscheiden, müssen Sie Folgendes beachten:

- Sie haben weniger Steuerungsmöglichkeiten hinsichtlich der Größe der UOW als bei der ersten Alternative. Infolgedessen können Sie bei der Ausführung des Geocoders nicht steuern, wie viele Sperren gehalten werden oder wie viele Protokolleinträge beim Betrieb des Geocoders erstellt werden.
- Falls der Geocoder einen Fehler feststellt, der einen Rollback erforderlich macht, müssen Sie den Geocoder erneut für alle Datensätze ausführen. Dies kann einen erheblichen Ressourcenaufwand verursachen, wenn die Tabelle sehr groß ist und der Fehler sowie der Rollback auftreten, nachdem die meisten Datensätze bereits verarbeitet wurden.

Geocoder für automatische Ausführung definieren

Sie können einen Geocoder so definieren, dass Daten automatisch umgesetzt werden, sobald sie in einer Tabelle hinzugefügt oder aktualisiert werden.

Vorbereitende Schritte

Vor dem Definieren eines Geocoders für die automatische Ausführung müssen Sie die folgenden Punkte beachten:

- Sie müssen Geocodierungsoperationen für alle räumlichen Spalten definieren, die mit Ausgabedaten des Geocoders gefüllt werden sollen.
- Ihre Benutzer-ID muss die folgenden Berechtigungen oder Zugriffsrechte besitzen:
  - Berechtigungen DBADM und DATAACCESS für die Datenbank, die die Tabelle enthält, für die Trigger zum Aufrufen des Geocoders definiert werden sollen
  - Zugriffsrecht CONTROL
  - Zugriffsrechte ALTER, SELECT und UPDATE
  - Erforderliche Zugriffsrechte zum Erstellen von Triggern für diese Tabelle

Informationen zu diesem Vorgang

Sie können einen Geocoder für die automatische Ausführung definieren, bevor Sie ihn im Stapelmodus aufrufen. Es ist deshalb möglich, dass eine automatische Geocodierung einer Geocodierung im Stapelbetrieb vorausgeht. In diesem Fall verarbeitet die Geocodierung im Stapelbetrieb wahrscheinlich dieselben Daten, die automatisch verarbeitet wurden. Diese Redundanz führt nicht dazu, dass Daten anschließend doppelt vorhanden sind. Wenn räumliche Daten zwei Mal generiert werden, überschreibt das zweite Datenergebnis das erste. Die Systemleistung kann dadurch jedoch beeinträchtigt werden.

Bevor Sie festlegen, ob die Adressendaten einer Tabelle im Stapelmodus oder im automatischen Modus geocodiert werden sollen, sollten Sie die folgenden Aspekte bedenken:

- Die Leistung ist bei der Geocodierung im Stapelbetrieb besser als bei der automatischen Geocodierung. Eine Stapelsitzung wird mit einer Initialisierung geöffnet und mit einer Bereinigung beendet. Bei der automatischen Geocodierung

wird jedes Datenelement in einer separaten Operation geocodiert, die jeweils mit einer Initialisierung beginnt und einer Bereinigung abgeschlossen wird.

- Generell ist eine räumliche Spalte, die durch eine automatische Geocodierung gefüllt wird, wahrscheinlich auf einem aktuelleren Stand als eine räumliche Spalte, die durch die Geocodierung im Stapelbetrieb gefüllt wird. Nach einer Stapelsitzung können sich Adressendaten ansammeln, die bis zu nächsten Sitzung nicht geocodiert werden. Ist jedoch die automatische Geocodierung bereits aktiviert, werden die Adressendaten geocodiert, sobald sie in der Datenbank gespeichert werden.

Vorgehensweise

Gehen Sie wie folgt vor, um einen Geocoder zur automatischen Ausführung einzurichten:

Wählen Sie die gewünschte Methode zum Definieren der automatischen Geocodierung aus:

- Setzen Sie den Befehl db2se enable_autogc ab.
- Führen Sie eine Anwendung aus, die die Prozedur DB2GSE.ST_ENABLE_AUTOGEOCODING aufruft.

Geocoder im Stapelmodus ausführen

Wenn Sie einen Geocoder im Stapelmodus ausführen, setzen Sie mehrere Datensätze in räumliche Daten um, die in einer bestimmten Spalte gespeichert werden.

Vorbereitende Schritte

Damit Sie einen Geocoder im Stapelmodus ausführen können, muss Ihre Benutzer-ID eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte besitzen:

- Berechtigung DATAACCESS für die Datenbank, die die Tabelle enthält, deren Daten geocodiert werden sollen.
- Zugriffsrecht CONTROL für diese Tabelle
- Zugriffsrecht UPDATE für diese Tabelle

Außerdem benötigen Sie für diese Tabelle das Zugriffsrecht SELECT, damit Sie die Anzahl der Datensätze angeben können, die vor jedem Commit verarbeitet werden sollen. Falls Sie Klauseln WHERE angeben, um die Zeilen zu begrenzen, für die der Geocoder ausgeführt wird, benötigen Sie unter Umständen auch das Zugriffsrecht SELECT für alle Tabellen und Sichten, auf die in diesen Klauseln verwiesen wird. Bitte wenden Sie sich diesbezüglich an Ihren Datenbankadministrator.

Informationen zu diesem Vorgang

Jedes Mal, bevor Sie einen Geocoder ausführen, um eine bestimmte räumliche Spalte zu füllen, können Sie Geocodierungsoperationen für diese Spalte definieren. Zur Konfiguration der Operationen gehört die Angabe, inwieweit bestimmte Voraussetzungen erfüllt werden müssen, wenn der Geocoder ausgeführt wird. Beispiel: Angenommen, DB2 Spatial Extender soll immer dann ein Commit veranlassen, nachdem 100 Eingabedatensätze durch den Geocoder verarbeitet wurden. Bei der Konfiguration der Operationen würden Sie den Wert 100 als erforderliche Anzahl definieren.

Nachdem Sie die Ausführung des Geocoders vorbereitet haben, können Sie jeden Wert, den Sie in diesem Zusammenhang definiert haben, außer Kraft setzen. Dies bleibt dann nur für die Dauer der Ausführung gültig.

Falls Sie keine Operationen definieren, müssen Sie immer dann, wenn Sie den Geocoder ausführen wollen, angeben, welche Voraussetzungen während der Ausführung erfüllt werden müssen.

Vorgehensweise

Gehen Sie wie folgt vor, um einen Geocoder im Stapelmodus auszuführen:

Legen Sie fest, wie ein Geocoder für die Ausführung im Stapelmodus aufgerufen werden soll:

- Setzen Sie den Befehl db2se run_gc ab.
- Führen Sie eine Anwendung aus, die die Prozedur DB2GSE.ST_RUN_GEOCODING aufruft.

Kapitel 10. DB2 Spatial Extender in Umgebung mit partitionierten Datenbanken

DB2 Spatial Extender kann effektiv in einer Umgebung mit partitionierten Datenbanken eingesetzt werden, um eine Analyse räumlicher Daten für große Kundendatentabellen auszuführen. Umgebungen mit partitionierten Datenbanken eignen sich zur Ausführung von Data-Warehousing-Anwendungen, die mit IBM InfoSphere Warehouse erstellt werden können.

Umgebungen mit partitionierten Datenbanken unterstützen eine leistungsfähige und hoch skalierbare Datenbankverarbeitung durch eine knotenübergreifende Parallelverarbeitung. Jeder Knoten verfügt über einen eigenen Prozessor, einen eigenen Hauptspeicher und einen eigenen Plattenspeicher.

In einer Umgebung mit partitionierten Datenbanken stehen Ihnen drei Möglichkeiten zum Speichern von DB2-Tabellen mit räumlichen Spalten zur Verfügung. Sie können sie auf einem einzelnen Knoten speichern, auf mehrere Knoten verteilen oder auf mehrere Knoten replizieren. Räumliche Indizes werden für jede dieser Möglichkeiten unterstützt. Welches Verfahren Sie wählen, ist abhängig von der Größe der Tabellen und von deren Verwendung in Abfragen räumlicher Daten. Weitere Informationen zur Partitionierung von Datenbanken finden Sie in „Umgebungen mit partitionierten Datenbanken" in *Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen*.

Räumliche Daten in einer Umgebung mit partitionierten Datenbanken erstellen und laden

Für die partitionierte Tabelle muss ein Partitionierungsschlüssel definiert sein, der dazu verwendet wird, die Verteilung von Zeilen auf die Partitionen zu steuern.

Vorbereitende Schritte

Stellen Sie sicher, dass die Datenbank für die Verarbeitung räumlicher Daten aktiviert ist, indem Sie räumliche Katalogtabellen sowie die räumlichen Typen und Funktionen in der Datenbank erstellen. Weitere Informationen hierzu finden Sie in „Datenbank für räumliche Operationen aktivieren" auf Seite 32.

Informationen zu diesem Vorgang

Eine optimale Leistung erzielen Sie, wenn Sie sicherstellen, dass die Tabellen, die räumliche Spalten enthalten, gleichmäßig auf die Partitionen verteilt sind. Dies erreichen Sie mithilfe des Standardhashalgorithmus und der Standardpartitionszuordnung.

Vorgehensweise

Die besten Ergebnisse erhalten Sie, indem Sie eine oder mehrere Spalten in der Anweisung CREATE TABLE als Partitionierungsschlüssel angeben. Wenn Sie keinen Partitionierungsschlüssel angeben, wählt DB2 die erste numerische Spalte bzw. die erste Zeichendatenspalte standardmäßig als Partitionierungsschlüssel aus. Mit dieser Standardeinstellung werden die Zeilen möglicherweise nicht gleichmäßig auf die Partitionen verteilt.

Beispiel

Das folgende Beispiel zeigt die Erstellung einer Tabelle, die räumliche Daten enthalten soll, und die Verwendung des Importdienstprogramms von DB2 Spatial Extender für die Angabe des Partitionierungsschlüssels.

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

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

Abfrageleistung für räumliche Daten in einer partitionierten Umgebung verbessern

Um eine optimale Leistung bei Abfragen in einer Umgebung mit partitionierten Datenbanken zu erzielen, müssen die Tabellenzeilen, die für die Erfüllung der Joinbedingung benötigt werden, benachbart angeordnet sein. Das heißt, diese Joins müssen Tabellenzeilen verwenden, die sich auf einem Knoten befinden, ohne auf Zeilen aus Tabellen bzw. Tabellenteilen zugreifen zu müssen, die sich auf einem anderen Knoten befinden.

Informationen zu diesem Vorgang

Räumliche Joins werden häufig in Anwendungen verwendet, die nach Kunden in bestimmten Polygonen suchen oder das Überschwemmungsrisiko von Kunden ermitteln. Im Folgenden ist ein Beispiel für eine Abfrage zur Ermittlung des Überschwemmungsrisikos dargestellt:

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

In diesem Beispiel ist die Tabelle `customers` relativ groß und auf mehrere Partitionen verteilt. Die Polygoninformationen zu Überschwemmungen haben nur einen geringen Umfang. Für eine effiziente Implementierung der Abfrage müssen Sie sicherstellen, dass die gesamte Überschwemmungspolygontabelle in jeder Partition verfügbar ist, die Kundendaten enthält.

Vorgehensweise

1. Importieren Sie die Polygondaten in eine Einzelpartition.
2. Erstellen Sie eine MQT (Materialized Query Table, gespeicherte Abfragetabelle), die in allen Partitionen repliziert wird, die die Kundendaten enthalten. Nach

dem Erstellen und Laden der Tabelle `floodpoly` können Sie beispielsweise die replizierte MQT mit einer Anweisung wie der folgenden erstellen:

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

REFRESH TABLE floodpoly;
```

Die replizierte MQT mit einer räumlichen Spalte kann benutzerverwaltet oder systemverwaltet sein, sie muss jedoch die Option REFRESH DEFERRED verwenden, nicht die Option REFRESH IMMEDIATE.

3. Damit der DB2-Abfragecompiler die MQT anstelle der Basistabelle verwendet, können Sie die folgenden DB2-Parameter festlegen:

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

Weitere Informationen zu diesen Parametern finden Sie in „Befehl UPDATE DATABASE CONFIGURATION" in *Command Reference*.

Nächste Schritte

Verwenden Sie die DB2-Explain-Tools, um zu ermitteln, wie effizient die Ausführung der Abfrage sein wird.

Kapitel 11. Indizes und Sichten für den Zugriff auf räumliche Daten verwenden

Verwenden Sie Indizes und Sichten für den Zugriff auf räumliche Spalten.

Bevor Sie räumliche Spalten abfragen, sollten Sie sich mit den Einzelheiten der Erstellung von Indizes und Sichten für den Zugriff auf räumliche Spalten auseinandersetzen. Zur Erstellung solcher Indizes müssen Sie die Eigenschaften der Indizes kennen, die Spatial Extender verwendet, um den Zugriff auf räumliche Spalten zu beschleunigen.

Räumliche Rasterindizes

Indizes können die Abfrageleistung von Anwendungen erheblich verbessern. Dies gilt insbesondere dann, wenn die abgefragten Tabellen eine große Anzahl von Zeilen enthalten. Wenn Sie entsprechende Indizes erstellen, die dann vom Abfrageoptimierungsprogramm zur Ausführung von Abfragen verwendet werden können, lässt sich dadurch die Anzahl der zu verarbeitenden Zeilen erheblich reduzieren.

DB2 Spatial Extender stellt einen Rasterindex zur Verfügung, der sich optimal zur Verarbeitung zweidimensionaler Daten eignet. Dieser Index wird auf den X- und Y-Dimensionen einer Geometrie erstellt.

Sie sollten sich mit den folgenden Aspekten eines Rasterindexes vertraut machen:

- Generieren des Indexes
- Verwendung räumlicher Funktionen in einer Abfrage
- Verwendung des räumlichen Rasterindexes durch eine Abfrage

Räumliche Rasterindizes generieren

DB2 Spatial Extender verwendet zum Generieren eines räumlichen Rasterindexes das minimal einschließende Rechteck (MBR) einer Geometrie.

Bei den meisten Geometrien ist das MBR ein Rechteck, das die Geometrie umgibt.

Ein räumlicher Rasterindex unterteilt einen Bereich in logische quadratische Rasterelemente mit einer festen Größe, die bei der Erstellung des Indexes angegeben wird. Der räumliche Index wird in einer räumlichen Spalte konstruiert; hierzu werden ein oder mehrere Einträge für die Schnittmengen der MBRs der einzelnen Geometrien mit den Rasterzellen vorgenommen. Ein Indexeintrag besteht aus der Rasterzellenkennung, dem MBR der Geometrie und der internen Kennung der Zeile, die die Geometrie enthält.

Sie können bis zu drei Ebenen von räumlichen Indizes (Rasterebenen) definieren. Die Verwendung mehrerer Rasterebenen ist insofern hilfreich, als sie eine Optimierung des Indexes für unterschiedliche Größen von räumlichen Daten ermöglicht.

Falls eine Geometrie vier oder mehr Rasterzellen schneidet, wird sie auf die nächsthöhere Stufe hochgestuft. Im Allgemeinen werden die größeren Geometrien auf einer höheren Stufe indexiert. Wenn eine Geometrie zehn oder mehr Rasterzellen der höchsten Rastergröße schneidet, wird eine integrierte Überlaufindexstufe verwendet. Die Überlaufstufe verhindert die Generierung überzähliger Indexeinträ-

ge. Zur Erzielung der optimalen Systemleistung sollten Sie die verwendeten Rastergrößen so festlegen, dass die Verwendung der Überlaufstufe vermieden wird.

Beispiel: Sind mehrere Rasterebenen vorhanden, versucht der Indexierungsalgorithmus, eine möglichst niedrige Rasterebene zu verwenden, um eine größtmögliche Auflösung der indexierten Daten zu erzielen. Wenn eine Geometrie mehr als vier Rasterzellen auf einer bestimmten Ebene geschnitten hat, wird sie auf die nächsthöhere Ebene hochgestuft (unter der Voraussetzung, dass eine weitere Ebene vorhanden ist). Daher schneidet ein räumlicher Index, der die drei Rasterebenen 10,0, 100,0 und 1000,0 aufweist, zunächst jede Geometrie mit dem Raster der Ebene 10,0. Wenn eine Geometrie mehr als vier Rasterzellen der Größe 10,0 geschnitten hat, wird sie hochgestuft und schneidet das Raster der Ebene 100,0. Entstehen auf der Ebene 100,0 mehr als vier Schnittmengen, wird die Geometrie auf die Ebene 1000,0 hochgestuft. Entstehen auf der Ebene 1000,0 mehr als 10 Schnittmengen, wird die Geometrie in der Überlaufebene indexiert.

Verwendung räumlicher Funktionen in einer Abfrage

Bei Verwendung räumlicher Funktionen in der Klausel WHERE berücksichtigt das DB2-Optimierungsprogramm räumliche Rasterindizes für den Zugriffsplan.

Die räumlichen Funktionen, die diesen Effekt auf das DB2-Optimierungsprogramm haben, sind im Einzelnen:

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

Verwendung des räumlichen Rasterindexes durch eine Abfrage

Wenn das Abfrageoptimierungsprogramm einen räumlichen Rasterindex auswählt, wird bei der Ausführung der Abfrage ein aus mehreren Schritten bestehender Filterprozess verwendet.

Der Filterprozess umfasst die folgenden Schritte:

1. Feststellen, welche Rasterzellen Überschneidungen mit dem Abfragefenster aufweisen. Das *Abfragefenster* ist die Geometrie, an der Sie interessiert sind und die als zweiter Parameter in einer räumlichen Funktion (siehe folgende Beispiele) angegeben wird.
2. Den Index nach Einträgen durchsuchen, die übereinstimmende Rasterzellenkennungen aufweisen.
3. Die MBR-Werte der Geometrie in den Indexeinträgen mit dem Abfragefenster vergleichen und alle Werte löschen, die sich außerhalb des Abfragefensters befinden.

4. Bei Bedarf weitere Analyseschritte durchführen. Die in den vorherigen Schritten ermittelten potenziellen Geometrien können weiter analysiert werden, um festzustellen, ob sie die Anforderungen für die räumlichen Funktionen (ST_Contains, ST_Distance, etc.) erfüllen. Bei der räumlichen Funktion EnvelopesIntersect wird dieser Schritt nicht ausgeführt. Sie erzielt normalerweise die beste Systemleistung.

In den folgenden Beispielen räumlicher Abfragen ist in der Spalte C.GEOMETRY ein räumlicher Rasterindex definiert:

```
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
```

Im ersten Beispiel definieren die vier Koordinatenwerte das Abfragefenster. Diese Koordinatenwerte geben die untere linke sowie die obere rechte Ecke (42,0 -73,0 und 43,0 -72,0) eines Rechtecks an.

Im zweiten Beispiel berechnet DB2 Spatial Extender den MBR der Geometrie, die in der Hostvariablen :geometry2 angegeben wurde, und verwendet diesen als Abfragefenster.

Beim Erstellen eines räumlichen Rasterindexes sollten Sie geeignete Rastergrößen für die Abfragefenstergrößen angeben, die in Ihrer räumlichen Anwendung am häufigsten verwendet werden. Wenn die Rastergröße höher ist, müssen Indexeinträge für Geometrien, die außerhalb des Abfragefensters liegen, durchsucht werden, da sich diese in Rasterzellen befinden, die das Abfragefenster überschneiden. Durch diese zusätzlichen Suchvorgänge wird die Systemleistung negativ beeinflusst. Eine geringere Rastergröße führt hingegen zur Generierung einer höheren Zahl von Indexeinträgen für die einzelnen Geometrien, wodurch dann auch eine größere Anzahl von Indexeinträgen durchsucht werden muss. Dies wirkt sich ebenfalls negativ auf die Abfrageleistung des Systems aus.

DB2 Spatial Extender stellt das Dienstprogramm Indexadvisor zur Verfügung, mit dem die Daten räumlicher Spalten analysiert werden und Vorschläge zu geeigneten Rastergrößen für häufig verwendete Abfragefenstergrößen erstellt werden können.

Überlegungen zur Anzahl der Indexstufen und Rastergrößen

Verwenden Sie den Indexadvisor zur Bestimmung geeigneter Rastergrößen für Ihre räumlichen Rasterindizes, da dies die beste Methode zur Optimierung der Indizes ist und für höchste Effizienz Ihrer räumlichen Abfragen sorgt.

Anzahl der Rasterebenen

Möglich sind bis zu drei Rasterebenen.

Für jede Rasterebene eines räumlichen Rasterindexes wird bei einer räumlichen Abfrage eine separate Indexsuche ausgeführt. Daher wird die Abfrage umso effizienter, je weniger Rasterebenen vorhanden sind.

Wenn die Werte in der räumlichen Spalte ungefähr dieselbe relative Größe besitzen, sollten Sie nur eine Rasterebene verwenden. Allerdings enthält eine typische räumliche Spalte keine Geometrien derselben relativen Größe, die Geometrien in

einer räumlichen Spalte können jedoch in der Regel nach Größe gruppiert werden. Dann können Sie die Rasterebenen auf diese Geometriengruppen abstimmen.

Angenommen, eine Tabelle enthält beispielsweise die Landparzellen eines Landkreises mit einer räumlichen Spalte, in der kleine städtische Parzellen, die von größeren ländlichen Parzellen umgeben sind, in Gruppen zusammengefasst sind. Da die Größen der Parzellen in zwei Gruppen (kleine städtische und größere ländliche) zusammengefasst werden können, könnten Sie in diesem Fall zwei Rasterebenen für den räumlichen Rasterindex angeben.

Größe von Rasterzellen

Generell gilt, dass die Rastergrößen so weit wie möglich herabgesetzt werden sollten, um die höchste Auflösung und gleichzeitig eine Minimierung der Anzahl von Indexeinträgen zu erreichen.

- Für die feinste Rastergröße sollte ein kleiner Wert verwendet werden, um den Gesamtindex für kleine Geometrien in der Spalte zu optimieren. Auf diese Weise wird der Systemaufwand vermieden, der durch die Auswertung von Geometrien entsteht, die sich nicht im Suchbereich befinden. Die feinste Rastergröße erzeugt allerdings auch die größte Anzahl von Indexeinträgen. Infolgedessen steigt die Anzahl der Indexeinträge, die bei Abfragen verarbeitet werden, ebenso wie der für den Index benötigte Speicher. Diese Faktoren verringern die Gesamtleistung.
- Durch eine Verwendung von größeren Rastergrößen kann der Index für größere Geometrien optimiert werden. Die größeren Rastergrößen erzeugen weniger Indexeinträge für große Geometrien, als dies bei den feinsten Rastergrößen der Fall ist. Daher ist der für den Index erforderliche Speicher geringer, und die Gesamtleistung wird verbessert.

Die folgenden Abbildungen zeigen die Auswirkungen unterschiedlicher Rastergrößen.

Abb. 13 auf Seite 81 zeigt eine Karte mit Landparzellen, wobei jede Parzelle durch eine Polygongeometrie dargestellt wird. Das schwarze Rechteck stellt das Abfragefenster dar. Sie wollen nun alle Geometrien ermitteln, deren minimal einschließendes Rechteck (MBR = Minimum Bounding Rectangle) eine Überschneidung mit dem Abfragefenster aufweist. Abb. 13 auf Seite 81 zeigt, dass 28 (in Rosa hervorgehobene) Geometrien einen MBR aufweisen, der sich mit dem Abfragefenster überschneidet.

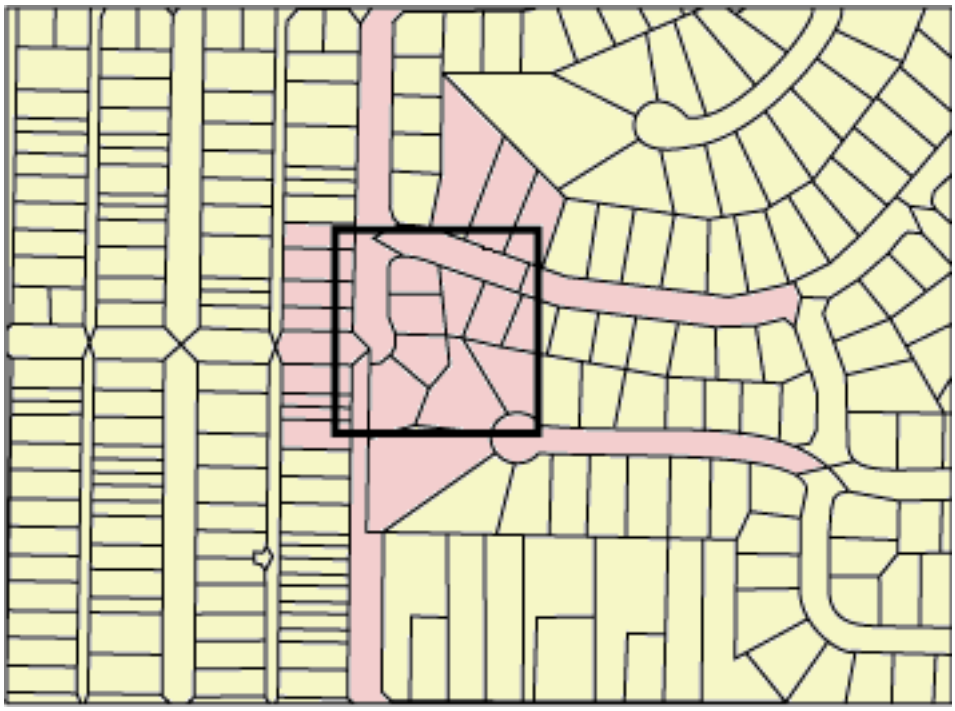


Abbildung 13. Benachbarte Landparzellen

Abb. 14 auf Seite 82 zeigt eine kleine Rastergröße (25), die eine gute Angleichung an das Abfragefenster erzielt.

- Die Abfrage gibt nur die 28 Geometrien aus, die hervorgehoben sind. Während der Abfrage müssen jedoch drei zusätzliche Geometrien überprüft und dann verworfen werden, deren MBR eine Überschneidung mit dem Abfragefenster aufweist.
- Diese niedrige Rastergröße ergibt zahlreiche Indexeinträge pro Geometrie. Während der Ausführung greift die Abfrage auf alle Indexeinträge für diese 31 Geometrien zu. Abb. 14 auf Seite 82 zeigt 256 Rasterzellen, die das Abfragefenster überlagern. Während der Ausführung der Abfrage wird jedoch auf 578 Indexeinträge zugegriffen, da zahlreiche Geometrien mit denselben Rasterzellen indexiert werden.

Beim vorliegenden Abfragefenster resultiert die geringe Rastergröße in einer übermäßig hohen Anzahl von zu überprüfenden Indexeinträgen.

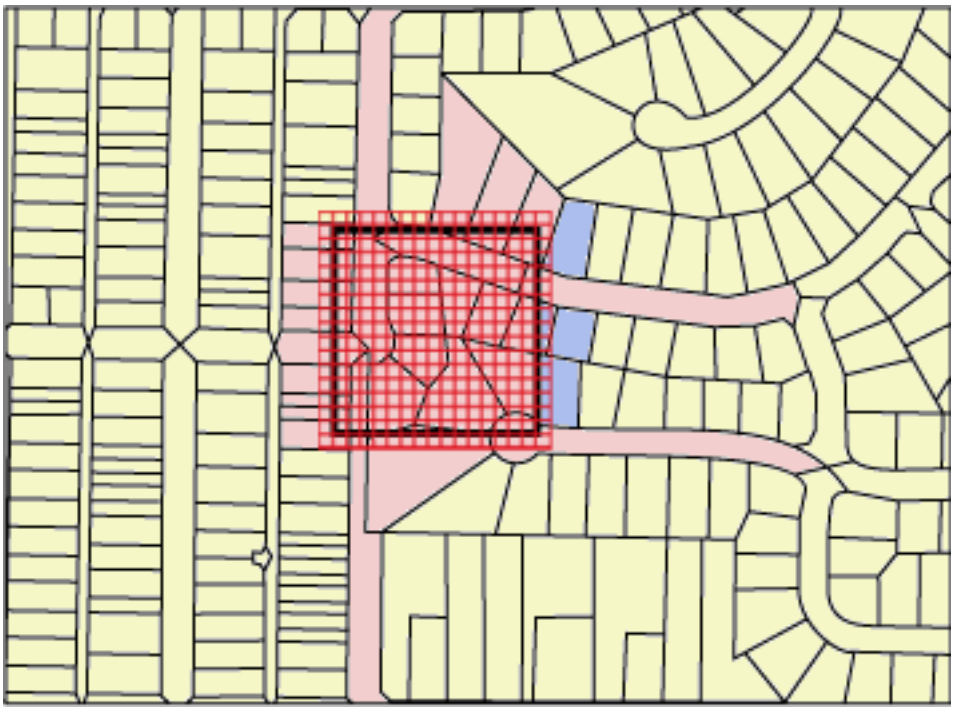


Abbildung 14. Geringe Rastergröße (25) auf Landparzellen

Abb. 15 auf Seite 83 zeigt eine hohe Rastergröße (400), die eine erheblich größere Fläche mit deutlich mehr Geometrien abdeckt, als im Abfragefenster enthalten sind.

- Diese hohe Rastergröße resultiert in nur einem Indexeintrag pro Geometrie, während der Abfrage müssen jedoch 59 zusätzliche Geometrien überprüft und verworfen werden, deren MBR eine Überschneidung mit der Rasterzelle aufweist.
- Während der Ausführung greift die Abfrage auf alle Indexeinträge der 28 Geometrien zu, die eine Überschneidung mit dem Abfragefenster aufweisen. Darüber hinaus wird auf die Indexeinträge der 59 zusätzlichen Geometrien zugegriffen. Dies ergibt 112 Indexeinträge.

Im vorliegenden Abfragefenster resultiert die hohe Rastergröße in einer übermäßig hohen Anzahl von zu überprüfenden Geometrien.

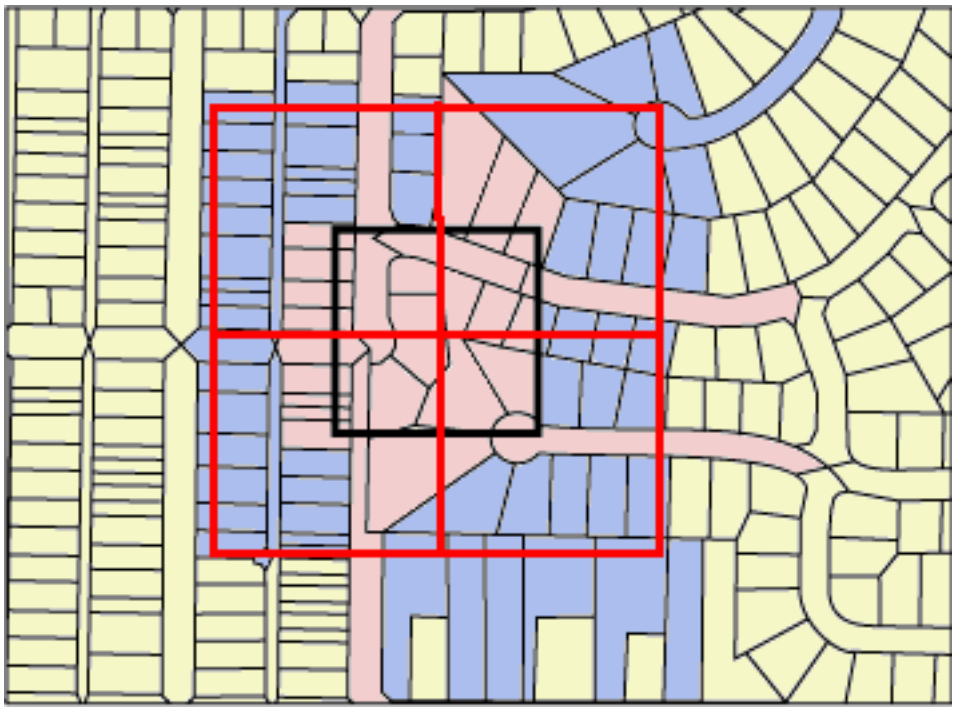


Abbildung 15. Hohe Rastergröße (400) auf Landparzellen

Abb. 16 auf Seite 84 zeigt eine mittlere Rastergröße (100), die eine gute Angleichung an das Abfragefenster erzielt.

- Die Abfrage gibt nur die 28 Geometrien zurück, die hervorgehoben sind. Während der Abfrage müssen jedoch fünf zusätzliche Geometrien überprüft und dann verworfen werden, deren MBR eine Überschneidung mit dem Abfragefenster aufweist.
- Während der Ausführung greift die Abfrage auf alle Indexeinträge der 28 Geometrien zu, die eine Überschneidung mit dem Abfragefenster aufweisen. Darüber hinaus wird auf die Indexeinträge der 5 zusätzlichen Geometrien zugegriffen. Dies ergibt 91 Indexeinträge.

Im vorliegenden Abfragefenster ist die mittlere Rastergröße am besten geeignet, da diese in deutlich weniger Indexeinträgen resultiert als die niedrige Rastergröße. Darüber hinaus müssen während der Abfrage weniger zusätzliche Geometrien überprüft werden als bei der hohen Rastergröße.

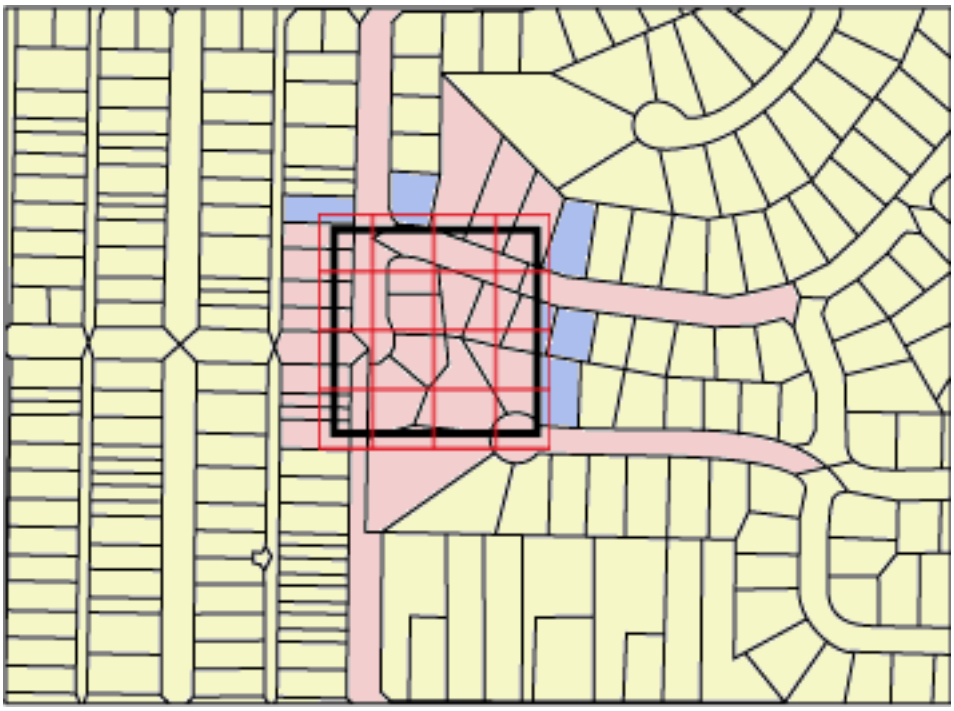


Abbildung 16. Mittlere Rastergröße (100) auf Landparzellen

Räumliche Rasterindizes erstellen

Erstellen Sie räumliche Rasterindizes zum Definieren zweidimensionaler Rasterindizes für räumliche Spalten, mit denen die Leistung räumlicher Abfragen optimiert werden kann.

Vorbereitende Schritte

Vor der Erstellung eines räumlichen Rasterindexes ist Folgendes zu berücksichtigen:

- Ihre Benutzer-ID muss über die Berechtigungen verfügen, die zur Ausführung der SQL-Anweisung CREATE INDEX erforderlich sind.
- Sie müssen die Werte kennen, die für den vollständig qualifizierten Namen des räumlichen Rasterindexes und die drei Rastergrößen angegeben werden sollen, die vom Index verwendet werden.

Empfehlungen:

- Bevor Sie einen räumlichen Rasterindex für eine Spalte erstellen, sollten Sie mithilfe des Indexadvisors die Indexparameter ermitteln. Der Indexadvisor kann die Daten der räumlichen Spalte analysieren und geeignete Rastergrößen für Ihren räumlichen Rasterindex vorschlagen.
- Wenn Sie die für die Spalte erforderlichen Daten mit einem einleitenden Ladevorgang bereitstellen wollen, müssen Sie den räumlichen Rasterindex erstellen, nachdem Sie den Ladeprozess abgeschlossen haben. Auf diese Weise können Sie die optimalen Rasterzellengrößen anhand der Merkmale der Daten oder durch die Verwendung des Indexadvisors auswählen. Außerdem wird die Leistung des

Ladeprozesses verbessert, wenn dieser vor der Indexerstellung ausgeführt wird, weil dann der räumliche Rasterindex während des Ladeprozesses nicht verwaltet werden muss.

Informationen zu diesem Vorgang

Sie können räumliche Rasterindizes erstellen, um die Abfrageleistung bei räumlichen Spalten zu verbessern. Beim Erstellen eines räumlichen Rasterindexes geben Sie die folgenden Informationen an:

- Name des räumlichen Rasterindexes.
- Name der räumlichen Spalte, für die der räumliche Rasterindex definiert werden soll.
- Die Kombination der drei Rastergrößen dient durch die Reduzierung der Gesamtanzahl an Indexeinträgen und der Anzahl an Indexeinträgen, die zur Ermittlung der Ergebnisse einer Abfrage durchsucht werden müssen, zur Leistungsoptimierung.

Diese Task beschreibt die Schritte zum Erstellen räumlicher Rasterindizes mithilfe der SQL-Anweisung CREATE INDEX. Räumliche Rasterindizes können auch mit einem GIS-Tool erstellt werden, das mit DB2 Spatial Extender arbeitet. Informationen zur Erstellung eines räumlichen Rasterindexes mit einem GIS-Tool finden Sie in der Dokumentation, die mit dem Tool ausgeliefert wird.

Vorgehensweise

Gehen Sie wie folgt vor, um räumliche Rasterindizes zu erstellen:

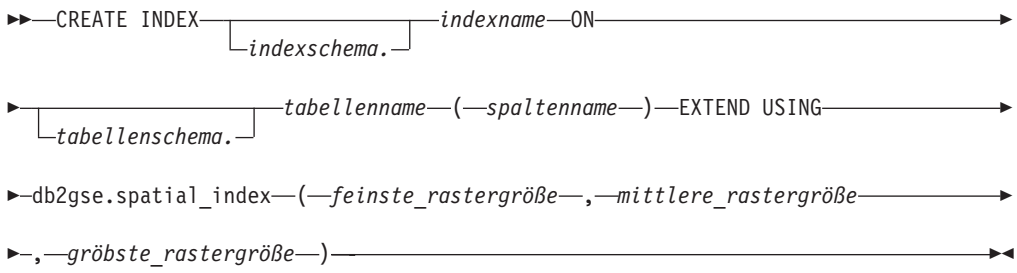
Setzen Sie den Befehl CREATE INDEX im DB2-Befehlszeilenprozessor ab und legen Sie dabei die Rasterindexerweiterung db2gse.spatial_index mit der Klausel EXTEND USING fest. Im folgenden Beispiel wird gezeigt, wie der räumliche Rasterindex TERRIDX für die Tabelle BRANCHES erstellt wird, die eine räumliche Spalte TERRITORY enthält.

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

Anweisung CREATE INDEX für den räumlichen Rasterindex

Verwenden Sie die Anweisung CREATE INDEX mit der Klausel EXTEND USING, um einen räumlichen Rasterindex zu erstellen.

Syntax



Parameter

indexschema.
Der Name des Schemas, zu dem der Index gehören soll, den Sie erstellen wollen. Wenn Sie keinen Namen angegeben, verwendet das DB2-Datenbanksystem den Schemanamen, der im Sonderregister CURRENT SCHEMA gespeichert ist.

indexname
Der Name des Rasterindexes (ohne Qualifikationsmerkmal), den Sie erstellen wollen.

tabellenschema.
Der Name des Schemas, zu dem die Tabelle gehört, die die in *spaltenname* angegebene Spalte enthält. Wenn Sie keinen Namen angeben, verwendet DB2 den Schemanamen, der im Sonderregister CURRENT SCHEMA gespeichert ist.

tabellenname
Der Name der Tabelle (ohne Qualifikationsmerkmal), die die in *spaltenname* angegebene Spalte enthält.

spaltenname
Der Name der räumlichen Spalte, für die der räumliche Rasterindex erstellt werden soll.

feinste_rastergröße, mittlere_rastergröße, gröbste_rastergröße
Die Rastergrößen für den räumlichen Rasterindex. Diese Parameter müssen die folgenden Bedingungen erfüllen:

- Der Wert für *feinste_rastergröße* muss größer als 0 sein.
- Der Wert für *mittlere_rastergröße* muss entweder größer als der Wert für *feinste_rastergröße* oder gleich 0 (null) sein.
- Der Wert für *gröbste_rastergröße* muss entweder größer als der Wert für *mittlere_rastergröße* oder gleich 0 (null) sein.

Wenn Sie den räumlichen Rasterindex mithilfe der Anweisung CREATE INDEX erstellen, wird die Gültigkeit der Rastergrößen geprüft, sobald die erste Geometrie indexiert wird. Falls die von Ihnen angegebenen Rastergrößen nicht die zuvor angegebenen Bedingungen ihrer Werte erfüllen, wird zu diesem Zeitpunkt in den folgenden Situationen eine Fehlerbedingung gemeldet:

- Wenn alle Geometrien in der räumlichen Spalte gleich null sind, erstellt DB2 Spatial Extender den Index, ohne dass die Gültigkeit der Rastergrößen überprüft werden muss. DB2 Spatial Extender überprüft die Rastergrößen, wenn Sie eine Geometrie mit einem Wert ungleich null in diese räumliche Spalte einfügen oder in dieser aktualisieren wollen. Wenn die angegebenen Rastergrößen ungültig sind, wird ein Fehler ausgegeben, sobald Sie eine solche Geometrie einfügen oder aktualisieren wollen.
- Wenn während der Indexerstellung Geometrien mit einem Wert ungleich null in der räumlichen Spalte vorhanden sind, überprüft DB2 Spatial Extender die Rastergrößen zu diesem Zeitpunkt. Wenn die angegebenen Rastergrößen ungültig sind, wird sofort ein Fehler ausgegeben, und der Index für das räumliche Raster wird nicht erstellt.

Beispiel

Die Anweisung CREATE INDEX im folgenden Beispiel dient zur Erstellung des Indexes TERRIDX für das räumliche Raster in der räumlichen Spalte TERRITORY der Tabelle BRANCHES:

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

Räumliche Rasterindizes mit dem Indexadvisor optimieren

DB2 Spatial Extender Spatial Extender stellt das Dienstprogramm Indexadvisor bereit, mit dem der Zugriff auf die räumlichen Daten optimiert werden kann.

Mit dem Indexadvisor können Sie Folgendes tun:

- Festlegen der geeigneten Rastergrößen für Ihre räumlichen Rasterindizes

  Der Indexadvisor analysiert die Geometrien in einer räumlichen Spalte und gibt Empfehlungen zu den optimalen Rastergrößen für Ihren räumlichen Rasterindex aus.
- Analysieren eines vorhandenen Rasterindexes

  Der Indexadvisor kann Statistikdaten erfassen und anzeigen, auf deren Basis Sie feststellen können, wie gut die momentan definierten Rasterzellengrößen zum Abrufen der räumlichen Daten geeignet sind.

Rastergrößen für räumliche Rasterindizes festlegen

Vor der Erstellung eines räumlichen Rasterindexes legen Sie mithilfe des Indexadvisors die benötigten Rastergrößen fest.

Vorbereitende Schritte

- Ihre Benutzer-ID muss das Zugriffsrecht SELECT für diese Tabelle besitzen.
- Wenn Ihre Tabelle mehr als eine Million Zeilen umfasst, sollten Sie die Klausel ANALYZE verwenden, um nur eine Untergruppe der Zeilen zu analysieren. Nur auf diese Weise können die Verarbeitungszeiten in einem akzeptablen Bereich gehalten werden.

Vorgehensweise

Ermittlung der geeigneten Rastergrößen für einen räumlichen Rasterindex:

1. Bestimmen Sie mithilfe des Indexadvisors eine empfohlene Rasterzellengröße für den Index, den Sie erstellen möchten.
   a. Geben Sie den Befehl zum Aufrufen des Indexadvisors mit dem Schlüsselwort ADVISE ein, um die Rasterzellengrößen anzufordern. Um den Indexadvisor z. B. für die Spalte SHAPE in der Tabelle COUNTIES aufzurufen, müssen Sie Folgendes eingeben:

      ```
      gseidx CONNECT TO meinedb USER benutzer_id USING kennwort GET GEOMETRY
      STATISTICS FOR COLUMN benutzer_id.counties(shape) ADVISE
      ```

      Einschränkung: Wenn Sie den Befehl **gseidx** von einer Eingabeaufforderung des Betriebssystems aus eingeben, müssen Sie den vollständigen Befehl in einer einzigen Zeile eingeben. Alternativ können Sie **gseidx**-Befehle über eine CLP-Datei ausführen. Dabei kann der Befehl über mehrere Zeilen aufgeteilt werden.

      Der Indexadvisor gibt dann die empfohlenen Rasterzellengrößen zurück. Beispielsweise gibt der Befehl **gseidx** mit dem oben genannten Schlüsselwort **ADVISE** die folgenden empfohlenen Zellengrößen für die Spalte SHAPE zurück:

      ```
      Abfragefenstergröße  Empfohlene Rastergrößen        Kosten
      -------------------  --------------------------     ------
                    0,1   0,7,       2,8,      14,0        2,7
                    0,2   0,7,       2,8,      14,0        2,9
      ```

```
 0,5   1,4,       3,5,      14,0        3,5
 1     1,4,       3,5,      14,0        4,8
 2     1,4,       3,5,      14,0        8,2
 5     1,4,       3,5,      14,0       24
10     2,8,       8,4,      21,0       66
20     4,2,      14,7,      37,0      190
50     7,0,      14,0,      70,0      900
100   42,0,         0,         0     2800
```

b. Wählen Sie in der Ausgabe von **gseidx** eine geeignete Abfragefenstergröße aus, und berücksichtigen Sie hierbei die Breite der Koordinaten, die in der Anzeige dargestellt werden.

   Im vorliegenden Beispiel werden die Koordinaten durch Längen- und Breitengradwerte in Dezimalgrad dargestellt. Wenn Ihre Kartendarstellung normalerweise eine Breite von 0,5 Grad (ca. 55 km) aufweist, suchen Sie die Zeile mit dem Wert 0,5 in der Spalte 'Afragefenstergr.'. Für diese Zeilen werden die Rastergrößen 1,4, 3,5 und 14,0 vorgeschlagen.

2. Erstellen Sie den Index mit den vorgeschlagenen Rastergrößen. Im Beispiel aus dem vorherigen Schritt können Sie die folgenden SQL-Anweisungen ausführen:

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

Statistikdaten für räumliche Rasterindizes analysieren

Anhand der Analyse von Statistikdaten zu einem vorhandenen räumlichen Rasterindex können Sie feststellen, ob der Index effizient ist oder aber durch einen effizienteren Index ersetzt werden sollte.

Vorbereitende Schritte

Bevor Sie die zu indexierenden Daten analysieren können, müssen die folgenden Bedingungen erfüllt werden:

- Ihre Benutzer-ID muss das Zugriffsrecht SELECT für diese Tabelle besitzen.
- Wenn Ihre Tabelle mehr als eine Million Zeilen umfasst, sollten Sie die Klausel ANALYZE verwenden, um nur eine Untergruppe der Zeilen zu analysieren. Nur auf diese Weise können die Verarbeitungszeiten in einem akzeptablen Bereich gehalten werden. Zur Verwendung der Klausel ANALYZE benötigen Sie einen Tabellenbereich USER TEMPORARY. Legen Sie als Seitengröße dieses Tabellenbereichs mindestens 8 KB fest, und stellen Sie sicher, dass Sie über USE-Zugriffsrechte für diesen Tabellenbereich verfügen. Mit folgenden DDL-Anweisungen können Sie z. B. einen Pufferpool erstellen, der über die gleiche Seitengröße verfügt wie der temporäre Benutzertabellenbereich, und jedem Benutzer das Zugriffsrecht USE erteilen:

```
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;
```

Informationen zu diesem Vorgang

Mit dem Indexadvisor können Sie Statistikdaten für vorhandene räumliche Rasterindizes abrufen. Analysieren Sie diese Statistikdaten und stellen Sie fest, ob Indizes ersetzt werden müssen.

Tipp: Ähnlich wichtig wie die Optimierung des Indexes ist das Überprüfen seiner Verwendung durch die abgesetzten Abfragen. Um festzustellen, ob ein räumlicher

Index verwendet wird, führen Sie ein Befehlszeilentool wie z. B. **db2exfmt** für Ihre Abfrage aus. Wenn im Abschnitt für den Zugriffsplan in der Ausgabe von Visual Explain der Operator EISCAN und der Name Ihres räumlichen Indexes angezeigt werden, verwendet die Abfrage diesen Index.

Vorgehensweise

Zum Analysieren von Statistikdaten für vorhandene räumliche Rasterindizes und um festzustellen, ob sie durch effizientere Indizes ersetzt werden sollten, gehen Sie wie folgt vor:

Rufen Sie Statistikdaten zu einem räumlichen Rasterindex ab, und ersetzen Sie ggf. den Index:

1. Lassen Sie vom Indexadvisor Statistikdaten erfassen, die auf den Rasterzellengrößen des vorhandenen Indexes basieren. Sie können Statistikdaten entweder für alle indexierten Daten oder für eine Untergruppe dieser Daten anfordern.
   - Um Statistikdaten für indexierte Daten in einer Untergruppe von Zeilen abzurufen, geben Sie den Befehl **gseidx** ein. Geben Sie außerdem zusätzlich zur Klausel für vorhandene Indizes und zum Schlüsselwort **DETAIL** das Schlüsselwort **ANALYZE** und seine Parameter an. Sie können entweder die Anzahl oder den Prozentsatz der Zeilen festlegen, die der Indexadvisor zum Abrufen der Statistikdaten analysieren soll. Um z. B. Statistikdaten für eine Untergruppe der Daten abzurufen, die mithilfe des Indexes COUNTIES_SHAPE_IDX indexiert wurden, müssen Sie Folgendes eingeben:

     ```
     gseidx CONNECT TO meinedb USER benutzer_id USING kennwort GET GEOMETRY
     STATISTICS FOR INDEX benutzer_id.counties_shape_idx DETAIL ANALYZE 25 PERCENT
      ADVISE
     ```

   - Um Statistikdaten für alle indexierten Daten abzurufen, geben Sie den Befehl **gseidx** ein. Geben Sie außerdem die Klausel für vorhandene Indizes an. Verwenden Sie das Schlüsselwort **DETAIL**. Um z. B. den Indexadvisor für den Index COUNTIES_SHAPE_IDX aufzurufen, müssen Sie Folgendes eingeben:

     ```
     gseidx CONNECT TO meinedb USER benutzer_id USING kennwort GET GEOMETRY
     STATISTICS FOR INDEX benutzer_id.counties_shape_idx DETAIL SHOW HISTOGRAM ADVISE
     ```

   Der Indexadvisor gibt Statistikdaten, ein Datenhistogramm und die empfohlenen Zellengrößen für den vorhandenen Index zurück. Mit dem zuvor verwendeten Befehl **gseidx** können für alle Daten, die mit COUNTIES_SHAPE_IDX indexiert wurden, die folgenden Statistikdaten zurückgegeben werden:

   ```
   Rasterebene 1
   ------------

   Rastergröße                   : 0,5
   Anzahl der Geometrien         : 2936
   Anzahl der Indexeinträge      : 12197

   Anzahl belegter Rasterzellen  : 2922
   Verhältnis Indexeintrag/Geometrie: 4,154292
   Verhältnis Geometrie/Rasterzelle: 1,004791
   Maximale Anzahl der Geometrien pro Rasterzelle: 14
   Minimale Anzahl der Geometrien pro Rasterzelle: 1

   Indexeinträge :   1      2      3      4      10
   ---------------   ------ ------ ------ ------ ------
   Absolut        :  86     564    72     1519   695
   Prozentsatz (%):  2,93   19,21  2,45   51,74  23,67
   ```

```
Rasterebene 2
-------------

Rastergröße                    : 0,0
Auf dieser Ebene wurden keine Geometrien indexiert.


Rasterebene 3
-------------

Rastergröße                    : 0,0
Auf dieser Ebene wurden keine Geometrien indexiert.


Rasterebene X
-------------

Anzahl der Geometrien          : 205
Anzahl der Indexeinträge       : 205
```

2. Ermitteln Sie, inwieweit die Rasterzellengrößen des vorhandenen Indexes die Abfrage vereinfachen. Werten Sie die im vorherigen Arbeitsschritt zurückgegebenen Statistikdaten aus.

   Tipp:

   - In der Statistik sollte für „Verhältnis Indexeintrag/Geometrie (Index Entry/Geometry ratio)" ein Wert im Bereich zwischen 1 und 4 ausgegeben werden, wobei Werte zu bevorzugen sind, die näher bei 1 liegen.
   - Die Anzahl der Indexeinträge pro Geometrie sollte für die höchste Rastergröße kleiner als 10 sein, um das Erreichen der Überlaufebene zu vermeiden.

     Die Darstellung des Abschnitts „Rasterebene X" in der Ausgabe des Indexadvisors gibt an, dass eine Überlaufebene vorhanden ist.

   Die im vorherigen Arbeitsschritt für COUNTIES_SHAPE_IDX abgerufenen Indexstatistikdaten geben an, dass die Rastergrößen (0,5, 0, 0) sich für die Daten in dieser Spalte nicht eignen. Dies hat folgende Gründe:

   - Für die Rasterebene 1 ist der Wert für „Verhältnis Indexeintrag/Geometrie" (`4,154292`) größer als der Richtwert von 4.

     Die Zeile „Indexeinträge" enthält die Werte 1, 2, 3, 4 und 10. Hierdurch wird die Anzahl der Indexeinträge pro Geometrie angegeben. Die Werte der Zeile „Absolut" unter der Spalte „Indexeinträge" gibt die Anzahl der Geometrien an, die diese spezielle Anzahl von Indexeinträgen aufweisen. Die Ausgabe im vorherigen Arbeitsschritt zeigt z. B. 1519 Geometrien mit 4 Indexeinträgen. Der Wert für „Absolut" lautet für 10 Indexeinträge 695, wodurch angezeigt wird, dass 695 Geometrien über 5 bis 10 Indexeinträge verfügen.
   - Die Darstellung des Abschnitts „Rasterebene X" zeigt, dass eine Überlaufindexebene vorhanden ist. Die Statistikdaten zeigen, dass 205 Geometrien über mehr als 10 Indexeinträge verfügen.

3. Wenn diese Statistikdaten nicht zufriedenstellend sind, überprüfen Sie die Daten im Abschnitt "Histogramm" und die entsprechenden Zeilen in den Spalten "Abfragefenstergröße" und "Empfohlene Rastergrößen" in der Ausgabe des Indexadvisors.

   a. Suchen Sie die MBR-Größe mit der höchsten Anzahl an Geometrien. Im Abschnitt „Histogramm" finden Sie die MBR-Größen und die Anzahl der Geometrien, die über diese MBR-Größe verfügen. Im folgenden Beispielhistogramm befindet sich die höchste Anzahl von Geometrien (437) in der MBR-Größe 0,5.

```
Histogramm:
----------
                    MBR-Größe           Geometrienzähler
                    ------------------- -------------------
                             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. Rufen Sie die Zeile 'Abfragefenstergröße' auf, die den Wert 0,5 enthält, um die empfohlenen Rastergrößen (1,4, 3,5, 14,0) zu ermitteln.

```
Abfragefenstergröße    Empfohlene Rastergrößen      Kosten
-------------------  --------------------------     ------
                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. Überprüfen Sie, ob die empfohlenen Größen den Richtlinien entsprechen, die in Schritt 2 aufgeführt sind. Führen Sie den Befehl **gseidx** mit den empfohlenen Rastergrößen aus:

```
gseidx CONNECT TO meinedb USER benutzer_id USING kennwort GET GEOMETRY
 STATISTICS FOR COLUMN benutzer_id.counties(shape) USING GRID SIZES (1.4, 3.5, 14.0)
Rasterebene 1
------------

Rastergröße                   : 1,4
Anzahl der Geometrien         : 3065
Anzahl der Indexeinträge      : 5951

Anzahl belegter Rasterzellen  : 513
Verhältnis Indexeintrag/Geometrie: 1,941599
Verhältnis Geometrie/Rasterzelle: 5,974659
Maximale Anzahl der Geometrien pro Rasterzelle: 42
Minimale Anzahl der Geometrien pro Rasterzelle: 1

Indexeinträge :  1      2      3      4      10
---------------  ------ ------ ------ ------ ------
Absolut        :   1180   1377   15     493    0
Prozentsatz (%):  38,50   44,93  0,49   16,08  0,00


Rasterebene 2
------------

Rastergröße                   : 3,5
Anzahl der Geometrien         : 61
Anzahl der Indexeinträge      : 143

Anzahl belegter Rasterzellen  : 56
Verhältnis Indexeintrag/Geometrie: 2,344262
Verhältnis Geometrie/Rasterzelle: 1,089286
Maximale Anzahl der Geometrien pro Rasterzelle: 10
Minimale Anzahl der Geometrien pro Rasterzelle: 1

Indexeinträge :  1      2      3      4      10
---------------  ------ ------ ------ ------ ------
Absolut        :  15     28     0      18     0
Prozentsatz (%):  24,59   45,90  0,00   29,51  0,00


Rasterebene 3
------------

Rastergröße                   : 14,0
Anzahl der Geometrien         : 15
Anzahl der Indexeinträge      : 28

Anzahl belegter Rasterzellen  : 9
Verhältnis Indexeintrag/Geometrie: 1,866667
Verhältnis Geometrie/Rasterzelle: 1,666667
Maximale Anzahl der Geometrien pro Rasterzelle: 10
Minimale Anzahl der Geometrien pro Rasterzelle: 1

Indexeinträge :  1      2      3      4      10
---------------  ------ ------ ------ ------ ------
Absolut        :  7      5      1      2      0
Prozentsatz (%):  46,67   33,33  6,67   13,33  0,00
```

In den Statistikdaten werden nun Werte dargestellt, die den geltenden Richtlinien entsprechen:

- Als Werte für „Verhältnis Indexeintrag/Geometrie" werden nun 1,941599 für die Rasterebene 1, 2,344262 für die Rasterebene 2 und 1,866667 für die Rasterebene 3 angegeben. Diese Werte liegen alle innerhalb des in den Richtlinien definierten Wertebereichs von 1 bis 4.

- Das Fehlen des Abschnitts „Rasterebene X" gibt an, dass die Überlaufebene keine Indexeinträge enthält.

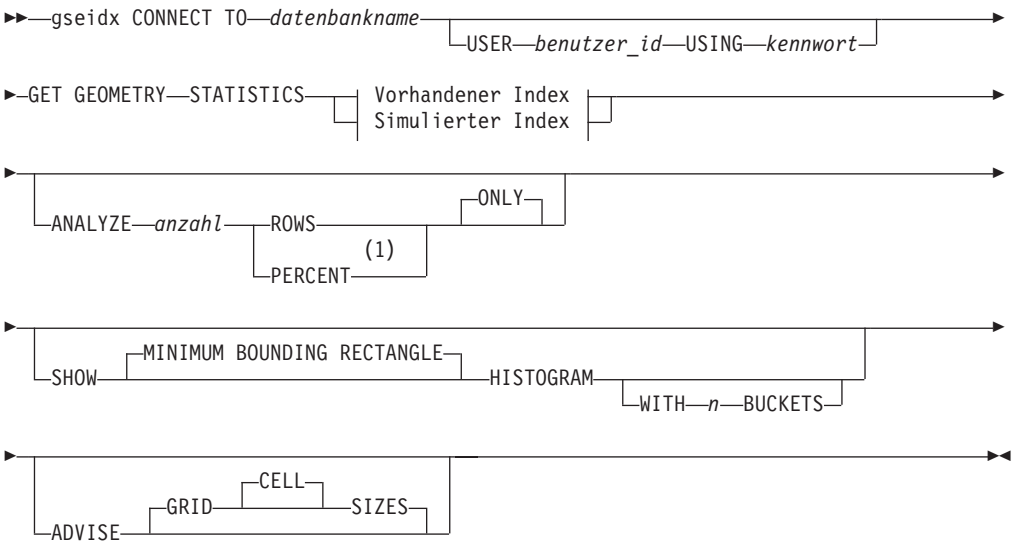
5. Löschen Sie den vorhandenen Index, und ersetzen Sie ihn durch einen Index, der die empfohlenen Rastergrößen angibt. Führen Sie für das Beispiel im vorherigen Schritt die folgenden DDL-Anweisungen aus:

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

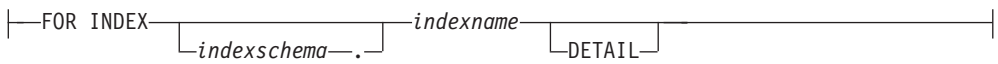
Befehl gseidx

Mit dem Befehl **gseidx** können Sie den Indexadvisor für räumliche Rasterindizes aufrufen.

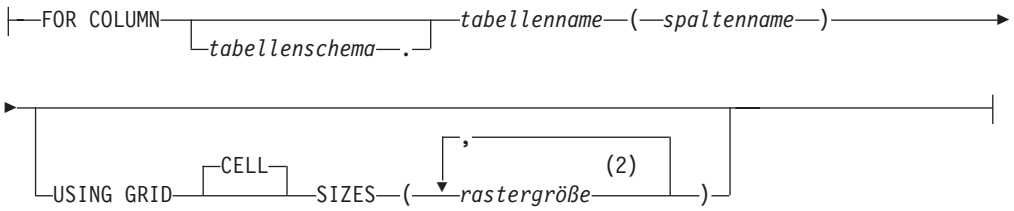
Syntax



vorhandener_index:



simulierter_index:



Anmerkungen:

1 Anstelle des Schlüsselworts PERCENT können Sie auch ein Prozentzeichen (%) angeben.

2 Sie können Zellengrößen für eine, zwei oder drei Rasterebenen angeben.

Parameter

datenbankname
Der Name der Datenbank, in der sich die räumliche Tabelle befindet.

benutzer_id
Die Benutzer-ID, die über die Berechtigung DATAACCESS für die Datenbank verfügt, in der der Index oder die Tabelle gespeichert ist, oder die die Berechtigung SELECT für die Tabelle hat. Wenn Sie sich in der DB2-Befehlsumgebung mit der Benutzer-ID des Datenbankeigners anmelden, müssen Sie im Befehl **gseidx** die Werte für *benutzer_id* und *kennwort* nicht angeben.

kennwort
Das Kennwort für die Benutzer-ID.

vorhandener_index

Dieser Parameter verweist auf einen vorhandenen Index, für den Statistikdaten erfasst werden sollen.

indexschema
Dieser Parameter gibt den Namen des Schemas an, das den vorhandenen Index enthält.

indexname
Dieser Parameter gibt den Namen ohne Qualifikationsmerkmal des vorhandenen Indexes an.

DETAIL
Dieses Schlüsselwort zeigt die folgenden Informationen zu den einzelnen Rasterebenen an:

- Die Größe der Rasterzellen.
- Die Anzahl der indexierten Geometrien.
- Die Anzahl der Indexeinträge.
- Die Anzahl der Rasterzellen, die Geometrien enthalten.
- Die durchschnittliche Anzahl von Indexeinträgen pro Geometrie.
- Die durchschnittliche Anzahl von Geometrien pro Rasterzelle.
- Die Anzahl von Geometrien in der Zelle, die die meisten Geometrien enthält.
- Die Anzahl von Geometrien in der Zelle, die die wenigsten Geometrien enthält.

simulierter_index

Dieser Parameter verweist auf eine Tabellenspalte und auf einen simulierten Index für diese Spalte.

tabellenschema
Dieser Parameter gibt den Namen des Schemas an, das die Tabelle mit der Spalte enthält, für die der simulierte Index gedacht ist.

tabellenname
Dieser Parameter gibt den Namen ohne Qualifikationsmerkmal der Tabelle mit der Spalte an, für die der simulierte Index gedacht ist.

spaltenname
Dieser Parameter gibt den Namen ohne Qualifikationsmerkmal der Tabellenspalte an, für die der simulierte Index gedacht ist.

rastergröße
Dieser Parameter gibt die Größen der Zellen auf den einzelnen

Rasterebenen (feinste Ebene, mittlere Ebene und gröbste Ebene) für einen simulierten Index an. Sie müssen für mindestens eine Ebene eine Zellengröße angeben. Wenn Sie eine Ebene nicht aufnehmen wollen, geben Sie entweder keine Rasterzellengröße für diese Ebene an oder aber eine Rasterzellengröße von 0,0 (null).

Bei Angabe des Parameters *rastergröße* gibt der Indexadvisor dieselben Statistikdaten wie bei Verwendung des Schlüsselworts DETAIL in der Klausel für den vorhandenen Index zurück.

ANALYZE anzahl ROWS | PERCENT ONLY
Geben Sie die ungefähre Menge oder den ungefähren Prozentsatz der Zeilen an, die zum Erfassen der Statistikdaten zu Daten verwendet werden. Wenn Ihre Tabelle mehr als eine Million Zeilen umfasst, sollten Sie die Klausel ANALYZE verwenden, um nur für eine Untergruppe der Daten Statistikdaten zu erfassen, sodass akzeptable Verarbeitungszeiten erzielt werden können.

SHOW MINIMUM BOUNDING RECTANGLE HISTOGRAM
Dieser Parameter ruft ein Diagramm auf, in dem die MBR-Größen der Geometrien sowie die Anzahl der Geometrien mit identischer MBR-Größe angegeben sind.

WITH n BUCKETS
Dieser Parameter gibt die Anzahl der Gruppierungen für die MBRs aller analysierten Geometrien an. Kleine MBRs werden mit anderen kleinen Geometrien in einer Gruppe zusammengefasst. Die größeren MBRs werden zusammen mit anderen größeren Geometrien gruppiert.

Wenn Sie diesen Parameter weglassen oder 0 Buckets angeben, zeigt der Indexadvisor logarithmische Bucketgrößen an. Als MBR-Größen können z. B. logarithmische Werte wie 1,0, 2,0, 3,0,... 10,0, 20,0, 30,0,... 100,0, 200,0, 300,0 etc. angegeben werden.

Wenn Sie eine Anzahl von Buckets angeben, die größer als 0 ist, zeigt der Indexadvisor Werte mit gleicher Größeneinteilung an. Die MBR-Größen können z. B. als Werte mit gleicher Größeneinteilung (8,0, 16,0, 24,0,... 320,0, 328,0, 336,0) angegeben werden.

Standardmäßig werden Buckets verwendet, deren Größe als logarithmische Werte angegeben sind.

ADVISE GRID CELL SIZES
Bei Verwendung dieses Parameters werden die bestmöglichen Rasterzellengrößen berechnet.

Hinweis zur Verwendung

Wenn Sie den Befehl **gseidx** von einer Eingabeaufforderung des Betriebssystems aus eingeben, müssen Sie den vollständigen Befehl in einer einzigen Zeile eingeben.

Beispiel

Mit dem folgenden Befehl wird die Rückgabe von ausführlichen Informationen zu einem vorhandenen Rasterindex angefordert, dessen Name COUNTIES_SHAPE_IDX lautet. Außerdem werden die geeigneten Rasterindexgrößen vorgeschlagen:

```
gseidx CONNECT TO meinedb USER benutzer_id USING kennwort GET GEOMETRY
STATISTICS FOR INDEX benutzer_id.counties_shape_idx DETAIL ADVISE
```

Über Sichten auf räumliche Spalten zugreifen

Sie können eine Sicht, die eine räumliche Spalte verwendet, genauso definieren, wie Sie Sichten in DB2 für andere Datentypen definieren.

Informationen zu diesem Vorgang

Dies ist besonders nützlich, wenn eine Tabelle über mehrere räumliche Spalten verfügt, da zahlreiche Darstellungsanwendungen nur mit einer Tabelle oder Sicht arbeiten können, die nur eine räumliche Spalte enthält. Die Sichtdefinition kann die entsprechende räumliche Spalte und andere nicht räumliche Spalten auswählen, die für die Anwendung verfügbar gemacht werden sollen.

Kapitel 12. Räumliche Informationen analysieren und generieren

Für die Analyse und Erstellung räumlicher Informationen ist es erforderlich, die Umgebungen zu kennen, in denen Abfragen übergeben werden können, sowie die Richtlinien für die Verwendung räumlicher Funktionen in Zusammenhang mit räumlichen Indizes. Sehen Sie sich Beispiele für die verschiedenen Typen von räumlichen Funktionen an, die Sie in einer Abfrage aufrufen können, um mehr über die Möglichkeiten in Spatial Extender zu erfahren.

Umgebungen zur Ausführung einer räumlichen Analyse

Sie können eine räumliche Analyse mit SQL und räumlichen Funktionen über den DB2-Befehlszeilenprozessor und über Anwendungsprogramme in allen von DB2 unterstützten Sprachen durchführen:

Beispiele für die Operationen von räumlichen Funktionen

DB2 Spatial Extender stellt Funktionen bereit, die verschiedene Operationen mit räumlichen Daten ausführen. Diese Funktionen können nach dem Typ der von ihnen ausgeführten Operation kategorisiert werden.

Tabelle 2 listet diese Kategorien zusammen mit Beispielen auf. Der Text nach Tabelle 2 gibt Aufschluss über die Codierung für diese Beispiele.

Tabelle 2. Räumliche Funktionen und Operationen

Kategorie der Funktion	Beispiel der Operation
Informationen zu spezifischen Geometrien zurückgeben	Umfang des Verkaufsbereichs von Verkaufsstelle 10 in Quadratkilometern zurückgeben
Vergleiche erstellen	Ermitteln, ob die Privatadresse eines Kunden im Verkaufsbereich von Verkaufsstelle 10 liegt
Neue Geometrien aus vorhandenen Geometrien ableiten	Verkaufsbereich einer Verkaufsstelle aus ihrem Standort ableiten
Geometrien in Datenaustauschformate umwandeln und umgekehrt	Kundeninformationen im GML-Format in eine Geometrie umwandeln, damit Informationen zu einer DB2-Datenbank hinzugefügt werden können.

Beispiel 1: Informationen zu spezifischen Geometrien zurückgeben

In diesem Beispiel gibt die Funktion ST_Area einen numerischen Wert zurück, der den Verkaufsbereich der Verkaufsstelle 10 darstellt. Die Funktion gibt den Bereich in denselben Einheiten zurück, die auch in dem Koordinatensystem verwendet werden, mit dessen Hilfe der Bereichsstandort definiert wird.

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

Das folgende Beispiel zeigt dieselbe Operation wie das vorherige Beispiel. ST_Area wird jedoch als Methode aufgerufen und gibt den Bereich als Anzahl von Quadratmeilen zurück.

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

Beispiel 2: Vergleiche erstellen

In diesem Beispiel vergleicht die Funktion ST_Within die Koordinaten der Geometrie für den Wohnsitz eines Kunden mit den Koordinaten einer Geometrie, die den Verkaufsbereich der Verkaufsstelle 10 darstellt. Die Ausgabe der Funktion gibt an, ob der Wohnsitz im Verkaufsbereich liegt.

```
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
```

Beispiel 3: Neue Geometrien aus vorhandenen Geometrien ableiten

In diesem Beispiel leitet die Funktion ST_Buffer eine Geometrie für den Verkaufsbereich einer Verkaufsstelle aus einer Geometrie ab, die den Standort der Verkaufsstelle darstellt.

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

Das folgende Beispiel zeigt dieselbe Operation wie das vorherige Beispiel. ST_Buffer wird jedoch als Methode aufgerufen.

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

Beispiel 4: Geometrien in Datenaustauschformate umwandeln und umgekehrt

In diesem Beispiel werden Kundeninformationen, die im GML-Format codiert sind, in eine Geometrie umgewandelt, um sie in einer DB2-Datenbank zu speichern.

```
     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) )
```

Funktionen, die zur Abfrageoptimierung Indizes verwenden

Eine spezielle Gruppe räumlicher Funktionen (sog. *Vergleichsfunktionen*) können zur Verbesserung der Abfrageleistung beitragen, indem sie räumliche Rasterindizes verwenden. Jede dieser Funktionen vergleicht zwei Geometrien miteinander.

Wenn die Ergebnisse der Vergleichsoperation bestimmte Kriterien erfüllen, gibt die Funktion einen Wert von 1 zurück. Ist dies nicht der Fall, gibt die Funktion den Wert 0 zurück. Kann die Vergleichsoperation nicht ausgeführt werden, gibt die Funktion einen Nullwert zurück.

Die Funktion ST_Overlaps vergleicht beispielsweise zwei Geometrien, deren Dimension identisch ist (z. B. zwei Linienfolgen oder zwei Polygone). Wenn sich die

Geometrien teilweise überlappen und wenn der durch die Überlappung belegte Bereich dieselbe Dimension wie die Geometrien aufweist, gibt ST_Overlaps den Wert 1 zurück.

Tabelle 3 zeigt, welche Vergleichsfunktionen einen räumlichen Rasterindex verwenden können:

Tabelle 3. Vergleichsfunktionen, die einen räumlichen Rasterindex verwenden können

Vergleichsfunktion	Verwendung eines räumlichen Rasterindexes möglich
EnvelopesIntersect	Ja
ST_Contains	Ja
ST_Crosses	Ja
ST_Distance	Ja
ST_EnvIntersects	Ja
ST_Equals	Ja
ST_Intersects	Ja
ST_MBRIntersects	Ja
ST_Overlaps	Ja
ST_Touches	Ja
ST_Within	Ja

Die Ausführung einer Funktion ist kosten- und speicherintensiv und kann daher einen erheblichen Verarbeitungsaufwand mit sich bringen. Außerdem ist ein Vergleich umso komplexer und zeitaufwändiger, je komplexer die zu vergleichenden Geometrien sind. Die weiter oben aufgeführten spezialisierten Funktionen können schneller ausgeführt werden, wenn die Geometrien über einen räumlichen Index lokalisiert werden. Wenn Sie eine solche Funktion für die Verwendung eines räumlichen Indexes aktivieren, sollten Sie die folgenden Regeln beachten:

- Die Funktion muss in einer Klausel WHERE angegeben werden. Wird sie in einer Klausel SELECT, HAVING oder GROUP BY angegeben, ist die Verwendung eines räumlichen Indexes nicht möglich.
- Die Funktion muss der Ausdruck sein, der links vom Vergleichselement steht.
- Der Operator, der im Vergleichselement für das Ergebnis der Funktion mit einem anderen Ausdruck verwendet wird, muss ein Gleichheitszeichen sein. Die Funktion ST_Distance bildet hierbei jedoch eine Ausnahme, da diese den Operator "kleiner als" verwenden muss.
- Der Ausdruck rechts vom Vergleichselement muss die Konstante 1 sein. Dies gilt allerdings nicht, wenn als Funktion auf der linken Seite ST_Distance angegeben ist.
- Die Operation muss eine Suche in einer räumlichen Spalte beinhalten, für die ein räumlicher Index definiert ist.

Beispiel:

```
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
```

Tabelle 4 auf Seite 100 zeigt richtige und falsche Beispiele für die Erstellung von räumlichen Abfragen, die einen räumlichen Index verwenden.

Tabelle 4. Beispiele für die Beachtung und Verletzung der Regeln zur Verwendung eines räumlichen Indexes durch räumliche Funktionen

Abfragen, die auf räumliche Funktionen verweisen	Verletzte Regeln

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Contains(s.sales_zone,`
`ST_Point(-121.8,37.3, 1)) = 1`
	In diesem Beispiel wurde keine Regel verletzt.

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Length(s.location) > 10`
	Die räumliche Funktion ST_Length vergleicht keine Geometrien und kann keinen räumlichen Index verwenden.

`SELECT *`
`FROM stores AS s`
`WHERE 1=db2gse.ST_Within(s.location,:BayArea)`
	Die Funktion muss ein Ausdruck sein, der links vom Vergleichselement steht.

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Contains(s.sales_zone,`
`ST_Point(-121.8,37.3, 1)) <> 0`
	Bei Vergleichen auf Gleichheit ist die ganzzahlige Konstante 1 zu verwenden.

`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`
	Keines der Argumente für die Funktion enthält einen räumlichen Index, sodass kein Index verwendet werden kann.

Kapitel 13. Anwendungen schreiben und Beispielprogramm verwenden

Um Anwendungen für Spatial Extender schreiben zu können, müssen Sie die Anforderungen kennen und das Beispielprogramm prüfen.

Kopfdatendatei von DB2 Spatial Extender in räumliche Anwendungen integrieren

DB2 Spatial Extender stellt eine Kopfdatendatei zur Verfügung, in der Konstanten definiert sind, die mit den gespeicherten Prozeduren und Funktionen von DB2 Spatial Extender verwendet werden können.

Informationen zu diesem Vorgang

Empfehlung:
Wenn Sie beabsichtigen, gespeicherte Prozeduren oder Funktionen von DB2 Spatial Extender über Programme aufzurufen, die in C oder C++ geschrieben sind, nehmen Sie diese Kopfdatendatei in Ihre räumlichen Anwendungen auf.

Vorgehensweise

Gehen Sie wie folgt vor, um die Kopfdatendatei von DB2 Spatial Extender in räumliche Anwendungen einzubinden:

1. Vergewissern Sie sich, dass Ihre DB2 Spatial Extender-Anwendungen die benötigten Definitionen in dieser Kopfdatendatei verwenden können.
   a. Nehmen Sie die Kopfdatendatei von DB2 Spatial Extender in Ihr Anwendungsprogramm auf. Die Kopfdatendatei heißt:

      ```
      db2gse.h
      ```

      Die Kopfdatendatei befindet sich im Verzeichnis *db2path*/include. Hierbei steht *db2path* für das Installationsverzeichnis des DB2-Datenbanksystems.
   b. Stellen Sie sicher, dass der Pfad für das Verzeichnis "include" in Ihrer Makefile mit der Kompilierungsoption angegeben ist.
2. Wenn Sie 64-Bit-Anwendungen für Windows auf einem 32-Bit-Windows-System erstellen, ändern Sie den Parameter `DB2_LIBS` in der Datei `samples/extenders/spatial/makefile.nt` so, dass er für 64-Bit-Anwendungen benutzt werden kann. Die erforderlichen Änderungen sind im folgenden Beispiel hervorgehoben:

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

Gespeicherte Prozeduren von DB2 Spatial Extender über eine Anwendung aufrufen

Falls Sie beabsichtigen, Anwendungsprogramme zu schreiben, die eine der gespeicherten Prozeduren von DB2 Spatial Extender aufrufen, verwenden Sie die SQL-Anweisung CALL, und geben Sie den Namen der gespeicherten Prozedur an.

Informationen zu diesem Vorgang

Gespeicherte Prozeduren von DB2 Spatial Extender werden erstellt, wenn Sie die Datenbank für räumliche Operationen aktivieren.

Vorgehensweise

Gehen Sie wie folgt vor, um gespeicherte Prozeduren von DB2 Spatial Extender über eine Anwendung aufzurufen:

1. Rufen Sie die gespeicherte Prozedur DB2GSE.ST_ENABLE_DB auf, um eine Datenbank für die Ausführung räumlicher Operationen zu aktivieren.

   Geben Sie den Namen der gespeicherten Prozedur wie folgt an:

   ```
   CALL DB2GSE!ST_ENABLE_DB
   ```

   Die Angabe `DB2GSE!` in diesem Aufruf stellt den Bibliotheksnamen von DB2 Spatial Extender dar. Die Prozedur ST_ENABLE_DB ist die einzige Prozedur, in deren Aufruf Sie ein Ausrufungszeichen aufnehmen müssen (`DB2GSE!`).

2. Rufen Sie weitere gespeicherte Prozeduren von DB2 Spatial Extender auf. Geben Sie den Namen der gespeicherten Prozedur im folgenden Format an, wobei `DB2GSE` für den Namen des Schemas für alle gespeicherten Prozeduren von DB2 Spatial Extender steht und *name_der_räumlichen_prozedur* für den Namen der gespeicherten Prozedur. Geben Sie im Aufruf kein Ausrufezeichen an.

   ```
   CALL DB2GSE.name_der_räumlichen_prozedur
   ```

   Die gespeicherten Prozeduren von DB2 Spatial Extender sind in der folgenden Tabelle aufgeführt.

Tabelle 5. Gespeicherte Prozeduren von DB2 Spatial Extender

Gespeicherte Prozedur	Beschreibung
ST_ALTER_COORDSYS	Aktualisiert ein Attribut eines Koordinatensystems in der Datenbank.
ST_ALTER_SRS	Aktualisiert ein Attribut eines räumlichen Bezugssystems in der Datenbank.
ST_CREATE_COORDSYS	Erstellt ein Koordinatensystem in der Datenbank.
ST_CREATE_SRS	Erstellt ein räumliches Bezugssystem in der Datenbank.
ST_DISABLE_AUTOGEOCODING	Gibt an, dass DB2 Spatial Extender die Synchronisierung einer geocodierten Spalte mit ihren zugehörigen Geocodierungsspalten stoppt.
ST_DISABLE_DB	Entfernt Ressourcen, mit denen DB2 Spatial Extender räumliche Daten speichern und Operationen für diese Daten unterstützen kann.
ST_DROP_COORDSYS	Löscht ein Koordinatensystem aus der Datenbank.
ST_DROP_SRS	Löscht ein räumliches Bezugssystem aus der Datenbank.
ST_ENABLE_AUTOGEOCODING	Gibt an, dass DB2 Spatial Extender eine geocodierte Spalte mit ihren zugehörigen Geocodierungsspalten synchronisieren soll.

Tabelle 5. Gespeicherte Prozeduren von DB2 Spatial Extender (Forts.)

Gespeicherte Prozedur	Beschreibung
ST_ENABLE_DB	Stellt einer Datenbank die Ressourcen zur Verfügung, die zum Speichern von räumlichen Daten und zur Unterstützung von Operationen benötigt werden.
ST_EXPORT_SHAPE	Exportiert ausgewählte Daten aus der Datenbank in eine Formdatei.
ST_IMPORT_SHAPE	Importiert eine Formdatei in eine Datenbank.
ST_REGISTER_GEOCODER	Registriert einen anderen Geocoder als den Geocoder DB2SE_USA_GEOCODER, der mit dem Produkt DB2 Spatial Extender ausgeliefert wird.
ST_REGISTER_SPATIAL_COLUMN	Registriert eine räumliche Spalte und ordnet ihr ein räumliches Bezugssystem zu.
ST_REMOVE_GEOCODING_SETUP	Entfernt alle Informationen der Geocodierungskonfiguration für die geocodierte Spalte.
ST_RUN_GEOCODING	Führt einen Geocoder im Stapelmodus aus.
ST_SETUP_GEOCODING	Ordnet einer zu geocodierenden Spalte einen Geocoder zu und definiert die entsprechenden Werte für die Geocodierungsparameter.
ST_UNREGISTER_GEOCODER	Nimmt die Registrierung eines Geocoders zurück.
ST_UNREGISTER_SPATIAL_COLUMN	Nimmt die Registrierung einer räumlichen Spalte zurück.

Beispielprogramm von DB2 Spatial Extender

Das Beispielprogramm von DB2 Spatial Extender trägt die Bezeichnung `runGseDemo` und dient zur Ausführung von zwei Aufgaben. Mit dem Beispielprogramm können Sie sich zum einen mit der Anwendungsprogrammierung für DB2 Spatial Extender vertraut machen und zum anderen die Installation von DB2 Spatial Extender prüfen.

Die Position des Programms 'runGSEdemo' variiert entsprechend des Betriebssystems, unter dem DB2 Spatial Extender installiert ist.

- Unter UNIX finden Sie das Programm runGseDemo in folgendem Pfad:
  `$HOME/sqllib/samples/extenders/spatial`

  Hierbei steht `$HOME` für das Ausgangsverzeichnis des Instanzeigners.
- Unter Windows® finden Sie das Programm runGseDemo in folgendem Pfad:
  `c:\Program Files\IBM\sqllib\samples\extenders\spatial`

  Hierbei steht `c:\Programme\IBM\sqllib` für das Verzeichnis, in dem Sie DB2 Spatial Extender installiert haben.

Das Beispielprogramm DB2 Spatial Extender `runGseDemo` vereinfacht die Anwendungsprogrammierung. Mit diesem Beispielprogramm können Sie eine Datenbank für räumliche Operationen aktivieren und räumliche Analysen mit Daten in dieser Datenbank ausführen. Die Datenbank enthält Tabellen mit fiktiven Informationen zu Kunden und zu Überschwemmungsgebieten. Anhand dieser Angaben können

Sie mit Spatial Extender experimentieren und ermitteln, welche Kunden möglicherweise mit Überschwemmungsschäden zu rechnen haben.

Mit dem Beispielprogramm können Sie

- sich mit den Schritten vertraut machen, die normalerweise zur Erstellung und Verwaltung einer für räumliche Operationen aktivierten Datenbank erforderlich sind.
- nachvollziehen, wie gespeicherte räumliche Prozeduren aus einem Anwendungsprogramm heraus aufgerufen werden.
- Mustercode ausschneiden und in eigene Anwendungen einfügen.

Mit dem folgenden Beispielprogramm können Sie Tasks für DB2 Spatial Extender codieren. Angenommen, Sie schreiben eine Anwendung, die gespeicherte Prozeduren von DB2 Spatial Extender über die Datenbankschnittstelle aufruft. In diesem Fall können Sie Code aus dem Beispielprogramm kopieren, um Ihre Anwendung anzupassen. Wenn Sie mit den Programmierungsschritten für DB2 Spatial Extender nicht vertraut sind, können Sie das Beispielprogramm ausführen und sich auf diese Weise jeden Schritt detailliert ansehen. Anweisungen zur Ausführung des Beispielprogramms finden Sie am Ende dieses Abschnitts unter „Zugehörige Tasks" .

In der folgenden Tabelle werden die einzelnen Schritte des Beispielprogramms beschrieben. In jedem Schritt führen Sie eine Aktion aus. Häufig wird diese Aktion anschließend umgekehrt oder rückgängig gemacht. Beispielsweise aktivieren Sie im ersten Schritt die räumliche Datenbank. Anschließend inaktivieren Sie die räumliche Datenbank. Auf diese Weise machen Sie sich mit vielen gespeicherten Prozeduren von DB2 Spatial Extender vertraut.

Tabelle 6. Schritte im Beispielprogramm von DB2 Spatial Extender

Schritte	Aktion und Beschreibung
Räumliche Datenbank aktivieren bzw. inaktivieren	• Räumliche Datenbank aktivieren Dies ist der erste Schritt, der für die Verwendung von DB2 Spatial Extender erforderlich ist. Eine Datenbank, die für räumliche Operationen aktiviert wurde, enthält eine Gruppe von räumlichen Typen, eine Gruppe von räumlichen Funktionen, eine Gruppe von räumlichen Vergleichselementen, neue Indextypen sowie eine Gruppe von Tabellen und Sichten für räumliche Kataloge. • Räumliche Datenbank inaktivieren Dieser Schritt wird normalerweise ausgeführt, wenn Sie die räumlichen Funktionen für die falsche Datenbank aktiviert haben oder wenn Sie in der entsprechenden Datenbank keine räumlichen Operationen mehr ausführen müssen. Beim Inaktivieren einer räumlichen Datenbank werden die Gruppe der räumlichen Typen, die Gruppe der räumlichen Funktionen, die Gruppe der räumlichen Vergleichselemente, die neuen Indextypen und die Gruppe der Tabellen und Sichten für räumliche Kataloge entfernt, die dieser Datenbank zugeordnet sind. • Räumliche Datenbank aktivieren Siehe vorherige Aktion.

Tabelle 6. Schritte im Beispielprogramm von DB2 Spatial Extender (Forts.)

Schritte	Aktion und Beschreibung
Koordinatensystem erstellen bzw. löschen	• Koordinatensystem namens NORTH_AMERICAN erstellen Mit diesem Schritt wird in der Datenbank ein neues Koordinatensystem erstellt. • Koordinatensystem namens NORTH_AMERICAN löschen Mit diesem Schritt wird das Koordinatensystem NORTH_AMERICAN aus der Datenbank gelöscht. • Koordinatensystem namens KY_STATE_PLANE erstellen Mit diesem Schritt wird ein neues Koordinatensystem namens KY_STATE_PLANE erstellt, das von dem im nächsten Schritt zu erstellenden räumlichen Bezugssystem verwendet wird.
Räumliches Bezugssystem erstellen bzw. löschen	• Räumliches Bezugssystem namens SRSDEMO1 erstellen Mit diesem Schritt wird ein neues räumliches Bezugssystem (Spatial Reference System - SRS) definiert, mit dem die Koordinaten interpretiert werden. Ein räumliches Bezugssystem enthält Geometriedaten in einem Format, das in einer Spalte einer für räumliche Operationen aktivierten Datenbank gespeichert werden kann. Nachdem das SRS für eine bestimmte räumliche Spalte registriert wurde, können die entsprechenden Koordinaten für diese räumliche Spalte in der zugeordneten Spalte der Tabelle CUSTOMERS gespeichert werden. • Räumliches Bezugssystem namens SRSDEMO1 löschen Diesen Schritt führen Sie aus, wenn Sie das räumliche Bezugssystem in der Datenbank nicht mehr benötigen. Beim Löschen eines räumlichen Bezugssystems wird seine Definition aus der Datenbank entfernt. • Räumliches Bezugssystem namens KY_STATE_SRS erstellen
Räumliche Tabellen erstellen und füllen	• Tabelle CUSTOMERS erstellen • Tabelle CUSTOMERS füllen Die Tabelle CUSTOMERS stellt Geschäftsdaten dar, die seit mehreren Jahren in der Datenbank gespeichert wurden. • Tabelle CUSTOMERS durch Hinzufügen der Spalte LOCATION ändern Die Anweisung ALTER TABLE fügt eine neue Spalte namens LOCATION hinzu. Diese Spalte hat den Typ ST_Point. Sie wird durch eine Geocodierung der Adress-Spalten in einem späteren Schritt ausgefüllt. • Tabelle OFFICES erstellen Die Tabelle OFFICES stellt neben anderen Daten die Vertriebszone für jede Niederlassung eines Versicherungsunternehmens dar. Die gesamte Tabelle wird in einem späteren Schritt mit den attributiven Daten aus einer Nicht-DB2-Datenbank ausgefüllt. In diesem nachfolgenden Schritt werden Attributdaten aus einer Formdatei in die Tabelle OFFICES importiert.

Tabelle 6. Schritte im Beispielprogramm von DB2 Spatial Extender (Forts.)

Schritte	Aktion und Beschreibung
Spalten füllen	• Adressdaten für die Spalte LOCATION der Tabelle CUSTOMERS mit dem Geocoder KY_STATE_GC geocodieren Mit diesem Schritt wird durch den Aufruf des Geocoderdienstprogramms eine räumliche Geocodierung im Stapelbetrieb ausgeführt. Eine Stapelgeocodierung wird normalerweise ausgeführt, wenn ein erheblicher Anteil der Tabelle geocodiert oder erneut geocodiert werden muss. • Die zuvor erstellte Tabelle OFFICES aus der Formdatei mithilfe des räumlichen Bezugsystems KY_STATE_SRS laden Mit diesem Schritt werden räumliche Daten in die Tabelle OFFICES geladen, die als Formdatei vorliegen. Da die Tabelle OFFICES vorhanden ist, hängt das Dienstprogramm LOAD die neuen Datensätze an die vorhandene Tabelle an. • Tabelle FLOODZONES aus der Formdatei mit dem räumlichen Bezugssystem KY_STATE_SRS erstellen und laden Mit diesem Schritt werden Daten in die die Tabelle FLOODZONES geladen, die als Formdatei vorliegen. Da die Tabelle nicht vorhanden ist, erstellt das Dienstprogramm LOAD die Tabelle, bevor die Daten geladen werden. • Tabelle REGIONS aus der Formdatei mit dem räumlichen Bezugssystem KY_STATE_SRS erstellen und laden
Geocoder registrieren bzw. Registrierung zurücknehmen	• Geocoder SAMPLEGC registrieren • Registrierung des Geocoders SAMPLEGC zurücknehmen • Geocoder KY_STATE_GC registrieren Mit diesen Schritten registrieren Sie den Geocoder SAMPLEGC und nehmen seine Registrierung zurück. Anschließend erstellen Sie einen neuen Geocoder namens KY_STATE_GC, der im Beispielprogramm verwendet werden soll.
Räumliche Indizes erstellen	• Räumlichen Rasterindex für die Spalte LOCATION der Tabelle CUSTOMERS erstellen • Räumlichen Rasterindex für die Spalte LOCATION der Tabelle CUSTOMERS löschen • Räumlichen Rasterindex für die Spalte LOCATION der Tabelle CUSTOMERS erstellen • Räumlichen Rasterindex für die Spalte LOCATION der Tabelle OFFICES erstellen • Räumlichen Rasterindex für die Spalte LOCATION der Tabelle FLOODZONES erstellen • Räumlichen Rasterindex für die Spalte LOCATION der Tabelle REGIONS erstellen Mit diesen Schritten werden die räumlichen Rasterindizes für die Tabellen CUSTOMERS, OFFICES, FLOODZONES und REGIONS erstellt.

Tabelle 6. Schritte im Beispielprogramm von DB2 Spatial Extender (Forts.)

Schritte	Aktion und Beschreibung
Automatische Geocodierung aktivieren	• Geocodierung der Spalte LOCATION in der Tabelle CUSTOMERS mit dem Geocoder KY_STATE_GC definieren Dieser Schritt ordnet der Spalte LOCATION in der Tabelle CUSTOMERS den Geocoder KY_STATE_GC zu und definiert die entsprechenden Werte für die Geocodierungsparameter. • Automatische Geocodierung für Spalte LOCATION in der Tabelle CUSTOMERS aktivieren Mit diesem Schritt wird der automatische Aufruf des Geocoders aktiviert. Durch die automatische Geocodierung werden die Spalten LOCATION, LATITUDE und LONGITUDE der Tabelle CUSTOMERS für spätere Einfüge- und Aktualisierungsoperationen miteinander synchronisiert.
Einfüge-, Aktualisierungs- und Löschoperationen für die Tabelle CUSTOMERS ausführen	Diese Schritte demonstrieren Einfüge-, Aktualisierungs- und Löschoperationen für die Spalten LATITUDE, LONGITUDE, STREET, CITY, STATE und ZIP in der Tabelle CUSTOMERS. Nachdem die automatische Geocodierung aktiviert wurde, werden Daten, die in diesen Spalten eingefügt oder aktualisiert wurden, automatisch in der Spalte LOCATION geocodiert. Dieser Prozess wurde im vorherigen Schritt aktiviert. • Einige Datensätze mit unterschiedlicher Straßenangabe einfügen • Einige Datensätze mit einer neuen Adresse aktualisieren • Alle Datensätze aus der Tabelle löschen
Automatische Geocodierung inaktivieren	Mit diesen Schritten wird der automatische Aufruf des Geocoders und der räumliche Index inaktiviert und so der nächste Schritt vorbereitet. Im anschließenden Schritt wird die gesamte Tabelle CUSTOMERS erneut geocodiert. • Automatische Geocodierung für die Spalte LOCATION in der Tabelle CUSTOMERS inaktivieren • Geocodierungskonfiguration für die Spalte LOCATION der Tabelle CUSTOMERS entfernen • Räumlichen Index für die Spalte LOCATION der Tabelle CUSTOMERS löschen **Empfehlung:** Wenn Sie große Mengen von Geodaten laden, sollten Sie vor dem Laden der Daten den räumlichen Index löschen und dann erneut erstellen, sobald das Laden der Daten abgeschlossen ist.
Sicht erstellen und räumliche Spalte in der Sicht registrieren	Mit diesen Schritten wird eine Sicht erstellt und ihre räumliche Spalte registriert. • Sicht HIGHRISKCUSTOMERS erstellen, die auf dem Join der Tabellen CUSTOMERS und FLOODZONES basiert • Räumliche Spalte der Sicht registrieren

Tabelle 6. Schritte im Beispielprogramm von DB2 Spatial Extender (Forts.)

Schritte	Aktion und Beschreibung
Räumliche Analyse ausführen	Mit diesen Schritten wird eine räumliche Analyse durchgeführt, die die räumlichen Vergleichselemente und Funktionen von DB2 SQL verwendet. Das DB2-Abfrageoptimierungsprogramm nutzt nach Möglichkeit den räumlichen Index für die räumlichen Spalten zur Verbesserung der Abfrageleistung. • Anzahl der Kunden ermitteln, die in jeder Region betreut werden (ST_Within) • Für Niederlassungen und Kunden in einer Region die Anzahl der Kunden ermitteln, die in einem bestimmten Umkreis der Niederlassungen wohnen (ST_Within, ST_Distance) • Durchschnittliche Einkommen und Prämien der Kunden für jede Region ermitteln (ST_Within) • Anzahl der Überschwemmungsgebiete ermitteln, die sich mit den einzelnen Niederlassungszonen überlappen (ST_Overlaps) • Nächstgelegene Niederlassung für einen bestimmten Kundenstandort ermitteln, unter der Voraussetzung, dass sich die Niederlassung im Zentrum der Niederlassungszone befindet (ST_Distance) • Kunden ermitteln, deren Wohnort nahe am Rand eines spezifischen Überschwemmungsgebiets liegt (ST_Buffer, ST_Intersects) • Diejenigen Kunden mit hohem Risiko ermitteln, die in einer angegebenen Entfernung zu einer bestimmten Niederlassung wohnen (ST_Within) Alle diese Schritte verwenden die interne Funktion sqlRunSpatialQueries.
Räumliche Daten in Formdateien exportieren	Dieser Schritt demonstriert, wie die Sicht HIGHRISKCUSTOMERS in Formdateien exportiert wird. Durch das Exportieren von Daten aus einem Datenbankformat in ein anderes Dateiformat können diese Informationen auch von anderen Tools (z. B. ArcExplorer for DB2) genutzt werden. • Sicht HIGHRISKCUSTOMERS in Formdateien exportieren

Kapitel 14. Fehler bei DB2 Spatial Extender erkennen

Um ein DB2 Spatial Extender-Problem zu ermitteln, müssen Sie die Ursache des Problems bestimmen.

Für die Fehlerbehebung stehen Ihnen bei DB2 Spatial Extender die folgenden Methoden zur Verfügung:

- Verwenden Sie die Nachrichteninformationen zur Fehlerdiagnose.
- Bei der Arbeit mit gespeicherten Prozeduren und Funktionen von Spatial Extender gibt DB2 Informationen darüber zurück, ob die gespeicherte Prozedur bzw. Funktion erfolgreich ausgeführt oder fehlerhaft beendet wurde. Die zurückgegebenen Informationen bestehen aus einem Nachrichtencode (in Form einer ganzen Zahl) und/oder einem Nachrichtentext. Dies ist von der Schnittstelle abhängig, die Sie bei der Arbeit mit DB2 Spatial Extender verwenden.
- Sie können die DB2-Benachrichtigungsdatei für die Systemverwaltung anzeigen, in der Diagnoseinformationen zu Fehlern aufgezeichnet werden.
- Wenn bei Spatial Extender wiederholt ein Fehler auftritt, der reproduziert werden kann, werden Sie möglicherweise von einem Mitarbeiter der IBM Kundenunterstützung gebeten, die Fehlerdiagnose mit der DB2-Tracefunktion vorzunehmen.

Hinweise zum Interpretieren der Nachrichten von DB2 Spatial Extender

Bei Kenntnis der Strukturierung von DB2 Spatial Extender-Nachrichten und zusätzlicher Informationsquellen können Sie einfacher bestimmen, ob die angeforderte räumliche Operation erfolgreich abgeschlossen wurde oder zu einem Fehler geführt hat.

Für die Arbeit mit DB2 Spatial Extender stehen Ihnen folgende Schnittstellen zur Verfügung:

- Gespeicherte Prozeduren von DB2 Spatial Extender
- Funktionen von DB2 Spatial Extender
- Befehlszeilenprozessor (CLP) von DB2 Spatial Extender

Alle diese Schnittstellen geben DB2 Spatial Extender-Nachrichten zurück.

Die folgende Tabelle erläutert anhand einer Beispielnachricht die einzelnen Bestandteile von DB2 Spatial Extender-Nachrichten:

```
GSE0000I: Die Operation wurde erfolgreich abgeschlossen.
```

Tabelle 7. Bestandteile von DB2 Spatial Extender-Nachrichten

Teil der Nachricht	Beschreibung
GSE	Die Nachrichten-ID. Alle DB2 Spatial Extender-Nachrichten beginnen mit dem dreistelligen Präfix GSE.
0000	Die Nachrichtennummer (vierstellige Nummer von 0000 bis 9999).

Tabelle 7. Bestandteile von DB2 Spatial Extender-Nachrichten (Forts.)

Teil der Nachricht	Beschreibung
I	Der Nachrichtentyp. Ein einzelner Buchstabe, der die Wertigkeit der Nachricht kenntlich macht: **C** Nachrichten über kritische Fehler **N** Nachrichten über nicht-kritische Fehler **W** Warnungen **I** Informationsnachrichten
Die Operation wurde erfolgreich abgeschlossen.	Die Erläuterung der Nachricht.

Die im Nachrichtentext angegebene Erläuterung ist eine Kurzbeschreibung. Sie können zusätzliche Informationen zu der Nachricht abrufen, die eine ausführliche Erklärung sowie Vorschläge für die Vermeidung oder Korrektur des Problems enthalten. So rufen Sie diese zusätzlichen Informationen auf:

1. Öffnen Sie eine Eingabeaufforderung des Betriebssystems.
2. Geben Sie den DB2-Hilfebefehl mit der Nachrichten-ID und der Nachrichtennummer ein, um zusätzliche Informationen zur Nachricht anzuzeigen. Beispiel:

   ```
   DB2 "? GSEnnnn"
   ```

   Hierbei steht *nnnn* für die Nachrichtennummer.

Sie können die GSE-Nachrichten-ID und den Buchstaben für den Nachrichtentyp in Großbuchstaben oder in Kleinbuchstaben eingeben. Durch Eingabe von `DB2 "? GSE0000I"` erzielen Sie dasselbe Ergebnis durch Eingabe von `db2 "? gse0000i"`.

Beim Eingeben des Befehls können Sie den Buchstaben nach der Nachrichtennummer auslassen. Beispielsweise erzielen Sie durch Eingabe von `DB2 "? GSE0000"` dasselbe Ergebnis wie durch Eingabe des Befehls `DB2 "? GSE0000I"`.

Angenommen, der Nachrichtencode lautet `GSE4107N`. Wenn Sie an der Eingabeaufforderung den Befehl `DB2 "? GSE4107N"` eingeben, werden die folgenden Informationen angezeigt:

```
GSE4107N Der verwendete Gittergrößenwert "<gittergröße>" ist nicht gültig.

Erläuterung: Die angegebene Gittergröße "<gittergröße>" ist nicht gültig.

 Eine der folgenden ungültigen Spezifikationen wurde beim Erstellen des
 Rasterindexes mit der Anweisung CREATE INDEX angegeben:

  - Eine Zahl kleiner als 0 (null) wurde als Rastergröße für die erste,
    zweite oder dritte Rasterebene angegeben.

  - 0 (null) wurde als Rastergröße für die erste Rasterebene
    angegeben.

  - Die für die zweite Rasterebene angegebene Rastergröße ist kleiner als
    die Rastergröße der ersten Rasterebene, ist aber nicht 0 (null).

  - Die für die dritte Rasterebene angegebene Rastergröße ist kleiner als
    die Rastergröße der zweiten Rasterebene, ist aber nicht 0 (null).

  - Die für die dritte Rasterebene angegebene Rastergröße ist größer als
    0 (null), aber die für die zweite Rasterebene angegebene Rastergröße ist 0 (null).

Benutzeraktion: Geben Sie einen gültigen Wert für die Rastergröße an.
```

```
msgcode: -4107

sqlstate: 38SC7
```

Wenn die Informationen aufgrund ihrer Länge nicht in einer einzigen Anzeige ausgegeben werden können und Ihr Betriebssystem das ausführbare Programm **more** und Pipes unterstützt, geben Sie den folgenden Befehl ein:

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

Bei Verwendung des Programms **more** wird nach dem Aufrufen der einzelnen Datenanzeigen eine Pause erzwungen, damit Sie die Informationen lesen können.

Ausgabeparameter von gespeicherten Prozeduren von DB2 Spatial Extender

Mithilfe der Ausgabeparameter von gespeicherten Prozeduren in DB2 Spatial Extender können Sie beim expliziten Aufrufen von gespeicherten Prozeduren in Anwendungsprogrammen oder über den DB2-Befehlszeilenprozessor Probleme diagnostizieren.

Gespeicherte Prozeduren von DB2 Spatial Extender haben zwei Ausgabeparameter - den Nachrichtencode und den Nachrichtentext. Der Parameterwert gibt Aufschluss darüber, ob eine gespeicherte Prozedur fehlgeschlagen ist oder erfolgreich ausgeführt wurde.

nachrichtencode

Der Parameter "nachrichtencode" gibt eine ganze Zahl an. Diese kann positiv, negativ oder null (0) sein. Positive Zahlen werden für Warnungen, negative Zahlen werden für Fehler (kritische und nicht-kritische Fehler) verwendet. Der Wert null (0) wird für Informationsnachrichten verwendet.

Der absolute Wert des Parameters "nachrichtencode" ist im Parameter "nachrichtentext" als Nachrichtennummer enthalten. Beispiel:

- Hat der Parameter "nachrichtencode" den Wert 0, lautet die Nachrichtennummer 0000.
- Hat der Parameter "nachrichtencode" den Wert -219, lautet die Nachrichtennummer 0219. Der negative Wert für "nachrichtencode" macht deutlich, dass die Nachricht auf einen kritischen oder nicht-kritischen Fehler hinweist.
- Hat der Parameter "nachrichtencode" den Wert +1036, lautet die Nachrichtennummer 1036. Der positive Wert für "nachrichtencode" macht deutlich, dass es sich bei der Nachricht um eine Warnung handelt.

Die Zahlenwerte des Parameters "nachrichtencode" für gespeicherte Prozeduren von DB2 Spatial Extender werden in drei Kategorien unterteilt, die in der folgenden Tabelle aufgeführt sind:

Tabelle 8. Nachrichtencodes für gespeicherte Prozeduren

Codes	Kategorie
0000 – 0999	Allgemeine Nachrichten
1000 – 1999	Verwaltungsnachrichten
2000 – 2999	Import- und Exportnachrichten

nachrichtentext

Der Parameter "nachrichtentext" besteht aus der Nachrichten-ID, der Nach-

richtennummer, dem Nachrichtentyp und der Erläuterung. Beispiel für einen Wert des Parameters "nachrichtentext" einer gespeicherten Prozedur:

```
GSE0219N  Eine EXECUTE IMMEDIATE-Anweisung
          ist fehlgeschlagen. SQLERROR = "<sql-fehler>".
```

Die im Parameter "nachrichtentext" angegebene Erläuterung ist eine Kurzbeschreibung. Sie können zusätzliche Informationen zu der Nachricht abrufen, die eine ausführliche Erklärung sowie Vorschläge für die Vermeidung oder Korrektur des Problems enthalten.

Eine ausführliche Erläuterung der einzelnen Bestandteile des Parameters finden Sie im Abschnitt "Hinweise zum Interpretieren der Nachrichten von DB2 Spatial Extender". Dort ist ebenfalls beschrieben, wie Sie zusätzliche Informationen zur Nachricht abrufen.

Zur Fehlerdiagnose für implizit aufgerufene gespeicherte Prozeduren durch Absetzen der DB2 Spatial Extender-Befehle verwenden Sie die Nachrichten, die vom Befehlszeilenprozessor von DB2 Spatial Extender zurückgegeben werden. Weitere Details finden Sie im Abschnitt „Hinweise zum Interpretieren der Nachrichten von DB2 Spatial Extender” auf Seite 109.

In Anwendungen mit gespeicherten Prozeduren arbeiten

Wenn Sie eine gespeicherte Prozedur von DB2 Spatial Extender in einer Anwendung aufrufen, empfangen Sie die Ausgabeparameter "nachrichtencode" und "nachrichtentext". Sie haben die folgenden Möglichkeiten:

- Programmierung der Anwendung für die Rückgabe der Ausgabeparameterwerte an den Anwendungsbenutzer
- Ausführung einer bestimmten Aktion gemäß dem zurückgegebenen Wert für "nachrichtencode"

Über die DB2-Befehlszeile mit gespeicherten Prozeduren arbeiten

Wenn Sie eine gespeicherte Prozedur von DB2 Spatial Extender über die DB2-Befehlszeile aufrufen, empfangen Sie die Ausgabeparameter "nachrichtencode" und "nachrichtentext". Diese Ausgabeparameter geben Aufschluss darüber, ob eine gespeicherte Prozedur fehlgeschlagen ist oder erfolgreich ausgeführt wurde.

Angenommen, Sie stellen eine Verbindung zu einer Datenbank her und wollen die gespeicherte Prozedur ST_DISABLE_DB aufrufen. Das folgende Beispiel verwendet einen DB2-Befehl CALL, um die Datenbank für räumliche Operationen zu inaktivieren, und zeigt die Ergebnisse für die Ausgabewerte. Für den Parameter FORCE wird am Ende des Befehls CALL der Wert 0 mit zwei Fragezeichen verwendet. Diese stehen für die Ausgabeparameter "nachrichtencode" und "nachrichtentext". Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

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

    Wert der Ausgabeparameter
    -------------------------
    Parametername   : MSGCODE
    Parameterwert   : 0

    Parametername   : MSGTEXT
    Parameterwert   : GSE0000I  Die Operation wurde erfolgreich abgeschlossen.

    Rückgabestatus = 0
```

Angenommen, für den Parameter "nachrichtentext" wurde der Wert GSE2110N zurückgegeben. Mit dem DB2-Hilfebefehl können Sie weitere Informationen zur Nachricht anzeigen. Beispiel:

```
"? GSE2110"
```

Die folgenden Angaben werden angezeigt:

```
GSE2110N Das räumliche Bezugssystem für die
            Geometrie in Zeile "<zeilennummer>" ist ungültig.
            Die numerische Kennung dieses räumlichen Bezugssystems
            ist "<srs-id>".

Erläuterung: Die zu exportierende Geometrie verwendet in Zeile
zeilennummer ein ungültiges räumliches Bezugssystem.
Die Geometrie kann nicht exportiert werden.

Benutzeraktion: Korrigieren Sie die angegebene Geometrie, oder
schließen Sie die Zeile von der Exportoperation aus,
indem Sie die SELECT-Anweisung entsprechend ändern.

nachrichtencode: -2110

sqlstate: 38S9A
```

Nachrichten von DB2 Spatial Extender-Funktionen

Nachrichten, die von Funktionen von DB2 Spatial Extender zurückgegeben werden, sind normalerweise in eine SQL-Nachricht eingebettet. Der in der Nachricht zurückgegebene Wert für SQLCODE gibt an, ob ein Fehler für die Funktion auftrat oder ob der Funktion eine Warnung zugeordnet ist.

Die folgenden Beispielnachrichten weisen auf einen Fehler und eine Warnung hin:

- Der SQLCODE -443 (Nachrichtennummer SQL0443) gibt an, dass bei der Funktion ein Fehler aufgetreten ist.
- Der SQLCODE +462 (Nachrichtennummer SQL0462) gibt an, dass für die Funktion eine Warnung vorhanden ist.

Die folgende Tabelle erläutert die wichtigen Bestandteile dieser Beispielnachricht:

```
DB21034E  Der Befehl
wurde als SQL-Anweisung verarbeitet, da es
sich um keinen gültigen Befehl des Befehlszeilenprozessors handelte.
Während der SQL-Verarbeitung wurde Folgendes ausgegeben:
SQL0443N  Die Routine "DB2GSE.GSEGEOMFROMWKT" (spezifischer Name
"GSEGEOMWKT1") gab einen SQLSTATE-Fehler zurück.
Der Diagnosetext lautet: "GSE3421N  Fläche ist nicht geschlossen.".
SQLSTATE=38SSL
```

Tabelle 9. Wichtige Bestandteile der Nachrichten von DB2 Spatial Extender-Funktionen

Nachrichtenteil	Beschreibung
SQL0443N	Der SQLCODE-Wert gibt den Fehlertyp an.
GSE3421N	Nachrichtennummer und Nachrichtentyp von DB2 Spatial Extender. Die Nachrichtennummern für Funktionen reichen von GSE3000 bis GSE3999. Außerdem können allgemeine Nachrichten zurückgegeben werden, wenn Sie mit Funktionen von DB2 Spatial Extender arbeiten. Allgemeine Nachrichten tragen die Nachrichtennummern GSE0001 bis GSE0999.
Fläche ist nicht geschlossen.	Erläuterung der DB2 Spatial Extender-Nachricht.

Tabelle 9. Wichtige Bestandteile der Nachrichten von DB2 Spatial Extender-Funktionen (Forts.)

Nachrichtenteil	Beschreibung
SQLSTATE=38SSL	Ein SQLSTATE-Wert, der den Fehler genauer angibt. Ein SQLSTATE-Wert wird für jede Anweisung oder Zeile zurückgegeben. • Die SQLSTATE-Werte für Fehler von DB2 Spatial Extender-Funktionen lauten 38S*xx*. Hierbei steht *x* für einen Buchstaben oder eine Ziffer. • Die SQLSTATE-Werte für Warnungen zu DB2 Spatial Extender-Funktionen lauten 01HS*x*. Hierbei steht *x* für einen Buchstaben oder eine Ziffer.

Beispiel für eine SQL0443-Fehlernachricht

Angenommen, Sie versuchen wie in der folgenden Anweisung dargestellt, Werte für ein Polygon (Fläche) in die Tabelle POLYGON_TABLE einzufügen:

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

Dies führt zu einer Fehlernachricht, weil Sie den Endwert zum Schließen des Polygons nicht angegeben haben. Die zurückgegebene Fehlernachricht lautet:

```
DB21034E  Der Befehl
wurde als SQL-Anweisung verarbeitet, da es
sich um keinen gültigen Befehl des Befehlszeilenprozessors handelte.
Während der SQL-Verarbeitung wurde Folgendes ausgegeben:
SQL0443N  Die Routine "DB2GSE.GSEGEOMFROMWKT" (spezifischer Name
"GSEGEOMWKT1") gab einen SQLSTATE-Fehler zurück.
Der Diagnosetext lautet: "GSE3421N  Fläche ist nicht geschlossen.".
SQLSTATE=38SSL
```

Die SQL-Nachrichtennummer `SQL0443N` gibt an, dass ein Fehler aufgetreten ist. Die Nachricht enthält den DB2 Spatial Extender-Nachrichtentext `GSE3421N Fläche ist nicht geschlossen.`

Führen Sie Folgendes aus, wenn Sie eine Nachricht dieses Typs empfangen:

1. Suchen Sie in der DB2- oder SQL-Fehlernachricht nach der GSE-Nachrichtennummer.
2. Verwenden Sie den DB2-Hilfebefehl (DB2 ?), um die Nachrichtenerläuterung und Benutzeraktion von DB2 Spatial Extender aufzurufen. In diesem Beispiel würden Sie hierzu den folgenden Befehl an einer Eingabeaufforderung des Betriebssystems eingeben:

   ```
   DB2 "? GSE3421"
   ```

Die Nachricht wird wiederholt. Außerdem werden eine ausführliche Erklärung und eine empfohlene Benutzeraktion angegeben.

Nachrichten des Befehlszeilenprozessors von DB2 Spatial Extender

Die Nachrichten des Befehlszeilenprozessors von DB2 Spatial Extender geben an, ob eine Operation erfolgreich ausgeführt wurde oder fehlgeschlagen ist. Darüber hinaus können sie Formdaten enthalten.

Der Befehlszeilenprozessor von DB2 Spatial Extender gibt Nachrichten für die folgenden Komponenten zurück:

- Implizit aufgerufene gespeicherte Prozeduren
- Forminformationen, falls das Unterbefehlsprogramm **shape_info** über den Befehlszeilenprozessor von DB2 Spatial Extender aufgerufen wurde. Hierbei handelt es sich um Informationsnachrichten.
- Upgradeoperationen.
- Import- und Exportoperationen für Formdaten an den und vom Client

Beispiele für Nachrichten gespeicherter Prozeduren, die vom Befehlszeilenprozessor von DB2 Spatial Extender zurückgegeben werden

Die meisten Nachrichten, die durch den Befehlszeilenprozessor von DB2 Spatial Extender zurückgegeben werden, beziehen sich auf gespeicherte Prozeduren von DB2 Spatial Extender. Wenn Sie eine gespeicherte Prozedur über den Befehlszeilenprozessor von DB2 Spatial Extender aufrufen, empfangen Sie einen Nachrichtentext, der angibt, ob die gespeicherte Prozedur erfolgreich ausgeführt wurde oder fehlgeschlagen ist.

Der Nachrichtentext besteht aus der Nachrichten-ID, der Nachrichtennummer, dem Nachrichtentyp und der Erläuterung. Wenn Sie beispielsweise eine Datenbank mit dem Befehl `db2se enable_db testdb` aktivieren, gibt der Befehlszeilenprozessor von DB2 Spatial Extender den folgenden Nachrichtentext zurück:

```
Die Datenbank wird aktiviert. Bitte warten...

GSE1036W Die Operation war erfolgreich. Werte bestimmter
          Datenbankmanager- und Datenbankkonfigurationsparameter
          müssen jedoch            erhöht werden.
```

Analog wird der folgende Nachrichtentext von DB2 Spatial Extender zurückgegeben, wenn Sie eine Datenbank mit dem Befehl `db2se disable_db testdb` inaktivieren:

```
GSE0000I   Die Operation wurde erfolgreich abgeschlossen.
```

Die im Nachrichtentext angegebene Erläuterung ist eine Kurzbeschreibung. Sie können zusätzliche Informationen zu der Nachricht abrufen, die eine ausführliche Erklärung sowie Vorschläge für die Vermeidung oder Korrektur des Problems enthalten. Die Schritte, mit denen diese Angaben abgerufen werden, sowie eine ausführliche Beschreibung der einzelnen Bestandteile des Nachrichtentextes finden Sie in einem gesonderten Abschnitt.

Angaben zur Diagnose der Parameter, die ausgegeben werden, wenn Sie gespeicherte Prozeduren aus einem Anwendungsprogramm heraus oder über die DB2-Befehlszeile aufrufen, enthält ein separater Abschnitt.

Beispielnachrichten für Formdaten, die vom Befehlszeilenprozessor von DB2 Spatial Extender zurückgegeben werden

Angenommen, Sie wollen Informationen zu einer Formdatei namens `office` anzeigen. Hierzu setzen Sie über den Befehlszeilenprozessor von DB2 Spatial Extender (db2se) den folgenden Befehl ab:

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

Daraufhin werden die folgenden Informationen ausgegeben:

```
Formdatei-Informationen
-----------------------
Dateicode                 = 9994
```

```
Dateilänge (16-Bit-Wörter) = 484
Formdateiversion           = 1000
Formtyp                    = 1 (ST_POINT)
Anzahl der Sätze                 = 31

Minimale X-Koordinate = -87.053834
Maximale X-Koordinate = -83.408752
Minimale Y-Koordinate = 36.939628
Maximale Y-Koordinate = 39.016477
Die Formen haben keine Z-Koordinaten.
Die Formen haben keine M-Koordinaten.

Es ist eine Formindexdatei (Erweiterung .shx) vorhanden.

Attributdatei-Informationen
--------------------------
Datenbankdateicode               = 3
Datum der letzten Aktualisierung = 1901-08-15
Anzahl der Sätze                 = 31
Anzahl der Byte in den Kopfdaten = 129
Anzahl der Byte in jedem Satz    = 39
Anzahl der Spalten               = 3

Spaltennummer  Spaltenname      Datentyp        Länge   Dezimal

            1   NAME             C (  Zeichen)      16         0
            2   EMPLOYEES        N ( Numerisch™)    11         0
            3   ID               N ( Numerisch)     11         0

Koordinatensystemdefinition: "GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433295]]"
```

Beispielnachrichten für Upgradeoperationen, die vom Befehlszeilenprozessor von DB2 Spatial Extender zurückgegeben werden

Beim Aufrufen von Befehlen, die Upgradeoperationen ausführen, werden Nachrichten zurückgegeben, die Aufschluss über den Erfolg oder das Fehlschlagen dieser Operation geben.

Beispiel: Sie führen ein Upgrade für die für räumliche Operationen aktivierte Datenbank *meine_datenbank* mithilfe des folgenden Befehls durch:

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

Der Befehlszeilenprozessor von DB2 Spatial Extender gibt daraufhin den folgenden Nachrichtentext zurück:

```
Ein Upgrade für die Datenbank wird durchgeführt. Bitte warten...
GSE0000I   Die Operation wurde erfolgreich abgeschlossen.
```

Trace für Fehler bei DB2 Spatial Extender mit dem Befehl 'db2trc' durchführen

Wenn bei DB2 Spatial Extender wiederholt ein Fehler auftritt, der reproduziert werden kann, können Sie mit der Tracefunktion von DB2 Informationen zu dem Fehler erfassen.

Vorbereitende Schritte

Vergewissern Sie sich, dass Sie über die korrekte Berechtigung zum Absetzen des Befehls **db2trc** verfügen. Auf UNIX-Betriebssystemen müssen Sie die Berechtigung SYSADM, SYSCTRL oder SYSMAINT besitzen, um einen Trace für eine DB2-Ins-

tanz durchführen zu können.Auf Windows-Betriebssystemen ist keine Sonderberechtigung erforderlich.

Informationen zu diesem Vorgang

Die DB2-Tracefunktion wird durch den Befehl **db2trc** aktiviert. Die DB2-Tracefunktion ermöglicht Folgendes:

- Traces für Ereignisse durchführen
- Tracedaten in einer Datei speichern
- Tracedaten in einem lesbaren Format formatieren

Einschränkungen

- Diese Funktion sollte nur auf Anweisung durch einen Mitarbeiter der technischen Unterstützung für DB2 aktiviert werden.
- Der Befehl **db2trc** muss an einer Eingabeaufforderung des Betriebssystems oder in einem Shell-Script eingegeben werden. Er kann weder in der Befehlszeilenschnittstelle (**db2se**) von DB2 Spatial Extender noch im Befehlszeilenprozessor (CLP) von DB2 verwendet werden.

Vorgehensweise

Gehen Sie wie folgt vor, um Probleme in DB2 Spatial Extender mithilfe der DB2-Tracefunktion zu beheben:

1. Beenden Sie alle anderen Anwendungen.
2. Aktivieren Sie den Trace.

   Der Mitarbeiter der technischen Unterstützung für DB2 teilt Ihnen die spezifischen Parameter für diesen Schritt mit. Der Basisbefehl lautet

   ```
   db2trc on
   ```

   Sie können die Tracedaten im Arbeitsspeicher oder in einer Datei speichern lassen. Die bevorzugte Methode ist das Speichern der Tracedaten im Arbeitsspeicher. Falls der reproduzierte Fehler die Workstation blockiert und einen Tracespeicherauszug verhindert, speichern Sie den Trace in einer Datei.
3. Reproduzieren Sie den Fehler.
4. Speichern Sie die Tracedaten unmittelbar nach Auftreten des Fehlers in einer Datei.

   Beispiel:

   ```
   db2trc dump january23trace.dmp
   ```

   Dieser Befehl erstellt im aktuellen Verzeichnis eine Datei (`january23trace.dmp`) mit dem von Ihnen angegebenen Namen und speichert die Trace-Informationen in dieser Datei. Durch Einfügen des Dateipfades können Sie ein anderes Verzeichnis angeben. Um die Speicherauszugsdatei beispielsweise im Verzeichnis `/tmp/spatial/errors` zu speichern, verwenden Sie die folgende Syntax:

   ```
   db2trc dump /tmp/spatial/errors/january23trace.dmp
   ```
5. Inaktivieren Sie den Trace.

   Beispiel:

   ```
   db2trc off
   ```
6. Formatieren Sie die Daten als ASCII-Datei. Sie können die Daten mit zwei Methoden sortieren:

   - Die Option **flw** sortiert die Daten nach Prozess bzw. Thread. Beispiel:

     ```
     db2trc flw january23trace.dmp january23trace.flw
     ```

- Die Option **fmt** listet alle Ereignisse in chronologischer Reihenfolge auf. Beispiel:

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

Benachrichtigungsdatei für Systemverwaltung

Diagnoseinformationen zu Fehlern werden in der Benachrichtigungsdatei für die Systemverwaltung aufgezeichnet. Diese Informationen werden zur Fehlerbestimmung verwendet und sind für die technische Unterstützung für DB2 gedacht.

Die Benachrichtigungsdatei für die Systemverwaltung enthält sowohl vom DB2-Datenbanksystem als auch von DB2 Spatial Extender protokollierte Textinformationen. Sie befindet sich in dem Verzeichnis, das durch den Konfigurationsparameter '**diagpath**' des Datenbankmanagers angegeben wird. Auf Systemen mit Windows-Betriebssystemen befindet sich die DB2-Benachrichtigungsdatei für die Systemverwaltung im Ereignisprotokoll und kann in der Windows-Ereignisanzeige geprüft werden.

Welche Informationen der DB2-Datenbankmanager im Verwaltungsprotokoll aufzeichnet, wird durch die Einstellungen für '**diaglevel**' und '**notifylevel**' bestimmt.

Rufen Sie die Datei auf der Maschine in einem Texteditor auf, auf der vermutlich ein Fehler aufgetreten ist. Die zuletzt aufgezeichneten Ereignisse befinden sich am Ende der Datei. Generell enthält jeder Eintrag die folgenden Teile:

- Eine Zeitmarke.
- Die Position, die den Fehler gemeldet hat. Anhand von Anwendungs-IDs können Sie in den Protokollen von Servern und Clients die Einträge zuordnen, die zu einer Anwendung gehören.
- Eine Diagnosenachricht, die normalerweise mit "DIA" oder "ADM" beginnt und den Fehler erläutert.
- Weitere Unterstützungsdaten (sofern verfügbar), beispielsweise Datenstrukturen des SQL-Kommunikationsbereichs und Zeiger auf die Position von zusätzlichen Speicherauszügen oder Tracedateien.

Wenn das Verhalten der Datenbank normal ist, sind diese Informationen nicht wichtig und können ignoriert werden.

Die Benachrichtigungsdatei für die Systemverwaltung nimmt ständig an Größe zu. Wenn sie zu groß wird, erstellen Sie ein Backup der Datei, und löschen Sie sie anschließen. Sobald das System diese Datei wieder benötigt, wird automatisch eine neue Datei generiert.

Kapitel 15. Katalogsichten

Sie können Katalogsichten für Spatial Extender verwenden, um nützliche Informationen zu Ihren räumlichen Daten abzurufen.

Die Spatial Extender-Katalogsichten enthalten folgende Informationen:

„Katalogsicht DB2GSE.ST_COORDINATE_SYSTEMS"
Koordinatensysteme, die verwendet werden können.

„Katalogsicht DB2GSE.ST_GEOMETRY_COLUMNS" auf Seite 120
Räumliche Spalten, die Sie ausfüllen oder aktualisieren können.

„Katalogsicht DB2GSE.ST_GEOCODERS" auf Seite 123 und „Katalogsicht DB2GSE.ST_GEOCODER_PARAMETERS" auf Seite 121
Geocoder, die Sie verwenden können.

„Katalogsicht DB2GSE.ST_GEOCODING" auf Seite 124 und „Katalogsicht DB2GSE.ST_GEOCODING_PARAMETERS" auf Seite 125
Spezifikationen zur Konfiguration eines Geocoders zur automatischen Ausführung und zur Vorabkonfiguration von Operationen, die während der Geocodierung im Stapelbetrieb ausgeführt werden sollen.

„Katalogsicht DB2GSE.ST_SIZINGS" auf Seite 127
Maximal zulässige Längen der Werte, die Sie Variablen zuordnen können.

„Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS" auf Seite 128
Räumliche Bezugssysteme, die Sie verwenden können.

„Katalogsicht DB2GSE.ST_UNITS_OF_MEASURE" auf Seite 131
Die Maßeinheiten (Meter, Meilen, Fuß usw.), in der von räumlichen Funktion generierte Abstände ausgedrückt werden können.

Katalogsicht DB2GSE.ST_COORDINATE_SYSTEMS

Sie können die Katalogsicht DB2GSE.ST_COORDINATE_SYSTEMS abfragen, um Informationen zu registrierten Koordinatensystemen abzurufen.

DB2 Spatial Extender führt zu den folgenden Zeitpunkten automatisch eine Registrierung von Koordinatensystemen im Spatial Extender-Katalog durch:

- Beim Aktivieren einer Datenbank für räumliche Operationen.
- Beim Definieren zusätzlicher Koordinatensysteme für die Datenbank durch die Benutzer.

Eine Beschreibung der Spalten in dieser Sicht finden Sie in der folgenden Tabelle.

Tabelle 10. Spalten in der Katalogsicht DB2GSE.ST_COORDINATE_SYSTEMS

Name	Datentyp	Dateneingabe optional?	Inhalt
COORDSYS_NAME	VARCHAR(128)	Nein	Name dieses Koordinatensystems. Der Name ist in der Datenbank eindeutig.

Tabelle 10. Spalten in der Katalogsicht DB2GSE.ST_COORDINATE_SYSTEMS (Forts.)

Name	Datentyp	Dateneingabe optional?	Inhalt
COORDSYS_TYPE	VARCHAR(128)	Nein	Typ dieses Koordinatensystems: **PROJECTED** Zweidimensional. **GEOGRAPHIC** Dreidimensional. Verwendet X- und Y-Koordinaten. **GEOCENTRIC** Dreidimensional. Verwendet X-, Y- und Z-Koordinaten. **UNSPECIFIED** Abstraktes Koordinatensystem oder nicht der realen Welt entstammendes Koordinatensystem. Der Wert für diese Spalte wird aus der Spalte DEFINITION übernommen.
DEFINITION	VARCHAR(2048)	Nein	WKT-Darstellung (WKT = Well-Known Text) der Definition dieses Koordinatensystems.
ORGANIZATION	VARCHAR(128)	Ja	Name der Organisation (z. B. eine Standardisierungsorganisation wie beispielsweise "European Petrol Survey Group" oder "ESPG"), die dieses Koordinatensystem definiert hat. Diese Spalte enthält einen Nullwert, wenn die Spalte ORGANIZATION_COORDSYS_ID einen Nullwert enthält.
ORGANIZATION_ COORDSYS_ID	INTEGER	Ja	Eine numerische Kennung, die diesem Koordinatensystem durch die Organisation zugeordnet wurde, die das Koordinatensystem definiert hat. Diese Kennung und der Wert in der Spalte ORGANIZATION bilden die eindeutige Kennzeichnung des Koordinatensystems. Dies gilt jedoch nicht, wenn die Kennung und der Spaltenwert jeweils null ist. Falls der Wert in der Spalte ORGANIZATION null ist, ist die Spalte ORGANIZATION_COORDSYS_ID ebenfalls null.
DESCRIPTION	VARCHAR(256)	Ja	Beschreibung des Koordinatensystems, das seine Anwendung angibt.

Katalogsicht DB2GSE.ST_GEOMETRY_COLUMNS

In der Katalogsicht DB2GSE.ST_GEOMETRY_COLUMNS können Sie Informationen zu allen räumlichen Spalten in allen Tabellen der Datenbank ermitteln, die räumliche Daten enthalten.

Falls eine räumliche Spalte zusammen mit einem räumlichen Bezugssystem registriert wurde, können Sie in der Sicht außerdem den Namen und die numerische Kennung des räumlichen Bezugssystems feststellen. Weitere Informationen zu räumlichen Spalten erhalten Sie durch Abfragen der Katalogsicht SYSCAT.COLUMN.

Falls eine räumliche Spalte zusammen mit einem räumlichen Bezugssystem registriert wurde und für **bereiche_berechnen** ein Wert angegeben wurde, können Sie in

der Sicht außerdem Informationen zu den geografischen Bereichen für die räumliche Spalte sowie das Datum ihrer letzten Berechnung abrufen.

Eine Beschreibung der Sicht DB2GSE.ST_GEOMETRY_COLUMNS finden Sie in der folgenden Tabelle.

Tabelle 11. Spalten in der Katalogsicht DB2GSE.ST_GEOMETRY_COLUMNS

Name	Datentyp	Dateneingabe optional?	Inhalt
TABLE_SCHEMA	VARCHAR(128)	Nein	Name des Schemas, zu dem die Tabelle, die diese räumliche Spalte enthält, gehört.
TABLE_NAME	VARCHAR(128)	Nein	Name ohne Qualifikationsmerkmal der Tabelle, die diese räumliche Spalte enthält.
COLUMN_NAME	VARCHAR(128)	Nein	Name dieser räumlichen Spalte. Die Kombination der Werte aus TABLE_SCHEMA, TABLE_NAME und COLUMN_NAME bildet die eindeutige Kennzeichnung der Spalte.
TYPE_SCHEMA	VARCHAR(128)	Nein	Name des Schemas, zu dem der deklarierte Datentyp dieser räumlichen Spalte gehört. Dieser Name wird aus dem DB2-Katalog übernommen.
TYPE_NAME	VARCHAR(128)	Nein	Name ohne Qualifikationsmerkmal des deklarierten Datentyps dieser räumlichen Spalte. Dieser Name wird aus dem DB2-Katalog übernommen.
SRS_NAME	VARCHAR(128)	Ja	Name des räumlichen Bezugssystems, das dieser räumlichen Spalte zugeordnet ist. Wenn der Spalte kein räumliches Bezugssystem zugeordnet ist, hat SRS_NAME einen Nullwert.
SRS_ID	INTEGER	Ja	Numerische Kennung des räumlichen Bezugssystems, das dieser räumlichen Spalte zugeordnet ist. Wenn der Spalte kein räumliches Bezugssystem zugeordnet ist, hat SRS_ID einen Nullwert.
MIN_X	DOUBLE	Ja	Minimaler *x*-Wert aller räumlichen Werte in der Spalte.
MIN_Y	DOUBLE	Ja	Minimaler *y*-Wert aller räumlichen Werte in der Spalte.
MIN_Z	DOUBLE	Ja	Minimaler *z*-Wert aller räumlichen Werte in der Spalte.
MAX_X	DOUBLE	Ja	Maximaler *x*-Wert aller räumlichen Werte in der Spalte.
MAX_Y	DOUBLE	Ja	Maximaler *y*-Wert aller räumlichen Werte in der Spalte.
MAX_Z	DOUBLE	Ja	Maximaler *z*-Wert aller räumlichen Werte in der Spalte.
MIN_M	DOUBLE	Ja	Minimaler *m*-Wert aller räumlichen Werte in der Spalte.
MAX_M	DOUBLE	Ja	Maximaler *m*-Wert aller räumlichen Werte in der Spalte.
EXTENT_TIME	TIMESTAMP	Ja	Zeitmarke der letzten Berechnung der Bereiche.

Katalogsicht DB2GSE.ST_GEOCODER_PARAMETERS

In der Katalogsicht DB2GSE.ST_GEOCODER_PARAMETERS können Sie Informationen zu den Parametern von registrierten Geocodern abrufen.

Wenn Sie weitere Informationen zu den Parametern von Geocodern benötigen, fragen Sie die Katalogsicht SYSCAT.ROUTINEPARMS ab.

Eine Beschreibung der Spalten in dieser Sicht finden Sie in der folgenden Tabelle.

Tabelle 12. Spalten in der Katalogsicht DB2GSE.ST_GEOCODER_PARAMETERS

Name	Datentyp	Dateneingabe optional?	Inhalt
GEOCODER_NAME	VARCHAR(128)	Nein	Name des Geocoders, zu dem dieser Parameter gehört.
ORDINAL	SMALLINT	Nein	Position dieses Parameters (d. h. des in der Spalte PARAMETER_NAME angegebenen Parameters) in der Kennung der Funktion, die als der in der Spalte GEOCODER_NAME angegebene Geocoder dient. Die Kombination der Werte in den Spalten GEOCODER_NAME und ORDINAL ergibt die eindeutige Kennzeichnung dieses Parameters. Ein Datensatz in der Katalogsicht SYSCAT.ROUTINEPARMS enthält ebenfalls Informationen zu diesem Parameter. Dieser Datensatz enthält einen Wert, der in der Spalte ORDINAL der Katalogsicht SYSCAT.ROUTINEPARMS angezeigt wird. Dieser Wert ist mit dem Wert identisch, der in der Spalte ORDINAL der Sicht DB2GSE.ST_GEOCODER_PARAMETERS angezeigt wird.
PARAMETER_NAME	VARCHAR(128)	Ja	Name dieses Parameters. Wenn kein Name angegeben wurde, als die Funktion, zu der dieser Parameter gehört, erstellt wurde, enthält die Spalte PARAMETER_NAME keinen Wert. Dieser Inhalt der Spalte PARAMETER_NAME wird aus dem DB2-Katalog übernommen.
TYPE_SCHEMA	VARCHAR(128)	Nein	Name des Schemas, zu dem dieser Parameter gehört. Dieser Name wird aus dem DB2-Katalog übernommen.
TYPE_NAME	VARCHAR(128)	Nein	Name ohne Qualifikationsmerkmal des Datentyps für die Werte, die diesem Parameter zugeordnet sind. Dieser Name wird aus dem DB2-Katalog übernommen.

Tabelle 12. Spalten in der Katalogsicht DB2GSE.ST_GEOCODER_PARAMETERS (Forts.)

Name	Datentyp	Dateneingabe optional?	Inhalt
PARAMETER_DEFAULT	VARCHAR(2048)	Ja	Standardwert, der diesem Parameter zugeordnet sein soll. DB2 interpretiert diesen Wert als SQL-Ausdruck. Wenn der Wert in Anführungszeichen gesetzt ist, wird er als Zeichenfolge an den Geocoder übergeben. Andernfalls wird durch die Auswertung des SQL-Ausdrucks ermittelt, welchen Datentyp der Parameter bei der Übergabe an den Geocoder annimmt. Wenn die Spalte PARAMETER_DEFAULT einen Nullwert enthält, wird dieser Nullwert an den Geocoder übergeben. Dem Standardwert kann ein Wert in der Katalogsicht DB2GSE.ST_GEOCODING_PARAMETERS entsprechen. Außerdem können die Eingabeparameter für die Prozedur ST_RUN_GEOCODING einen entsprechenden Wert enthalten. Wenn einer der entsprechenden Werte vom Standardwert abweicht, wird der Standardwert durch den entsprechenden Wert außer Kraft gesetzt.
DESCRIPTION	VARCHAR(256)	Ja	Beschreibung des Parameters, die seine Anwendung angibt.

Katalogsicht DB2GSE.ST_GEOCODERS

Falls Sie den Benutzern weitere Geocoder zur Verfügung stellen wollen, müssen Sie diese Geocoder registrieren. Um Informationen zu registrierten Geocodern abzurufen, verwenden Sie die Katalogsicht DB2GSE.ST_GEOCODERS.

Informationen zu den Parametern der Geocoder können Sie durch Abfragen der Katalogsicht DB2GSE.ST_GEOCODER_PARAMETERS und der Katalogsicht SYSCAT.ROUTINEPARMS ermitteln. Wenn Sie Informationen zu Funktionen benötigen, die als Geocoder verwendet werden, erhalten Sie diese durch Abfragen der Katalogsicht SYSCAT.ROUTINES.

Eine Beschreibung der Spalten in der Sicht DB2GSE.ST_GEOCODERS finden Sie in der folgenden Tabelle.

Tabelle 13. Spalten in der Katalogsicht DB2GSE.ST_GEOCODERS

Name	Datentyp	Dateneingabe optional?	Inhalt
GEOCODER_NAME	VARCHAR(128)	Nein	Name dieses Geocoders. Er ist in der Datenbank eindeutig.
FUNCTION_SCHEMA	VARCHAR(128)	Nein	Name des Schemas, zu dem die Funktion gehört, die für diesen Geocoder verwendet wird.
FUNCTION_NAME	VARCHAR(128)	Nein	Name ohne Qualifikationsmerkmal der Funktion, die für diesen Geocoder verwendet wird.

Tabelle 13. Spalten in der Katalogsicht DB2GSE.ST_GEOCODERS (Forts.)

Name	Datentyp	Dateneingabe optional?	Inhalt
SPECIFIC_NAME	VARCHAR(128)	Nein	Spezifischer Name der Funktion, die für diesen Geocoder verwendet wird. Die Kombination der Werte aus FUNCTION_SCHEMA und SPECIFIC_NAME dient zur eindeutigen Kennzeichnung der Funktion, die für diesen Geocoder verwendet wird.
RETURN_TYPE_SCHEMA	VARCHAR(128)	Nein	Name des Schemas, zu dem der Datentyp für den Ausgabeparameter dieses Geocoders gehört. Dieser Name wird aus dem DB2-Katalog übernommen.
RETURN_TYPE_NAME	VARCHAR(128)	Nein	Name ohne Qualifikationsmerkmal des Datentyps für den Ausgabeparameter dieses Geocoders. Dieser Name wird aus dem DB2-Katalog übernommen.
VENDOR	VARCHAR(256)	Ja	Name des Lieferanten, der diesen Geocoder erstellt hat.
DESCRIPTION	VARCHAR(256)	Ja	Beschreibung des Geocoders, der seine Anwendung angibt.

Katalogsicht DB2GSE.ST_GEOCODING

Wenn Sie Geocodierungsoperationen definieren, werden die Einzelheiten Ihrer Einstellungen automatisch im Katalog von DB2 Spatial Extender aufgezeichnet. Um diese Einzelheiten zu ermitteln, können Sie die Katalogsichten DB2GSE.ST_GEOCODING und DB2GSE.ST_GEOCODING_PARAMETERS abfragen.

Die Katalogsicht DB2GSE.ST_GEOCODING, die im Abschnitt Tabelle 14 beschrieben ist, enthält Einzelheiten zu allen Einstellungen, beispielsweise die Anzahl der Datensätze, die ein Geocoder vor jedem Commit verarbeiten kann.

Die Katalogsicht DB2GSE.ST_GEOCODING_PARAMETERS enthält Einzelheiten, die für jeden Geocoder spezifisch sind. Beispielsweise der Mindestübereinstimmungsgrad von eingegebenen Adressen mit tatsächlichen Adressen, der Voraussetzung für die Geocodierung der Eingabe durch den Geocoder ist. Diese Mindestanforderung, die auch als Mindestübereinstimmungsquote bezeichnet wird, wird in der Katalogsicht DB2GSE.ST_GEOCODING_PARAMETERS aufgezeichnet.

Tabelle 14. Spalten in der Katalogsicht DB2GSE.ST_GEOCODING

Name	Datentyp	Dateneingabe optional?	Inhalt
TABLE_SCHEMA	VARCHAR(128)	Nein	Name des Schemas, zu dem die Tabelle gehört, die die in der Spalte COLUMN_NAME angegebene Spalte enthält.
TABLE_NAME	VARCHAR(128)	Nein	Name ohne Qualifikationsmerkmal des Schemas, das die in der Spalte COLUMN_NAME angegebene Spalte enthält.

Tabelle 14. Spalten in der Katalogsicht DB2GSE.ST_GEOCODING (Forts.)

Name	Datentyp	Dateneingabe optional?	Inhalt
COLUMN_NAME	VARCHAR(128)	Nein	Name der räumlichen Spalte, die gemäß den in dieser Katalogsicht angezeigten Spezifikationen gefüllt werden soll. Die Kombination der Werte aus TABLE_SCHEMA, TABLE_NAME und COLUMN_NAME dient zur eindeutigen Kennzeichnung der räumlichen Spalte.
GEOCODER_NAME	VARCHAR(128)	Nein	Name des Geocoders, der Daten für die in der Spalte COLUMN_NAME angegebene Spalte erzeugen soll. Einer räumlichen Spalte kann jeweils nur ein Geocoder zugeordnet sein.
MODE	VARCHAR(128)	Nein	Modus für den Geocodierungsprozess: **BATCH** Nur die Geocodierung im Stapelbetrieb ist aktiviert. **AUTO** Die automatische Geocodierung ist definiert und aktiviert. **INVALID** Es wurde eine Inkonsistenz in den räumlichen Katalogtabellen festgestellt. Der Geocodierungseintrag ist ungültig.
SOURCE_COLUMNS	VARCHAR(10000)	Ja	Namen der Tabellenspalten, die für die automatische Geocodierung definiert wurden. Bei jeder Aktualisierung dieser Spalten fordert ein Trigger die Geocodierung der aktualisierten Daten durch den Geocoder an.
WHERE_CLAUSE	VARCHAR(10000)	Ja	Suchbedingung innerhalb einer Klausel WHERE. Diese Bedingung gibt an, dass der Geocoder bei einer Ausführung im Stapelmodus nur Daten in einer angegebenen Untermenge von Datensätzen geocodieren soll.
COMMIT_COUNT	INTEGER	Ja	Die Anzahl der Zeilen, die bei der Geocodierung im Stapelbetrieb verarbeitet werden sollen, bevor ein Commit abgesetzt wird. Falls der Wert in der Spalte COMMIT_COUNT 0 ist oder kein Wert angegeben ist, werden keine Commits abgesetzt.

Katalogsicht DB2GSE.ST_GEOCODING_PARAMETERS

Wenn Sie Geocodierungsoperationen für einen bestimmten Geocoder definieren, sind die geocoderspezifischen Aspekte der Einstellungen über die Katalogsicht DB2GSE.ST_GEOCODING_PARAMETERS abrufbar.

Eine Operation kann beispielsweise darin bestehen, eingegebene Adressen mit Bezugsdaten zu vergleichen.

Wenn Sie Operationen für einen Geocoder definieren, geben Sie an, wie groß der Übereinstimmungsgrad, der auch als Mindestübereinstimmungsquote bezeichnet wird, sein soll. Ihre Spezifikation wird dann im Katalog aufgezeichnet.

Um die geocoderspezifischen Aspekte der Einstellungen für Geocodierungsoperationen zu ermitteln, fragen Sie die Katalogsicht DB2GSE.ST_GEOCODING_PARAMETERS ab. Diese Sicht ist in der folgenden Tabelle beschrieben.

In der Katalogsicht DB2GSE.ST_GEOCODER_PARAMETERS sind bestimmte Standardwerte für Konfigurationen von Geocodierungsoperationen verfügbar. Werte in der Spalte DB2GSE.ST_GEOCODING_PARAMETERS setzen die Standardwerte außer Kraft.

Tabelle 15. Spalten in der Katalogsicht DB2GSE.ST_GEOCODING_PARAMETERS

Name	Datentyp	Dateneingabe optional?	Inhalt
TABLE_SCHEMA	VARCHAR(128)	Nein	Name des Schemas, zu dem die Tabelle gehört, die die in der Spalte COLUMN_NAME angegebene Spalte enthält.
TABLE_NAME	VARCHAR(128)	Nein	Name ohne Qualifikationsmerkmal der Tabelle, die die räumliche Spalte enthält.
COLUMN_NAME	VARCHAR(128)	Nein	Name der räumlichen Spalte, die gemäß den in dieser Katalogsicht angezeigten Spezifikationen gefüllt werden soll. Die Kombination der Werte aus TABLE_SCHEMA, TABLE_NAME und COLUMN_NAME dient zur eindeutigen Kennzeichnung dieser räumlichen Spalte.
ORDINAL	SMALLINT	Nein	Position dieses Parameters (d. h. des in der Spalte PARAMETER_NAME angegebenen Parameters) in der Kennung der Funktion, die als Geocoder für die in der Spalte COLUMN_NAME angegebene Spalte verwendet wird. Ein Datensatz in der Katalogsicht SYSCAT.ROUTINEPARMS enthält ebenfalls Informationen zu diesem Parameter. Dieser Datensatz enthält einen Wert, der in der Spalte ORDINAL der Katalogsicht SYSCAT.ROUTINEPARMS angezeigt wird. Dieser Wert ist mit dem Wert identisch, der in der Spalte ORDINAL der Sicht DB2GSE.ST_GEOCODING_PARAMETERS angezeigt wird.
PARAMETER_NAME	VARCHAR(128)	Ja	Name eines Parameters in der Definition des Geocoders. Wenn bei der Definition des Geocoders kein Name angegeben wurde, enthält die Spalte PARAMETER_NAME einen Nullwert. Dieser Inhalt der Spalte PARAMETER_NAME wird aus dem DB2-Katalog übernommen.

Tabelle 15. Spalten in der Katalogsicht DB2GSE.ST_GEOCODING_PARAMETERS (Forts.)

Name	Datentyp	Dateneingabe optional?	Inhalt
PARAMETER_VALUE	VARCHAR(2048)	Ja	Der Wert, der diesem Parameter zugeordnet ist. DB2 interpretiert diesen Wert als SQL-Ausdruck. Wenn der Wert in Anführungszeichen gesetzt ist, wird er als Zeichenfolge an den Geocoder übergeben. Andernfalls wird durch die Auswertung des SQL-Ausdrucks ermittelt, welchen Datentyp der Parameter bei der Übergabe an den Geocoder annimmt. Wenn die Spalte PARAMETER_VALUE einen Nullwert enthält, wird dieser Nullwert an den Geocoder übergeben. Die Spalte PARAMETER_VALUE entspricht der Spalte PARAMETER_DEFAULT in der Katalogsicht DB2GSE.ST_GEOCODER_PARAMETERS. Falls die Spalte PARAMETER_VALUE einen Wert enthält, setzt dieser Wert den Standardwert in der Spalte PARAMETER_DEFAULT außer Kraft. Enthält die Spalte PARAMETER_VALUE keinen Wert, wird der Standardwert verwendet.

Katalogsicht DB2GSE.ST_SIZINGS

In der Katalogsicht DB2GSE.ST_SIZINGS können Sie Informationen zu unterstützten Variablen und deren maximaler Länge abrufen:

Diese Katalogsicht gibt die folgenden Informationen zurück:

- Alle durch DB2 Spatial Extender unterstützten Variablen, beispielsweise den Namen des Koordinatensystems, den Namen des Geocoders und Variablen, denen WKT-Darstellungen (WKT = Well-Known Text) von räumlichen Daten zugeordnet werden können.
- Die zulässige Höchstlänge (sofern bekannt) von Werten, die diesen Variablen zugeordnet sind, beispielsweise die zulässigen Höchstlängen für die Namen von Koordinatensystemen, die Namen von Geocodern und von WKT-Darstellungen räumlicher Daten.

Eine Beschreibung der Spalten in dieser Sicht finden Sie in der folgenden Tabelle.

Tabelle 16. Spalten in der Katalogsicht DB2GSE.ST_SIZINGS

Name	Datentyp	Dateneingabe optional?	Inhalt
VARIABLE_NAME	VARCHAR(128)	Nein	Begriff, der eine Variable kennzeichnet. Der Begriff ist in der Datenbank eindeutig.

Tabelle 16. Spalten in der Katalogsicht DB2GSE.ST_SIZINGS (Forts.)

Name	Datentyp	Dateneingabe optional?	Inhalt
SUPPORTED_VALUE	INTEGER	Ja	Zulässige Höchstlänge der Werte, die der in der Spalte VARIABLE_NAME angezeigten Variablen zugeordnet sind. Gültige Werte in der Spalte SUPPORTED_VALUE: **Numerischer Wert ungleich 0** Die zulässige Höchstlänge von Werten, die dieser Variablen zugeordnet sind. **0** Entweder ist jede beliebige Länge zulässig, oder die zulässige Länge kann nicht ermittelt werden. **NULL** DB2 Spatial Extender unterstützt diesen Wert nicht.
DESCRIPTION	VARCHAR(128)	Ja	Beschreibung dieser Variablen.

Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS

Sie können die Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS abfragen, um Informationen zu registrierten räumlichen Bezugssystemen abzurufen.

DB2 Spatial Extender führt zu den folgenden Zeitpunkten automatisch eine Registrierung von räumlichen Bezugssystemen im Spatial Extender-Katalog durch:

- Wenn Sie eine Datenbank für räumliche Operationen aktivieren, werden fünf räumliche Standardbezugssysteme registriert.
- Wenn Benutzer zusätzliche räumliche Bezugssysteme erstellen.

Damit Sie die Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS optimal nutzen können, müssen Sie zunächst einmal wissen, dass jedem räumlichen Bezugssystem ein Koordinatensystem zugeordnet ist. Das räumliche Bezugssystem ist zum einen dazu gedacht, Koordinaten, die aus dem Koordinatensystem abgeleitet wurden, in Werte umzuwandeln, die von DB2 mit einem Maximum an Effizienz verarbeitet werden können. Zum anderen soll es die maximale mögliche Ausdehnung des Gebietes definieren, auf das sich diese Koordinaten beziehen können.

Um Name und Typ des Koordinatensystems zu ermitteln, das einem bestimmten räumlichen Bezugssystem zugeordnet ist, fragen Sie die Spalten COORDSYS_NAME und COORDSYS_TYPE der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS ab. Wenn Sie weitere Informationen zum Koordinatensystem benötigen, fragen Sie die Katalogsicht DB2GSE.ST_COORDINATE_SYSTEMS ab.

Tabelle 17. Spalten in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS

Name	Datentyp	Dateneingabe optional?	Inhalt
SRS_NAME	VARCHAR(128)	Nein	Name des räumlichen Bezugssystems. Dieser Name ist in der Datenbank eindeutig.

Tabelle 17. Spalten in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS (Forts.)

Name	Datentyp	Dateneingabe optional?	Inhalt
SRS_ID	INTEGER	Nein	Numerische Kennung des räumlichen Bezugssystems. Jedes räumliche Bezugssystem hat eine eindeutige numerische Kennung. Räumliche Funktionen geben räumliche Bezugssysteme durch deren numerische Kennungen und nicht durch deren Namen an.
X_OFFSET	DOUBLE	Nein	Offset, das von allen X-Koordinaten einer Geometrie subtrahiert werden soll. Die Subtraktion ist ein Schritt in dem Prozess, mit dem die Koordinaten der Geometrie in Werte umgewandelt werden, die DB2 mit einem Maximum an Effizienz verarbeiten kann. In einem anschließenden Schritt wird der bei der Subtraktion entstehende Wert mit dem Maßstabsfaktor multipliziert, der in der Spalte X_SCALE angegeben ist.
X_SCALE	DOUBLE	Nein	Maßstabsfaktor, mit dem das Ergebnis multipliziert werden soll, das durch die Subtraktion eines Offsets von einer X-Koordinate entsteht. Dieser Faktor ist mit dem Wert identisch, der in der Spalte Y_SCALE angezeigt wird.
Y_OFFSET	DOUBLE	Nein	Offset, das von allen Y-Koordinaten einer Geometrie subtrahiert werden soll. Die Subtraktion ist ein Schritt in dem Prozess, mit dem die Koordinaten der Geometrie in Werte umgewandelt werden, die DB2 mit einem Maximum an Effizienz verarbeiten kann. In einem anschließenden Schritt wird der bei der Subtraktion entstehende Wert mit dem Maßstabsfaktor multipliziert, der in der Spalte Y_SCALE angegeben ist.
Y_SCALE	DOUBLE	Nein	Maßstabsfaktor, mit dem das Ergebnis multipliziert werden soll, das durch die Subtraktion eines Offsets von einer Y-Koordinate entsteht. Dieser Faktor ist mit dem Wert identisch, der in der Spalte X_SCALE angezeigt wird.
Z_OFFSET	DOUBLE	Nein	Offset, das von allen Z-Koordinaten einer Geometrie subtrahiert werden soll. Die Subtraktion ist ein Schritt in dem Prozess, mit dem die Koordinaten der Geometrie in Werte umgewandelt werden, die DB2 mit einem Maximum an Effizienz verarbeiten kann. In einem anschließenden Schritt wird der bei der Subtraktion entstehende Wert mit dem Maßstabsfaktor multipliziert, der in der Spalte Z_SCALE angegeben ist.
Z_SCALE	DOUBLE	Nein	Maßstabsfaktor, mit dem das Ergebnis multipliziert werden soll, das durch die Subtraktion eines Offsets von einer Z-Koordinate entsteht.

Tabelle 17. Spalten in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS (Forts.)

Name	Datentyp	Dateneingabe optional?	Inhalt
M_OFFSET	DOUBLE	Nein	Offset, das von allen Bemaßungen subtrahiert werden soll, die einer Geometrie zugeordnet sind. Die Subtraktion ist ein Schritt in dem Prozess, mit dem die Bemaßungen in Werte umgewandelt werden, die DB2 mit einem Maximum an Effizienz verarbeiten kann. In einem anschließenden Schritt wird der bei der Subtraktion entstehende Wert mit dem Maßstabsfaktor multipliziert, der in der Spalte M_SCALE angegeben ist.
M_SCALE	DOUBLE	Nein	Maßstabsfaktor, mit dem das Ergebnis multipliziert werden soll, das durch die Subtraktion eines Offsets von einer Bemaßung entsteht.
MIN_X	DOUBLE	Nein	Kleinstmöglicher Wert für X-Koordinaten in den Geometrien, auf die dieses räumliche Bezugssystem angewendet wird. Dieser Wert wird von den Werten in den Spalten X_OFFSET und X_SCALE abgeleitet.
MAX_X	DOUBLE	Nein	Größtmöglicher Wert für X-Koordinaten in den Geometrien, auf die dieses räumliche Bezugssystem angewendet wird. Dieser Wert wird von den Werten in den Spalten X_OFFSET und X_SCALE abgeleitet.
MIN_Y	DOUBLE	Nein	Kleinstmöglicher Wert für Y-Koordinaten in den Geometrien, auf die dieses räumliche Bezugssystem angewendet wird. Dieser Wert wird von den Werten in den Spalten Y_OFFSET und Y_SCALE abgeleitet.
MAX_Y	DOUBLE	Nein	Größtmöglicher Wert für Y-Koordinaten in den Geometrien, auf die dieses räumliche Bezugssystem angewendet wird. Dieser Wert wird von den Werten in den Spalten Y_OFFSET und Y_SCALE abgeleitet.
MIN_Z	DOUBLE	Nein	Kleinstmöglicher Wert für Z-Koordinaten in Geometrien, auf die dieses räumliche Bezugssystem angewendet wird. Dieser Wert wird von den Werten in den Spalten Z_OFFSET und Z_SCALE abgeleitet.
MAX_Z	DOUBLE	Nein	Größtmöglicher Wert für Z-Koordinaten in Geometrien, auf die dieses räumliche Bezugssystem angewendet wird. Dieser Wert wird von den Werten in den Spalten Z_OFFSET und Z_SCALE abgeleitet.
MIN_M	DOUBLE	Nein	Kleinstmöglicher Wert für Bemaßungen, die mit Geometrien gespeichert werden können, auf die dieses räumliche Bezugssystem angewendet wird. Dieser Wert wird von den Werten in den Spalten M_OFFSET und M_SCALE abgeleitet.
MAX_M	DOUBLE	Nein	Größtmöglicher Wert für Bemaßungen, die mit Geometrien gespeichert werden können, auf die dieses räumliche Bezugssystem angewendet wird. Dieser Wert wird von den Werten in den Spalten M_OFFSET und M_SCALE abgeleitet.
COORDSYS_NAME	VARCHAR(128)	Nein	Der kennzeichnende Name des Koordinatensystems, auf dem dieses räumliche Bezugssystem basiert.
COORDSYS_TYPE	VARCHAR(128)	Nein	Der Typ des Koordinatensystems, auf dem dieses räumliche Bezugssystem basiert.

Tabelle 17. Spalten in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS (Forts.)

Name	Datentyp	Dateneingabe optional?	Inhalt
ORGANIZATION	VARCHAR(128)	Ja	Der Name der Organisation (z. B. eine Standardisierungsorganisation), die das Koordinatensystem, auf dem dieses räumliche Bezugssystem basiert, definiert hat. In der Spalte ORGANIZATION ist kein Wert angegeben, wenn die Spalte ORGANIZATION_COORSYS_ID keinen Wert enthält.
ORGANIZATION_ COORDSYS_ID	INTEGER	Ja	Der Name der Organisation (z. B. eine Standardisierungsorganisation), die das Koordinatensystem, auf dem dieses räumliche Bezugssystem basiert, definiert hat. In der Spalte ORGANIZATION_COORDSYS_ID ist kein Wert angegeben, wenn die Spalte ORGANIZATION keinen Wert enthält.
DEFINITION	VARCHAR(2048)	Nein	WKT-Darstellung (WKT = Well-Known Text) der Definition des Koordinatensystems.
DESCRIPTION	VARCHAR(256)	Ja	Beschreibung des räumlichen Bezugssystems.

Katalogsicht DB2GSE.ST_UNITS_OF_MEASURE

Verwenden Sie die Katalogsicht DB2GSE.ST_UNITS_OF_MEASURE, um zu bestimmen, welche Maßeinheiten zur Auswahl stehen.

Bestimmte räumliche Funktionen akzeptieren Werte oder geben Werte zurück, die einen spezifischen Abstand angeben. In manchen Fällen können Sie die Einheit auswählen, in der dieser Abstand ausgedrückt wird. Die Funktion ST_Distance gibt beispielsweise den Mindestabstand zwischen zwei angegebenen Geometrien zurück. Manchmal kann es sinnvoll sein, wenn die Funktion ST_Distance den Abstand in Form von Meilen zurückgibt. In anderen Fällen ist es möglicherweise besser, wenn der Abstand in Metern ausgedrückt wird.

Eine Beschreibung der Spalten in dieser Sicht finden Sie in der folgenden Tabelle.

Tabelle 18. Spalten in der Sicht DB2GSE.ST_UNITS_OF_MEASURE

Name	Datentyp	Dateneingabe optional?	Inhalt
UNIT_NAME	VARCHAR(128)	Nein	Name der Maßeinheit. Dieser Name ist in der Datenbank eindeutig.
UNIT_TYPE	VARCHAR(128)	Nein	Typ der Maßeinheit. Gültige Werte: **LINEAR** Die Maßeinheit ist linear. **ANGULAR** Die Maßeinheit ist winklig.

Tabelle 18. Spalten in der Sicht DB2GSE.ST_UNITS_OF_MEASURE (Forts.)

Name	Datentyp	Dateneingabe optional?	Inhalt
CONVERSION_FACTOR	DOUBLE	Nein	Numerischer Wert, mit dem diese Maßeinheit in ihre Basiseinheit umgewandelt wird. Die Basiseinheit für lineare Maßeinheiten ist METER. Die Basiseinheit für winklige Maßeinheiten ist RADIAN (Radiant). Die Basiseinheit selbst hat den Umwandlungsfaktor 1,0.
DESCRIPTION	VARCHAR(256)	Ja	Beschreibung der Maßeinheit.

Katalogsicht DB2GSE.SPATIAL_REF_SYS

Verwenden Sie die Katalogsicht DB2GSE.SPATIAL_REF_SYS, um Informationen zu räumlichen Bezugssystemen abzurufen, die in DB2 Spatial Extender registriert sind.

Wichtig: Diese Katalogsicht ist veraltet und wurde durch die „Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS" auf Seite 128 ersetzt.

Eine Beschreibung der Spalten in dieser Sicht finden Sie in der folgenden Tabelle.

Tabelle 19. Spalten in der Katalogsicht DB2GSE.SPATIAL_REF_SYS

Name	Datentyp	Dateneingabe optional?	Inhalt
SRID	INTEGER	Nein	Benutzerdefinierte Kennung für dieses räumliche Bezugssystem.
SR_NAME	VARCHAR(64)	Nein	Name dieses räumlichen Bezugssystems.
CSID	INTEGER	Nein	Numerische Kennung für das Koordinatensystem, das diesem räumlichen Bezugssystem zugrunde liegt.
CS_NAME	VARCHAR(64)	Nein	Name des Koordinatensystems, das diesem räumlichen Bezugssystem zugrunde liegt.
AUTH_NAME	VARCHAR(256)	Ja	Name der Organisation, die die Standards für dieses räumliche Bezugssystem festlegt.
AUTH_SRID	INTEGER	Ja	Die Kennung, die die in der Spalte AUTH_NAME angegebene Organisation diesem räumlichen Bezugssystem zuordnet.
SRTEXT	VARCHAR(2048)	Nein	Anmerkungstext für dieses räumliche Bezugssystem.
FALSEX	FLOAT	Nein	Eine Zahl, die von einem negativen X-Koordinatenwert subtrahiert wird, um auf diese Weise eine nicht negative Zahl zu erhalten (eine positive Zahl oder null).
FALSEY	FLOAT	Nein	Eine Zahl, die von einem negativen Y-Koordinatenwert subtrahiert wird, um auf diese Weise eine nicht negative Zahl zu erhalten (eine positive Zahl oder null).
XYUNITS	FLOAT	Nein	Eine Zahl, die bei Multiplikation mit einer dezimalen X-Koordinate oder einer dezimalen Y-Koordinate eine ganze Zahl ergibt, die als 32-Bit-Datenelement gespeichert werden kann.

Tabelle 19. Spalten in der Katalogsicht DB2GSE.SPATIAL_REF_SYS (Forts.)

Name	Datentyp	Dateneingabe optional?	Inhalt
FALSEZ	FLOAT	Nein	Eine Zahl, die von einem negativen Z-Koordinatenwert subtrahiert wird, um auf diese Weise eine nicht negative Zahl zu erhalten (eine positive Zahl oder null).
ZUNITS	FLOAT	Nein	Eine Zahl, die bei Multiplikation mit einer dezimalen Z-Koordinate eine ganze Zahl ergibt, die als 32-Bit-Datenelement gespeichert werden kann.
FALSEM	FLOAT	Nein	Eine Zahl, die von einer negativen Bemaßung subtrahiert wird, um auf diese Weise eine nicht negative Zahl zu erhalten (eine positive Zahl oder null).
MUNITS	FLOAT	Nein	Eine Zahl, die bei Multiplikation mit einer dezimalen Bemaßung eine ganze Zahl ergibt, die als 32-Bit-Datenelement gespeichert werden kann.

Kapitel 16. Befehle von DB2 Spatial Extender

Verwenden Sie diese Befehle, um DB2 Spatial Extender einzurichten und Projekte zu entwickeln, die mit räumlichen Daten arbeiten.

Befehle zur Konfiguration von DB2 Spatial Extender und Entwicklung von Projekten aufrufen

Über den DB2 Spatial Extender-Befehlszeilenprozessor (CLP) namens "db2se" können Sie DB2 Spatial Extender konfigurieren und Projekte erstellen, die räumliche Daten verwenden. Der folgende Abschnitt erläutert, wie Sie Befehle von DB2 Spatial Extender mit dem Befehlszeilenprozessor **db2se** ausführen.

Voraussetzungen

Damit Sie **db2se**-Befehle absetzen können, müssen Sie dazu berechtigt sein. Informationen über die erforderliche Berechtigung für einen bestimmten Befehl finden Sie im Abschnitt zu den Berechtigungen für jeden Befehl.

Geben Sie **db2se**-Befehle über eine Eingabeaufforderung des Betriebssystems ein.

So können Sie ermitteln, welche db2se-Befehle und -Parameter verfügbar sind:

- Geben Sie `db2se` oder `db2se -h` ein. Drücken Sie anschließend die Eingabetaste. Daraufhin wird eine Liste der db2se-Unterbefehle angezeigt.
- Geben Sie `db2se` und einen Befehl oder `db2se` und einen Befehl gefolgt von dem Parameter **-h** ein. Drücken Sie abschließend die Eingabetaste. Daraufhin wird die für den Unterbefehl erforderliche Syntax angezeigt. Für diese Syntax gilt Folgendes:
  - Vor jedem Parameter steht ein Silbentrennungsstrich. Auf den Parameter folgt ein Platzhalter für den Parameterwert.
  - In eckigen Klammern angezeigte Parameter sind optional. Die übrigen Parameter sind erforderlich.

Um einen db2se-Befehl abzusetzen, geben Sie `db2se` ein. Anschließend geben Sie einen Befehl sowie die für den Befehl erforderlichen Parameter und Parameterwerte ein. Drücken Sie abschließend die Eingabetaste.

Möglicherweise müssen Sie die Benutzer-ID und das Kennwort eingeben, mit dem Sie auf die soeben angegebene Datenbank zugreifen können. Geben Sie beispielsweise die Benutzer-ID und das Kennwort ein, wenn Sie mit einem anderen als Ihrem eigenen Benutzereintrag eine Verbindung zur Datenbank herstellen wollen. Vor der Benutzer-ID müssen Sie immer den Parameter **-userId** und vor dem Kennwort immer den Parameter **-pw** angeben. Wenn Sie Benutzer-ID und Kennwort nicht angeben, werden standardmäßig die aktuellen Werte für Benutzer-ID und Kennwort verwendet.

Bei von Ihnen eingegebenen Werten wird die Groß-/Kleinschreibung standardmäßig nicht beachtet. Wenn Sie Werte angeben möchten, bei denen die Groß-/Kleinschreibung beachtet werden soll, setzen Sie diese in doppelte Anführungszeichen. Wenn Sie zum Beispiel den Tabellennamen mytable in Kleinbuchstaben angeben wollen, geben Sie Folgendes ein: "mytable".

Möglicherweise müssen Sie ein Escapezeichen vor den Anführungszeichen verwenden, damit sie nicht durch die Eingabeaufforderung (Shell) interpretiert werden. Geben Sie zum Beispiel \"mytable\" an um auf die Tabelle mit dem Namen "mytable" zu verweisen. Wenn ein Wert, bei dem die Groß-/Kleinschreibung beachtet werden muss, durch einen anderen solchen Wert qualifiziert wird, müssen Sie die beiden Werte einzeln begrenzen. Beispiel: "myschema"." mytable". Schließen Sie die Zeichenfolgen in doppelte Anführungszeichen ein. Beispiel: "select * from newtable".

Wichtig: Schließen Sie nicht die gesamte Liste der Parameternamen und -werte in Anführungszeichen ein.

Die meisten **db2se**-Befehle, mit Ausnahme des Befehls **db2se shape_info**, führen eine entsprechende gespeicherte Prozedur auf dem DB2-Server aus. Dieser Befehl wird auf dem Client ausgeführt und greift möglicherweise über eine für räumliche Operationen aktivierte Datenbank auf Informationen von Koordinatensystemen und räumlichen Bezugssystemen zu.

Die Befehle **db2se import_shape** und **db2se export_shape** können ebenfalls auf dem Client ausgeführt werden. Dies ist nützlich, da es den Zugriff auf lokale Dateien auf dem Client ermöglicht.

Befehl 'db2se alter_cs'

Der Befehl **db2se alter_cs** aktualisiert die Definition eines Koordinatensystems.

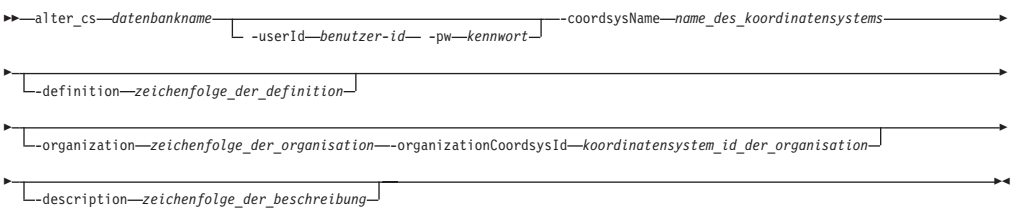
Mithilfe dieses Befehls können Sie die Definitionszeichenfolge, den Organisationsnamen, die Koordinatensystem-ID der Organisation und die Beschreibung in einem Koordinatensystem, das über den Befehl **db2se create_cs** oder die gespeicherte Prozedur ST_CREATE_COORDSYS definiert wird, aktualisieren. Die Informationen zum Koordinatensystem sind in der Katalogsicht DB2GSE.ST_COORDINATE_SYSTEMS abrufbar.

Berechtigung

Die Benutzer-ID muss über die Berechtigungen DBADM und DATAACCESS für die Datenbank verfügen, die für räumliche Operationen aktiviert ist, um diesen Befehl auszuführen.

Befehlssyntax

Befehl 'db2se alter_cs'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie das Koordinatensystem ändern möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-coordsysName *name_des_koordinatensystems*
Dieser Parameter gibt die eindeutige Kennzeichnung des Koordinatensystems an. Die maximale Länge für diesen Parameter beträgt 128 Zeichen.

Der Wert für *name_des_koordinatensystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-definition *zeichenfolge_der_definition*
Dieser Parameter definiert das Koordinatensystem. Normalerweise stellt der Lieferant des Koordinatensystems die Informationen für diesen Parameter zur Verfügung. Die maximale Länge für diesen Parameter beträgt 2048 Zeichen.

-organization *zeichenfolge_der_organisation*
Definiert den Namen der Organisation, die das Koordinatensystem definiert und seine Definition zur Verfügung gestellt hat (z. B. "European Petroleum Survey Group (EPSG)"). Die maximale Länge für diesen Parameter beträgt 128 Zeichen.

Die Kombination der Parameter *zeichenfolge_der_organisation* und *koordinatensystem_id_der_organisation* gibt das Koordinatensystem eindeutig an.

-organizationCoordsysId *koordinatensystem_id_der_organisation*
Dieser Parameter gibt eine numerische Kennung an. Dieser Wert wird von der im Parameter *zeichenfolge_der_organisation* angegebenen Einheit zugeordnet. Er ist nicht zwangsläufig gegenüber allen anderen Koordinatensystemen eindeutig.

Die Kombination der Parameter *zeichenfolge_der_organisation* und *koordinatensystem_id_der_organisation* gibt das Koordinatensystem eindeutig an.

-description *zeichenfolge_der_beschreibung*
Dieser Parameter beschreibt das Koordinatensystem, indem seine Anwendung erläutert wird. Die maximale Länge für diesen Parameter beträgt 256 Zeichen.

Hinweise zur Verwendung

Falls Sie die Definition des Koordinatensystems mithilfe dieses Befehls ändern und räumliche Daten vorhanden sind, denen ein räumliches Bezugssystem zugeordnet ist, das auf diesem Koordinatensystem basiert, können die räumlichen Daten unbeabsichtigterweise geändert werden. Wenn räumliche Daten betroffen sind, müssen Sie sicherstellen, dass die geänderten räumlichen Daten weiterhin exakt und gültig sind.

Beispiel

Im folgenden Beispiel wird die Definition eines Koordinatensystems namens MYCOORDSYS mit einen neuen Organisationsnamen aktualisiert.

```
db2se alter_cs mydb -coordsysName mycoordsys -organization myNeworganizationb
```

Befehl 'db2se alter_srs'

Der Befehl **db2se alter_srs** aktualisiert die Definition eines räumlichen Bezugssystems.

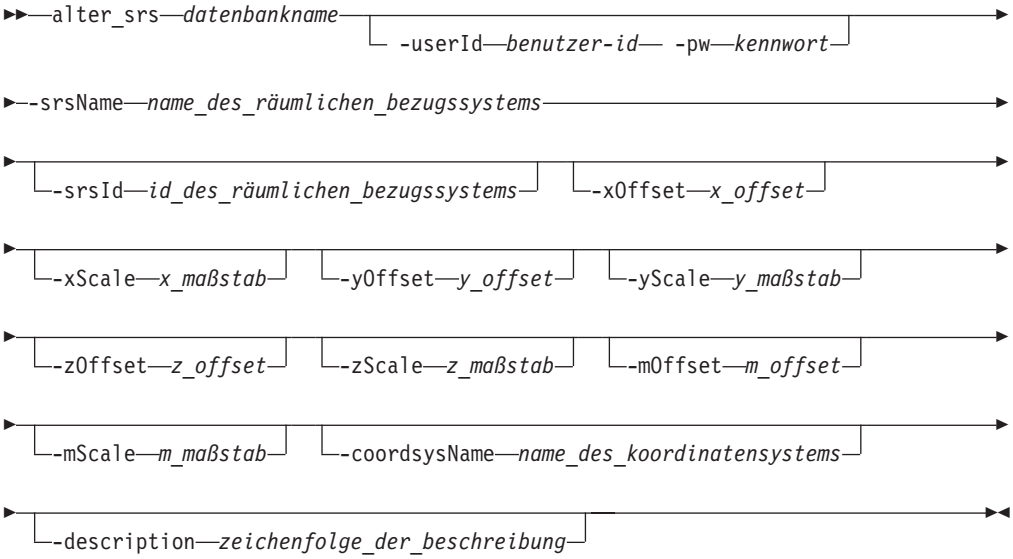
Mithilfe dieses Befehls können Sie das Offset, den Maßstab oder den Namen des Koordinatensystems eines räumlichen Bezugssystems ändern. Die Informationen zum Koordinatensystem sind in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS abrufbar.

Berechtigung

Die Benutzer-ID muss über die Berechtigungen DBADM und DATAACCESS für die Datenbank verfügen, die für räumliche Operationen aktiviert ist, um diesen Befehl auszuführen.

Befehlssyntax

Befehl 'db2se alter_srs'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie die Definition eines räumlichen Bezugssystems aktualisieren möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-srsName *name_des_räumlichen_bezugssystems*
Gibt das räumliche Bezugssystem an, das Sie aktualisieren möchten. Der Wert für *name_des_räumlichen_bezugssystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-srsId *id_des_räumlichen_bezugssystems*
Dieser Parameter gibt die eindeutige Kennung des räumlichen Bezugssystems an. Diese Kennung wird für unterschiedliche räumliche Funktionen als Eingabeparameter verwendet.

-xOffset *x_offset*
Dieser Parameter gibt den Offset für alle X-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor *x_maßstab* angewendet wird.

-xScale *x_maßstab*
Dieser Parameter gibt den Maßstabsfaktor für alle X-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset *x_offset* subtrahiert wurde.

-yOffset *y_offset*
Dieser Parameter gibt den Offset für alle Y-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor *y_maßstab* angewendet wird.

-yScale *y_maßstab*
Dieser Parameter gibt den Maßstabsfaktor für alle Y-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset *y_offset* subtrahiert wurde. Dieser Maßstabsfaktor muss mit dem Wert für *x_maßstab* identisch sein.

-zOffset *z_offset*
Dieser Parameter gibt den Offset für alle Z-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor *z_maßstab* angewendet wird.

-zScale *z_maßstab*
Dieser Parameter gibt den Maßstabsfaktor für alle Z-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset *z_offset* subtrahiert wurde.

-mOffset *m_offset*
Dieser Parameter gibt den Offset für alle M-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor *m_maßstab* angewendet wird.

-mScale *m_maßstab*
Dieser Parameter gibt den Maßstabsfaktor für alle M-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Um-

wandlung von externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset *m_offset* subtrahiert wurde.

-coordsysName *name_des_koordinatensystems*
Dieser Parameter ist die eindeutige Kennzeichnung des Koordinatensystems, auf dem dieses räumliche Bezugssystem basiert. Das Koordinatensystem muss in der Sicht DB2GSE.ST_COORDINATE_SYSTEMS aufgeführt sein.

Der Wert für *name_des_koordinatensystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen. Die maximale Länge für diesen Parameter beträgt 128 Zeichen.

-description *zeichenfolge_der_beschreibung*
Dieser Parameter beschreibt das Koordinatensystem, indem seine Anwendung erläutert wird. Die maximale Länge für diesen Parameter beträgt 256 Zeichen.

Hinweise zur Verwendung

Wenn Sie mit diesem Befehl die Parameter für Offset und Maßstab oder den Parameter 'name_des_koordinatensystems' des räumlichen Bezugssystems ändern und bereits räumliche Daten vorhanden sind, die diesem räumlichen Bezugssystem zugeordnet sind, werden diese räumlichen Daten möglicherweise unbeabsichtigt geändert. Wenn räumliche Daten betroffen sind, müssen Sie sicherstellen, dass die geänderten räumlichen Daten weiterhin exakt und gültig sind.

Beispiel

Im folgenden Beispiel wird ein räumliches Bezugssystem namens MYSRS durch eine andere Beschreibung geändert.

```
db2se alter_srs mydb -srsName mysrs -description
  "Dies ist mein eigenes räumliches Bezugssystem."
```

Im folgenden Beispiel wird ein räumliches Bezugssystem namens MYSRS_35 durch einen anderen Wert für xOffset und eine andere Beschreibung geändert, weil es keine räumlichen Daten gibt, die MYSRS_35 verwenden. Wenn räumliche Daten MYSRS_35 verwenden, werden diese Daten unbrauchbar.

```
db2se alter_srs mydb -srsName mysrs_35 -xOffset 35
        -description "Dies ist mein eigenes räumliches Bezugssystem mit xOffset=35."
```

Befehl 'db2se create_cs'

Der Befehl **db2se create_cs** erstellt ein Koordinatensystem.

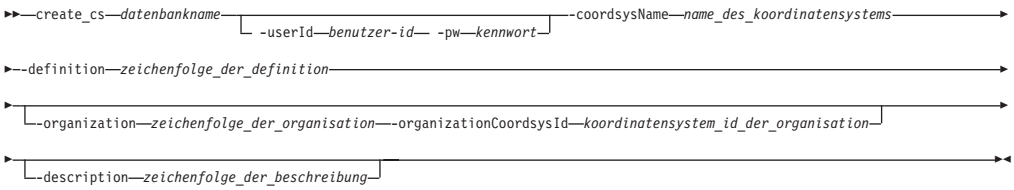
Dieser Befehl speichert Informationen zu einem neuen Koordinatensystem in der Datenbank. Die Informationen zum Koordinatensystem sind in der Katalogsicht DB2GSE.ST_COORDINATE_SYSTEMS abrufbar.

Berechtigung

Die Benutzer-ID muss über die Berechtigungen DBADM und DATAACCESS für die Datenbank verfügen, die für räumliche Operationen aktiviert ist, um diesen Befehl auszuführen.

Befehlssyntax

Befehl 'db2se create_cs'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Der Name der Datenbank, für die Sie das Koordinatensystem erstellen möchten.

-userId *benutzer-id*
Der Name der Datenbankbenutzer-ID, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Das Kennwort für *benutzer-id*.

-coordsysName *name_des_koordinatensystems*
Dieser Parameter gibt die eindeutige Kennzeichnung des Koordinatensystems an. Die maximale Länge für diesen Parameter beträgt 128 Zeichen.

Der Wert für *name_des_koordinatensystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-definition *zeichenfolge_der_definition*
Dieser Parameter definiert das Koordinatensystem. Normalerweise stellt der Lieferant des Koordinatensystems die Informationen für diesen Parameter zur Verfügung. Die maximale Länge für diesen Parameter beträgt 2048 Zeichen.

-organization *zeichenfolge_der_organisation*
Definiert den Namen der Organisation, die das Koordinatensystem definiert und seine Definition zur Verfügung gestellt hat (z. B. "European Petroleum Survey Group (EPSG)").

Die Kombination der Parameter *zeichenfolge_der_organisation* und *koordinatensystem_id_der_organisation* gibt das Koordinatensystem eindeutig an. Die maximale Länge für diesen Parameter beträgt 128 Zeichen.

-organizationCoordsysId *koordinatensystem_id_der_organisation*
Dieser Parameter gibt eine numerische Kennung an. Dieser Wert wird von der im Parameter *zeichenfolge_der_organisation* angegebenen Einheit zugeordnet. Er ist nicht zwangsläufig gegenüber allen anderen Koordinatensystemen eindeutig.

Die Kombination der Parameter *zeichenfolge_der_organisation* und *koordinatensystem_id_der_organisation* gibt das Koordinatensystem eindeutig an.

-description *zeichenfolge_der_beschreibung*
Dieser Parameter beschreibt das Koordinatensystem, indem seine Anwendung erläutert wird. Die maximale Länge für diesen Parameter beträgt 256 Zeichen.

Hinweise zur Verwendung

DB2 Spatial Extender stellt mehr als 4000 Koordinatensysteme zur Verfügung: Das Erstellen eines Koordinatensystems ist nur selten erforderlich. Sie müssen auch ein räumliches Bezugssystem erstellen, das auf dem neuen Koordinatensystem basiert.

Beispiel

Im folgenden Beispiel wird ein Koordinatensystem namens MYCOORDSYS erstellt.

```
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]]
```

Befehl 'db2se create_srs'

Der Befehl **db2se create_srs** erstellt ein räumliches Bezugssystem.

Mithilfe dieses Befehls können Sie die Definition eines räumlichen Bezugssystems erstellen. Ein räumliches Bezugssystem ist durch das Koordinatensystem, die Genauigkeit und die Koordinatenbereiche, die in dem angegebenen räumlichen Bezugssystem dargestellt sind, definiert. Die Bereiche setzen sich aus den kleinsten und größten Koordinatenwerten für die X-, Y-, Z- und M-Koordinaten zusammen. Die Informationen zum Koordinatensystem sind in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS abrufbar.

Dieser Befehl erstellt räumliche Bezugssysteme mithilfe zweier Methoden:

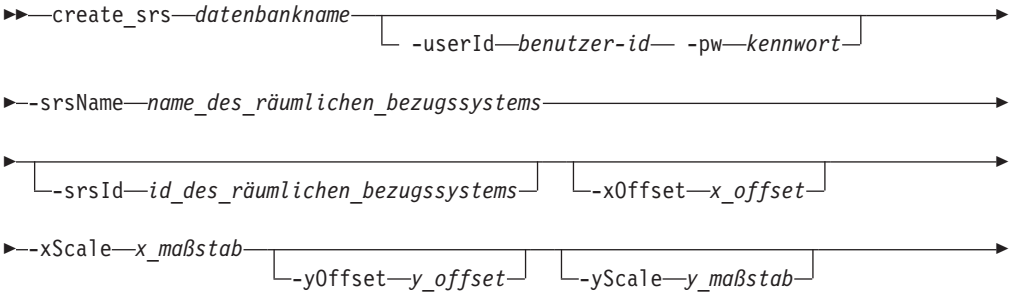
- Verwendung von Umwandlungsfaktoren (Offsetwerte und Maßstabsfaktoren)
- Verwendung von Bereichen und Genauigkeitswerten (Umwandlungsfaktoren werden berechnet)

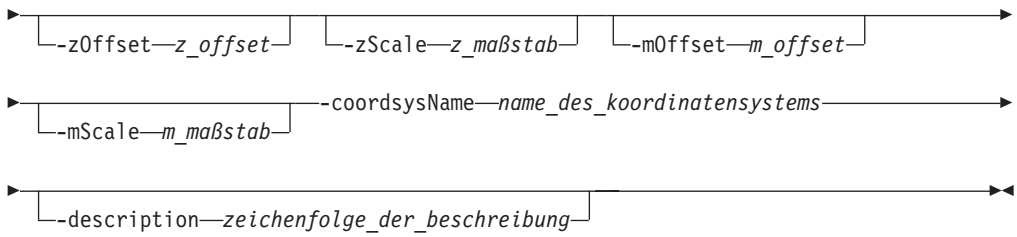
Berechtigung

Die Benutzer-ID muss über die Berechtigungen DBADM und DATAACCESS für die Datenbank verfügen, die für räumliche Operationen aktiviert ist, um diesen Befehl auszuführen.

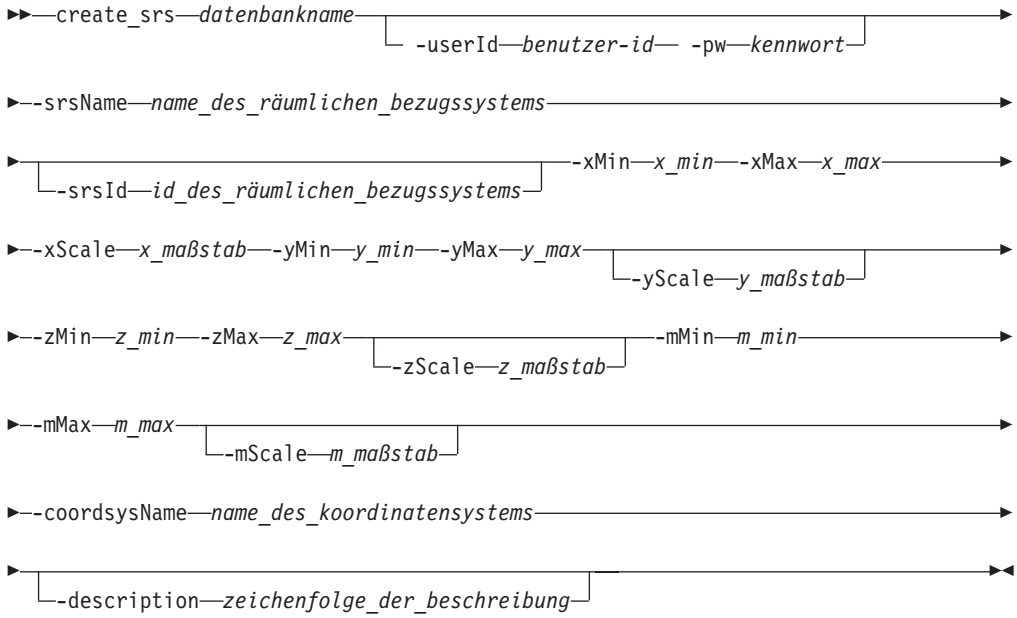
Befehlssyntax

Befehl 'db2se create_srs' (Verwendung von Umwandlungsfaktoren)





Befehl 'db2se create_srs' (Verwendung von Bereichen und Genauigkeitswerten)



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie die Definition eines räumlichen Bezugssystems erstellen möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-srsName *name_des_räumlichen_bezugssystems*
Gibt das räumliche Bezugssystem an, das Sie erstellen möchten. Der Wert für *name_des_räumlichen_bezugssystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-srsId *id_des_räumlichen_bezugssystems*
Dieser Parameter gibt die eindeutige Kennung des räumlichen Bezugssystems an. Diese Kennung wird für unterschiedliche räumliche Funktionen als Eingabeparameter verwendet.

-xMin *x_min*
Gibt den kleinstmöglichen X-Koordinatenwert für alle Geometrien an, die das angegebene räumliche Bezugssystem verwenden.

-xMax *x_max*
Gibt den größtmöglichen X-Koordinatenwert für alle Geometrien an, die das angegebene räumliche Bezugssystem verwenden. Abhängig von dem Wert *x_maßstab* ist der von der Sicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS zurückgegebene maximale X-Koordinatenwert möglicherweise größer als der unter *x_max* angegebene Wert. Gültig ist der Wert, der von der Sicht zurückgegeben wird.

-xOffset *x_offset*
Dieser Parameter gibt den Offset für alle X-Koordinaten von Geometrien an, die in dem angegebenen räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor *x_maßstab* angewendet wird.

-xScale *x_maßstab*
Dieser Parameter gibt den Maßstabsfaktor für alle X-Koordinaten von Geometrien an, die in dem angegebenen räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset *x_offset* subtrahiert wurde.

-yMin *y_min*
Gibt den kleinstmöglichen Y-Koordinatenwert für alle Geometrien an, die das angegebene räumliche Bezugssystem verwenden.

-yMax *y_max*
Gibt den größtmöglichen Y-Koordinatenwert für alle Geometrien an, die das angegebene räumliche Bezugssystem verwenden. Abhängig von dem Wert *y_maßstab* ist der von der Sicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS zurückgegebene maximale Y-Koordinatenwert möglicherweise größer als der angegebene Wert *y_max*. Gültig ist der Wert, der von der Sicht zurückgegeben wird.

-yOffset *y_offset*
Dieser Parameter gibt den Offset für alle Y-Koordinaten von Geometrien an, die in dem angegebenen räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor *y_maßstab* angewendet wird.

-yScale *y_maßstab*
Dieser Parameter gibt den Maßstabsfaktor für alle Y-Koordinaten von Geometrien an, die in dem angegebenen räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset *y_offset* subtrahiert wurde. Dieser Maßstabsfaktor muss mit dem Wert für *x_maßstab* identisch sein.

-zMin *z_min*
Gibt den kleinstmöglichen Z-Koordinatenwert für alle Geometrien an, die das angegebene räumliche Bezugssystem verwenden.

-zMax *z_max*
Gibt den größtmöglichen Z-Koordinatenwert für alle Geometrien an, die das angegebene räumliche Bezugssystem verwenden. Abhängig von dem Wert *z_maßstab* ist der von der Sicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS zurückgegebene maximale Z-Koordinatenwert möglicherweise größer als der angegebene Wert *z_max*. Gültig ist der Wert, der von der Sicht zurückgegeben wird.

-zOffset *z_offset*
Dieser Parameter gibt den Offset für alle Z-Koordinaten von Geometrien an, die in dem angegebenen räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor *z_maßstab* angewendet wird.

-zScale *z_maßstab*
Dieser Parameter gibt den Maßstabsfaktor für alle Z-Koordinaten von Geometrien an, die in dem angegebenen räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset *z_offset* subtrahiert wurde.

-mMin *m_min*
Gibt den kleinstmöglichen M-Koordinatenwert für alle Geometrien an, die das angegebene räumliche Bezugssystem verwenden.

-mMax *m_max*
Gibt den größtmöglichen M-Koordinatenwert für alle Geometrien an, die das angegebene räumliche Bezugssystem verwenden. Abhängig von dem Wert *m_maßstab* ist der von der Sicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS zurückgegebene maximale M-Koordinatenwert möglicherweise größer als der angegebene Wert *m_max*. Gültig ist der Wert, der von der Sicht zurückgegeben wird.

-mOffset *m_offset*
Dieser Parameter gibt den Offset für alle M-Koordinaten von Geometrien an, die in dem angegebenen räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor *m_maßstab* angewendet wird.

-mScale *m_maßstab*
Dieser Parameter gibt den Maßstabsfaktor für alle M-Koordinaten von Geometrien an, die in dem angegebenen räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset *m_offset* subtrahiert wurde.

-coordsysName *name_des_koordinatensystems*
Dieser Parameter ist die eindeutige Kennzeichnung des Koordinatensystems, auf dem das angegebene räumliche Bezugssystem basiert. Das Koordinatensystem muss in der Sicht DB2GSE.ST_COORDINATE_SYSTEMS aufgeführt sein.

Der Wert für *name_des_koordinatensystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen. Die maximale Länge für diesen Parameter beträgt 128 Zeichen.

-description *zeichenfolge_der_beschreibung*
Dieser Parameter beschreibt das Koordinatensystem, indem seine Anwendung erläutert wird. Die maximale Länge für diesen Parameter beträgt 256 Zeichen.

Hinweise zur Verwendung

DB2 Spatial Extender bietet Unterstützung für fünf räumliche Bezugssysteme und mehr als 4000 Koordinatensysteme, zwischen denen Sie wählen können. Das Erstellen eines räumlichen Bezugssystems ist nur selten erforderlich.

Beispiel

Im folgenden Beispiel wird ein räumliches Bezugssystem namens MYSRS erstellt, indem die Befehlssyntax für Umrechnungsfaktoren verwendet wird.

```
db2se create_srs mydb -srsName mysrs -srsID 100
        -xOffset -180 -xScale 1000000 -yOffset -90
        -coordsysName \"GCS_North_American_1983\"
```

Befehl 'db2se disable_autogc'

Der Befehl **db2se disable_autogc** inaktiviert die automatische Geocodierung.

Dieser Befehl verhindert die Synchronisierung einer geocodierten Spalte mit ihren zugehörigen Geocodierungsspalten durch DB2 Spatial Extender. Eine Geocodierungsspalte wird als Eingabe für den Geocoder verwendet. Bei jeder Einfügung oder Aktualisierung von Werten in der/den Geocodierungsspalte(n) werden Trigger aktiviert. Diese Trigger rufen den zugeordneten Geocoder auf, damit dieser die eingefügten oder aktualisierten Werte geocodiert und die Ergebnisdaten in die geocodierte Spalte stellt. Informationen zu geocodierten Spalten sind in der Katalogsicht DB2GSE.ST_GEOCODING abrufbar.

Berechtigung

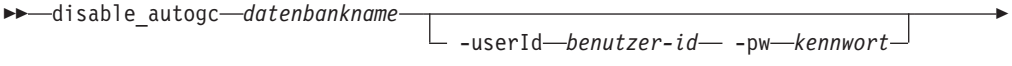
Die Benutzer-ID muss über eine der folgenden Berechtigungen zur Ausführung dieses Befehls verfügen:

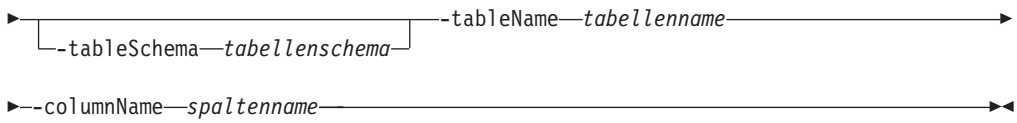
- Berechtigungen DBADM und DATAACCESS für die Datenbank mit der Tabelle, für die Trigger definiert sind, die gelöscht werden.
- Berechtigung CONTROL für diese Tabelle und Berechtigung DROPIN für das Schema DB2GSE.
- Berechtigungen ALTER und UPDATE für diese Tabelle und Berechtigung DROPIN für das Schema DB2GSE.

Außerdem muss die Benutzer-ID über die erforderlichen Berechtigungen für den DB2-Server verfügen, um Ausnahme- und Nachrichtendateien zu erstellen oder in diese Dateien zu schreiben.

Befehlssyntax

Befehl 'db2se disable_autogc'





Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie die automatische Geocodierung inaktivieren möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-tableSchema *tabellenschema*
Gibt den Schemanamen für die im Parameter *tabellenname* angegebene Tabelle an. Falls Sie keinen Schemanamen angeben, wird als Schemaname für die Tabelle bzw. Sicht der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

-tableName *tabellenname*
Gibt den Namen ohne Qualifikationsmerkmal der Tabelle an, die die angegebene Spalte enthält, für die die automatische Geocodierung inaktiviert werden soll. Die für die Synchronisierung der geocodierten Spalte erstellten Trigger werden gelöscht. Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-columnName *spaltenname*
Gibt den Namen der verwalteten geocodierten Spalte an, für die Sie die automatische Geocodierung inaktivieren möchten. Der Wert für *spaltenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Beispiel

Im folgenden Beispiel wird die automatische Geocodierung für eine geocodierte Spalte namens "MYCOLUMN" in der Tabelle "MYTABLE" inaktiviert.

```
db2se disable_autogc mydb -tableName mytable -columnName mycolumn
```

Befehl 'db2se disable_db'

Der Befehl **db2se disable_db** entfernt Ressourcen, die DB2 Spatial Extender das Speichern räumlicher Daten und die Unterstützung von Operationen mit diesen Daten ermöglichen.

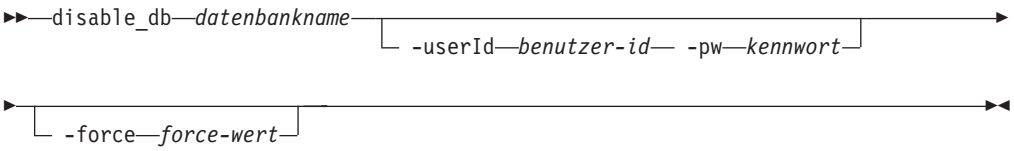
Zu diesen Ressourcen gehören räumliche Datentypen, Typen von räumlichen Indizes, Katalogsichten, bereitgestellte Funktionen und gespeicherte Prozeduren, die bei der Aktivierung der Unterstützung von räumlichen Operationen auf einer Datenbank erstellt wurden.

Berechtigung

Die Benutzer-ID muss über die Berechtigungen DBADM und DATAACCESS für die Datenbank verfügen, die für räumliche Operationen aktiviert ist, um diesen Befehl auszuführen.

Befehlssyntax

Befehl 'db2se disable_db'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie DB2 Spatial Extender inaktivieren möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-force *force-wert*
Dieser Parameter gibt an, dass Sie eine Datenbank für räumliche Operationen inaktivieren wollen, obwohl unter Umständen Datenbankobjekte vorhanden sind, die von den räumlichen Typen oder den räumlichen Funktionen abhängig sind. Bei solchen abhängigen Datenbankobjekten kann es sich um Tabellen, Sichten, Integritätsbedingungen, Trigger, generierte Spalten, Methoden, Funktionen, Prozeduren und andere Datentypen (untergeordnete Typen oder strukturierte Typen mit einem räumlichen Attribut) handeln.

Falls Sie einen Wert ungleich null für den Parameter **-force** angeben, wird die Datenbank inaktiviert, und alle Ressourcen von DB2 Spatial Extender werden entfernt. Wenn Sie für *force-wert* 0 (null) angeben, wird die Datenbank nur inaktiviert, wenn Datenbankobjekte unabhängig von räumlichen Typen oder räumlichen Funktionen sind.

Hinweise zur Verwendung

Wenn Sie bereits räumliche Spalten definiert haben, aber trotzdem die Datenbank für räumliche Operationen inaktivieren wollen, müssen Sie einen anderen Wert als 0 (null) für den Parameter 'force' angeben. Dann werden alle räumlichen Ressourcen aus der Datenbank entfernt, von denen keine anderen Ressourcen abhängig sind.

Beispiel

Im folgenden Beispiel wird die Unterstützung für räumliche Operationen in der Datenbank *MYDB* auch dann inaktiviert, wenn Datenbankobjekte mit Abhängigkeiten von räumlichen Typen oder Funktionen vorhanden sind.

```
db2se disable_db mydb -force 1
```

Befehl 'db2se drop_cs'

Der Befehl **db2se drop_cs** löscht die Definition eines Koordinatensystems.

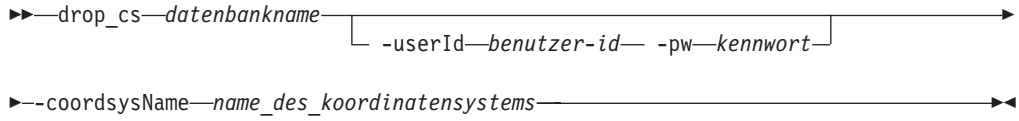
Dieser Befehl löscht ein Koordinatensystem aus der Datenbank. Die Informationen sind in der Katalogsicht DB2GSE.ST_COORDINATE_SYSTEMS nicht länger abrufbar.

Berechtigung

Die Benutzer-ID muss über die Berechtigungen DBADM und DATAACCESS für die Datenbank verfügen, die für räumliche Operationen aktiviert ist, um diesen Befehl auszuführen.

Befehlssyntax

Befehl 'db2se drop_cs'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie das Koordinatensystem löschen möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für die im Parameter *benutzer-id* angegebene Benutzer-ID an.

-coordsysName *name_des_koordinatensystems*
Gibt das Koordinatensystem, das Sie löschen möchten, eindeutig an.

Der Wert für *name_des_koordinatensystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen. Die maximale Länge für diesen Parameter beträgt 128 Zeichen.

Hinweise zur Verwendung

Ein Koordinatensystem, auf dem ein räumliches Bezugssystem basiert, kann nicht gelöscht werden.

Beispiel

Im folgenden Beispiel wird ein Koordinatensystem namens MYCOORDSYS gelöscht.

```
db2se drop_cs mydb -coordsysName mycoordsys
```

Befehl 'db2se drop_srs'

Der Befehl **db2se drop_srs** löscht die Definition eines räumlichen Bezugssystems.

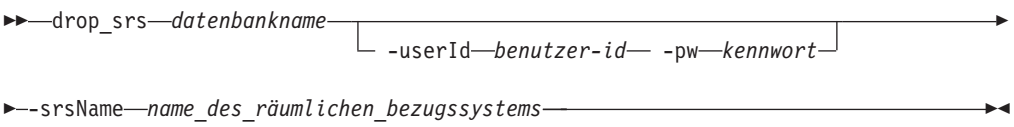
Die Informationen zum räumlichen Bezugssystem sind in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS nicht länger abrufbar.

Berechtigung

Die Benutzer-ID verfügt über die Berechtigungen DBADM und DATAACCESS für die Datenbank, die für räumliche Operationen aktiviert ist, um diesen Befehl auszuführen.

Befehlssyntax

Befehl 'db2se drop_srs'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie die Definition eines räumlichen Bezugssystems löschen möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-srsName *name_des_räumlichen_bezugssystems*
Gibt das räumliche Bezugssystem an, das Sie löschen möchten. Der Wert für *name_des_räumlichen_bezugssystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Hinweise zur Verwendung

Wenn Sie räumliche Spalten registriert haben, die ein räumliches Bezugssystem verwenden, können Sie das räumliche Bezugssystem nicht löschen.

Beispiel

Im folgenden Beispiel wird ein räumliches Bezugssystem namens MYSRS gelöscht.

```
db2se drop_srs mydb -srsName mysrs
```

Befehl 'db2se enable_autogc'

Der Befehl **db2se enable_autogc** aktiviert die automatische Geocodierung.

Dieser Befehl aktiviert DB2 Spatial Extender, um eine geocodierte Spalte mit ihren zugehörigen Geocodierungsspalten zu synchronisieren. Eine Geocodierungsspalte wird als Eingabe für den Geocoder verwendet. Bei jeder Einfügung oder Aktualisierung von Werten in der/den Geocodierungsspalte(n) werden Trigger aktiviert. Diese Trigger rufen den zugeordneten Geocoder auf, damit dieser die eingefügten oder aktualisierten Werte geocodiert und die Ergebnisdaten in die geocodierte Spalte stellt. Informationen zu geocodierten Spalten sind in der Katalogsicht DB2GSE.ST_GEOCODING abrufbar.

Berechtigung

Die Benutzer-ID muss über eine der folgenden Berechtigungen zur Ausführung dieses Befehls verfügen:

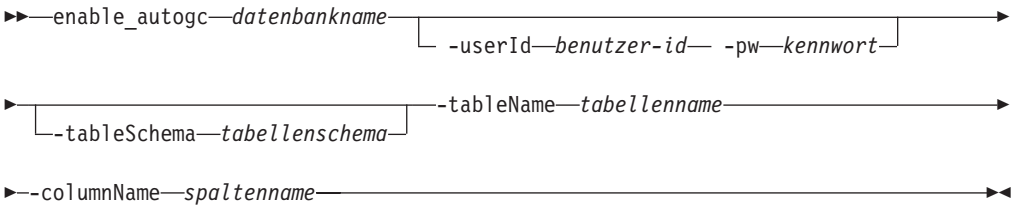
- Berechtigungen DBADM und DATAACCESS für die Datenbank mit der Tabelle, für die die Trigger definiert sind, die durch diese gespeicherte Prozedur erstellt werden.
- Zugriffsrecht CONTROL für die Tabelle
- Zugriffsrecht ALTER für die Tabelle

Falls die Berechtigungs-ID der Anweisung die Berechtigung DBADM nicht besitzt, muss sie - sofern der Trigger vorhanden ist - alle folgenden Zugriffsrechte besitzen (das Zugriffsrecht PUBLIC oder Gruppenzugriffsrechte werden nicht berücksichtigt):

- Zugriffsrecht SELECT für die Tabelle, für die die automatische Geocodierung aktiviert ist, oder Berechtigung DATAACCESS.
- Erforderliche Zugriffsrechte für die Auswertung von SQL-Ausdrücken, die in der Geocodierungskonfiguration für die Parameter angegeben sind.

Befehlssyntax

Befehl 'db2se enable_autogc'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie die automatische Geocodierung aktivieren möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-tableSchema *tabellenschema*
Gibt den Schemanamen für die im Parameter *tabellenname* angegebene Tabelle an. Falls Sie keinen Schemanamen angeben, wird als Schemaname für die Tabelle bzw. Sicht der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

-tableName *tabellenname*
Gibt den Namen ohne Qualifikationsmerkmal der Tabelle an, die die angegebene Spalte enthält, für die die automatische Geocodierung aktiviert werden soll. Die Trigger für die Synchronisierung der geocodierten Spalte werden erstellt. Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-columnName *spaltenname*
Dieser Parameter gibt die Spalte an, in der die geocodierten Daten eingefügt oder aktualisiert werden sollen. Diese Spalte wird als geocodierte Spalte bezeichnet. Der Wert für *spaltenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Hinweise zur Verwendung

Vor der Aktivierung der automatischen Geocodierung müssen Sie die Konfiguration der Geocodierung ausführen, um die Parameterwerte für den Geocoder und die Geocodierung anzugeben. Außerdem werden die Geocodierungsspalten angegeben, die mit den geocodierten Spalten synchronisiert werden sollen.

Die automatische Geocodierung kann nur für Tabellen aktiviert werden, für die INSERT- und UPDATE-Trigger erstellt werden können. Infolgedessen kann die automatische Geocodierung nicht für Sichten oder Kurznamen aktiviert werden.

Beispiel

Im folgenden Beispiel wird die automatische Geocodierung für eine Spalte namens "MYCOLUMN" in der Tabelle "MYTABLE" definiert.

```
db2se enable_autogeocoding mydb -tableName mytable -columnName mycolumn
```

Befehl 'db2se enable_db'

Der Befehl **db2se enable_db** stellt einer Datenbank die Ressourcen zur Verfügung, die zum Speichern von räumlichen Daten und zur Unterstützung von Operationen benötigt werden.

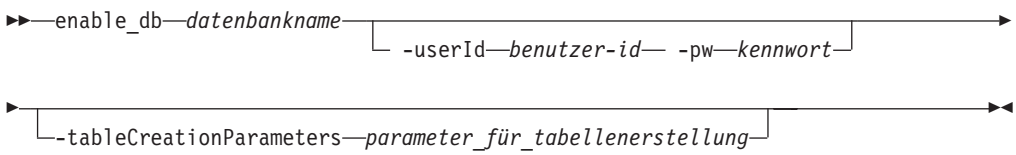
Zu diesen Ressourcen gehören räumliche Datentypen, Typen von räumlichen Indizes, Katalogsichten, bereitgestellte Funktionen und andere gespeicherte Prozeduren.

Berechtigung

Die Benutzer-ID muss über die Berechtigungen DBADM, DATAACCESS und CTRLACCESSDBADM für die Datenbank verfügen, die für räumliche Operationen aktiviert ist, um diesen Befehl auszuführen.

Befehlssyntax

Befehl 'db2se enable_db'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie DB2 Spatial Extender aktivieren möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-tableCreationParameters *parameter_für_tabellenerstellung*
Gibt Parameter für die Anweisungen CREATE TABLE an, die zur Erstellung der DB2 Spatial Extender-Katalogtabellen verwendet werden. Verwenden Sie die Parametersyntax für die Anweisung CREATE TABLE. Im folgenden Beispiel wird unter dem Betriebssystem Windows ein Tabellenbereich angegeben, in dem die Tabellen erstellt werden sollen, und ein Tabellenbereich, in dem die Tabellenindizes erstellt werden sollen:

```
-tableCreationParameters "IN tabellenbereichsname INDEX IN indextabellenbereichsname"
```

Die maximale Länge für diesen Parameter beträgt 32.672 Zeichen.

Hinweise zur Verwendung

Stellen Sie sicher, dass ein Tabellenbereich für temporäre Systemtabellen mit einer Seitengröße von mindestens 8 KB und einer Mindestgröße von 500 Seiten vorhanden ist. Dies ist eine Voraussetzung für die erfolgreiche Ausführung des Befehls **db2se enable_db**.

Beispiel

Im folgenden Beispiel wird eine Datenbank namens "MYDB" für räumliche Operationen aktiviert.

```
db2se enable_db mydb
```

Befehl 'db2se export_shape'

Der Befehl **db2se export_shape** exportiert eine räumliche Spalte und ihre zugehörige Tabelle in eine Formdatei.

Mithilfe dieses Befehls können Sie die Definition eines räumlichen Bezugssystems erstellen. Ein räumliches Bezugssystem ist durch das Koordinatensystem, die Genauigkeit und die Koordinatenbereiche, die in dem angegebenen räumlichen Be-

zugssystem dargestellt sind, definiert. Die Bereiche sind die kleinstmöglichen und größtmöglichen Koordinatenwerte für die X-, Y-, Z- und M-Koordinaten. Die Informationen zum Koordinatensystem sind in der Katalogansicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS abrufbar.

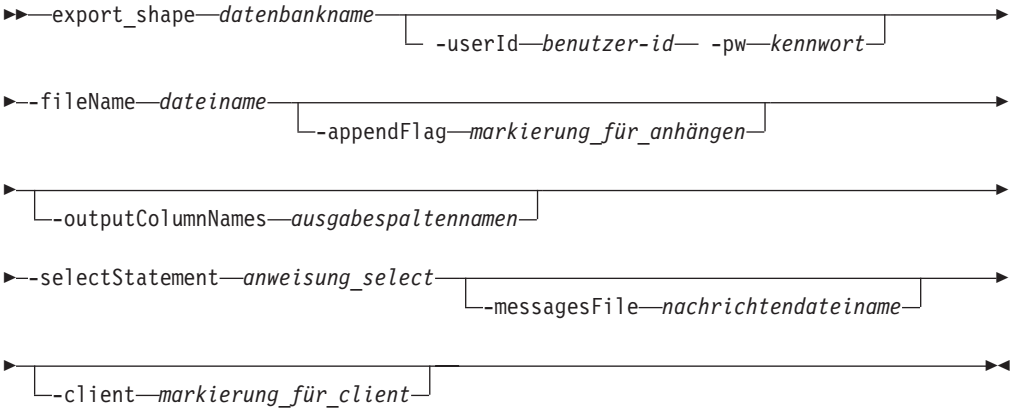
Berechtigung

Der Benutzer muss die Zugriffsrechte besitzen, die für eine erfolgreiche Ausführung der Anweisung SELECT, über die die Daten exportiert werden, erforderlich sind.

Außerdem muss die DB2-Instanzeigner-ID über die erforderlichen Berechtigungen für den DB2-Server verfügen, um Ausnahme- und Nachrichtendateien zu erstellen oder in diese Dateien zu schreiben.

Befehlssyntax

Befehl 'db2se export_shape'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, die die zu exportierende Tabelle enthält.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-fileName *dateiname*
Dieser Parameter gibt den vollständigen Pfadnamen der Formdatei an, in die die Daten exportiert werden sollen. Eine vollständige Liste der Dateien, die auf den DB2-Server geschrieben werden, finden Sie unter „Hinweise zur Verwendung" auf Seite 156. Die maximale Länge für diesen Parameter beträgt 256 Zeichen.

Beim Export in eine neue Datei können Sie die optionale Dateierweiterung .shp oder .SHP angeben. Wenn Sie .shp oder .SHP als Dateierweiterung angeben, erstellt DB2 Spatial Extender die Datei mit dem angegebenen Wert für *dateiname*.

Falls Sie die optionale Dateierweiterung nicht angeben, erstellt DB2 Spatial Extender die Datei unter dem Namen *dateiname*.shp.

Wenn Daten beim Exportieren an eine vorhandene Datei angehängt werden sollen, sucht DB2 Spatial Extender zunächst nach einer exakten Übereinstimmung mit dem Namen, den Sie mit dem Parameter **-fileName** angegeben haben. Falls DB2 Spatial Extender keine exakte Übereinstimmung feststellt, wird zunächst nach einer Datei mit der Erweiterung .shp und dann nach einer Datei mit der Erweiterung .SHP gesucht. Wenn der Wert *markierung_für_anhängen* angibt, dass Daten beim Exportieren nicht an eine vorhandene Datei angehängt werden, die Datei aber bereits vorhanden ist, gibt DB2 Spatial Extender einen Fehler zurück und überschreibt die Datei nicht.

-appendFlag *markierung_für_anhängen*
Dieser Parameter gibt an, ob die zu exportierenden Daten an eine vorhandene Formdatei angehängt werden sollen. Folgende Parameterwerte sind möglich:

- Ein Wert ungleich null für *markierung_für_anhängen*, um anzugeben, dass die Daten an eine vorhandene Formdatei angehängt werden sollen. Wenn die vorhandene Dateistruktur nicht mit den exportierten Daten übereinstimmt, wird ein Fehler zurückgegeben.
- Ein Wert 0 (null) für *markierung_für_anhängen*, um anzugeben, dass die Daten in eine neue Datei exportiert werden sollen. Vorhandene Dateien werden von DB2 Spatial Extender nicht überschrieben.

-outputColumnNames *ausgabespaltennamen*
Gibt einen oder mehrere (durch Kommas voneinander getrennte) Spaltennamen an, die in der dBASE-Ausgabedatei für nicht räumliche Spalten verwendet werden. Wenn für diesen Parameter kein Wert angegeben wird, werden die Spaltennamen aus der Anweisung SELECT verwendet.

Sofern die Spaltennamen nicht in doppelte Anführungszeichen gesetzt sind, werden sie in Großbuchstaben umgewandelt. Die Anzahl der angegebenen Spalten muss mit der Anzahl der Spalten übereinstimmen, die von der im Parameter *anweisung_select* angegebenen Anweisung SELECT zurückgegeben werden. Die räumliche Spalte ist in diesem Wert nicht enthalten.

Die maximale Länge für diesen Parameter beträgt 32.672 Zeichen.

-selectStatement *anweisung_select*
Dieser Parameter gibt den Subselect an, der die zu exportierenden Daten zurückgibt. Der Subselect muss sich auf genau eine räumliche Spalte und eine beliebige Anzahl von Attributspalten beziehen. Die maximale Länge für diesen Parameter beträgt 32.672 Zeichen.

-messagesFile *nachrichtendateiname*
Gibt den vollständigen Pfadnamen der Datei auf dem DB2-Server an, in die DB2 Spatial Extender Nachrichten über die Exportoperation schreibt. Wenn Sie diesen Parameter angeben und die Datei bereits vorhanden ist, wird ein Fehler zurückgegeben, und die Exportoperation wird beendet. Wenn Sie diesen Parameter nicht angeben, erstellt DB2 Spatial Extender keine Nachrichtendatei.

Folgende Nachrichtentypen werden in die Nachrichtendatei geschrieben:

- Informationsnachrichten, beispielsweise eine Zusammenfassung der Exportoperation
- Fehlernachrichten für Daten, die nicht exportiert werden konnten, beispielsweise aufgrund eines abweichenden Koordinatensystems

Die maximale Länge für diesen Parameter beträgt 256 Zeichen.

-client *markierung_für_client*
Gibt an, ob die Exportoperation auf dem Client oder dem DB2-Server stattfindet und wo die Dateien erstellt werden. Folgende Parameterwerte sind möglich:

- 0 (null), um anzugeben, dass die Exportoperation auf dem DB2-Server ausgeführt wird und die Dateien auf dem DB2-Server erstellt werden.
- 1, um anzugeben, dass die Exportoperation auf dem Client ausgeführt wird und die Dateien auf dem Client erstellt werden.

Wenn Sie keinen Wert für diesen Parameter angeben, lautet der Standardwert 0 (null).

Hinweise zur Verwendung

Sie können immer nur eine räumliche Spalte auf einmal exportieren.

Sie können den Exportprozess auf dem Client ausführen, auf dem auch der Befehl ausgeführt wird. Dies ist in vielen Fällen praktischer, da kein Zugriff auf das DB2-Serverdateisystem erforderlich ist.

Der Befehl **db2se export_shape** erstellt die folgenden vier Dateien bzw. schreibt in diese Dateien:

- die eigentliche Formdatei (Erweiterung .shp)
- die Formindexdatei (Erweiterung .shx)
- eine dBASE-Datei, die Daten für nicht-räumliche Spalten enthält (Erweiterung .dbf). Diese Datei wird nur dann erstellt, wenn tatsächlich Attributspalten exportiert werden müssen.
- eine Projektionsdatei, die das Koordinatensystem angibt, das den räumlichen Daten zugeordnet ist, falls das Koordinatensystem nicht "UNSPECIFIED" lautet (Erweiterung .prj). Das Koordinatensystem wird dem ersten räumlichen Datensatz entnommen. Falls in nachfolgenden Datensätzen andere Koordinatensysteme angegeben sind, tritt ein Fehler auf.

Die folgende Tabelle gibt Aufschluss darüber, wie DB2-Datentypen in dBASE-Attributdateien gespeichert sind. Alle nicht angegebenen DB2-Datentypen werden nicht unterstützt.

Tabelle 20. Speicherung von DB2-Datentypen in Attributdateien

SQL-Typ	Typ bei .dbf	Länge bei .dbf	Dezimalstellen bei .dbf	Kommentare
SMALLINT	N	6	0	
INTEGER	N	11	0	
BIGINT	N	20	0	
DECIMAL	N	Genauigkeit+2	Maßstab	
REAL FLOAT(1) über FLOAT(24)	F	14	6	
DOUBLE FLOAT(25) über FLOAT(53)	F	19	9	
CHARACTER, VARCHAR, LONG VARCHAR und DATALINK	C	*länge*	0	länge ≤ 255
DATE	D	8	0	
TIME	C	8	0	
TIMESTAMP	C	26	0	

Alle Synonyme für Datentypen und einzigartige Datentypen, die auf den in der vorstehenden Tabelle aufgeführten Typen basieren, werden unterstützt.

Beispiel

Im folgenden Beispiel werden eine räumliche Spalte namens MYCOLUMN und ihre zugehörige Tabelle, MYTABLE, in die Formdatei `myshapefile` exportiert, die sich auf dem Client befindet.

```
db2se export_shape mydb -fileName /home/myaccount/myshapefile
        -selectStatement "select * from mytable" -client 1
```

Befehl 'db2se import_shape'

Der Befehl **db2se import_shape** importiert eine Formdatei in eine Datenbank, die für räumliche Operationen aktiviert ist.

Dieser Befehl kann Form- und Attributdaten in eine vorhandene Tabelle oder eine neue Tabelle importieren.

Berechtigung

Außerdem muss die DB2-Instanzeigner-ID über die erforderlichen Berechtigungen für den DB2-Server verfügen, um Ausnahme- und Nachrichtendateien zu erstellen oder in diese Dateien zu schreiben.

Für die Benutzer-ID sind zusätzliche Berechtigungen erforderlich, um diesen Befehl auszuführen. Die Anforderungen sind abhängig davon, ob Sie in eine vorhandene Tabelle oder eine neue Tabelle importieren.

Anforderungen für den Import in eine vorhandene Tabelle

Die Benutzer-ID muss über eine der folgenden Berechtigungen verfügen:

- DATAACCESS
- Zugriffsrecht CONTROL für die Tabelle bzw. Sicht
- Zugriffsrecht INSERT und SELECT für die Tabelle bzw. Sicht

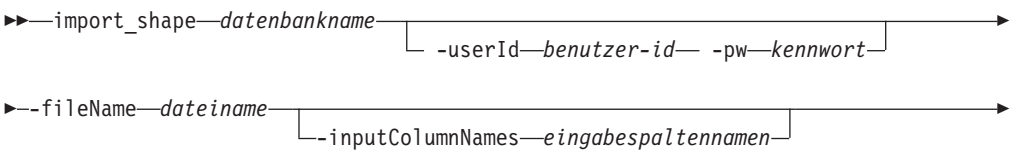
Anforderungen für den Import in eine neue Tabelle

Die Benutzer-ID muss über eine der folgenden Berechtigungen verfügen:

- DBADM und DATAACCESS
- Berechtigung CREATETAB für die Datenbank
- Berechtigung IMPLICIT_SCHEMA für die Datenbank, falls der Schemaname der Tabelle nicht vorhanden ist
- Zugriffsrecht CREATEIN für das Schema, falls das Schema der Tabelle vorhanden ist

Befehlssyntax

Befehl 'db2se import_shape'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie die Formdatei importieren möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-fileName *dateiname*
Dieser Parameter gibt den vollständigen Pfadnamen der Formdatei an, in die die Daten importiert werden sollen. Wenn Sie `.shp` oder `.SHP` als Dateierweiterung angeben, sucht DB2 Spatial Extender zunächst nach einer exakten Übereinstimmung mit dem Namen, den Sie mit dem Parameter **-fileName** angegeben haben. Falls DB2 Spatial Extender keine exakte Übereinstimmung feststellt, wird zunächst nach einer Datei mit der Erweiterung `.shp` und dann nach einer Datei mit der Erweiterung `.SHP` gesucht. Eine vollständige Liste der Dateien, die auf den DB2-Server geschrieben werden, finden Sie unter „Hinweise zur Verwendung” auf Seite 163.

Die maximale Länge für diesen Parameter beträgt 256 Zeichen.

-inputColumnNames *eingabespaltennamen*
Dieser Parameter gibt eine Liste der Attributspalten an, die aus der dBASE-Datei importiert werden sollen. Wenn für diesen Parameter kein Wert angegeben wird, werden alle Spalten in der Datei importiert. Verwenden Sie eines der folgenden Formate, um eine Liste mit Attributen anzugeben:

- Eine durch Kommas getrennte Liste von Spaltennamen, die aus der dBASE-Datei importiert werden soll, wie im folgenden Beispiel gezeigt:

  ```
  N(SPALTE1,SPALTE5,SPALTE3,SPALTE7)
  ```

  Sofern die Spaltennamen nicht in doppelte Anführungszeichen gesetzt sind, werden sie in Großbuchstaben umgewandelt. Die resultierenden Namen müssen genau mit den Spaltennamen in der dBASE-Datei übereinstimmen.
- Eine durch Kommas getrennte Liste von Spaltennummern, die aus der dBASE-Datei importiert werden soll, wie im folgenden Beispiel gezeigt:

  ```
  P(1,5,3,7)
  ```

  Die Spaltennummerierung beginnt bei 1. Jede Nummer in der Liste muss durch ein Komma abgegrenzt werden.
- Eine leere Zeichenfolge `""`, um anzugeben, dass keine Attributdaten importiert werden sollen.

Die maximale Länge für diesen Parameter beträgt 32.672 Zeichen.

-srsName *name_des_räumlichen_bezugssystems*
Dieser Parameter gibt das räumliche Bezugssystem an, das für die Geometrien, die in die räumliche Spalte importiert werden, verwendet werden soll. Der Wert für *name_des_räumlichen_bezugssystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Die räumliche Spalte ist nicht registriert. Das räumliche Bezugssystem muss vorhanden sein, bevor die Daten importiert werden. Der Importprozess nimmt keine implizite Erstellung des räumlichen Bezugssystems vor, vergleicht jedoch das Koordinatensystem des räumlichen Bezugssystems mit dem Koordinatensystem, das in der Datei `.prj` (sofern zusammen mit der Formdatei verfügbar) angegeben ist.

Außerdem wird während des Importprozesses überprüft, ob der Bereich der Daten in der Formdatei im angegebenen räumlichen Bezugssystem dargestellt werden kann. Der Importprozess überprüft also, ob die Bereiche innerhalb der kleinsten und größten X-, Y-, Z- und M-Koordinaten des räumlichen Bezugssystems liegen.

-tableSchema *tabellenschema*
Dieser Parameter gibt den Schemanamen für die im Parameter *tabellenname* angegebene Tabelle an. Falls Sie keinen Schemanamen angeben, wird als Schemaname für die Tabelle bzw. Sicht der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

-tableName *tabellenname*
Dieser Parameter gibt den Namen ohne Qualifikationsmerkmal der Tabelle an, in die die Daten in der Formdatei importiert werden sollen. Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-tableAttrColumns *attributspalten*
Dieser Parameter gibt die Namen der Tabellenspalten an, in denen die Attributdaten aus der dBASE-Datei gespeichert werden sollen. Wenn für diesen Parameter kein Wert angegeben wird, werden die Namen der Spalten in der dBASE-Datei verwendet.

Die Anzahl der angegebenen Spalten muss mit der Anzahl der Spalten übereinstimmen, die aus der dBASE-Datei importiert werden sollen. Wenn die Tabelle vorhanden ist, müssen die Spaltendefinitionen mit den ankommenden Daten identisch sein. Erläuterungen zur Zuordnung von Attributdatentypen zu den entsprechenden DB2-Datentypen finden Sie unter „Hinweise zur Verwendung” auf Seite 163.

Sofern die Spaltennamen nicht in doppelte Anführungszeichen gesetzt sind, werden sie in Großbuchstaben umgewandelt. Die maximale Länge für diesen Parameter beträgt 32.672 Zeichen.

-createTableFlag *markierung_erstellen*
Dieser Parameter gibt an, ob der Importprozess eine Tabelle erstellen soll. Folgende Parameterwerte sind möglich:

- Ein Wert ungleich 0 (null) für *markierung_erstellen*, um eine Tabelle zu erstellen. Falls die Tabelle vorhanden ist, wird ein Fehler zurückgegeben.
- Ein Wert 0 (null) für *markierung_erstellen*, um eine vorhandene Tabelle zu verwenden.

Wenn für diesen Parameter kein Wert angegeben ist, wird eine neue Tabelle erstellt.

-tableCreationParameters *parameter_für_tabellenerstellung*
Dieser Parameter gibt alle Optionen an, die zu der Anweisung CREATE TABLE hinzugefügt werden sollen, mit der die Tabelle, die im Parameter *tabellenname* angegeben ist, erstellt wird.

Verwenden Sie zur Angabe der Optionen für CREATE TABLE die Syntax der Anweisung CREATE TABLE. So können Sie im Parameter *parameter_für_tabellenerstellung* beispielsweise einen Tabellenbereich angeben, in dem die Tabellen, Indizes und großen Objekte erstellt werden sollen:

```
IN tabellenbereichsname INDEX IN indextabellenbereichsname LONG IN
langer_tabellenbereichsname
```

Die maximale Länge für diesen Parameter beträgt 32.672 Zeichen.

-spatialColumn *räumliche_spalte*
Dieser Parameter gibt den Namen der räumlichen Spalte in der Tabelle an, in die die Formdaten importiert werden sollen.

Bei einer neuen Tabelle gibt dieser Parameter den Namen der neuen räumlichen Spalte an, die erstellt werden soll. Bei einer vorhandenen Tabelle gibt dieser Parameter den Namen einer vorhandenen räumlichen Spalte in der Tabelle an.

Der Wert für *räumliche_spalte* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-typeSchema *typschema*
Dieser Parameter gibt den Schemanamen des räumlichen Datentyps an, der im Wert *typname* angegeben ist. Wenn für diesen Parameter kein Wert angegeben ist, wird DB2GSE als Schemaname verwendet.

Der Wert für *typschema* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-typeName *typname*
Dieser Parameter gibt den Namen des Datentyps an, der für die räumlichen Werte verwendet werden soll. Wenn für diesen Parameter kein Wert angegeben ist, wird der Datentyp anhand der Formdatei ermittelt. Folgende Datentypen sind möglich:

- ST_Point
- ST_MultiPoint
- ST_MultiLineString
- ST_MultiPolygon

Formdateien lassen definitionsgemäß nur eine Unterscheidung zwischen Punkten und Mehrpunktangaben zu. Es kann nicht zwischen Polygonen und Multipolygonen oder zwischen Linienfolgen und Mehrlinienfolgen unterschieden werden.

Wenn Sie Daten in eine neue Tabelle importieren wollen, wird der Datentyp *typname* auch für die räumliche Spalte verwendet. In diesem Fall kann der Datentyp auch ein übergeordneter Typ von ST_Point, ST_MultiPoint, ST_MultiLineString oder ST_MultiPolygon sein.

Der Wert für *typname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-inlineLength *inlinelänge*
Dieser Parameter gibt für eine neue Tabelle an, wie viele Byte in der Tabelle maximal für die räumliche Spalte zugeordnet werden dürfen. Wenn für diesen Parameter kein Wert angegeben ist, wird der Standardwert für die Inlinelänge verwendet.

Räumliche Datensätze, die die im Parameter *inlinelänge* angegebene Größe überschreiten, werden im LOB-Tabellenbereich separat gespeichert. Der Zugriff auf diesen Tabellenbereich kann möglicherweise länger dauern.

Die folgenden Größen werden üblicherweise für unterschiedliche räumliche Typen benötigt:

- **Einzelpunkt:** 292 Byte.
- **Mehrpunktangabe, Linie oder Polygon:** Der Wert sollte so groß wie möglich sein. Achten Sie darauf, dass die Gesamtanzahl der Byte in einer Zeile den Grenzwert für die Seitengröße des Tabellenbereichs, für den die Tabelle erstellt wird, nicht überschreitet.

Eine vollständige Beschreibung des Werts *inlinelänge* finden Sie in der DB2-Dokumentation zur SQL-Anweisung CREATE TABLE. Mit der Tabellenfunktion ADMIN_EST_INLINE_LENGTH können Sie eine Schätzung bezüglich der für Geometrien in vorhandenen Tabellen erforderlichen Inlinelänge vornehmen.

-idColumn *id_spalte*
Dieser Parameter gibt den Namen einer zu erstellenden Spalte an, in der für jede Datenzeile eine eindeutige Nummer gespeichert werden soll. Die eindeutigen Werte für diese Spalte werden während des Importprozesses automatisch generiert. Der im Parameter *id_spalte* angegebene Name darf nicht mit einem der Spaltennamen in der dBASE-Datei identisch sein. Für einige räumliche Tools ist eine Spalte mit einer eindeutigen ID erforderlich.

Die Voraussetzungen und Auswirkungen dieses Parameters sind davon abhängig, ob die Tabelle bereits vorhanden ist.

- Bei einer vorhandenen Tabelle kann der Typ des Parameters *id_spalte* jeder beliebige Integertyp sein, z. B. INTEGER, SMALLINT oder BIGINT.
- Bei einer neuen Tabelle ist die Spalte wie folgt definiert:
  ```
  INTEGER NOT NULL PRIMARY KEY
  ```

  Wenn für *id_spalte_ist_identitätsspalte* ein Wert ungleich 0 (null) angegeben ist, wird die Definition wie folgt erweitert:

```
INTEGER NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY
( START WITH 1 INCREMENT BY 1 )
```

Der Wert für *id_spalte* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-idColumnIsIdentity *id_spalte_ist_identitätsspalte*
Dieser Parameter gibt an, ob die in *id_spalte* angegebene Spalte mit der Klausel IDENTITY erstellt werden soll. Wenn für *id_spalte_ist_identitätsspalte* ein Wert ungleich 0 (null) angegeben ist, wird die Spalte *id_spalte* als Identitätsspalte erstellt. Bei bereits vorhandenen Tabellen wird dieser Parameter ignoriert.

-restartCount *zähler_für_neustart*
Gibt an, dass die Importoperation bei Datensatz *n* + 1 gestartet werden soll. Die ersten *n* Datensätze werden übersprungen. Wenn für diesen Parameter kein Wert angegeben ist, werden alle Datensätze, beginnend mit dem Datensatz Nr. 1, importiert.

-commitScope *commitbereich*
Dieser Parameter gibt an, dass ein Commit durchgeführt werden soll, nachdem mindestens *n* Datensätze importiert wurden. Wenn für diesen Parameter kein Wert angegeben ist, wird am Ende der Operation ein Commit durchgeführt. Dies kann in umfangreichen Protokolldateien und Datenverlusten resultieren, wenn Operationen unterbrochen werden.

-exceptionFile *ausnahmedatei*
Dieser Parameter gibt den vollständigen Pfadnamen der Formdatei an, in der die Formdaten geschrieben werden sollen, die nicht importiert werden konnten. Wenn für den Parameter kein Wert angegeben ist, wird keine Ausnahmedatei erstellt.

Falls Sie einen Wert für den Parameter angeben und eine optionale Dateierweiterung hinzufügen wollen, verwenden Sie .shp oder .SHP. Wenn Sie keine Erweiterung angeben, wird die Erweiterung `.shp` an die Ausnahmedatei *ausnahmedatei* angehängt.

Die Ausnahmedatei enthält den vollständigen Satz Zeilen für die fehlgeschlagene Einfügeanweisung. Eine Einfügeanweisung kann mehrere Zeilen hinzufügen. Beispiel: Angenommen, eine Zeile kann nicht importiert werden, weil die Formdaten nicht richtig codiert sind. Eine einzelne Einfügeanweisung versucht, 20 Zeilen (einschließlich der Zeile mit den falschen Formdaten) zu importieren. Da die Einfügeanweisung fehlschlägt, wird das gesamte Set von 20 Zeilen in die Ausnahmedatei geschrieben.

Datensätze werden nur dann in die Ausnahmedatei geschrieben, wenn sie korrekt identifiziert werden können (beispielsweise dann, wenn der Formdatensatztyp nicht gültig ist). Manchmal sind Formdaten (Dateien .shp) und Formindizes (Dateien .shx) auf eine Weise beschädigt, bei der die entsprechenden Datensätze nicht identifiziert werden können. In einem solchen Fall können keine Datensätze in die Ausnahmedatei geschrieben werden und das Problem wird in Form einer Fehlernachricht gemeldet.

Falls Sie für diesen Parameter einen Wert angeben, werden auf dem DB2-Server vier Dateien erstellt. Eine Erläuterung dieser Dateien finden Sie unter „Hinweise zur Verwendung" auf Seite 163.

Wenn die Datei *ausnahmedatei* bereits vorhanden ist, gibt der Befehl einen Fehler zurück.

Die maximale Länge für diesen Parameter beträgt 256 Zeichen.

-messagesFile *nachrichtendateiname*
Gibt den vollständigen Pfadnamen der Datei auf dem DB2-Server an, in die DB2 Spatial Extender Nachrichten über die Importoperation schreibt. Wenn Sie diesen Parameter nicht angeben, erstellt DB2 Spatial Extender keine Nachrichtendatei.

Folgende Nachrichtentypen werden in die Nachrichtendatei geschrieben:

- Informationsnachrichten, beispielsweise eine Zusammenfassung der Importoperation
- Fehlernachrichten für Daten, die nicht importiert werden konnten, beispielsweise aufgrund eines abweichenden Koordinatensystems. Diese Fehlernachrichten entsprechen den Formdaten, die in der Ausnahmedatei *ausnahmedatei* gespeichert werden.

Wenn die Datei *nachrichtendateiname* bereits vorhanden ist, gibt der Befehl einen Fehler zurück.

Die maximale Länge für diesen Parameter beträgt 256 Zeichen.

-client *markierung_für_client*
Dieser Parameter gibt an, ob die Importoperation auf dem Client oder dem DB2-Server stattfindet und wo die Dateien erstellt werden. Folgende Parameterwerte sind möglich:

- 0 (null), um anzugeben, dass die Importoperation auf dem DB2-Server ausgeführt wird und auf die Dateien über den DB2-Server zugegriffen wird.
- 1, um anzugeben, dass die Importoperation auf dem Client ausgeführt wird und auf die Dateien über den Client zugegriffen wird.

Wenn Sie keinen Wert für diesen Parameter angeben, lautet der Standardwert 0 (null).

Hinweise zur Verwendung

Sie können den Importprozess auf dem Client ausführen, auf dem auch der Befehl ausgeführt wird. Dies ist in vielen Fällen praktischer, da kein Zugriff auf das DB2-Serverdateisystem erforderlich ist.

Der Befehl **db2se import_shape** erstellt die folgenden vier Dateien bzw. schreibt in diese Dateien:

- die eigentliche Formdatei (Erweiterung .shp). Diese Datei ist erforderlich.
- die Formindexdatei (Erweiterung .shx). Diese Datei ist optional. Wenn sie vorhanden ist, kann die Leistung der Importoperation möglicherweise gesteigert werden.
- eine dBASE-Datei, die Attributdaten enthält (Erweiterung .dbf). Diese Datei ist nur dann erforderlich, wenn Attributdaten importiert werden sollen.
- die Projektionsdatei, die das Koordinatensystem der Formdaten angibt (Erweiterung .prj). Diese Datei ist optional. Wenn sie vorhanden ist, wird das in ihr definierte Koordinatensystem mit dem Koordinatensystem des räumlichen Bezugssystems verglichen, das im Parameter *id_des_räumlichen_bezugssystems* angegeben ist.

Die folgende Tabelle beschreibt, wie dBASE-Attributdatentypen den DB2-Datentypen zugeordnet werden. Alle nicht angegebenen Attributdatentypen werden nicht unterstützt.

Tabelle 21. Beziehung zwischen DB2-Datentypen und dBASE-Attributdatentypen

Typ bei .dbf	Länge bei .dbf (siehe Anmerkung)	Dezimalstellen bei .dbf (siehe Anmerkung)	SQL-Typ	Kommentare
N	< 5	0	SMALLINT	
N	< 10	0	INTEGER	
N	< 20	0	BIGINT	
N	*länge*	*dezimalstellen*	DECIMAL (*länge*,*dezimalstellen*)	*länge*<32
F	*länge*	*dezimalstellen*	REAL	*länge + dezimalstellen* < 7
F	*länge*	*dezimalstellen*	DOUBLE	
C	*länge*		CHAR(*länge*)	
L			CHAR(1)	
D			DATE	

Anmerkung: Diese Tabelle enthält die folgenden beiden Variablen, die in den Kopfdaten der dBASE-Datei definiert sind:

- Die Variable *länge* stellt die Gesamtlänge der Spalte in der dBASE-Datei dar. DB2 Spatial Extender verwendet diesen Wert, um
  - die Genauigkeit für den SQL-Datentyp DECIMAL oder die Länge für den SQL-Datentyp CHAR zu definieren,
  - den zu verwendenden Integer- oder Gleitkommatyp zu ermitteln.
- Die Variable *dezimalstellen* gibt an, wie viele Stellen rechts vom Dezimalzeichen der Spalte in der dBASE-Datei maximal zulässig sind. DB2 Spatial Extender verwendet diesen Wert, um die Anzahl der Kommastellen für den SQL-Datentyp DECIMAL zu definieren.

Beispiel: Angenommen, die dBASE-Datei enthält eine Datenspalte, deren Länge (*länge*) mit dem Wert 20 definiert ist. Für die Anzahl der Stellen rechts vom Dezimalzeichen (*dec*) wird der Wert 5 angenommen. Beim Importieren von Daten aus dieser Spalte leitet DB2 Spatial Extender aus den Werten der Variablen *länge* und *dezimalstellen* den folgenden SQL-Datentyp ab: DECIMAL(20,5).

Beispiel

Der folgende Befehl importiert die Daten aus der Formdatei `myfile`, die auf dem Client gespeichert ist, in die Tabelle MYTABLE. Die räumlichen Daten aus der Datei `myfile` werden in die Spalte MYCOLUMN der Tabelle MYTABLE eingefügt.

```
db2se import_shape mydb -fileName myfile -srsName NAD83_SRS_1
        -tableName mytable -spatialColumnName mycolumn -client 1
```

Befehl 'db2se register_gc'

Der Befehl **`db2se register_gc`** registriert einen Geocoder.

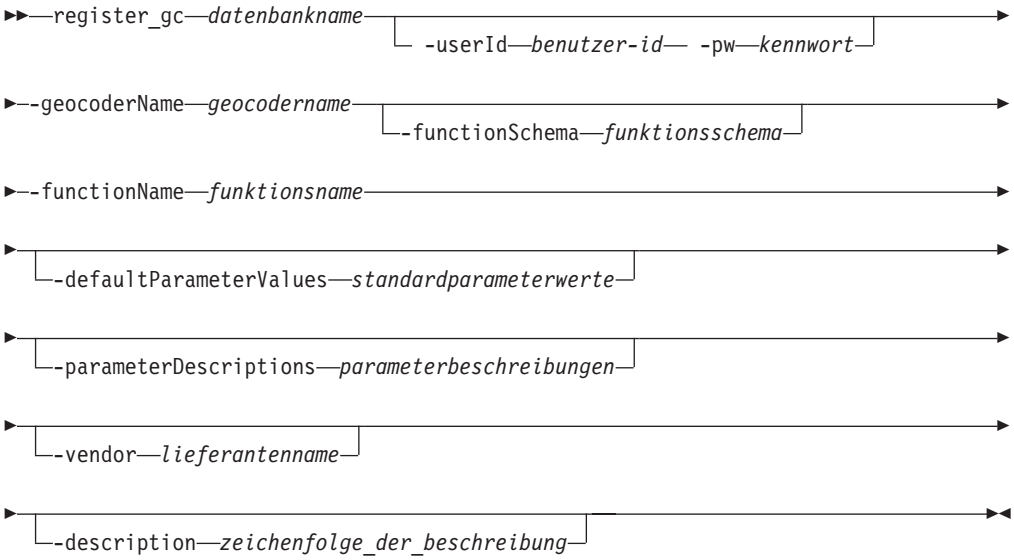
Dieser Befehl registriert einen Geocoder, damit dieser für die Geocodierung von Werten in einer Tabelle und einem Speicher oder für die Aktualisierung der resultierenden Geometriewerte verwendet werden kann. Die Informationen zu registrierten Geocodern sind in der Katalogsicht DB2GSE.ST_GEOCODERS abrufbar.

Berechtigung

Die Benutzer-ID muss über die Berechtigungen DBADM und DATAACCESS für die Datenbank verfügen, die für räumliche Operationen aktiviert ist, um diesen Befehl auszuführen.

Befehlssyntax

Befehl 'db2se register_gc'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie einen Geocoder registrieren möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-geocoderName *geocodername*
Gibt den Geocoder, den Sie registrieren möchten, eindeutig an. Der Wert für *geocodername* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen. Die maximale Länge für diesen Parameter beträgt 128 Zeichen.

-functionSchema *funktionsschema*
Gibt den Schemanamen für die Funktion an, die diesen Geocoder implementiert. Falls Sie für diesen Parameter keinen Wert angeben, wird als Schemaname für die Funktion der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

Der Wert für *funktionsschema* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-functionName *funktionsname*
Gibt den Namen ohne Qualifikationsmerkmal der Funktion an, die diesen Geocoder implementiert. Die Funktion muss bereits erstellt worden und in der Katalogsicht SYSCAT.ROUTINES aufgeführt sein.

Der Wert *funktionsname* muss zusammen mit dem implizit oder explizit definierten Wert *funktionsschema* die eindeutige Kennzeichnung der Funktion ergeben.

Der Wert für *funktionsname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-defaultParameterValues *standardparameterwerte*
Gibt eine Liste der Standardwerte der Geocodierungsparameter für die Geocoderfunktion an.

Sie müssen die Parameterwerte in der Reihenfolge angeben, in der die Parameter durch die Funktion definiert sind, und die einzelnen Werte durch ein Komma voneinander abgrenzen. Beispiel:

```
standardwert_für_parameter1,standardwert_für_parameter2,...
```

Jeder Parameterwert muss ein SQL-Ausdruck sein. Beachten Sie bei der Angabe der Standardparameterwerte die folgenden Richtlinien:

- Ein Zeichenfolgewert muss in einfache Anführungszeichen gesetzt werden.
- Ein Zahlenwert darf nicht in einfache Anführungszeichen gesetzt werden.
- Wenn der Parameterwert NULL ist, muss er in den korrekten Typ umgesetzt werden. Geben Sie beispielsweise den folgenden Ausdruck an, um für einen ganzzahligen Parameter den Wert NULL festzulegen:

  ```
  CAST(NULL AS INTEGER)
  ```

- Wenn der Geocodierungsparameter eine Geocodierungsspalte sein soll, geben Sie einen Standardwert für den Parameter an.

Verwenden Sie zwei aufeinanderfolgende Kommas (`...,,...`), um Standardwerte für Parameter auszuschließen, die Sie bei der Konfiguration der Geocodierung oder der Ausführung der Geocodierung im Stapelmodus angeben, indem Sie den Parameter **-parameterValues** mit dem Befehl **db2se setup_gc** oder dem Befehl **db2se run_gc** verwenden.

Die maximale Länge für diesen Parameter beträgt 32.672 Zeichen.

-parameterDescriptions *parameterbeschreibungen*
Dieser Parameter gibt eine Liste der Beschreibungen der Geocodierungsparameter für die Geocoderfunktion an. Die maximale Länge für diesen Parameter beträgt 32.672 Zeichen.

Jede angegebene Parameterbeschreibung erläutert die Bedeutung und die Verwendung des Parameters. Sie kann bis zu 256 Zeichen lang sein. Die Beschreibungen der Parameter müssen durch Kommas voneinander abgegrenzt werden und in der Reihenfolge erscheinen, in der die Parameter durch die Funktion definiert sind. Um innerhalb der Beschreibung eines Parameters ein Komma zu verwenden, setzen Sie die Zeichenfolge in einfache oder doppelte Anführungszeichen. Beispiel:

```
beschreibung1,'beschreibung2, die ein komma enthält',beschreibung3
```

-vendor *lieferantenname*
Gibt den Namen des Lieferanten an, der den Geocoder implementiert hat. Die maximale Länge für diesen Parameter beträgt 128 Zeichen.

-description *zeichenfolge_der_beschreibung*
Dieser Parameter beschreibt den Geocoder, indem seine Anwendung erläutert wird. Die maximale Länge für diesen Parameter beträgt 256 Zeichen.

Hinweise zur Verwendung

Der Rückgabetyp der Geocoderfunktion muss mit dem Datentyp der geocodierten Spalte identisch sein. Als Geocodierungsparameter können Spaltennamen (so genannte Geocodierungsspalten) verwendet werden, die vom Geocoder benötigte Daten enthalten. Die Geocoderparameter können beispielsweise Adressen oder einen Wert mit einer bestimmten Bedeutung für den Geocoder (z. B. die Mindestübereinstimmungsquote) angeben. Wenn es sich bei einem Geocodierungsparameter um einen Spaltennamen handelt, muss die Spalte in derselben Tabelle oder Sicht wie die geocodierte Spalte enthalten sein.

Der Rückgabetyp der Geocoderfunktion dient als Datentyp für die geocodierte Spalte. Der Rückgabetyp kann ein beliebiger DB2-Datentyp, benutzerdefinierter Typ oder strukturierter Typ sein. Wenn ein benutzerdefinierter oder ein strukturierter Typ zurückgegeben wird, muss die Geocoderfunktion dafür sorgen, dass ein gültiger Wert des entsprechenden Datentyps zurückgegeben wird. Gibt die Geocoderfunktion Werte eines räumlichen Typs zurück (also ST_Geometry oder einer seiner untergeordneten Typen), muss die Geocoderfunktion dafür sorgen, dass eine gültige Geometrie erstellt wird. Die Geometrie muss mithilfe eines vorhandenen räumlichen Bezugssystems dargestellt werden. Die Geometrie ist gültig, wenn Sie die räumliche Funktion ST_IsValid für die Geometrie aufrufen und der Wert 1 zurückgegeben wird. Die von der Geocoderfunktion zurückgegebenen Daten werden in der geocodierten Spalte aktualisiert oder in die Spalte eingefügt. Dies ist davon abhängig, welche Operation (INSERT oder UPDATE) die Generierung des geocodierten Werts verursacht hat.

Beispiel

Im folgenden Beispiel wird ein Geocoder namens „mygeocoder" registriert, der durch eine Funktion mit dem Namen „myschema.myfunction" implementiert wird.

```
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"
```

Befehl 'db2se register_spatial_column'

Der Befehl **db2se register_spatial_column** registriert eine räumliche Spalte und ordnet ihr ein räumliches Bezugssystem zu.

Beim Registrieren einer räumlichen Spalte wird (sofern möglich) eine Integritätsbedingung für die Tabelle erstellt, die sicherstellen soll, dass alle Geometrien das angegebene räumliche Bezugssystem verwenden.

Mithilfe dieses Befehls können Sie auch die Informationen zum räumlichen Bezugssystem aktualisieren.

Die Informationen zu registrierten räumlichen Spalten und räumlichen Bereichen sind in der Katalogsicht DB2GSE.ST_GEOMETRY_COLUMNS abrufbar.

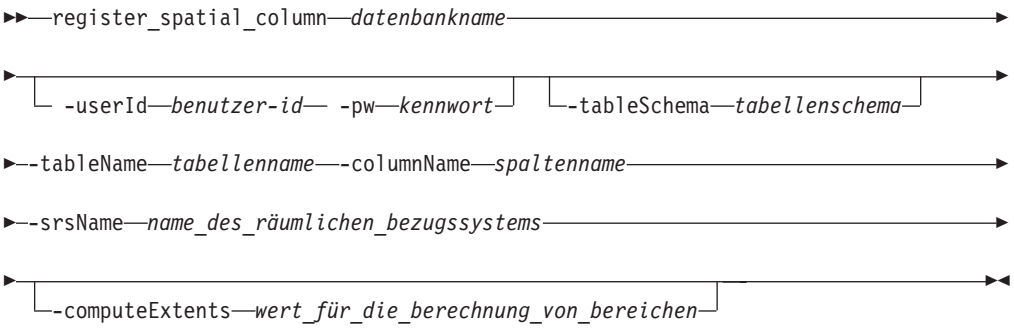
Berechtigung

Die Benutzer-ID muss über eine der folgenden Berechtigungen zur Ausführung dieses Befehls verfügen:

- Berechtigungen DBADM und DATAACCESS für die Datenbank mit der Tabelle, zu der die räumliche Spalte gehört, die registriert werden soll
- Zugriffsrecht CONTROL für diese Tabelle
- Zugriffsrecht ALTER für diese Tabelle

Befehlssyntax

Befehl 'db2se register_spatial_column'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie eine räumliche Spalte registrieren möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-tableSchema *tabellenschema*
Gibt den Schemanamen für die im Parameter *tabellenname* angegebene Tabelle an. Falls Sie keinen Schemanamen angeben, wird als Schemaname für die Tabelle bzw. Sicht der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

-tableName *tabellenname*
Gibt den Namen ohne Qualifikationsmerkmal der Tabelle an, zu der die registrierte Spalte gehört. Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-columnName *spaltenname*
Gibt den Namen der zu registrierenden Spalte an. Der Wert für *spaltenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-srsName *name_des_räumlichen_bezugssystems*
Gibt das räumliche Bezugssystem an, das für diese räumliche Spalte verwendet

werden soll. Der Wert für *name_des_räumlichen_bezugssystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

computeExtents *wert_für_die_berechnung_von_bereichen*
Gibt an, ob die geografischen Bereiche einer angegebenen Spalte berechnet und in der Katalogsicht DB2GSE.ST_GEOMETRY_COLUMNS zur Verfügung gestellt werden sollen. Folgende Parameterwerte sind möglich:

- Ein Wert größer als 0 (null), um die geografischen Bereiche zu berechnen.
- Null, 0 oder ein negativer Wert, um diese Berechnung zu verhindern.

Wenn Sie für diesen Parameter keinen Wert angeben, wird der Standardwert 0 (null) angenommen. Es wird keine Bereichsberechnung durchgeführt.

Beispiel

Im folgenden Beispiel wird eine räumliche Spalte namens MYCOLUMN in der Tabelle MYTABLE mit dem räumlichen Bezugssystem „USA_SRS_1" registriert.

```
db2se register_spatial_column mydb -tableName mytable -columnName mycolumn
  -srsName USA_SRS_1
```

Befehl 'db2se remove_gc_setup'

Mit dem Befehl **db2se remove_gc_setup** werden alle Informationen zur Konfiguration der Geocodierung für eine geocodierte Spalte entfernt.

Die Informationen, die der angegebenen geocodierten Spalte zugeordnet sind, sind in den Katalogsichten DB2GSE.ST_GEOCODING und DB2GSE.ST_GEOCODING_PARAMETERS nicht länger verfügbar.

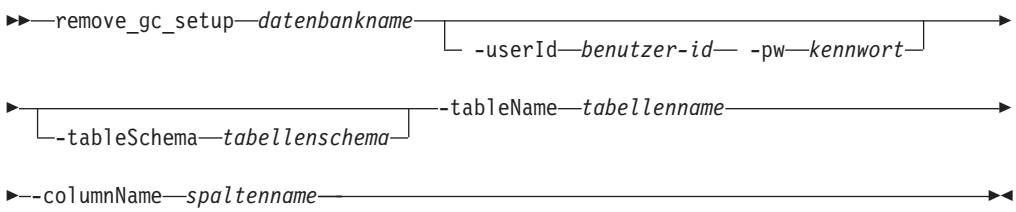
Berechtigung

Die Benutzer-ID muss über eine der folgenden Berechtigungen zur Ausführung dieses Befehls verfügen:

- Berechtigung DATAACCESS für die Datenbank mit der Tabelle, für die der angegebene Geocoder ausgeführt werden soll.
- Zugriffsrecht CONTROL für diese Tabelle
- Zugriffsrecht UPDATE für diese Tabelle

Befehlssyntax

Befehl 'db2se remove_gc_setup'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie alle Informationen zur Konfiguration der Geocodierung für eine geocodierte Spalte entfernen möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-tableSchema *tabellenschema*
Gibt den Schemanamen für die im Parameter *tabellenname* angegebene Tabelle an. Falls Sie keinen Schemanamen angeben, wird als Schemaname für die Tabelle bzw. Sicht der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

-tableName *tabellenname*
Gibt den Namen ohne Qualifikationsmerkmal der Tabelle für den im Parameter *spaltenname* angegebenen Spaltennamen an. Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-columnName *spaltenname*
Gibt den Namen der Spalte an, aus der die Konfiguration der Geocodierung entfernt werden soll. Der Wert für *spaltenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Hinweise zur Verwendung

Wenn die automatische Geocodierung für die geocodierte Spalte aktiviert ist, kann die Geocodierungskonfiguration nicht entfernt werden.

Beispiel

Im folgenden Beispiel wird eine Einrichtung für Geocodierungsoperationen entfernt, die auf eine räumliche Spalte namens "MYCOLUMN" in der Tabelle "MYTABLE" angewendet werden.

```
db2se remove_gc_setup mydb -tableName mytable -columnName mycolumn
```

Befehl 'db2se restore_indexes'

Der Befehl **db2se restore_indexes** stellt die räumlichen Indizes wieder her, die Sie zuvor mit dem Befehl **db2se save_indexes** in einer für räumliche Operationen aktivierten Datenbank gespeichert hatten.

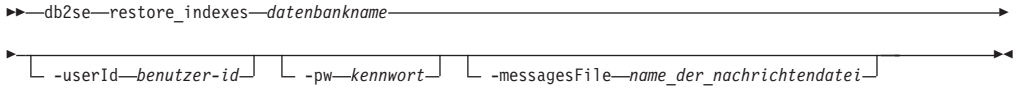
Dieser Befehl wird eingesetzt, um räumliche Indizes nach dem Upgrade von einer 32-Bit- auf eine 64-Bit-Instanz erneut zu erstellen.

Berechtigung

Berechtigungen DBADM und DATAACCESS für die Datenbank, die für räumliche Operationen aktiviert ist.

Befehlssyntax

Befehl 'db2se restore_indexes'



Befehlsparameter

datenbankname
Der Name der Datenbank, für die ein Upgrade durchgeführt werden soll.

-userId benutzer_id
Die Datenbankbenutzer-ID, die entweder die Berechtigung SYSADM oder die Berechtigung DBADM für die zu aktualisierende Datenbank besitzt.

-pw kennwort
Ihr Benutzerkennwort.

-messagesFile name_der_nachrichtendatei
Der Name der Datei, die den Bericht über die Migrationsaktionen enthält. Der von Ihnen angegebene Dateiname muss ein vollständig qualifizierter Name auf dem Server sein.

Befehl 'db2se save_indexes'

Der Befehl 'db2se save_indexes' speichert die räumlichen Indizes, die in einer für räumliche Operationen aktivierten Datenbank definiert sind.

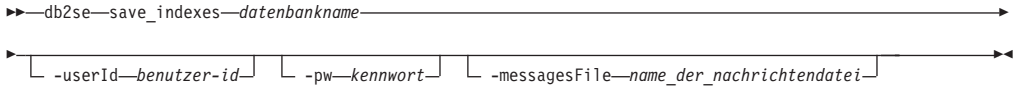
Dieser Befehl wird eingesetzt, um Definitionen aktueller räumlicher Indizes beim Upgrade von einer 32-Bit- auf eine 64-Bit-Instanz zu sichern.

Berechtigung

Berechtigungen DBADM und DATAACCESS für die Datenbank, die für räumliche Operationen aktiviert ist.

Befehlssyntax

Befehl 'db2se save_indexes'



Befehlsparameter

datenbankname
Der Name der Datenbank, für die ein Upgrade durchgeführt werden soll.

-userId benutzer_id
Die Datenbankbenutzer-ID, die entweder die Berechtigung SYSADM oder die Berechtigung DBADM für die zu aktualisierende Datenbank besitzt.

-pw kennwort
Ihr Benutzerkennwort.

-messagesFile name_der_nachrichtendatei
Der Name der Datei, die den Bericht über die Migrationsaktionen enthält. Der von Ihnen angegebene Dateiname muss ein vollständig qualifizierter Name auf dem Server sein.

Befehl 'db2se run_gc'

Der Befehl **db2se run_gc** führt einen Geocoder für eine geocodierte Spalte im Stapelmodus aus.

Die Informationen, die der angegebenen geocodierten Spalte zugeordnet sind, sind in den Katalogsichten DB2GSE.ST_GEOCODING und DB2GSE.ST_GEOCODING_PARAMETERS nicht länger verfügbar.

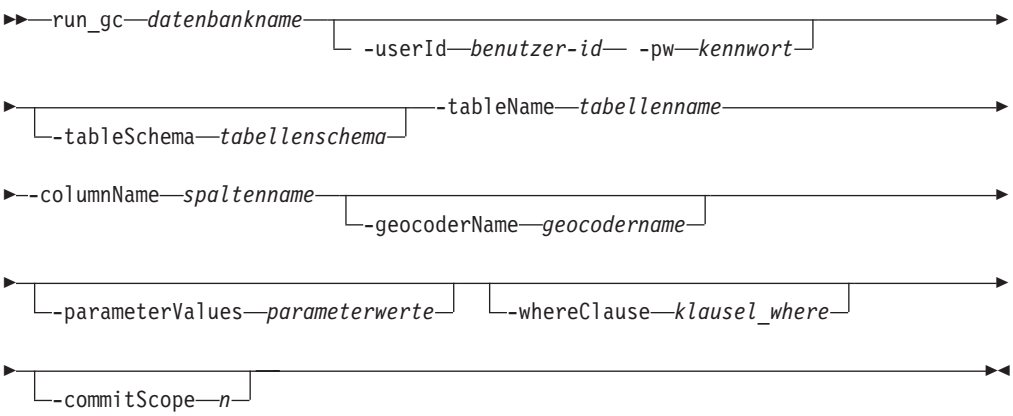
Berechtigung

Die Benutzer-ID muss über eine der folgenden Berechtigungen zur Ausführung dieses Befehls verfügen:

- Berechtigung DATAACCESS für die Datenbank mit der Tabelle, für die der angegebene Geocoder ausgeführt werden soll.
- Zugriffsrecht CONTROL für diese Tabelle
- Zugriffsrecht UPDATE für diese Tabelle

Befehlssyntax

Befehl 'db2se run_gc'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie einen Geocoder für eine geocodierte Spalte im Stapelmodus ausführen möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-tableSchema *tabellenschema*
Gibt den Schemanamen für die im Parameter *tabellenname* angegebene Tabelle an. Falls Sie keinen Schemanamen angeben, wird als Schemaname für die Tabelle bzw. Sicht der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

-tableName *tabellenname*
Gibt den Namen ohne Qualifikationsmerkmal der Tabelle für den im Parameter *spaltenname* angegebenen Spaltennamen an. Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-columnName *spaltenname*
Gibt den Namen der Spalte an, in der die geocodierten Daten eingefügt oder aktualisiert werden sollen. Der Wert für *spaltenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-geocoderName *geocodername*
Gibt den Namen des Geocoders, der die Geocodierung vornehmen soll, eindeutig an. Der Wert für *geocodername* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen. Die maximale Länge für diesen Parameter beträgt 128 Zeichen.

-parameterValues *parameterwerte*
Gibt eine Liste der Geocodierungsparameterwerte für die Geocoderfunktion an. Wenn für diesen Parameter kein Wert angegeben ist, werden entweder die Parameterwerte verwendet, die beim Konfigurieren des Geocoders angegeben wurden, oder die Standardparameterwerte, die bei der Registrierung des Geocoders angegeben wurden.

Sie müssen die Parameterwerte in der Reihenfolge angeben, in der die Parameter durch die Funktion definiert sind, und die einzelnen Werte durch ein Komma voneinander trennen. Beispiel:

```
standardwert_für_parameter1,standardwert_für_parameter2,...
```

Jeder Parameterwert muss ein SQL-Ausdruck sein. Beachten Sie bei der Angabe der Standardparameterwerte die folgenden Richtlinien:

- Ein Zeichenfolgewert muss in einfache Anführungszeichen gesetzt werden.
- Ein Zahlenwert darf nicht in einfache Anführungszeichen gesetzt werden.
- Wenn der Parameterwert NULL ist, muss er in den korrekten Typ umgesetzt werden. Geben Sie beispielsweise den folgenden Ausdruck an, um für einen ganzzahligen Parameter den Wert NULL festzulegen: `CAST(NULL AS INTEGER)`
- Wenn der Geocodierungsparameter eine Geocodierungsspalte sein soll, geben Sie einen Standardwert für den Parameter an.

Verwenden Sie zwei aufeinanderfolgende Kommas (`...,,...`), um Werte für Parameter auszuschließen, für die Sie bei der Konfiguration oder der Registrierung des Geocoders einen Wert angegeben haben.

Die maximale Länge für diesen Parameter beträgt 32.672 Zeichen.

-whereClause *klausel_where*
Gibt den Text für eine Suchbedingung der Klausel WHERE an, um nach den Datensätzen zu filtern, die geocodiert werden sollen. Wenn für diesen Parameter kein Wert angegeben ist, wird der während der Konfiguration der Geocodierung angegebene Wert von *klausel_where* verwendet. Wenn während der Konfiguration der Geocodierung kein Wert für *klausel_where* angegeben wurde, werden alle Zeilen in der Tabelle geocodiert.

Die angegebene Klausel kann sich auf eine beliebige Spalte in der Tabelle oder Sicht beziehen, für die der Geocoder ausgeführt werden soll.

Das Schlüsselwort WHERE darf im Parameter *klausel_where* nicht angegeben werden.

Die maximale Länge für diesen Parameter beträgt 32.672 Zeichen.

-commitScope *n*
Dieser Parameter gibt an, dass ein Commit durchgeführt werden soll, nachdem jeweils *n* Datensätze geocodiert wurden. Wenn für diesen Parameter kein Wert angegeben ist, wird der während der Konfiguration der Geocodierung angegebene Wert für den Parameter **-commitScope** verwendet. Wenn für diesen Parameter kein Wert angegeben ist und auch während der Konfiguration der Geocodierung kein Wert angegeben wurde, wird am Ende der Operation ein Commit durchgeführt. Ein COMMIT am Ende der Operation kann in umfangreichen Protokolldateien und Datenverlusten resultieren, wenn Operationen unterbrochen werden.

Beispiel

Im folgenden Beispiel wird ein Geocoder im Stapelmodus ausgeführt, um eine Spalte namens "MYCOLUMN" in einer Tabelle "MYTABLE" zu füllen.

```
db2se run_gc mydb -tableName mytable -columnName mycolumn
```

Befehl 'db2se setup_gc'

Der Befehl **db2se setup_gc** ordnet einer Spalte, die geocodiert werden soll, einen Geocoder zu und definiert die entsprechenden Geocodierungsparameter.

Die Informationen zu dieser Konfiguration können Sie in den Katalogsichten DB2GSE.ST_GEOCODING und DB2GSE.ST_GEOCODING_PARAMETERS abfragen.

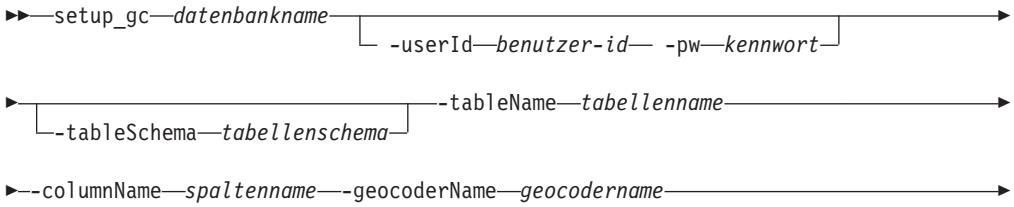
Berechtigung

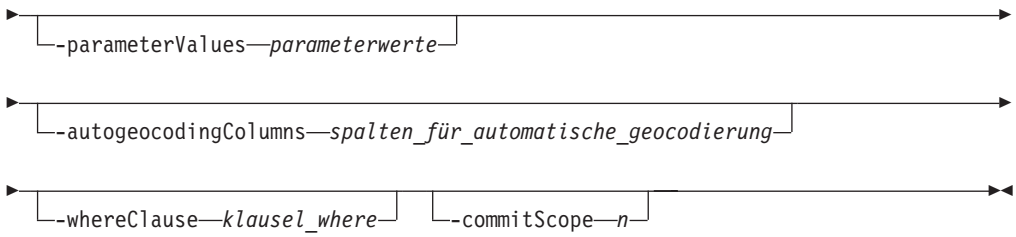
Die Benutzer-ID muss über eine der folgenden Berechtigungen zur Ausführung dieses Befehls verfügen:

- Berechtigung DATAACCESS für die Datenbank mit der Tabelle, für die der angegebene Geocoder ausgeführt werden soll.
- Zugriffsrecht CONTROL für diese Tabelle
- Zugriffsrecht UPDATE für diese Tabelle

Befehlssyntax

Befehl 'db2se setup_gc'





Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie einen Geocoder konfigurieren möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-tableSchema *tabellenschema*
Gibt den Schemanamen für die im Parameter *tabellenname* angegebene Tabelle an. Falls Sie keinen Schemanamen angeben, wird als Schemaname für die Tabelle bzw. Sicht der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

-tableName *tabellenname*
Gibt den Namen ohne Qualifikationsmerkmal der Tabelle für den im Parameter *spaltenname* angegebenen Spaltennamen an. Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-columnName *spaltenname*
Gibt den Namen der Spalte an, in der die geocodierten Daten eingefügt oder aktualisiert werden sollen. Der Wert für *spaltenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-geocoderName *geocodername*
Gibt den Namen eines bereits registrierten Geocoders, der die Geocodierung vornehmen soll, eindeutig an. Der Wert für *geocodername* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen. Die maximale Länge für diesen Parameter beträgt 128 Zeichen.

-parameterValues *parameterwerte*
Dieser Parameter gibt eine Liste der Geocodierungsparameterwerte für die Geocoderfunktion an. Wenn für diesen Parameter kein Wert angegeben wird, werden die Standardparameterwerte verwendet, die beim Registrieren des Geocoders angegeben wurde.

Sie müssen die Parameterwerte in der Reihenfolge angeben, in der die Parameter durch die Funktion definiert sind, und die einzelnen Werte durch ein Komma voneinander abgrenzen. Beispiel:

```
standardwert_für_parameter1,standardwert_für_parameter2,...
```

Jeder Parameterwert muss ein SQL-Ausdruck sein. Beachten Sie bei der Angabe der Standardparameterwerte die folgenden Richtlinien:

- Ein Zeichenfolgewert muss in einfache Anführungszeichen gesetzt werden.
- Ein Zahlenwert darf nicht in einfache Anführungszeichen gesetzt werden.
- Wenn der Parameterwert NULL ist, muss er in den korrekten Typ umgesetzt werden. Geben Sie beispielsweise den folgenden Ausdruck an, um für einen ganzzahligen Parameter den Wert NULL festzulegen: `CAST(NULL AS INTEGER)`
- Wenn der Geocodierungsparameter eine Geocodierungsspalte sein soll, geben Sie einen Standardwert für den Parameter an.

Verwenden Sie zwei aufeinanderfolgende Kommas (`...,,...`), um Werte für Parameter auszuschließen, für die Sie bei der Konfiguration oder der Registrierung des Geocoders einen Wert angegeben haben.

Die maximale Länge für diesen Parameter beträgt 32.672 Zeichen.

-whereClause *klausel_where*
Gibt den Text für eine Suchbedingung der Klausel WHERE an, um nach den Datensätzen zu filtern, die geocodiert werden sollen. Wenn für diesen Parameter kein Wert angegeben wird, werden alle Zeilen in der Tabelle geocodiert.

Die angegebene Klausel kann sich auf eine beliebige Spalte in der Tabelle oder Sicht beziehen, für die der Geocoder ausgeführt werden soll.

Das Schlüsselwort WHERE darf im Parameter *klausel_where* nicht angegeben werden.

Die maximale Länge für diesen Parameter beträgt 32.672 Zeichen.

-commitScope *n*
Dieser Parameter gibt an, dass ein COMMIT durchgeführt werden soll, nachdem jeweils *n* Datensätze geocodiert wurden. Wenn für diesen Parameter kein Wert angegeben ist, wird am Ende der Operation ein Commit durchgeführt. Ein COMMIT am Ende der Operation kann in umfangreichen Protokolldateien und Datenverlusten resultieren, wenn Operationen unterbrochen werden.

Hinweise zur Verwendung

Dieser Befehl ruft keine Geocodierungsoperation auf. Sie ist vielmehr eine schnelle Methode, um Parametereinstellungen für die Spalte anzugeben, die geocodiert werden soll. Die Parametereinstellungen, die bei der Konfiguration der Geocodierung angegeben werden, setzen alle Standardparameterwerte außer Kraft, die bei der Registrierung des Geocoders angegeben wurden.

Sie müssen diesen Befehl ausführen, bevor Sie die Geocodierung aktivieren. Das Definieren der Parameter für die Geocodierung ist erforderlich, bevor die automatische Geocodierung aktiviert wird.

Sie können diesen Befehl absetzen, bevor Sie die Geocodierung im Stapelmodus ausführen. Wenn Sie keine Parameterwerte für die Ausführung der Geocodierung im Stapelmodus festlegen, werden die bei der Konfiguration der Geocodierung angegebenen Parameterwerte verwendet. Wenn Sie Parameterwerte angeben, setzen diese Werte die bei der Konfiguration der Geocodierung angegebenen Parameterwerte außer Kraft.

Beispiel

Im folgenden Beispiel werden Geocodierungsoperationen definiert, die eine räumliche Spalte namens "MYCOLUMN" in der Tabelle "MYTABLE" füllen.

```
db2se setup_gc mydb -tableName mytable -columnName mycolumn
        -parameterValues "address,city,state,zip,2,90,70,20,1.1,'meter',4.."
        -autogeocodingColumns address,city,state,zip
        -commitScope 10
```

Befehl 'db2se shape_info'

Der Befehl **db2se shape_info** zeigt Informationen zu einer Formdatei und ihrem Inhalt an. Für eine angegebene Datenbank kann dieser Befehl optional alle kompatiblen Koordinatensysteme und räumlichen Bezugssysteme angeben, die in der Formdatei enthalten sind.

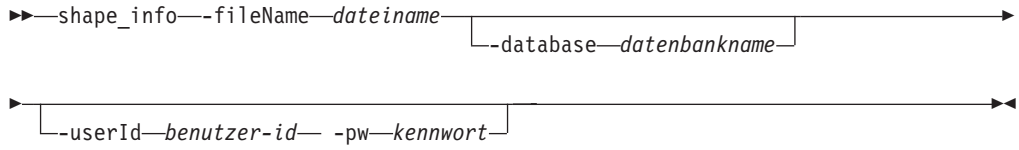
Berechtigung

Die Benutzer-ID muss über die erforderlichen Berechtigungen für den DB2-Server verfügen, um die Formdateien zu lesen.

Wenn der Parameter **-database** angegeben wird, muss die Benutzer-ID über die Berechtigung SELECT für die Katalogsicht DB2GSE.ST_COORDINATE_SYSTEMS verfügen.

Befehlssyntax

Befehl 'db2se shape_info'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

-fileName *dateiname*
Gibt den vollständigen Pfadnamen der Formdatei an, für die Informationen angezeigt werden sollen.

DB2 Spatial Extender sucht zunächst nach einer exakten Übereinstimmung mit dem Namen, den Sie mit dem Parameter **-fileName** angegeben haben. Falls DB2 Spatial Extender keine exakte Übereinstimmung feststellt, wird zunächst nach einer Datei mit der Erweiterung `.shp` und dann nach einer Datei mit der Erweiterung `.SHP` gesucht.

Die maximale Länge für diesen Parameter beträgt 256 Zeichen.

-database *datenbankname*
Gibt den Namen der Datenbank an, für die Sie alle kompatiblen Koordinatensysteme und räumlichen Bezugssysteme suchen möchten, die in der Formdatei *dateiname* enthalten sind.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

Beispiel

Im folgenden Beispiel werden Informationen zu einer Formdatei namens MYFILE angezeigt, die sich im aktuellen Verzeichnis befindet.

```
db2se shape_info -fileName myfile
```

Im folgenden Beispiel werden Informationen zu einer UNIX-Beispielformdatei mit dem Namen offices angezeigt. Der Parameter *–database* sucht alle kompatiblen Koordinatensysteme und räumlichen Bezugssysteme in der angegeben Datenbank (hier MYDB).

```
db2se shape_info -fileName ~/sqllib/samples/extenders/spatial/data/offices
-database myDB
```

Befehl 'db2se unregister_gc'

Der Befehl **db2se unregister_gc** nimmt die Registrierung eines Geocoders zurück.

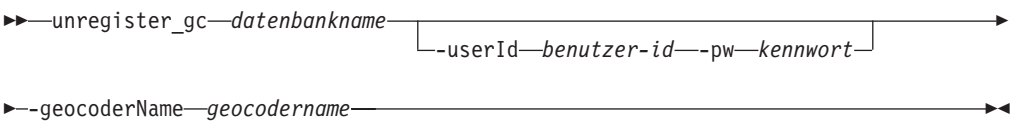
Informationen zum Geocoder, dessen Registrierung zurückgenommen werden soll, finden Sie in der Katalogsicht DB2GSE.ST_GEOCODERS.

Berechtigung

Die Benutzer-ID muss über die Berechtigungen DBADM und DATAACCESS für die Datenbank verfügen, die für räumliche Operationen aktiviert ist, um diesen Befehl auszuführen.

Befehlssyntax

Befehl 'db2se unregister_gc'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie die Registrierung eines Geocoders aufheben möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-geocoderName *geocodername*
Gibt den Geocoder, für den Sie die Registrierung aufheben möchten, eindeutig an. Der Wert für *geocodername* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen. Die maximale Länge für diesen Parameter beträgt 128 Zeichen.

Hinweise zur Verwendung

Die Registrierung eines Geocoders kann nicht zurückgenommen werden, wenn er in der Geocodierungskonfiguration einer Spalte angegeben ist. In den Katalogsichten DB2GSE.ST_GEOCODING und DB2GSE.ST_GEOCODING_PARAMETERS können Sie ermitteln, ob ein Geocoder in der Geocodierungskonfiguration für eine Spalte angegeben ist.

Beispiel

Im folgenden Beispiel wird die Registrierung eines Geocoders namens MYGEOCODER zurückgenommen.

```
db2se unregister_gc mydb -geocoderName mygeocoder
```

Befehl 'db2se unregister_spatial_column'

Der Befehl **db2se unregister_spatial_column** nimmt die Registrierung einer räumlichen Spalte zurück.

Dieser Befehl entfernt die Registrierung auf folgende Weise:

- Die Zuordnung des räumlichen Bezugssystems zur räumlichen Spalte wird entfernt. In der Katalogsicht DB2GSE.ST_GEOMETRY_COLUMNS ist die räumliche Spalte zwar weiterhin angegeben, aber der Spalte ist kein räumliches Bezugssystem mehr zugeordnet.
- Bei einer Basistabelle wird die Integritätsbedingung gelöscht, die für diese Tabelle eingerichtet wurde, um sicherzustellen, dass alle Geometriewerte in dieser räumlichen Spalte dasselbe räumliche Bezugssystem verwenden.

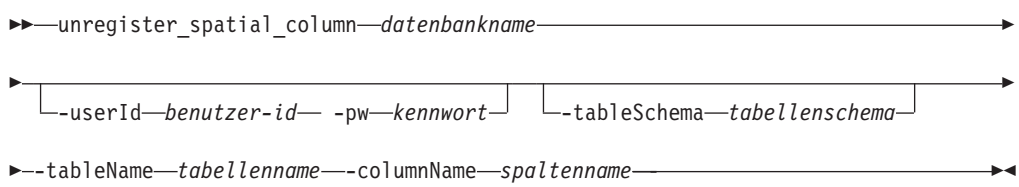
Berechtigung

Die Benutzer-ID muss über eine der folgenden Berechtigungen zur Ausführung dieses Befehls verfügen:

- Berechtigungen DBADM und DATAACCESS für die Datenbank mit der Tabelle, zu der die räumliche Spalte gehört, die registriert werden soll
- Zugriffsrecht CONTROL für diese Tabelle
- Zugriffsrecht ALTER für diese Tabelle

Befehlssyntax

Befehl 'db2se unregister_spatial_column'



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Gibt den Namen der Datenbank an, für die Sie die Registrierung einer räumlichen Spalte entfernen möchten.

-userId *benutzer-id*
Gibt die Datenbankbenutzer-ID an, die über die Berechtigung DATAACCESS für die durch den Parameter *datenbankname* angegebene Datenbank verfügt.

-pw *kennwort*
Gibt das Kennwort für *benutzer-id* an.

-tableSchema *tabellenschema*
Gibt den Schemanamen für die im Parameter *tabellenname* angegebene Tabelle an. Falls Sie keinen Schemanamen angeben, wird als Schemaname für die Tabelle bzw. Sicht der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

-tableName *tabellenname*
Gibt den Namen ohne Qualifikationsmerkmal der Tabelle für den im Parameter *spaltenname* angegebenen Spaltennamen an. Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

-columnName *spaltenname*
Gibt die Spalte an, für die die Registrierung zurückgenommen werden soll. Der Wert für *spaltenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Beispiel

Im folgenden Beispiel wird die Registrierung einer räumlichen Spalte namens "MYCOLUMN" in der Tabelle "MYTABLE" zurückgenommen.

```
db2se unregister_spatial_column mydb -tableName mytable -columnName mycolumn
```

Befehl 'db2se upgrade'

Der Befehl **db2se upgrade** aktualisiert eine räumlich aktivierte Datenbank von Version 9.5 oder Version 9.7 auf Version 10.1.

Dieser Befehl löscht möglicherweise räumliche Indizes und erstellt diese erneut, um das Upgrade durchzuführen; abhängig von der Größe der verwendeten Tabellen kann dies einige Zeit in Anspruch nehmen. So werden die Indizes beispielsweise gelöscht und erneut erstellt, wenn die Daten von einer 32-Bit-Instanz auf eine 64-Bit-Instanz migriert werden.

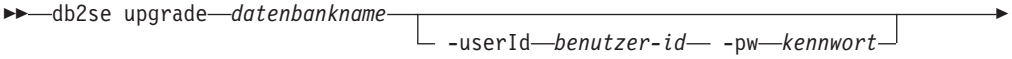
Tipp: Führen Sie den Befehl **db2se upgrade** mit der Option `-force 0` aus, und geben Sie eine Nachrichtendatei an, um zu ermitteln, für welche Indizes ein Upgrade erforderlich ist, ohne dass eine zusätzliche Upgradeverarbeitung durchgeführt werden muss.

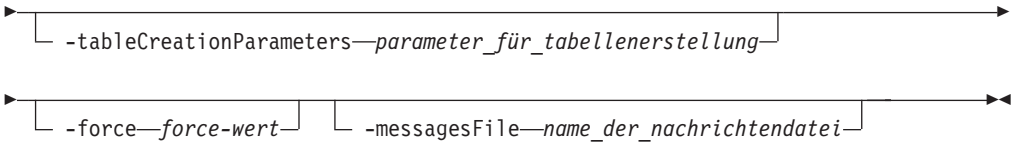
Berechtigung

Berechtigungen DBADM und DATAACCESS für die für räumliche Operationen aktivierte Datenbank, für die ein Upgrade durchgeführt werden soll

Befehlssyntax

Befehl db2se upgrade





Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Der Name der Datenbank, für die ein Upgrade durchgeführt werden soll.

-userId benutzer_id
Die Datenbankbenutzer-ID, die über die Berechtigung DATAACCESS für die Datenbank verfügt, für die das Upgrade durchgeführt wird.

-pw kennwort
Ihr Benutzerkennwort.

-tableCreationParameters parameter_für_die_tabellenerstellung
Die für die Erstellung der Spatial Extender-Katalogtabellen zu verwendenden Parameter.

-force force_wert

- 0: Standardwert. Die Durchführung des Upgrades wird versucht, aber gestoppt, wenn benutzerdefinierte Objekte, wie beispielsweise Sichten, Funktionen, Trigger oder räumliche Indizes, Spatial Extender-Objekte referenzieren.
- 1: Automatisches Sichern und automatischer Restore von anwendungsdefinierten Objekten. Sichern und Restore von räumlichen Indizes bei Bedarf.
- 2: Automatisches Sichern und automatischer Restore von anwendungsdefinierten Objekten. Sichern von Informationen für räumliche Indizes, jedoch kein automatischer Restore von räumlichen Indizes.

-messagesFile name_der_nachrichtendatei
Der Name der Datei, die den Bericht über die Upgradeaktionen enthält. Der von Ihnen angegebene Dateiname muss ein vollständig qualifizierter Name auf dem Server sein.

Tipp: Diesen Parameter angeben, um Unterstützung bei der Behebung von Upgradefehlern zu erhalten.

Einschränkung: Die Angabe einer vorhandenen Datei ist nicht möglich.

Hinweise zur Verwendung

Der Befehl **db2se upgrade** prüft eine Reihe von Bedingungen und gibt eine oder mehrere der folgenden Fehlernachrichten zurück, wenn eine oder mehrere dieser Bedingungen nicht erfüllt sind:

- Die Datenbank ist gegenwärtig nicht für räumliche Operationen aktiviert.
- Der Datenbankname ist ungültig.
- Es bestehen andere Verbindungen zur Datenbank. Das Upgrade kann nicht fortgesetzt werden.
- Der Spatial Extender-Katalog ist nicht konsistent.

- Der Benutzer ist nicht berechtigt.
- Das Kennwort ist ungültig.
- Für einige Benutzerobjekte konnte kein Upgrade durchgeführt werden.

Stellen Sie sicher, dass ein Tabellenbereich für temporäre Systemtabellen mit einer Seitengröße von mindestens 8 KB und einer Mindestgröße von 500 Seiten vorhanden ist. Dies ist eine Voraussetzung für die erfolgreiche Ausführung des Befehls **db2se upgrade**.

Befehl 'db2se migrate'

Mit dem Befehl **db2se migrate** kann eine Migration einer für räumliche Operationen aktivierten Datenbank auf Version 9.7 durchgeführt werden. Dieser Befehl gilt als veraltet und wird in einem zukünftigen Release entfernt. Verwenden Sie stattdessen den Befehl **db2se upgrade**.

Dieser Befehl löscht möglicherweise räumliche Indizes und erstellt diese erneut, um die Migration durchzuführen; abhängig von der Größe der verwendeten Tabellen kann dies einige Zeit in Anspruch nehmen. So werden die Indizes beispielsweise gelöscht und erneut erstellt, wenn die Daten von einer 32-Bit-Instanz auf eine 64-Bit-Instanz migriert werden.

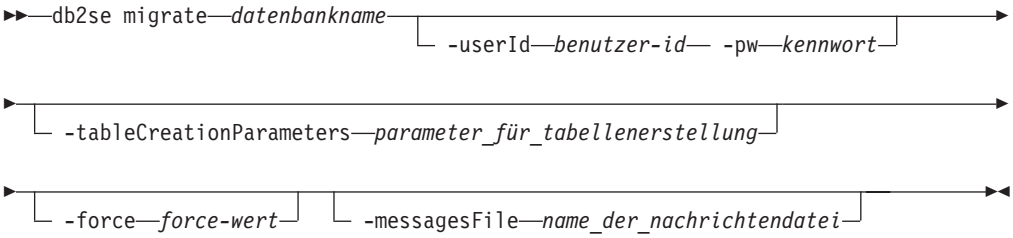
Tipp: Führen Sie den Befehl **db2se migrate** mit der Option `-force 0` aus, und geben Sie eine Nachrichtendatei an, um zu ermitteln, welche Indizes migriert werden müssen, ohne dass eine zusätzliche Migrationsverarbeitung durchgeführt werden muss.

Berechtigung

Berechtigung SYSADM oder DBADM für die zu migrierende Datenbank, die für räumliche Operationen aktiviert ist.

Befehlssyntax

Befehl db2se migrate



Befehlsparameter

Die Angaben haben die folgende Bedeutung:

datenbankname
Der Name der Datenbank, die migriert werden soll.

-userId benutzer_id
Die Datenbankbenutzer-ID, die entweder die Berechtigung SYSADM oder die Berechtigung DBADM für die zu migrierende Datenbank besitzt.

-pw kennwort
Ihr Benutzerkennwort.

-tableCreationParameters parameter_für_die_tabellenerstellung
Die für die Erstellung der Spatial Extender-Katalogtabellen zu verwendenden Parameter.

-force force_wert

- 0: Standardwert. Die Ausführung der Migration wird versucht, aber gestoppt, wenn anwendungsdefinierte Objekte, wie beispielsweise Sichten, Funktionen, Trigger oder räumliche Indizes, auf Spatial Extender-Objekten basieren.
- 1: Automatisches Sichern und automatischer Restore von anwendungsdefinierten Objekten. Sichern und Restore von räumlichen Indizes bei Bedarf.
- 2: Automatisches Sichern und automatischer Restore von anwendungsdefinierten Objekten. Sichern von Informationen für räumliche Indizes, jedoch kein automatischer Restore von räumlichen Indizes.

-messagesFile name_der_nachrichtendatei
Der Name der Datei, die den Bericht über die Migrationsaktionen enthält. Der von Ihnen angegebene Dateiname muss ein vollständig qualifizierter Name auf dem Server sein.

Tipp: Diesen Parameter angeben, um Unterstützung bei der Behebung von Migrationsfehlern zu erhalten.

Einschränkung: Die Angabe einer vorhandenen Datei ist nicht möglich.

Während der Migration wird möglicherweise mindestens eine der folgenden Fehlernachrichten ausgegeben:

- Die Datenbank ist gegenwärtig nicht für räumliche Operationen aktiviert.
- Der Datenbankname ist ungültig.
- Es bestehen andere Verbindungen zur Datenbank. Die Ausführung ist nicht möglich.
- Der Spatial Extender-Katalog ist nicht konsistent.
- Der Benutzer ist nicht berechtigt.
- Das Kennwort ist ungültig.
- Einige Benutzerobjekte konnten nicht migriert werden.

Kapitel 17. Gespeicherte Prozeduren

Mithilfe der gespeicherten Prozeduren von DB2 Spatial Extender können Sie DB2 Spatial Extender konfigurieren und Projekte erstellen, die räumliche Daten verwenden.

Wenn Sie DB2 Spatial Extender oder den DB2-Befehlszeilenprozessor konfigurieren, rufen Sie implizit diese gespeicherten Prozeduren auf. Wenn Sie z. B. den Befehlszeilenprozessorbefehl **db2se create_srs** absetzen, wird die Prozedur DB2GSE.ST_CREATE_SRS aufgerufen.

Alternativ dazu können Sie die gespeicherten Prozeduren von DB2 Spatial Extender explizit in einem Anwendungsprogramm aufrufen.

Vor dem Aufrufen der meisten gespeicherten Prozeduren von DB2 Spatial Extender für eine Datenbank müssen folgende Schritte ausgeführt werden:

1. Stellen Sie sicher, dass ein Tabellenbereich für temporäre Systemtabellen mit einer Seitengröße von mindestens 8 KB und einer Mindestgröße von 500 Seiten vorhanden ist. Dies ist eine Voraussetzung für die erfolgreiche Ausführung der Prozedur ST_ENABLE_DB STORED bzw. des Befehls **db2se enable_db**.
2. Aktivieren Sie die Datenbank für räumliche Operationen, indem Sie die Prozedur ST_ENABLE_DB aufrufen. Details hierzu finden Sie in „Prozedur ST_ENABLE_DB" auf Seite 210.

Nachdem eine Datenbank für räumliche Operationen aktiviert ist, können Sie eine beliebige gespeicherte Prozedur von DB2 Spatial Extender entweder implizit oder explizit für diese Datenbank aufrufen, falls eine Verbindung zur Datenbank besteht.

Dieses Kapitel enthält Abschnitte zu allen gespeicherten Prozeduren von DB2 Spatial Extender:

- „Prozedur ST_ALTER_COORDSYS" auf Seite 186
- „Prozedur ST_ALTER_SRS" auf Seite 188
- „Prozedur ST_CREATE_COORDSYS" auf Seite 192
- „Prozedur ST_CREATE_SRS" auf Seite 194
- „Prozedur ST_DISABLE_AUTOGEOCODING" auf Seite 202
- „Prozedur ST_DISABLE_DB" auf Seite 203
- „Prozedur ST_DROP_COORDSYS" auf Seite 205
- „Prozedur ST_DROP_SRS" auf Seite 206
- „Prozedur ST_ENABLE_AUTOGEOCODING" auf Seite 207
- „Prozedur ST_ENABLE_DB" auf Seite 210
- „Prozedur ST_EXPORT_SHAPE" auf Seite 211
- „Prozedur ST_IMPORT_SHAPE" auf Seite 215
- „Prozedur ST_REGISTER_GEOCODER" auf Seite 224
- „Prozedur ST_REGISTER_SPATIAL_COLUMN" auf Seite 228
- „Prozedur ST_REMOVE_GEOCODING_SETUP" auf Seite 230
- „Prozedur ST_RUN_GEOCODING" auf Seite 232
- „Prozedur ST_SETUP_GEOCODING" auf Seite 235

- „Prozedur ST_UNREGISTER_GEOCODER" auf Seite 239
- „Prozedur ST_UNREGISTER_SPATIAL_COLUMN" auf Seite 240

Die Implementierungen der gespeicherten Prozeduren sind in der Bibliothek db2gse des Servers von DB2 Spatial Extender archiviert.

Prozedur ST_ALTER_COORDSYS

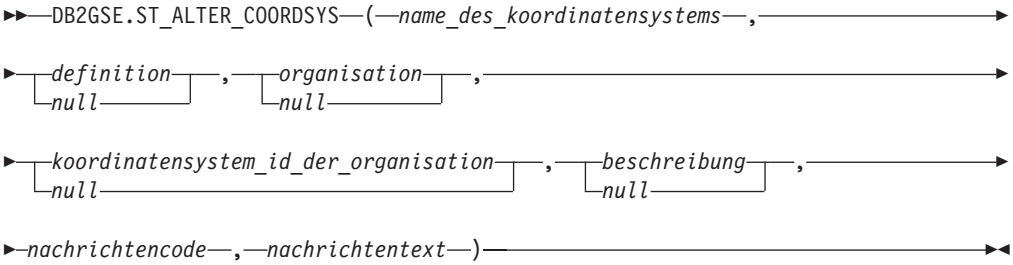
Mit dieser gespeicherten Prozedur können Sie die Definition eines Koordinatensystems in der Datenbank aktualisieren. Sobald die gespeicherte Prozedur verarbeitet wird, werden die Informationen zum Koordinatensystem in der Katalogsicht DB2GSE.ST_COORDINATE_SYSTEMS aktualisiert.

Achtung: Bei der Verwendung dieser gespeicherten Prozedur sollten Sie äußerst vorsichtig vorgehen. Falls Sie die Definition des Koordinatensystems mithilfe dieser gespeicherten Prozedur ändern und räumliche Daten vorhanden sind, denen ein räumliches Bezugssystem zugeordnet ist, das auf diesem Koordinatensystem basiert, können die räumlichen Daten unbeabsichtigterweise geändert werden. Wenn räumliche Daten betroffen sind, müssen Sie sicherstellen, dass die geänderten räumlichen Daten weiterhin exakt und gültig sind.

Berechtigung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss die Berechtigung DBADM besitzen.

Syntax



Parameterbeschreibungen

name_des_koordinatensystems
Dieser Parameter gibt die eindeutige Kennzeichnung des Koordinatensystems an. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *name_des_koordinatensystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

definition
Dieser Parameter definiert das Koordinatensystem. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird die Definition des Koordinatensystems nicht geändert.

Dieser Parameter hat den Datentyp VARCHAR(2048).

organisation
Dieser Parameter gibt den Namen der Organisation an, die das Koordinatensystem definiert und seine Definition zur Verfügung gestellt hat (z. B. "European Petroleum Survey Group (EPSG)"). Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein.

Wenn Sie für diesen Parameter den Wert NULL angeben, wird die Organisation des Koordinatensystems nicht geändert. Wenn Sie für diesen Parameter einen Wert angeben, kann der Parameter *koordinatensystem_id_der_organisation* nicht den Wert NULL haben. In diesem Fall wird das Koordinatensystem durch die Kombination der Werte für *organisation* und *koordinatensystem_id_der_organisation* eindeutig gekennzeichnet.

Dieser Parameter hat den Datentyp VARCHAR(128).

koordinatensystem_id_der_organisation
Dieser Parameter gibt eine numerische Kennung an, die dem Koordinatensystem durch die Einheit zugeordnet wurde, die im Parameter *organisation* angegeben wird. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein.

Wenn Sie für diesen Parameter den Wert NULL angeben, muss auch für den Parameter *organisation* der Wert NULL angegeben sein. In diesem Fall wird die Koordinatensystem-ID der Organisation nicht geändert. Wenn Sie für diesen Parameter einen Wert angeben, kann der Parameter *organisation* nicht den Wert NULL haben. In diesem Fall wird das Koordinatensystem durch die Kombination der Werte für *organisation* und *koordinatensystem_id_der_organisation* eindeutig gekennzeichnet.

Dieser Parameter hat den Datentyp INTEGER.

beschreibung
Dieser Parameter beschreibt das Koordinatensystem, indem seine Anwendung erläutert wird. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, werden die Beschreibungsinformationen zum Koordinatensystem nicht geändert.

Dieser Parameter hat den Datentyp VARCHAR(256).

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_ALTER-_COORDSYS über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird ein Koordinatensystem namens NORTH_AMERICAN_TEST über den DB2-Befehl CALL aktualisiert. Dieser Befehl CALL ordnet dem Parameter *koordinatensystem-id* den Wert 1002 zu:

```
call DB2GSE.ST_ALTER_COORDSYS('NORTH_AMERICAN_TEST',NULL,NULL,1002,NULL,?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_ALTER_SRS

Mit dieser gespeicherten Prozedur können Sie die Definition eines räumlichen Bezugssystems in der Datenbank aktualisieren. Bei der Verarbeitung dieser gespeicherten Prozedur werden Informationen zum räumlichen Bezugssystem in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aktualisiert.

Intern speichert DB2 Spatial Extender die Koordinatenwerte als positive ganze Zahlen. Auf diese Weise kann die Auswirkung von Rundungsfehlern (die stark vom tatsächlichen Wert bei Operationen mit Gleitkommazahlen abhängen) reduziert werden. Außerdem kann so die Leistung von räumlichen Operationen erheblich gesteigert werden.

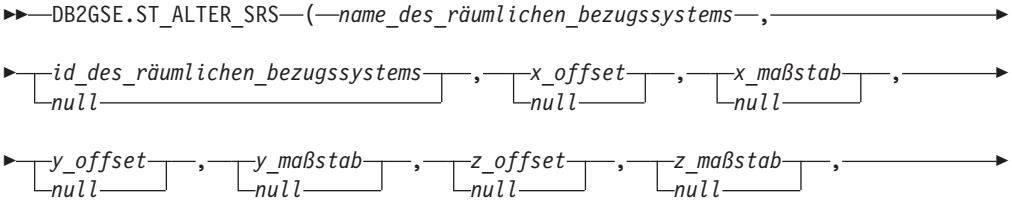
Einschränkung: Ein räumliches Bezugssystem kann nicht geändert werden, wenn es von einer registrierten räumlichen Spalte verwendet wird.

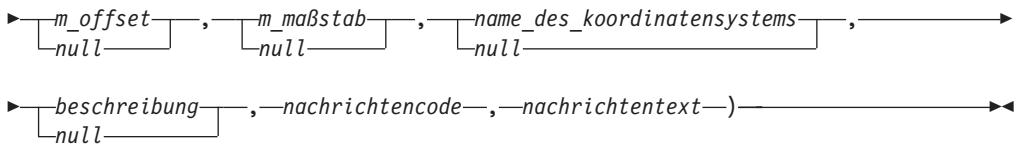
Achtung: Bei der Verwendung dieser gespeicherten Prozedur sollten Sie äußerst vorsichtig vorgehen. Wenn Sie mit dieser gespeicherten Prozedur die Parameter für Offset, Maßstab oder *name_des_koordinatensystems* des räumlichen Bezugssystems ändern und bereits räumliche Daten vorhanden sind, die diesem räumlichen Bezugssystem zugeordnet sind, werden diese räumlichen Daten möglicherweise unbeabsichtigt geändert. Wenn räumliche Daten betroffen sind, müssen Sie sicherstellen, dass die geänderten räumlichen Daten weiterhin exakt und gültig sind.

Berechtigung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss die Berechtigung DBADM besitzen.

Syntax





Parameterbeschreibungen

name_des_räumlichen_bezugssystems
Dieser Parameter gibt das räumliche Bezugssystem an. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *name_des_räumlichen_bezugssystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

id_des_räumlichen_bezugssystems
Dieser Parameter gibt die eindeutige Kennung des räumlichen Bezugssystems an. Diese Kennung wird für unterschiedliche räumliche Funktionen als Eingabeparameter verwendet. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird die numerische Kennung des räumlichen Bezugssystems nicht geändert.

Dieser Parameter hat den Datentyp INTEGER.

x_offset
Dieser Parameter gibt den Offset für alle X-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert für diesen Parameter in der Definition des räumlichen Bezugssystems nicht geändert.

Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor (Wert für *x_scale*) angewendet wird. (*WKT* steht für "Well-Known Text" und *WKB* für "Well-Known Binary".)

Dieser Parameter hat den Datentyp DOUBLE.

x_maßstab
Dieser Parameter gibt den Maßstabsfaktor für alle X-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert für diesen Parameter in der Definition des räumlichen Bezugssystems nicht geändert.

Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WBT- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset (Wert für *x_offset*) subtrahiert wurde.

Dieser Parameter hat den Datentyp DOUBLE.

y_offset
Dieser Parameter gibt den Offset für alle Y-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Sie müssen zwar ei-

nen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert für diesen Parameter in der Definition des räumlichen Bezugssystems nicht geändert.

Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor (Wert für *y_scale*) angewendet wird.

Dieser Parameter hat den Datentyp DOUBLE.

y_maßstab
Dieser Parameter gibt den Maßstabsfaktor für alle Y-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert für diesen Parameter in der Definition des räumlichen Bezugssystems nicht geändert.

Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB- Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset (Wert für *y_offset*) subtrahiert wurde. Dieser Maßstabsfaktor muss mit dem Wert für *x_maßstab* identisch sein.

Dieser Parameter hat den Datentyp DOUBLE.

z_offset
Dieser Parameter gibt den Offset für alle Z-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert für diesen Parameter in der Definition des räumlichen Bezugssystems nicht geändert.

Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WBT-, Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor (Wert für *z_scale*) angewendet wird.

Dieser Parameter hat den Datentyp DOUBLE.

z_maßstab
Dieser Parameter gibt den Maßstabsfaktor für alle Z-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert für diesen Parameter in der Definition des räumlichen Bezugssystems nicht geändert.

Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB-, Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset (Wert für *z_offset*) subtrahiert wurde.

Dieser Parameter hat den Datentyp DOUBLE.

m_offset
Dieser Parameter gibt den Offset für alle M-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert für diesen Parameter in der Definition des räumlichen Bezugssystems nicht geändert.

Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB-, Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor (Wert für *m_scale*) angewendet wird.

Dieser Parameter hat den Datentyp DOUBLE.

m_maßstab
Dieser Parameter gibt den Maßstabsfaktor für alle M-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert für diesen Parameter in der Definition des räumlichen Bezugssystems nicht geändert.

Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB-, Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset (Wert für *m_offset*) subtrahiert wurde.

Dieser Parameter hat den Datentyp DOUBLE.

name_des_koordinatensystems
Dieser Parameter ist die eindeutige Kennzeichnung des Koordinatensystems, auf dem dieses räumliche Bezugssystem basiert. Das Koordinatensystem muss in der Sicht ST_COORDINATE_SYSTEMS aufgeführt sein. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird das Koordinatensystem, das für dieses räumlichen Bezugssystem verwendet wird, nicht geändert.

Der Wert für *name_des_koordinatensystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

beschreibung
Dieser Parameter beschreibt das räumliche Bezugssystem, indem seine Anwendung erläutert wird. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, werden die Beschreibungsinformationen zum räumlichen Bezugssystem nicht geändert.

Dieser Parameter hat den Datentyp VARCHAR(256).

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der

Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_ALTER_SRS über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird mit einem DB2-Befehl CALL der Parameter *beschreibung* eines räumlichen Bezugssystems namens SRSDEMO geändert:

```
call DB2GSE.ST_ALTER_SRS('SRSDEMO',NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
         NULL,NULL,'RBZ für GSE-Demoprogramm: Tabelle OFFICES',?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

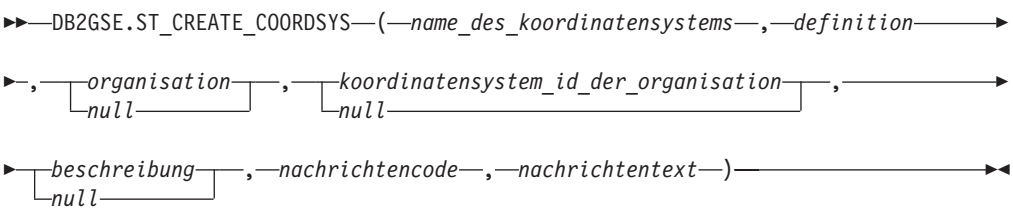
Prozedur ST_CREATE_COORDSYS

Mit dieser gespeicherten Prozedur können Sie ein neues Koordinatensystem löschen. Sobald die gespeicherte Prozedur verarbeitet wird, werden die Informationen zum Koordinatensystem zur Katalogsicht DB2GSE.ST_COORDINATE_SYSTEMS hinzugefügt.

Berechtigung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss die Berechtigung DBADM besitzen.

Syntax



Parameterbeschreibungen

name_des_koordinatensystems
Dieser Parameter gibt die eindeutige Kennzeichnung des Koordinatensystems an. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *name_des_koordinatensystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

definition
Dieser Parameter definiert das Koordinatensystem. Für diesen Parameter müs-

sen Sie einen anderen Wert als NULL angeben. Normalerweise stellt der Lieferant des Koordinatensystems die Informationen für diesen Parameter zur Verfügung.

Dieser Parameter hat den Datentyp VARCHAR(2048).

organisation
Dieser Parameter gibt den Namen der Organisation an, die das Koordinatensystem definiert und seine Definition zur Verfügung gestellt hat (z. B. "European Petroleum Survey Group (EPSG)"). Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein.

Wenn für diesen Parameter der Wert NULL angegeben wird, muss für den Parameter *koordinatensystem_id_der_organisation* ebenfalls der Wert NULL angegeben sein. Wenn Sie für diesen Parameter einen Wert angeben, kann der Parameter *koordinatensystem_id_der_organisation* nicht den Wert NULL haben. In diesem Fall wird das Koordinatensystem durch die Kombination der Werte für *organisation* und *koordinatensystem_id_der_organisation* eindeutig gekennzeichnet.

Dieser Parameter hat den Datentyp VARCHAR(128).

koordinatensystem_id_der_organisation
Dieser Parameter gibt eine numerische Kennung an. Dieser Wert wird von der im Parameter *organisation* angegebenen Einheit zugeordnet. Er ist nicht zwangsläufig gegenüber allen anderen Koordinatensystemen eindeutig. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein.

Wenn für diesen Parameter der Wert NULL angegeben wird, muss der Wert des Parameters *organisation* ebenfalls NULL sein. Wenn Sie für diesen Parameter einen Wert angeben, kann der Parameter *organisation* nicht den Wert NULL haben. In diesem Fall wird das Koordinatensystem durch die Kombination der Werte für *organisation* und *koordinatensystem_id_der_organisation* eindeutig gekennzeichnet.

Dieser Parameter hat den Datentyp INTEGER.

beschreibung
Dieser Parameter beschreibt das Koordinatensystem, indem seine Anwendung erläutert wird. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, werden keine Beschreibungsangaben über das Koordinatensystem aufgezeichnet.

Dieser Parameter hat den Datentyp VARCHAR(256).

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der

Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_CREATE_COORDSYS über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird mit einem DB2-Befehl CALL ein Koordinatensystem erstellt, wobei die folgenden Parameterwerte verwendet werden:

- Parameter *name_des_koordinatensystems*: NORTH_AMERICAN_TEST
- Parameter *definition*:

```
GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",
     SPHEROID["GRS_1980",6378137.0,298.257222101]],
PRIMEM["Greenwich",0.0],
UNIT["Degree",0.0174532925199433]]
```

- Parameter *organisation*: EPSG
- Parameter *koordinatensystem_id_der_organisation*: 1001
- Parameter *beschreibung*: Testkoordinatensysteme

```
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',?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_CREATE_SRS

Mit der gespeicherten Prozedur ST_CREATE_SRS können Sie ein räumliches Bezugssystem erstellen.

Ein räumliches Bezugssystem ist durch das Koordinatensystem, die Genauigkeit und die Koordinatenbereiche, die in diesem räumlichen Bezugssystem dargestellt sind, definiert. Die Bereiche sind die kleinstmöglichen und größtmöglichen Koordinatenwerte für die X-, Y-, Z- und M-Koordinaten.

Intern speichert DB2 Spatial Extender die Koordinatenwerte als positive ganze Zahlen. Auf diese Weise kann die Auswirkung von Rundungsfehlern (die stark vom tatsächlichen Wert bei Operationen mit Gleitkommazahlen abhängen) reduziert werden. Außerdem kann so die Leistung von räumlichen Operationen erheblich gesteigert werden.

Für diese gespeicherte Prozedur gibt es zwei Varianten:

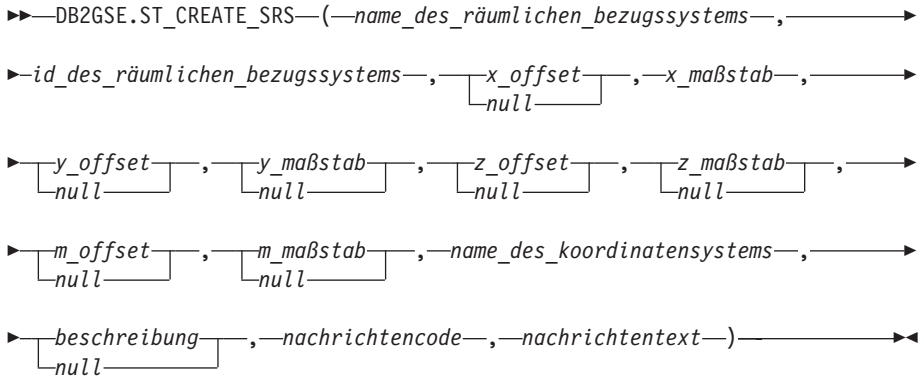
- Die erste Variante verwendet die Umwandlungsfaktoren (Abstände und Maßstabsfaktoren) als Eingabeparameter.
- Die zweite Variante verwendet als Eingabeparameter die Bereiche und die Genauigkeit und berechnet die Umwandlungsfaktoren intern.

Berechtigung

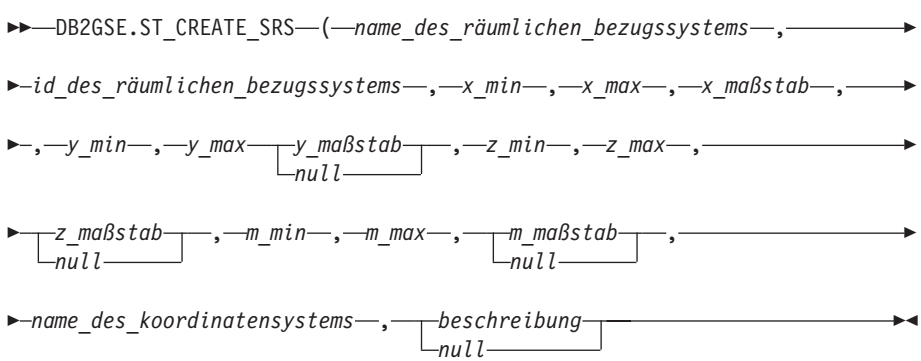
Es ist keine Berechtigung erforderlich.

Syntax

Verwendung von Umwandlungsfaktoren (Variante 1)



Verwendung des größtmöglichen Bereichs (Variante 2)



Parameterbeschreibungen

Verwendung von Umwandlungsfaktoren (Variante 1)

srs_name
Dieser Parameter gibt das räumliche Bezugssystem an. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *name_des_räumlichen_bezugssystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

id_des_räumlichen_bezugssystems
Dieser Parameter gibt die eindeutige Kennung des räumlichen Bezugssystems an. Diese numerische Kennung wird für unterschiedliche räumliche Funktionen als Eingabeparameter verwendet. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Dieser Parameter hat den Datentyp INTEGER.

x_offset
Dieser Parameter gibt den Offset für alle X-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor (Wert für *x_scale*) angewendet wird. (*WKT* steht für "Well-Known Text" und *WKB* für "Well-Known Binary".) Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert null verwendet.

Dieser Parameter hat den Datentyp DOUBLE.

x_maßstab
Dieser Parameter gibt den Maßstabsfaktor für alle X-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WBT- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset (Wert für *x_offset*) subtrahiert wurde. Der Wert für *x_offset* wird entweder explizit angegeben, oder es wird für *x_offset* der Standardwert 0 verwendet. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Dieser Parameter hat den Datentyp DOUBLE.

y_offset
Dieser Parameter gibt den Offset für alle Y-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor (Wert für *y_scale*) angewendet wird. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert null verwendet.

Dieser Parameter hat den Datentyp DOUBLE.

y_maßstab
Dieser Parameter gibt den Maßstabsfaktor für alle Y-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB- Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset (Wert für *y_offset*) subtrahiert wurde. Der Wert für *y_offset* wird entweder explizit angegeben, oder es wird für *y_offset* der Standardwert 0 verwendet. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn für diesen Parameter der Wert NULL angegeben ist, wird der Wert des Parameters *x_maßstab* verwendet. Geben Sie für diesen Parameter einen anderen Wert als NULL an, muss der von Ihnen angegebene Wert mit dem Wert des Parameters *x_maßstab* identisch sein.

Dieser Parameter hat den Datentyp DOUBLE.

z_offset
Dieser Parameter gibt den Offset für alle Z-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WBT-, Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor (Wert für *z_scale*) angewendet wird. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert null verwendet.

Dieser Parameter hat den Datentyp DOUBLE.

z_maßstab
Dieser Parameter gibt den Maßstabsfaktor für alle Z-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB-, Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset (Wert für *z_offset*) subtrahiert wurde. Der Wert für *z_offset* wird entweder explizit angegeben, oder es wird für *z_offset* der Standardwert 0 verwendet. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert 1 verwendet.

Dieser Parameter hat den Datentyp DOUBLE.

m_offset
Dieser Parameter gibt den Offset für alle M-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB-, Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Offset subtrahiert, bevor der Maßstabsfaktor (Wert für *m_scale*) angewendet wird. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert null verwendet.

Dieser Parameter hat den Datentyp DOUBLE.

m_maßstab
Dieser Parameter gibt den Maßstabsfaktor für alle M-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB-, Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset (Wert für *m_offset*) subtrahiert wurde. Der Wert für *m_offset* wird entweder explizit angegeben, oder es wird für *m_offset* der Standardwert 0 verwendet. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert 1 verwendet.

Dieser Parameter hat den Datentyp DOUBLE.

name_des_koordinatensystems
Dieser Parameter ist die eindeutige Kennzeichnung des Koordinatensystems, auf dem dieses räumliche Bezugssystem basiert. Das

Koordinatensystem muss in der Sicht ST_COORDINATE_SYSTEMS aufgeführt sein. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *name_des_koordinatensystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

Beschreibung
Dieser Parameter beschreibt das räumliche Bezugssystem, indem der Zweck der Anwendung erläutert wird. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, werden keine Beschreibungsangaben aufgezeichnet.

Dieser Parameter hat den Datentyp VARCHAR(256).

Verwendung des größtmöglichen Bereichs (Variante 2)

srs_name
Dieser Parameter gibt das räumliche Bezugssystem an. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *name_des_räumlichen_bezugssystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

id_des_räumlichen_bezugssystems
Dieser Parameter gibt die eindeutige Kennung des räumlichen Bezugssystems an. Diese numerische Kennung wird für unterschiedliche räumliche Funktionen als Eingabeparameter verwendet. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Dieser Parameter hat den Datentyp INTEGER.

x_min Dieser Parameter gibt den kleinstmöglichen X-Koordinatenwert für alle Geometrien an, die dieses räumliche Bezugssystem verwenden. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Dieser Parameter hat den Datentyp DOUBLE.

x_max Dieser Parameter gibt den größtmöglichen X-Koordinatenwert für alle Geometrien an, die dieses räumliche Bezugssystem verwenden. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Abhängig vom Wert für *x_maßstab* kann der in der Sicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS angezeigte Wert größer als der hier angegebene Wert sein. Gültig ist der Wert, der in der Sicht angezeigt wird.

Dieser Parameter hat den Datentyp DOUBLE.

x_maßstab
Dieser Parameter gibt den Maßstabsfaktor für alle X-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WBT- und Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset (Wert für *x_offset*) subtrahiert wurde. Die Berechnung des Offsets (Wert für *x_offset*) basiert auf dem Wert für *x_min*. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Wenn sowohl für *x_maßstab* als auch für *y_maßstab* ein Wert angegeben wird, müssen die beiden Werte identisch sein.

Dieser Parameter hat den Datentyp DOUBLE.

y_min Dieser Parameter gibt den kleinstmöglichen Y-Koordinatenwert für alle Geometrien an, die dieses räumliche Bezugssystem verwenden. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Dieser Parameter hat den Datentyp DOUBLE.

y_max Dieser Parameter gibt den größtmöglichen Y-Koordinatenwert für alle Geometrien an, die dieses räumliche Bezugssystem verwenden. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Abhängig vom Wert für *y_maßstab* kann der in der Sicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS angezeigte Wert größer als der hier angegebene Wert sein. Gültig ist der Wert, der in der Sicht angezeigt wird.

Dieser Parameter hat den Datentyp DOUBLE.

y_maßstab
Dieser Parameter gibt den Maßstabsfaktor für alle Y-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB- Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset (Wert für *y_offset*) subtrahiert wurde. Die Berechnung des Offsets (Wert für *y_offset*) basiert auf dem Wert für *y_min*. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn für diesen Parameter der Wert NULL angegeben ist, wird der Wert des Parameters *x_maßstab* verwendet. Wenn sowohl für *y_maßstab* als auch für *x_maßstab* ein Wert angegeben wird, müssen die beiden Werte identisch sein.

Dieser Parameter hat den Datentyp DOUBLE.

z_min Dieser Parameter gibt den kleinstmöglichen Z-Koordinatenwert für alle Geometrien an, die dieses räumliche Bezugssystem verwenden. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Dieser Parameter hat den Datentyp DOUBLE.

z_max Dieser Parameter gibt den größtmöglichen Z-Koordinatenwert für

alle Geometrien an, die dieses räumliche Bezugssystem verwenden. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Abhängig vom Wert für *z_maßstab* kann der in der Sicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS angezeigte Wert größer als der hier angegebene Wert sein. Gültig ist der Wert, der in der Sicht angezeigt wird.

Dieser Parameter hat den Datentyp DOUBLE.

z_maßstab
Dieser Parameter gibt den Maßstabsfaktor für alle Z-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB-, Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset (Wert für *z_offset*) subtrahiert wurde. Die Berechnung des Offsets (Wert für *z_offset*) basiert auf dem Wert für *z_min*. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert 1 verwendet.

Dieser Parameter hat den Datentyp DOUBLE.

m_min
Dieser Parameter gibt den kleinstmöglichen M-Koordinatenwert für alle Geometrien an, die dieses räumliche Bezugssystem verwenden. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Dieser Parameter hat den Datentyp DOUBLE.

m_max
Dieser Parameter gibt den größtmöglichen M-Koordinatenwert für alle Geometrien an, die dieses räumliche Bezugssystem verwenden. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Abhängig vom Wert für *m_maßstab* kann der in der Sicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS angezeigte Wert größer als der hier angegebene Wert sein. Gültig ist der Wert, der in der Sicht angezeigt wird.

Dieser Parameter hat den Datentyp DOUBLE.

m_maßstab
Dieser Parameter gibt den Maßstabsfaktor für alle M-Koordinaten von Geometrien an, die in diesem räumlichen Bezugssystem dargestellt sind. Bei der Umwandlung von Geometrien aus externen Darstellungen (WKT-, WKB-, Formdarstellung) in die interne Darstellung von DB2 Spatial Extender wird der Maßstabsfaktor (durch Multiplikation) angewendet, nachdem der Offset (Wert für *m_offset*) subtrahiert wurde. Die Berechnung des Offsets (Wert für *m_offset*) basiert auf dem Wert für *m_min*. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert 1 verwendet.

Dieser Parameter hat den Datentyp DOUBLE.

name_des_koordinatensystems
Dieser Parameter ist die eindeutige Kennzeichnung des Koordinatensystems, auf dem dieses räumliche Bezugssystem basiert. Das Koordinatensystem muss in der Sicht ST_COORDINATE_SYSTEMS aufgeführt sein. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *name_des_koordinatensystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

Beschreibung
Dieser Parameter beschreibt das räumliche Bezugssystem, indem der Zweck der Anwendung erläutert wird. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, werden keine Beschreibungsangaben aufgezeichnet.

Dieser Parameter hat den Datentyp VARCHAR(256).

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_CREATE_SRS über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird mit einem DB2-Befehl CALL ein räumliches Bezugssystem namens SRSDEMO erstellt, wobei die folgenden Parameterwerte verwendet werden:

- *id_des_räumlichen_bezugssystems*: 1000000
- *x_offset*: -180
- *x_maßstab*: 1000000
- *y_offset*: -90
- *y_maßstab*: 1000000

```
call DB2GSE.ST_CREATE_SRS('SRSDEMO',1000000,
                          -180,1000000, -90, 1000000,
                           0, 1, 0, 1,'NORTH_AMERICAN',
                          'RBZ für GSE-Demoprogramm: Tabelle CUSTOMER',?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_DISABLE_AUTOGEOCODING

Mit dieser gespeicherten Prozedur können Sie angeben, dass DB2 Spatial Extender die Synchronisierung einer geocodierten Spalte mit ihrer bzw. ihren zugeordneten Geocodierungsspalte(n) stoppen soll.

Eine *Geocodierungsspalte* wird als Eingabe für den Geocoder verwendet.

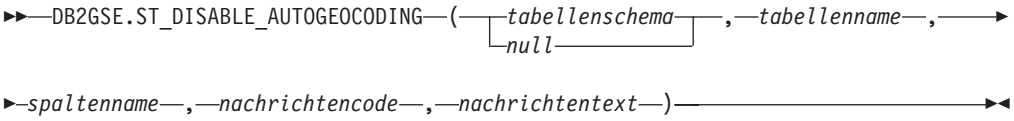
Berechtigung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte besitzen:

- Berechtigungen DBADM und DATAACCESS für die Datenbank mit der Tabelle, für die Trigger definiert sind, die gelöscht werden.
- Zugriffsrecht CONTROL für diese Tabelle
- Zugriffsrechte ALTER und UPDATE für diese Tabelle

Anmerkung: Für die Zugriffsrechte CONTROL und ALTER müssen Sie die Berechtigung DROPIN für das Schema DB2GSE besitzen.

Syntax



Parameterbeschreibungen

tabellenschema
: Dieser Parameter gibt den Namen des Schemas an, zu dem die Tabelle bzw. Sicht, die im Parameter *tabellenname* angegeben ist, gehört. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Falls Sie für diesen Parameter den Wert NULL angeben, wird als Schemaname für die Tabelle bzw. Sicht der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

  Der Wert für *tabellenschema* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in Anführungszeichen setzen.

  Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

tabellenname
: Dieser Parameter gibt den Namen ohne Qualifikationsmerkmal der Tabelle an, für die die zu löschenden Trigger definiert sind. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

spaltenname
Dieser Parameter gibt den Namen der geocodierten Spalte an, die durch die zu löschenden Trigger verwaltet wird. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *spaltenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_DISABLE_AUTOGEOCODING über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird mit einem DB2-Befehl CALL die automatische Geocodierung für die Spalte LOCATION in der Tabelle namens CUSTOMERS inaktiviert:

```
call DB2GSE.ST_DISABLE_AUTOGEOCODING(NULL,'CUSTOMERS','LOCATION',?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_DISABLE_DB

Mit dieser gespeicherten Prozedur können Sie Ressourcen entfernen, die DB2 Spatial Extender das Speichern räumlicher Daten und die Unterstützung von Operationen mit diesen Daten ermöglichen.

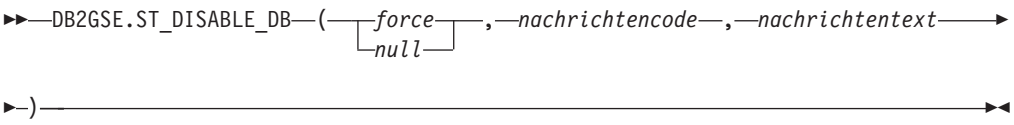
Diese gespeicherte Prozedur hilft Ihnen bei der Lösung von Problemen, die nach der Aktivierung der Datenbank für räumliche Operationen auftreten. Angenom-

men, Sie haben beispielsweise eine Datenbank, für räumliche Operationen aktiviert und beschließen dann, stattdessen eine andere Datenbank mit DB2 Spatial Extender zu verwenden. Für den Fall, dass Sie noch keine räumlichen Spalten definiert oder räumliche Daten importiert haben, können Sie diese gespeicherte Prozedur aufrufen, um alle räumlichen Ressourcen aus der ersten Datenbank zu entfernen. Aufgrund der gegenseitigen Abhängigkeit zwischen räumlichen Spalten und Typdefinitionen ist das Löschen von Typdefinitionen nicht möglich, wenn Spalten des entsprechenden Typs vorhanden sind. Wenn Sie bereits räumliche Spalten definiert haben, aber trotzdem die Datenbank für räumliche Operationen inaktivieren wollen, müssen Sie einen anderen Wert als 0 (null) für den Parameter *force_wert* angeben. Dann werden alle räumlichen Ressourcen aus der Datenbank entfernt, von denen keine anderen Ressourcen abhängig sind.

Berechtigung

Die Benutzer-ID, unter der diese gespeicherter Prozedur aufgerufen wird, muss die Berechtigung DBADM für die Datenbank besitzen, aus der die Ressourcen von DB2 Spatial Extender-Ressourcen entfernt werden sollen.

Syntax



Parameterbeschreibungen

force
Dieser Parameter gibt an, dass Sie eine Datenbank für räumliche Operationen inaktivieren wollen, obwohl unter Umständen Datenbankobjekte vorhanden sind, die von den räumlichen Typen oder den räumlichen Funktionen abhängig sind. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Falls Sie einen anderen Wert als 0 (null) oder NULL für den Parameter *force* angeben, wird die Datenbank inaktiviert, und alle Ressourcen von DB2 Spatial Extender werden nach Möglichkeit entfernt. Wenn Sie den Wert null oder NULL angeben, wird die Datenbank nicht aktiviert, falls Datenbankobjekte von räumlichen Typen oder räumlichen Funktionen abhängig sind. Bei solchen abhängigen Datenbankobjekten kann es sich um Tabellen, Sichten, Integritätsbedingungen, Trigger, generierte Spalten, Methoden, Funktionen, Prozeduren und andere Datentypen (untergeordnete Typen oder strukturierte Typen mit einem räumlichen Attribut) handeln.

Dieser Parameter hat den Datentyp SMALLINT.

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_DISABLE_DB über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird die Datenbank mit einem DB2-Befehl CALL für räumliche Operationen inaktiviert, wobei für den Parameter *force* der Wert 1 angegeben wird:

```
call DB2GSE.ST_DISABLE_DB(1,?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_DROP_COORDSYS

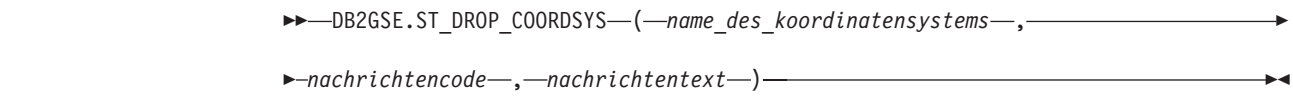
Mit dieser gespeicherten Prozedur können Sie Informationen zu einem Koordinatensystem aus der Datenbank löschen. Sobald die gespeicherte Prozedur verarbeitet wird, sind die Informationen zum Koordinatensystem nicht länger über die Katalogsicht DB2GSE.ST_COORDINATE_SYSTEMS abrufbar.

Einschränkung:
Ein Koordinatensystem, auf dem ein räumliches Bezugssystem basiert, kann nicht gelöscht werden.

Berechtigung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss die Berechtigung DBADM besitzen.

Syntax



Parameterbeschreibungen

name_des_koordinatensystems
Dieser Parameter gibt die eindeutige Kennzeichnung des Koordinatensystems an. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *name_des_koordinatensystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_DROP-_COORDSYS über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird ein Koordinatensystem namens NORTH_AMERICAN_TEST über einen DB2-Befehl CALL aus der Datenbank gelöscht:

```
call DB2GSE.ST_DROP_COORDSYS('NORTH_AMERICAN_TEST',?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_DROP_SRS

Mit dieser gespeicherten Prozedur können Sie ein räumliches Bezugssystem löschen.

Sobald diese gespeicherte Prozedur verarbeitet wird, werden Informationen zum räumlichen Bezugssystem aus der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS entfernt.

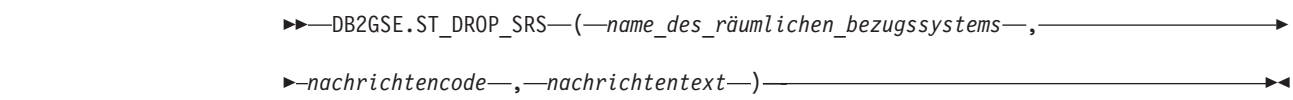
Einschränkung: Ein räumliches Bezugssystem kann nicht gelöscht werden, wenn eine räumliche Spalte registriert ist, die dieses räumliche Bezugssystem verwendet.

Wichtig:
Bei der Verwendung dieser gespeicherten Prozedur sollten Sie äußerst vorsichtig vorgehen. Wenn Sie ein räumliches Bezugssystem mit dieser gespeicherten Prozedur löschen und dieses räumliche Bezugssystem räumlichen Daten zugeordnet ist, können Sie für die entsprechenden räumlichen Daten keine räumlichen Operationen mehr ausführen.

Berechtigung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss die Berechtigung DBADM besitzen.

Syntax



Parameterbeschreibungen

name_des_räumlichen_bezugssystems
Dieser Parameter gibt das räumliche Bezugssystem an. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *name_des_räumlichen_bezugssystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_DROP_SRS über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird mit einem DB2-Befehl CALL ein räumliches Bezugssystem namens SRSDEMO gelöscht:

```
call DB2GSE.ST_DROP_SRS('SRSDEMO',?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_ENABLE_AUTOGEOCODING

Mit dieser gespeicherten Prozedur können Sie angeben, dass DB2 Spatial Extender eine geocodierte Spalte mit ihrer bzw. ihren zugeordneten Geocodierungsspalte(n) synchronisieren soll.

Eine *Geocodierungsspalte* wird als Eingabe für den Geocoder verwendet. Bei jeder Einfügung oder Aktualisierung von Werten in der/den Geocodierungsspalte(n) werden Trigger aktiviert. Diese Trigger rufen den zugeordneten Geocoder auf, damit dieser die eingefügten oder aktualisierten Werte geocodiert und die Ergebnisdaten in die geocodierte Spalte stellt.

Einschränkung: Die automatische Geocodierung kann nur für Tabellen aktiviert werden, für die INSERT- und UPDATE-Trigger erstellt werden können. Infolgedessen kann die automatische Geocodierung nicht für Sichten oder Kurznamen aktiviert werden.

Vorbedingung: Bevor Sie die automatische Geocodierung aktivieren, müssen Sie die Geocodierung konfigurieren. Hierzu rufen Sie die Prozedur ST_SETUP_GEOCODING auf. Bei der Konfiguration der Geocodierung werden Parameterwerte für den Geocoder und die Geocodierung angegeben. Außerdem werden die Geocodierungsspalten angegeben, die mit den geocodierten Spalten synchronisiert werden sollen.

Berechtigung

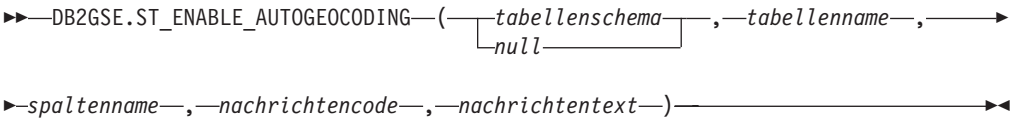
Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte besitzen:

- Berechtigung DBADM für die Datenbank mit der Tabelle, für die die Trigger definiert sind, die durch diese gespeicherte Prozedur erstellt werden.
- Zugriffsrecht CONTROL für die Tabelle
- Zugriffsrecht ALTER für die Tabelle

Falls die Berechtigungs-ID der Anweisung die Berechtigung DBADM nicht besitzt, muss sie - sofern der Trigger vorhanden ist - alle folgenden Zugriffsrechte besitzen (das Zugriffsrecht PUBLIC oder Gruppenzugriffsrechte werden nicht berücksichtigt):

- Zugriffsrecht SELECT für die Tabelle, für die die automatische Geocodierung aktiviert ist, oder Berechtigung DATAACCESS
- Erforderliche Zugriffsrechte für die Auswertung von SQL-Ausdrücken, die in der Geocodierungskonfiguration für die Parameter angegeben sind

Syntax



Parameterbeschreibungen

tabellenschema
Dieser Parameter gibt den Namen des Schemas an, zu dem die Tabelle, die im Parameter *tabellenname* angegeben ist, gehört. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Falls Sie für diesen Parameter den Wert NULL angeben, wird als Schemaname für die Tabelle der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

Der Wert für *tabellenschema* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

tabellenname
Dieser Parameter gibt den Namen ohne Qualifikationsmerkmal der Tabelle an, die die Spalte enthält, in der die geocodierten Daten eingefügt oder aktualisiert werden sollen. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

spaltenname
Dieser Parameter gibt die Spalte an, in der die geocodierten Daten eingefügt oder aktualisiert werden sollen. Diese Spalte wird als geocodierte Spalte bezeichnet. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *spaltenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_ENABLE_AUTOGEOCODING über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird mit einem DB2-Befehl CALL die automatische Geocodierung für die Spalte LOCATION in der Tabelle namens CUSTOMERS aktiviert:

```
call DB2GSE.ST_ENABLE_AUTOGEOCODING(NULL,'CUSTOMERS','LOCATION',?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_ENABLE_DB

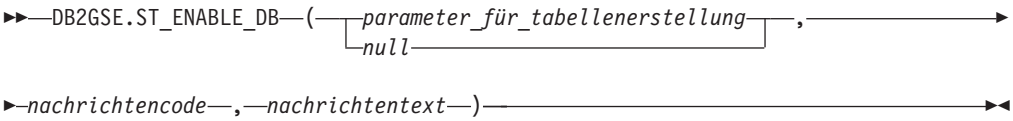
Mit dieser gespeicherten Prozedur können Sie einer Datenbank die Ressourcen zur Verfügung stellen, die zum Speichern von räumlichen Daten und zur Unterstützung räumlicher Operationen erforderlich sind. Zu diesen Ressourcen gehören räumliche Datentypen, Typen von räumlichen Indizes, Katalogsichten, bereitgestellte Funktionen und andere gespeicherte Prozeduren.

Diese gespeicherte Prozedur ersetzt DB2GSE.gse_enable_db.

Berechtigung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss die Berechtigung DBADM für die zu aktivierende Datenbank besitzen.

Syntax



Parameterbeschreibungen

parameter_für_tabellenerstellung
Dieser Parameter gibt alle Optionen an, die zu den Anweisungen CREATE TABLE für die Katalogtabellen von DB2 Spatial Extender hinzugefügt werden sollen. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, werden keine Optionen zu den Anweisungen CREATE TABLE hinzugefügt.

Verwenden Sie zur Angabe der Optionen die Syntax der DB2-Anweisung CREATE TABLE. So können Sie beispielsweise einen Tabellenbereich angeben, in dem die Tabellen erstellt werden sollen:

```
IN tsName INDEX IN indexTsName
```

Dieser Parameter hat den Datentyp VARCHAR(32K).

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_ENABLE_DB über die CLI (Call Level Interface - Schnittstelle auf Aufrufebene) aufgerufen wird:

```
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;

    /* Umgebungskennung zuordnen */
    rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);

    /* Datenbankkennung zuordnen */
    rc = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);

    /* Verbindung zur Datenbank "testdb" herstellen */
    rc = SQLConnect(hdbc, (SQLCHAR *)"testdb", SQL_NTS, (SQLCHAR *)uid,SQL_NTS,
                                    (SQLCHAR *)pwd, SQL_NTS);


    /* Anweisungskennung zuordnen */
    rc = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt) ;

  /* SQL-Anweisung für Aufruf der gespeicherten Prozedur ST_ENABLE_DB    */
  /* der Anweisungskennung zuordnen und Anweisung zur Vorbereitung an DBMS senden. */
  rc = SQLPrepare(hstmt, "call DB2GSE!ST_ENABLE_DB(?,?,?)", SQL_NTS);

    /* 1. Parametermarke in der SQL-Rufanweisung - Eingabeparameter   */
    /* für Tabellenerstellungsparameter - an Variable                 */
    /* table_creation_parameters binden.                              */
    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]);

    /* 2. Parametermarke in der SQL-Rufanweisung - Ausgabeparameter */
    /* für zurückgegebenen Nachrichtencode - an Variable msg_code   */
  /* binden.                                                        */
    ind[1] = 0;
    rc = SQLBindParameter(hstmt, 2, SQL_PARAM_OUTPUT, SQL_C_LONG,
                                    SQL_INTEGER, 0, 0, &msg_code, 4, &ind[1]);

    /* 3. Parametermarke in der SQL-Rufanweisung - Ausgabeparameter */
    /* für zurückgegebenen Nachrichtentext - an Variable msg_text   */
  /* binden.                                                        */
    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);
```

Prozedur ST_EXPORT_SHAPE

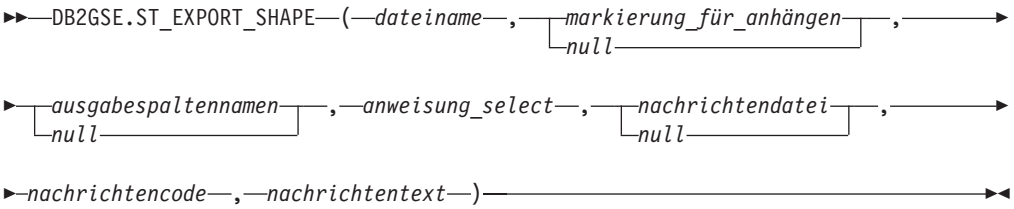
Mit dieser gespeicherten Prozedur können Sie eine räumliche Spalte und ihre zugeordnete Tabelle in eine Formdatei exportieren.

Berechtigung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss die Zugriffsrechte besitzen, die für eine erfolgreiche Ausführung der Anweisung SELECT, über die die Daten exportiert werden, erforderlich sind.

Die gespeicherte Prozedur, deren Ausführung als Prozess erfolgt, dessen Eigner der DB2-Instanzeigner ist, muss die Zugriffsrechte auf der Servermaschine besitzen, die zum Erstellen von oder Schreiben in Formdateien erforderlich sind.

Syntax



Parameterbeschreibungen

dateiname
Dieser Parameter gibt den vollständigen Pfadnamen der Formdatei an, in die die Daten exportiert werden sollen. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Mit der gespeicherten Prozedur ST_EXPORT_SHAPE können Sie für den Export eine neue Datei angeben oder die exportierten Daten an eine vorhandene Datei anhängen lassen:

- Beim Export in eine neue Datei können Sie die optionale Dateierweiterung .shp oder .SHP angeben. Wenn Sie .shp oder .SHP als Dateierweiterung angeben, erstellt DB2 Spatial Extender die Datei mit dem angegebenen Wert für *dateiname*. Falls Sie die optionale Dateierweiterung nicht angeben, erstellt DB2 Spatial Extender die Datei mit dem für *dateiname* angegebenen Namen und mit der Erweiterung .shp.
- Wenn Daten beim Exportieren an eine vorhandene Datei angehängt werden sollen, sucht DB2 Spatial Extender zunächst nach einer exakten Übereinstimmung mit dem Namen, den Sie für den Parameter *dateiname* angegeben haben. Falls DB2 Spatial Extender keine exakte Übereinstimmung feststellt, wird zunächst nach einer Datei mit der Erweiterung .shp und dann nach einer Datei mit der Erweiterung .SHP gesucht.

Wenn der Wert für den Parameter *markierung_für_anhängen* angibt, dass die Daten nicht an eine vorhandene Datei angehängt werden sollen, die im Parameter *dateiname* bezeichnete Datei jedoch bereits vorhanden ist, gibt DB2 Spatial Extender einen Fehler zurück. Ein Überschreiben der Datei findet nicht statt.

Eine Liste der Dateien, die auf der Servermaschine geschrieben werden, finden Sie unter Hinweise zur Verwendung. Die gespeicherte Prozedur, deren Ausführung als Prozess erfolgt, dessen Eigner der DB2-Instanzeigner ist, muss die Zugriffsrechte auf der Servermaschine besitzen, die zum Erstellen von oder Schreiben in Formdateien erforderlich sind.

Dieser Parameter hat den Datentyp VARCHAR(256).

markierung_für_anhängen
Dieser Parameter gibt an, ob die zu exportierenden Daten an eine vorhandene

Formdatei angehängt werden sollen. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. So geben Sie an, ob die Daten an eine vorhandene Formdaten angehängt werden sollen:

- Wenn die Daten an eine vorhandene Formdatei angehängt werden sollen, geben Sie einen anderen Wert als 0 (null) oder NULL an. In diesem Fall muss die Dateistruktur mit den exportierten Daten übereinstimmen. Andernfalls wird ein Fehler zurückgegeben.
- Wenn Sie die Daten in eine neue Datei exportieren wollen, geben Sie den Wert null oder NULL an. In diesem Fall werden vorhandene Dateien von DB2 Spatial Extender nicht überschrieben.

Dieser Parameter hat den Datentyp SMALLINT.

ausgabespaltennamen
Dieser Parameter gibt einen oder mehrere (durch Kommas voneinander getrennte) Spaltennamen an, die in der dBASE-Ausgabedatei für nicht-räumliche Spalten verwendet werden sollen. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn für diesen Parameter den Wert NULL hat, werden die Namen verwendet, die aus der Anweisung SELECT abgeleitet werden.

Falls Sie diesen Parameter angeben, die Spaltennamen jedoch nicht in doppelte Anführungszeichen setzen, werden die Spaltennamen in Großbuchstaben umgewandelt. Die Anzahl der angegebenen Spalten muss mit der Anzahl der Spalten übereinstimmen, die von der im Parameter *anweisung_select* angegebenen Anweisung SELECT zurückgegeben werden. Die räumliche Spalte ist in diesem Wert nicht enthalten.

Dieser Parameter hat den Datentyp VARCHAR(32K).

anweisung_select
Dieser Parameter gibt den Subselect an, der die zu exportierenden Daten zurückgibt. Der Subselect muss sich auf genau eine räumliche Spalte und eine beliebige Anzahl von Attributspalten beziehen. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Dieser Parameter hat den Datentyp VARCHAR(32K).

nachrichtendatei
Dieser Parameter gibt den vollständigen Pfadnamen der Datei (auf der Servermaschine) an, in der die Nachrichten über die Exportoperation gespeichert werden sollen. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird keine Datei für die DB2 Spatial Extender-Nachrichten erstellt.

Die folgenden Nachrichten können an diese Nachrichtendatei gesendet werden:

- Informationsnachrichten, beispielsweise eine Zusammenfassung der Exportoperation
- Fehlernachrichten für Daten, die nicht exportiert werden konnten, beispielsweise aufgrund eines abweichenden Koordinatensystems

Die gespeicherte Prozedur, deren Ausführung als Prozess erfolgt, dessen Eigner der DB2-Instanzeigner ist, muss die Zugriffsrechte auf dem Server besitzen, die zum Erstellen der Datei erforderlich sind.

Dieser Parameter hat den Datentyp VARCHAR(256).

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Hinweise zur Verwendung

Sie können immer nur eine räumliche Spalte auf einmal exportieren.

Die gespeicherte Prozedur ST_EXPORT_SHAPE erstellt die folgenden vier Dateien bzw. schreibt in diese Dateien:

- die eigentliche Formdatei (Erweiterung .shp)
- die Formindexdatei (Erweiterung .shx)
- eine dBASE-Datei, die Daten für nicht-räumliche Spalten enthält (Erweiterung .dbf). Diese Datei wird nur dann erstellt, wenn tatsächlich Attributspalten exportiert werden müssen.
- eine Projektionsdatei, die das Koordinatensystem angibt, das den räumlichen Daten zugeordnet ist, falls das Koordinatensystem nicht "UNSPECIFIED" lautet (Erweiterung .prj). Das Koordinatensystem wird dem ersten räumlichen Datensatz entnommen. Falls in nachfolgenden Datensätzen andere Koordinatensysteme angegeben sind, tritt ein Fehler auf.

Die folgende Tabelle gibt Aufschluss darüber, wie DB2-Datentypen in dBASE-Attributdateien gespeichert sind. Alle nicht angegebenen DB2-Datentypen werden nicht unterstützt.

Tabelle 22. Speicherung von DB2-Datentypen in Attributdateien

SQL-Typ	Typ bei .dbf	Länge bei .dbf	Dezimalstellen bei .dbf	Kommentare
SMALLINT	N	6	0	
INTEGER	N	11	0	
BIGINT	N	20	0	
DECIMAL	N	Genauigkeit+2	Maßstab	
REAL FLOAT(1) über FLOAT(24)	F	14	6	
DOUBLE FLOAT(25) über FLOAT(53)	F	19	9	

Tabelle 22. Speicherung von DB2-Datentypen in Attributdateien (Forts.)

SQL-Typ	Typ bei .dbf	Länge bei .dbf	Dezimalstellen bei .dbf	Kommentare
CHARACTER, VARCHAR, LONG VARCHAR und DATALINK	C	*länge*	0	länge ≤ 255
DATE	D	8	0	
TIME	C	8	0	
TIMESTAMP	C	26	0	

Alle Synonyme für Datentypen und einzigartige Datentypen, die auf den in der vorstehenden Tabelle aufgeführten Typen basieren, werden unterstützt.

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_EXPORT_SHAPE über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel werden mit einem DB2-Befehl CALL alle Zeilen aus der Tabelle CUSTOMERS in eine zu erstellende Formdatei namens `/tmp/exportdatei` exportiert:

```
call DB2GSE.ST_EXPORT_SHAPE('/tmp/exportdatei',0,NULL,
          'select * from customers','/tmp/export_msg',?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_IMPORT_SHAPE

Mit dieser gespeicherten Prozedur können Sie eine Formdatei in eine Datenbank importieren, die für räumliche Operationen aktiviert wurde.

Je nach dem Wert, der für den Parameter *markierung_für_tabellenerstellung* angegeben ist, gibt es für die Ausführung dieser gespeicherten Prozedur zwei Methoden:

- DB2 Spatial Extender kann eine Tabelle mit einer räumlichen Spalte und mit Attributspalten erstellen und dann die Daten der Datei in die Spalten der Tabelle laden.
- Andernfalls können die Form- und Attributdaten in eine vorhandene Tabelle geladen werden, deren räumliche Spalte und Attributspalten mit denen der Datendateien übereinstimmen.

Berechtigung

Der Eigner der DB2-Instanz muss die Zugriffsrechte auf der Servermaschine besitzen, die für das Lesen der Eingabedateien und das optionale Schreiben von Fehlerdateien benötigt werden. Welche weiteren Berechtigungen erforderlich sind, ist davon abhängig, ob die Daten in eine vorhandene Tabelle oder in eine neue Tabelle importiert werden sollen.

- Beim **Importieren in eine vorhandene Tabelle** muss die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte besitzen:
  - DATAACCESS
  - Zugriffsrecht CONTROL für die Tabelle bzw. Sicht

  - Zugriffsrecht INSERT und SELECT für die Tabelle bzw. Sicht
- Beim **Importieren in eine neue Tabelle** muss die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte besitzen:
  - DBADM
  - Berechtigung CREATETAB für die Datenbank

  Außerdem muss die Benutzer-ID eine der folgenden Berechtigungen besitzen:
  - Berechtigung IMPLICIT_SCHEMA für die Datenbank, falls der Schemaname der Tabelle nicht vorhanden ist
  - Zugriffsrecht CREATEIN für das Schema, falls das Schema der Tabelle vorhanden ist

Syntax

Parameterbeschreibungen

dateiname
: Dieser Parameter gibt den vollständigen Pfadnamen der Formdatei an, die importiert werden soll. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

  Wenn Sie die optionale Dateierweiterung angeben wollen, verwenden Sie .shp oder .SHP. DB2 Spatial Extender sucht zunächst nach einer exakten Übereinstimmung mit dem angegebenen Dateinamen. Falls DB2 Spatial Extender keine exakte Übereinstimmung feststellt, wird zunächst nach einer Datei mit der Erweiterung .shp und dann nach einer Datei mit der Erweiterung .SHP gesucht.

  Eine Liste der erforderlichen Dateien, die auf der Servermaschine gespeichert sein müssen, finden Sie unter Hinweise zur Verwendung. Die gespeicherte Prozedur, deren Ausführung als Prozess erfolgt, dessen Eigner der DB2-Instanzeigner ist, muss die Zugriffsrechte auf dem Server besitzen, die zum Lesen der Dateien erforderlich sind.

Dieser Parameter hat den Datentyp VARCHAR(256).

attributspalten_in_eingabedatei
Dieser Parameter gibt eine Liste der Attributspalten an, die aus der dBASE-Datei importiert werden sollen. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, werden keine Spalten importiert. Falls die dBASE-Datei nicht vorhanden ist, muss für diesen Parameter eine leere Zeichenfolge oder der Wert NULL angegeben werden.

Wenn Sie für diesen Parameter einen anderen Wert als NULL angeben wollen, verwenden Sie eine der folgenden Spezifikationen:

- **Geben Sie die Namen der Attributspalten in einer Liste an.** Das folgende Beispiel veranschaulicht, wie die Namen der Attributspalten, die aus der dBASE-Datei importiert werden sollen, in einer Liste angegeben werden:
  ```
  N(SPALTE1,SPALTE5,SPALTE3,SPALTE7)
  ```

  Sofern ein Spaltenname nicht in doppelte Anführungszeichen gesetzt ist, wird er in Großbuchstaben umgewandelt. Jeder Name in der Liste muss durch ein Komma abgegrenzt werden. Die resultierenden Namen müssen genau mit den Spaltennamen in der dBASE-Datei übereinstimmen.
- **Geben Sie die Nummern der Attributspalten in einer Liste an.** Das folgende Beispiel veranschaulicht, wie die Nummern der Attributspalten, die aus der dBASE-Datei importiert werden sollen, in einer Liste angegeben werden:
  ```
  P(1,5,3,7)
  ```

  Die Spaltennummerierung beginnt bei 1. Jede Nummer in der Liste muss durch ein Komma abgegrenzt werden.
- **Geben Sie an, dass keine Attributdaten importiert werden sollen.** Geben Sie die Zeichen "" an. Hierbei handelt es sich um eine leere Zeichenfolge, die explizit angibt, dass DB2 Spatial Extender *keine* Attributdaten importieren soll.

Dieser Parameter hat den Datentyp VARCHAR(32K).

name_des_räumlichen_bezugssystems
Dieser Parameter gibt das räumliche Bezugssystem an, das für die Geometrien, die in die räumliche Spalte importiert werden, verwendet werden soll. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Die räumliche Spalte wird nicht registriert. Das räumliche Bezugssystem muss vorhanden sein, bevor die Daten importiert werden. Der Importprozess nimmt keine implizite Erstellung des räumlichen Bezugssystems vor, vergleicht jedoch das Koordinatensystem des räumlichen Bezugssystems mit dem Koordinatensystem, das in der Datei .prj (sofern zusammen mit der Formdatei verfügbar) angegeben ist. Außerdem wird während des Importprozesses überprüft, ob der Bereich der Daten in der Formdatei im angegebenen räumlichen Bezugssystem dargestellt werden kann. Der Importprozess überprüft also, ob die Bereiche innerhalb der kleinstmöglichen und größtmöglichen X-, Y-, Z- und M-Koordinaten des räumlichen Bezugssystems liegen.

Der Wert für *name_des_räumlichen_bezugssystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

tabellenschema
Dieser Parameter gibt den Namen des Schemas an, zu dem die Tabelle, die im

Parameter *tabellenname* angegeben ist, gehört. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Falls Sie für diesen Parameter den Wert NULL angeben, wird als Schemaname für die Tabelle bzw. Sicht der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

Der Wert für *tabellenschema* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

tabellenname
Dieser Parameter gibt den Namen ohne Qualifikationsmerkmal der Tabelle an, in die die importierte Formdatei geladen werden soll. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

attributspalten_in_tabelle
Dieser Parameter gibt die Namen der Tabellenspalten an, in denen die Attributdaten aus der dBASE-Datei gespeichert werden sollen. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn für diesen Parameter der Wert NULL verwendet wird, werden die Namen der Spalten in der dBASE-Datei verwendet.

Wird dieser Parameter angegeben, muss die Anzahl der Namen mit der Anzahl der Spalten identisch sein, die aus der dBASE-Datei importiert werden sollen. Wenn die Tabelle vorhanden ist, müssen die Spaltendefinitionen mit den ankommenden Daten identisch sein. Erläuterungen zur Zuordnung von Attributdatentypen zu den entsprechenden DB2-Datentypen finden Sie unter Hinweise zur Verwendung.

Dieser Parameter hat den Datentyp VARCHAR(32K).

markierung_für_tabellenerstellung
Dieser Parameter gibt an, ob der Importprozess eine neue Tabelle erstellen soll. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn für diesen Parameter kein Wert oder ein anderer Wert als 0 (null) angegeben ist, wird eine neue Tabelle erstellt. (Falls die Tabelle bereits vorhanden ist, wird ein Fehler zurückgegeben.) Hat dieser Parameter den Wert null, wird keine Tabelle erstellt, und die Tabelle muss vorhanden sein.

Dieser Parameter hat den Datentyp INTEGER.

parameter_für_tabellenerstellung
Dieser Parameter gibt alle Optionen an, die zu der Anweisung CREATE TABLE hinzugefügt werden sollen, mit der eine Tabelle für die zu importierenden Daten erstellt wird. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, werden keine Optionen zur Anweisung CREATE TABLE hinzugefügt.

Verwenden Sie zur Angabe der Optionen für CREATE TABLE die Syntax der DB2-Anweisung CREATE TABLE. So können Sie beispielsweise einen Tabellenbereich angeben, in dem die Tabellen erstellt werden sollen:

```
IN tabellenbereichsname INDEX IN indextabellenbereichsname LONG IN
langer_tabellenbereichsname
```

Dieser Parameter hat den Datentyp VARCHAR(32K).

räumliche_spalte
Dieser Parameter gibt den Namen der räumlichen Spalte in der Tabelle an, in die die Formdaten geladen werden sollen. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Bei einer neuen Tabelle gibt dieser Parameter den Namen der neuen räumlichen Spalte an, die erstellt werden soll. Andernfalls gibt dieser Parameter den Namen einer vorhandenen räumlichen Spalte in der Tabelle an.

Der Wert für *räumliche_spalte* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

typschema
Dieser Parameter gibt den Schemanamen des räumlichen Datentyps (angegeben durch den Parameter *typname*) an, der beim Erstellen einer räumlichen Spalte in einer neuen Tabelle verwendet werden soll. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert DB2GSE verwendet.

Der Wert für *typschema* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

typname
Dieser Parameter gibt den Namen des Datentyps an, der für die räumlichen Werte verwendet werden soll. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Datentyp anhand der Formdatei ermittelt. Die folgenden Typen sind möglich:

- ST_Point
- ST_MultiPoint
- ST_MultiLineString
- ST_MultiPolygon

Bitte beachten Sie, dass Formdateien definitionsgemäß nur eine Unterscheidung zwischen Punkten und Mehrpunktangaben, nicht jedoch zwischen Polygonen und Multipolygonen oder zwischen Linienfolgen und Mehrlinienfolgen zulassen.

Wenn Sie Daten in eine noch nicht vorhandene Tabelle importieren wollen, wird dieser Datentyp auch für die räumliche Spalte verwendet. In diesem Fall kann der Datentyp auch ein übergeordneter Typ von ST_Point, ST_MultiPoint, ST_MultiLineString oder ST_MultiPolygon sein.

Der Wert für *typname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

inlinelänge
Dieser Parameter gibt für eine neue Tabelle an, wie viele Byte in der Tabelle maximal für die räumliche Spalte zugeordnet werden dürfen. Sie müssen zwar

einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird in der Anweisung CREATE TABLE keine explizite Option INLINE LENGTH verwendet. Stattdessen werden implizit die DB2-Standardwerte verwendet.

Räumliche Datensätze, die diese Größe überschreiten, werden im LOB-Tabellenbereich separat gespeichert. Der Zugriff auf diesen Tabellenbereich kann möglicherweise länger dauern.

Die folgenden Größen werden üblicherweise für unterschiedliche räumliche Typen benötigt:

- **Einzelpunkt:** 292.
- **Mehrpunktangabe, Linie oder Polygon:** Der Wert sollte so groß wie möglich sein. Achten Sie darauf, dass die Gesamtanzahl der Byte in einer Zeile den Grenzwert für die Seitengröße des Tabellenbereichs, für den die Tabelle erstellt wird, nicht überschreitet.

Eine vollständige Beschreibung dieses Wertes finden Sie in der DB2-Dokumentation zur SQL-Anweisung CREATE TABLE. Angaben dazu, wie Sie die Anzahl der Inlinegeometrien für vorhandene Tabellen ermitteln und die Inlinelänge ändern können, finden Sie in den Informationen zum Dienstprogramm "db2dart".

Dieser Parameter hat den Datentyp INTEGER.

id_spalte
Dieser Parameter gibt den Namen einer zu erstellenden Spalte an, in der für jede Datenzeile eine eindeutige Nummer gespeichert werden soll. (Bei ESRI-Tools muss die Spalte mit SE_ROW_ID benannt werden.) Die eindeutigen Werte für diese Spalte werden während des Importprozesses automatisch generiert. Sie müssen zwar einen Wert für diesen Parameter angeben, aber dieser Wert kann NULL sein, falls keine Spalte (mit einer eindeutigen ID in jeder Zeile) in der Tabelle vorhanden ist oder falls Sie keine derartige Spalte zu einer neu erstellten Tabelle hinzufügen wollen. Wenn Sie für diesen Parameter den Wert NULL angeben, werden keine Spalten erstellt oder mit eindeutigen Nummer gefüllt.

Einschränkung: Der Name für *id_spalte* darf nicht mit einem der Spaltennamen in der dBASE-Datei identisch sein.

Die Voraussetzungen und Auswirkungen dieses Parameters sind davon abhängig, ob die Tabelle bereits vorhanden ist.

- **Bei einer vorhandenen Tabelle** kann der Typ des Parameters *id_spalte* jeder beliebige Integertyp sein (INTEGER, SMALLINT oder BIGINT).
- **Bei einer neu zu erstellenden Tabelle** wird die Spalte zur Tabelle hinzugefügt, wenn diese durch die gespeicherte Prozedur erstellt wird. Die Spalte wird folgendermaßen definiert:

  ```
  INTEGER NOT NULL PRIMARY KEY
  ```

  Falls der Parameter *id_spalte_ist_identitätsspalte* einen anderen Wert als 0 (null) oder NULL hat, wird die Definition wie folgt erweitert:

  ```
  INTEGER NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY
  ( START WITH 1 INCREMENT BY 1 )
  ```

Der Wert für *id_spalte* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

id_spalte_ist_identitätsspalte
Dieser Parameter gibt an, ob die in *id_spalte* angegebene Spalte mit der Klausel IDENTITY erstellt werden soll. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Hat dieser Parameter den Wert null oder NULL, wird die Spalte nicht als Identitätsspalte erstellt. Wenn der Parameter einen anderen Wert als 0 (null) oder NULL, wird die Spalte als Identitätsspalte erstellt. Bei bereits vorhandenen Tabellen wird dieser Parameter ignoriert.

Dieser Parameter hat den Datentyp SMALLINT.

zähler_für_neustart
Gibt an, dass eine Importoperation bei Datensatz *n* + 1 gestartet werden soll. Die ersten *n* Datensätze werden übersprungen. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, werden alle Datensätze (beginnend mit dem Datensatz Nr. 1) importiert.

Dieser Parameter hat den Datentyp INTEGER.

commitbereich
Dieser Parameter gibt an, dass ein Commit durchgeführt werden soll, nachdem mindestens *n* Datensätze importiert wurden. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird der Wert 0 (null) verwendet, und am Ende der Operation wird ein COMMIT durchgeführt. Dies kann in umfangreichen Protokolldateien und Datenverlusten resultieren, wenn Operationen unterbrochen werden.

Dieser Parameter hat den Datentyp INTEGER.

ausnahmedatei
Dieser Parameter gibt den vollständigen Pfadnamen der Formdatei an, in der die Formdaten gespeichert werden sollen, die nicht importiert werden konnten. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, werden keine Dateien erstellt.

Falls Sie einen Wert für den Parameter angeben und die optionale Dateierweiterung angeben wollen, verwenden Sie .shp oder .SHP. Wenn für die Erweiterung der Wert NULL angegeben ist, wird die Erweiterung .shp angehängt.

Die Ausnahmedatei enthält den vollständigen Zeilenblock, für den eine einzelne Einfügeanweisung fehlgeschlagen ist. Beispiel: Angenommen, eine Zeile kann nicht importiert werden, weil die Formdaten nicht richtig codiert sind. Eine einzelne Einfügeanweisung versucht, 20 Zeilen (einschließlich der fehlerhaften Zeile) zu importieren. Aufgrund der fehlerhaften Einzelzeile wird der gesamte Block von 20 Zeilen in die Ausnahmedatei geschrieben.

Datensätze werden nur dann in die Ausnahmedatei geschrieben, wenn sie korrekt identifiziert werden können (beispielsweise dann, wenn der Formdatensatztyp nicht gültig ist). Manchmal sind Formdaten (Dateien .shp) und Formindizes (Dateien .shx) auf eine Weise beschädigt, bei der die entsprechenden Datensätze nicht identifiziert werden können. In einem solchen Fall werden keine Datensätze in die Ausnahmedatei geschrieben, und das Problem wird in Form einer Fehlernachricht gemeldet.

Falls Sie für diesen Parameter einen Wert angeben, werden auf der Servermaschine vier Dateien erstellt. Eine Erläuterung dieser Dateien finden Sie unter Hinweise zur Verwendung. Die gespeicherte Prozedur, deren Ausführung als Prozess erfolgt, dessen Eigner der DB2-Instanzeigner ist, muss die Zugriffsrech-

te auf dem Server besitzen, die zum Erstellen der Datei erforderlich sind. Wenn die Dateien bereits vorhanden sind, gibt die gespeicherte Prozedur einen Fehler zurück.

Dieser Parameter hat den Datentyp VARCHAR(256).

nachrichtendatei
Dieser Parameter gibt den vollständigen Pfadnamen der Datei (auf der Servermaschine) an, in der die Nachrichten über die Importoperation gespeichert werden sollen. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird keine Datei für die DB2 Spatial Extender-Nachrichten erstellt.

Die folgenden Nachrichten können in diese Nachrichtendatei geschrieben werden:

- Informationsnachrichten, beispielsweise eine Zusammenfassung der Importoperation
- Fehlernachrichten für Daten, die nicht importiert werden konnten, beispielsweise aufgrund eines abweichenden Koordinatensystems

Diese Nachrichten entsprechen den Formdaten, die in der Ausnahmedatei (durch den Parameter *ausnahmedatei* angegeben) gespeichert werden.

Die gespeicherte Prozedur, deren Ausführung als Prozess erfolgt, dessen Eigner der DB2-Instanzeigner ist, muss die Zugriffsrechte auf dem Server besitzen, die zum Erstellen der Datei erforderlich sind. Wenn die Datei bereits vorhanden ist, gibt die gespeicherte Prozedur einen Fehler zurück.

Dieser Parameter hat den Datentyp VARCHAR(256).

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Hinweise zur Verwendung

Die gespeicherte Prozedur ST_IMPORT_SHAPE verwendet zwischen einer und vier Dateien:

- die eigentliche Formdatei (Erweiterung .shp). Diese Datei ist erforderlich.
- die Formindexdatei (Erweiterung .shx). Diese Datei ist optional. Wenn sie vorhanden ist, kann die Leistung der Importoperation möglicherweise gesteigert werden.

- eine dBASE-Datei, die Attributdaten enthält (Erweiterung .dbf). Diese Datei ist nur dann erforderlich, wenn Attributdaten importiert werden sollen.
- die Projektionsdatei, die das Koordinatensystem der Formdaten angibt (Erweiterung .prj). Diese Datei ist optional. Wenn sie vorhanden ist, wird das in ihr definierte Koordinatensystem mit dem Koordinatensystem des räumlichen Bezugssystems verglichen, das im Parameter *id_des_räumlichen_bezugssystems* angegeben ist.

Die folgende Tabelle beschreibt, wie dBASE-Attributdatentypen den DB2-Datentypen zugeordnet werden. Alle nicht angegebenen Attributdatentypen werden nicht unterstützt.

Tabelle 23. Beziehung zwischen DB2-Datentypen und dBASE-Attributdatentypen

Typ bei .dbf	Länge bei .dbf (siehe Anmerkung)	Dezimalstellen bei .dbf (siehe Anmerkung)	SQL-Typ	Kommentare
N	< 5	0	SMALLINT	
N	< 10	0	INTEGER	
N	< 20	0	BIGINT	
N	*länge*	*dezimalstellen*	DECIMAL (*länge*,*dezimalstellen*)	*länge*<32
F	*länge*	*dezimalstellen*	REAL	*länge* + *dezimalstellen* < 7
F	*länge*	*dezimalstellen*	DOUBLE	
C	*länge*		CHAR(*länge*)	
L			CHAR(1)	
D			DATE	

Anmerkung: Diese Tabelle enthält die folgenden beiden Variablen, die in den Kopfdaten der dBASE-Datei definiert sind:

- Die Variable *länge* stellt die Gesamtlänge der Spalte in der dBASE-Datei dar. DB2 Spatial Extender verwendet diesen Wert, um
  - die Genauigkeit für den SQL-Datentyp DECIMAL oder die Länge für den SQL-Datentyp CHAR zu definieren,
  - den zu verwendenden Integer- oder Gleitkommatyp zu ermitteln.
- Die Variable *dezimalstellen* gibt an, wie viele Stellen rechts vom Dezimalzeichen der Spalte in der dBASE-Datei maximal zulässig sind. DB2 Spatial Extender verwendet diesen Wert, um die Anzahl der Kommastellen für den SQL-Datentyp DECIMAL zu definieren.

Beispiel: Angenommen, die dBASE-Datei enthält eine Datenspalte, deren Länge (*länge*) mit dem Wert 20 definiert ist. Für die Anzahl der Stellen rechts vom Dezimalzeichen (*dec*) wird der Wert 5 angenommen. Beim Importieren von Daten aus dieser Spalte leitet DB2 Spatial Extender aus den Werten der Variablen *länge* und *dezimalstellen* den folgenden SQL-Datentyp ab: DECIMAL(20,5).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_IMPORT_SHAPE über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird mit einem DB2-Befehl CALL eine Formdatei namens `/tmp/officesShape` in die Tabelle OFFICES importiert:

```
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',?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_REGISTER_GEOCODER

Mit dieser gespeicherten Prozedur können Sie einen Geocoder registrieren.

Vorbedingungen: Vor der Registrierung eines Geocoders sollten Sie die folgenden Punkte beachten:

- Vergewissern Sie sich, dass die Funktion, die den Geocoder implementiert, bereits erstellt wurde. Jede Geocoderfunktion kann als Geocoder mit einem eindeutig gekennzeichneten Geocodernamen registriert werden.
- Erfragen Sie beim Lieferanten des Geocoders die folgenden Angaben:
  - SQL-Anweisung, die die Funktion erstellt
  - Die Parameterwerte, die zum Aufrufen der Prozedur ST_CREATE_SRS verwendet werden, damit geometrische Daten unterstützt werden können
  - Informationen zur Registrierung des Geocoders, beispielsweise:
    - Beschreibung des Geocoders
    - Beschreibung der Parameter für den Geocoder
    - Standardwerte für die Geocoderparameter

Der Rückgabetyp der Geocoderfunktion muss mit dem Datentyp der geocodierten Spalte identisch sein. Als Geocodierungsparameter können Spaltennamen (so genannte *Geocodierungsspalten*) verwendet werden, die vom Geocoder benötigte Daten enthalten. Die Geocoderparameter können beispielsweise Adressen oder einen Wert mit einer bestimmten Bedeutung für den Geocoder (z. B. die Mindestübereinstimmungsquote) angeben. Wenn es sich bei einem Geocodierungsparameter um einen Spaltennamen handelt, muss die Spalte in derselben Tabelle oder Sicht wie die geocodierte Spalte enthalten sein.

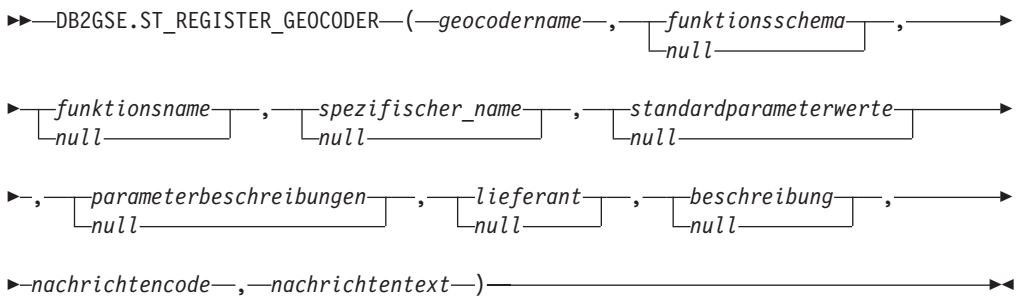
Der Rückgabetyp der Geocoderfunktion dient als Datentyp für die geocodierte Spalte. Der Rückgabetyp kann ein beliebiger DB2-Datentyp, benutzerdefinierter Typ oder strukturierter Typ sein. Wenn ein benutzerdefinierter oder ein strukturierter Typ zurückgegeben wird, muss die Geocoderfunktion dafür sorgen, dass ein gültiger Wert des entsprechenden Datentyps zurückgegeben wird. Gibt die Geocoderfunktion Werte eines räumlichen Typs zurück (also ST_Geometry oder einer seiner untergeordneten Typen), muss die Geocoderfunktion dafür sorgen, dass eine gültige Geometrie erstellt wird. Die Geometrie muss mithilfe eines vorhandenen räumlichen Bezugssystems dargestellt werden. Die Geometrie ist gültig, wenn Sie die räumliche Funktion ST_IsValid für die Geometrie aufrufen und der Wert 1 zurückgegeben wird. Die von der Geocoderfunktion zurückgegebenen Daten werden in der geocodierten Spalte aktualisiert oder in die Spalte eingefügt. Dies ist davon abhängig, welche Operation (INSERT oder UPDATE) die Generierung des geocodierten Werts verursacht hat.

In der Katalogsicht DB2GSE.ST_GEOCODERS können Sie ermitteln, ob ein Geocoder bereits registriert ist.

Berechtigung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss die Berechtigung DBADM für die zu aktivierende Datenbank besitzen.

Syntax



Parameterbeschreibungen

geocodername
Dieser Parameter kennzeichnet den Geocoder auf eindeutige Weise. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *geocodername* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

funktionsschema
Dieser Parameter gibt den Namen des Schemas für die Funktion an, die diesen Geocoder implementiert. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Falls Sie für diesen Parameter den Wert NULL angeben, wird als Schemaname für die Funktion der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

Der Wert für *funktionsschema* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

funktionsname
Dieser Parameter gibt den Namen ohne Qualifikationsmerkmal der Funktion an, die diesen Geocoder implementiert. Die Funktion muss bereits erstellt worden sein und in SYSCAT.ROUTINES aufgeführt sein.

Für diesen Parameter können Sie den Wert NULL angeben, wenn Sie einen Wert für den Parameter *spezifischer_name* angeben. Falls der Parameter *spezifischer_name* nicht angegeben ist, muss der Wert für *funktionsname* zusammen mit dem implizit oder explizit definierten Wert für *funktionsschema* die eindeutige Kennzeichnung der Funktion ergeben. Wird der Parameter *funktionsname* nicht angegeben, ruft DB2 Spatial Extender den Wert für *funktionsname* aus der Katalogsicht SYSCAT.ROUTINES ab.

Der Wert für *funktionsname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

spezifischer_name
Dieser Parameter gibt den spezifischen Namen der Funktion an, die den Geocoder implementiert. Die Funktion muss bereits erstellt worden sein und in SYSCAT.ROUTINES aufgeführt sein.

Für diesen Parameter können Sie den Wert NULL angeben, wenn der Parameter *funktionsname* angegeben ist und die Kombination der Werte für *funktionsschema* und *funktionsname* die Geocoderfunktion eindeutig kennzeichnet. Falls der Name der Geocoderfunktion überlastet ist, kann der Parameter *spezifischer_name* nicht NULL sein. (Ein Funktionsname ist *überlastet*, wenn eine oder mehrere andere Funktionen denselben Namen, jedoch keine identischen Parameter oder Parameterdatentypen verwenden.)

Der Wert für *spezifischer_name* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

standardparameterwerte
Dieser Parameter gibt eine Liste der Standardwerte der Geocodierungsparameter für die Geocoderfunktion an. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn der gesamte Wert für *standardparameterwerte* NULL ist, sind alle Standardparameterwerte Nullwerte.

Bei der Angabe von Parameterwerten verwenden Sie die Reihenfolge, in der die Parameter durch die Funktion definiert sind, und grenzen Sie die einzelnen Werte durch ein Komma voneinander ab. Beispiel:

```
standardwert_für_parameter1,standardwert_für_parameter2,...
```

Jeder Parameterwert ist ein SQL-Ausdruck, Beachten Sie die folgenden Richtlinien:

- Ein Zeichenfolgewert muss in einfache Anführungszeichen gesetzt werden.
- Ein Zahlenwert darf nicht in einfache Anführungszeichen gesetzt werden.
- Wenn der Parameterwert NULL ist, muss er in den korrekten Typ umgesetzt werden. Statt lediglich NULL anzugeben, geben Sie beispielsweise Folgendes an:

  ```
  CAST(NULL AS INTEGER)
  ```

- Wenn der Geocodierungsparameter eine Geocodierungsspalte sein soll, geben Sie keinen Standardwert für den Parameter an.

Falls ein Parameterwert nicht angegeben wird, Sie also zwei aufeinander folgende Kommas (`...,,...`) angeben, muss dieser Parameter entweder beim Definieren der Geocodierung oder bei der Ausführung der Geocodierung im Stapelmodus mit dem Parameter *parameterwerte* der jeweiligen gespeicherten Prozedur angegeben werden.

Dieser Parameter hat den Datentyp VARCHAR(32K).

parameterbeschreibungen
Dieser Parameter gibt eine Liste der Beschreibungen der Geocodierungsparameter für die Geocoderfunktion an. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein.

Wenn der gesamte Wert für *parameterbeschreibungen* NULL ist, sind alle Parameterbeschreibungen Nullwerte. Jede angegebene Parameterbeschreibung erläutert die Bedeutung und die Verwendung des Parameters. Sie kann bis zu 256 Zeichen lang sein. Die Beschreibungen der Parameter müssen durch Kommas voneinander abgegrenzt werden und in der Reihenfolge erscheinen, in der die Parameter durch die Funktion definiert sind. Wenn Sie innerhalb der Beschrei-

bung eines Parameters ein Komma verwenden wollen, setzen Sie die Zeichenfolge in einfache oder doppelte Anführungszeichen. Beispiel:

```
beschreibung1,'beschreibung2, die ein komma enthält',beschreibung3
```

Dieser Parameter hat den Datentyp VARCHAR(32K).

lieferant
Dieser Parameter gibt den Namen des Lieferanten an, der den Geocoder implementiert hat. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, werden keine Informationen zum Lieferanten aufgezeichnet, der den Geocoder implementiert hat.

Dieser Parameter hat den Datentyp VARCHAR(128).

beschreibung
Dieser Parameter beschreibt den Geocoder, indem seine Anwendung erläutert wird. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, werden keine Beschreibungsangaben über den Geocoder aufgezeichnet.

Dieser Parameter hat den Datentyp VARCHAR(256).

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel geht davon aus, dass ein Geocoder erstellt werden soll, der Breitengrad- und Längengradwerte als Eingabe verwendet und in Form von räumlichen Daten des Typs ST_Point geocodiert. Hierzu muss zunächst eine Funktion namens lat_long_gc_func erstellt werden. Anschließend wird ein Geocoder namens SAMPLEGC erstellt, der die Funktion lat_long_gc_func verwendet.

Beispiel für die SQL-Anweisung, die die Funktion lat_long_gc_func zur Rückgabe von ST_Point erstellt:

```
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)
```

Nachdem die Funktion erstellt wurde, können Sie sie als Geocoder registrieren. Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_REGISTER_GEOCODER über den Befehl CALL des DB2-Befehlszeilenprozessors aufgerufen wird und einen Geocoder namens SAMPLEGC mit der Funktion 'lat_long_gc_func' registriert:

```
call DB2GSE.ST_REGISTER_GEOCODER ('SAMPLEGC',NULL,'LAT_LONG_GC_FUNC',',,1'
                                                ,NULL,'My Company','Latitude/Longitude to
ST_Point Geocoder'?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_REGISTER_SPATIAL_COLUMN

Mit dieser gespeicherten Prozedur können Sie eine räumliche Spalte registrieren und ihr ein räumliches Bezugssystem zuordnen.

Mithilfe dieser gespeicherten Prozedur können Sie auch die geografischen Bereiche der räumlichen Spalte berechnen.

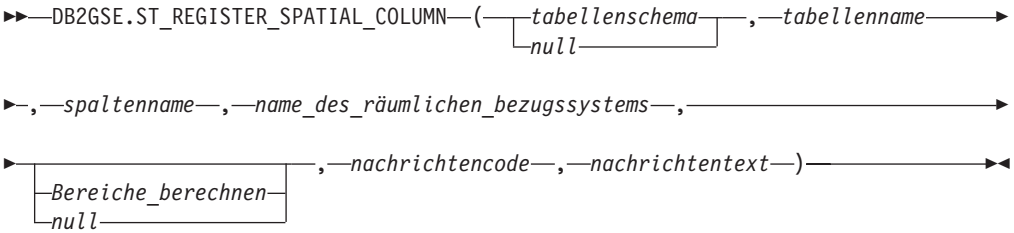
Nachdem diese gespeicherte Prozedur verarbeitet wurde, können Informationen zur registrierten räumlichen Spalte und zu den geografischen Bereichen in der Katalogsicht DB2GSE.ST_GEOMETRY_COLUMNS abgerufen werden. Beim Registrieren einer räumlichen Spalte wird (sofern möglich) eine Integritätsbedingung für die Tabelle erstellt, die sicherstellen soll, dass alle Geometrien das angegebene räumliche Bezugssystem verwenden.

Berechtigung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte besitzen:

- Berechtigung DBADM für die Datenbank mit der Tabelle, zu der die räumliche Spalte gehört, die registriert werden soll
- Zugriffsrecht CONTROL für diese Tabelle
- Zugriffsrecht ALTER für diese Tabelle

Syntax



Parameterbeschreibungen

tabellenschema
Dieser Parameter gibt den Namen des Schemas an, zu dem die Tabelle bzw. Sicht, die im Parameter *tabellenname* angegeben ist, gehört. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Falls

Sie für diesen Parameter den Wert NULL angeben, wird als Schemaname für die Tabelle bzw. Sicht der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

Der Wert für *tabellenschema* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

tabellenname
Dieser Parameter gibt den Namen ohne Qualifikationsmerkmal der Tabelle bzw. der Sicht an, zu der die registrierte Spalte gehört. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

spaltenname
Dieser Parameter gibt den Namen der zu registrierenden Spalte an. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *spaltenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

name_des_räumlichen_bezugssystems
Dieser Parameter gibt das räumliche Bezugssystem an, das für diese räumliche Spalte verwendet werden soll. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *name_des_räumlichen_bezugssystems* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

Bereiche_berechnen
Gibt an, ob die geografischen Bereiche einer angegebenen Spalte berechnet und in der Katalogsicht DB2GSE.ST_GEOMETRY_COLUMNS zur Verfügung gestellt werden sollen. Folgende Parameterwerte sind möglich:

- Ein Wert größer als 0 (null), um die geografischen Bereiche zu berechnen.
- Null, 0 oder ein negativer Wert, um diese Berechnung zu verhindern.

Wenn Sie für diesen Parameter keinen Wert angeben, wird der Standardwert 0 (null) angenommen. Es wird keine Bereichsberechnung durchgeführt.

Dieser Parameter hat den Datentyp INTEGER.

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausge-

führt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_REGISTER_SPATIAL_COLUMN über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird die räumliche Spalte namens LOCATION in der Tabelle CUSTOMERS mit einem DB2-Befehl CALL registriert. In diesem Befehl CALL wird für den Parameter *name_des_räumlichen_bezugssystems* der Wert USA_SRS_1 verwendet:

```
call DB2GSE.ST_REGISTER_SPATIAL_COLUMN(NULL,'CUSTOMERS','LOCATION',
          'USA_SRS_1',?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_REMOVE_GEOCODING_SETUP

Mit dieser gespeicherten Prozedur können Sie alle Informationen zur Konfiguration der Geocodierung für die geocodierte Spalte entfernen.

Diese gespeicherte Prozedur entfernt Informationen, die der angegebenen geocodierten Spalte zugeordnet sind, aus den Katalogsichten DB2GSE.ST_GEOCODING und DB2GSE.ST_GEOCODING_PARAMETERS.

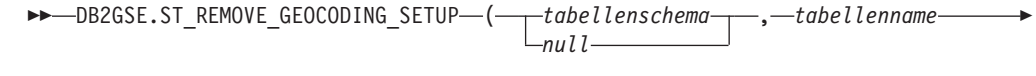
Einschränkung:
Wenn die automatische Geocodierung für die geocodierte Spalte aktiviert ist, kann die Geocodierungskonfiguration nicht entfernt werden.

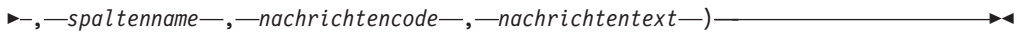
Berechtigung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte besitzen:

- Berechtigung DATAACCESS für die Datenbank mit der Tabelle, für die der angegebene Geocoder ausgeführt werden soll.
- Zugriffsrecht CONTROL für diese Tabelle
- Zugriffsrecht UPDATE für diese Tabelle

Syntax





Parameterbeschreibungen

tabellenschema
Dieser Parameter gibt den Namen des Schemas an, zu dem die Tabelle bzw. Sicht, die im Parameter *tabellenname* angegeben ist, gehört. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Falls Sie für diesen Parameter den Wert NULL angeben, wird als Schemaname für die Tabelle bzw. Sicht der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

Der Wert für *tabellenschema* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

tabellenname
Dieser Parameter gibt den Namen ohne Qualifikationsmerkmal der Tabelle bzw. Sicht an, die die Spalte enthält, in der die geocodierten Daten eingefügt oder aktualisiert werden sollen. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

spaltenname
Dieser Parameter gibt den Namen der Spalte an, in der die geocodierten Daten eingefügt oder aktualisiert werden sollen. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *spaltenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_REMOVE_GEOCODING_SETUP über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird die Konfiguration der Geocodierung für die Tabelle CUSTOMER und die Spalte namens LOCATION mit einem DB2-Befehl CALL entfernt:

```
call DB2GSE.ST_REMOVE_GEOCODING_SETUP(NULL, 'CUSTOMERS', 'LOCATION',?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_RUN_GEOCODING

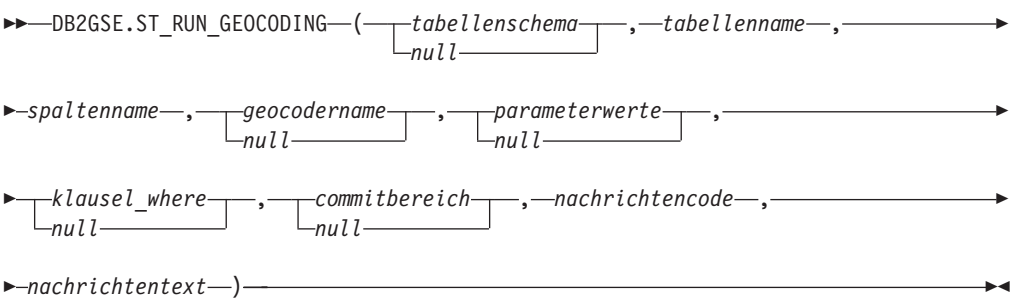
Mit dieser gespeicherten Prozedur können Sie einen Geocoder für eine geocodierte Spalte im Stapelmodus ausführen.

Berechtigung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte besitzen:

- Berechtigung DATAACCESS für die Datenbank mit der Tabelle, für die der angegebene Geocoder ausgeführt werden soll.
- Zugriffsrecht CONTROL für diese Tabelle
- Zugriffsrecht UPDATE für diese Tabelle

Syntax



Parameterbeschreibungen

tabellenschema

Dieser Parameter gibt den Namen des Schemas an, zu dem die Tabelle bzw. Sicht, die im Parameter *tabellenname* angegeben ist, gehört. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Falls Sie für diesen Parameter den Wert NULL angeben, wird als Schemaname für die Tabelle bzw. Sicht der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

Der Wert für *tabellenschema* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

tabellenname
Dieser Parameter gibt den Namen ohne Qualifikationsmerkmal der Tabelle bzw. Sicht an, die die Spalte enthält, in der die geocodierten Daten eingefügt oder aktualisiert werden sollen. Wenn der Name einer Sicht angegeben wird, muss es sich um eine aktualisierbare Sicht handeln. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

spaltenname
Dieser Parameter gibt den Namen der Spalte an, in der die geocodierten Daten eingefügt oder aktualisiert werden sollen. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *spaltenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

geocodername
Dieser Parameter gibt den Namen des Geocoders an, der die Geocodierung vornehmen soll. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird die Geocodierung durch den Geocoder ausgeführt, der beim Definieren der Geocodierung angegeben wurde.

Der Wert für *geocodername* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

parameterwerte
Dieser Parameter gibt eine Liste der Geocodierungsparameterwerte für die Geocoderfunktion an. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn der Wert für den gesamten Parameter *parameterwerte* NULL ist, werden entweder die Parameterwerte verwendet, die beim Definieren des Geocoders angegeben wurden, oder aber die Standardparameterwerte für den Geocoder, falls der Geocoder nicht definiert wurde.

Bei der Angabe von Parameterwerten verwenden Sie die Reihenfolge, in der die Parameter durch die Funktion definiert sind, und grenzen Sie die einzelnen Werte durch ein Komma voneinander ab. Beispiel:

```
wert_für_parameter1,wert_für_parameter2,...
```

Jeder Parameterwert kann ein Spaltenname, eine Zeichenfolge, ein numerischer Wert oder der Wert NULL sein.

Jeder Parameterwert ist ein SQL-Ausdruck, Beachten Sie die folgenden Richtlinien:

- Falls ein Parameterwert der Name einer Geocodierungsspalte ist, muss sich die Spalte in der gleichen Tabelle bzw. Sicht wie die geocodierte Spalte befinden.

- Handelt es sich bei dem Parameterwert um eine Zeichenfolge, muss sie in einfache Anführungszeichen gesetzt werden.
- Ein Zahlenwert darf nicht in einfache Anführungszeichen gesetzt werden.
- Wenn der Parameter den Wert NULL hat, muss er in den korrekten Typ umgesetzt werden. Statt lediglich NULL anzugeben, geben Sie beispielsweise Folgendes an:

  ```
  CAST(NULL AS INTEGER)
  ```

Falls ein Parameterwert nicht angegeben wird, Sie also zwei aufeinander folgende Kommas (`...,,...`) angeben, muss dieser Parameter entweder beim Definieren der Geocodierung oder bei der Ausführung der Geocodierung im Stapelmodus mit dem Parameter *parameterwerte* der jeweiligen gespeicherten Prozedur angegeben werden.

Dieser Parameter hat den Datentyp VARCHAR(32K).

klausel_where
Dieser Parameter gibt den Hauptteil der Klausel WHERE an, die eine Einschränkung für die Gruppe der zu geocodierenden Datensätze definiert. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein.

Falls der Parameter *klausel_where* den Wert NULL hat, ist das resultierende Verhalten davon abhängig, ob die Geocodierung für die (im Parameter *spaltenname* angegebene) Spalte definiert wurde, bevor die gespeicherte Prozedur ausgeführt wird. Hat der Parameter *klausel_where* den Wert NULL und trifft eine der folgenden Bedingungen zu, gilt Folgendes:

- Wurde beim Definieren der Geocodierung ein Wert angegeben, wird dieser Wert für den Parameter *klausel_where* verwendet.
- Wurde entweder die Geocodierung nicht definiert oder beim Definieren der Geocodierung kein Wert angegeben, wird keine Klausel WHERE verwendet.

Die angegebene Klausel kann sich auf eine beliebige Spalte in der Tabelle oder Sicht beziehen, für die der Geocoder ausgeführt werden soll. Das Schlüsselwort WHERE darf nicht angegeben werden.

Dieser Parameter hat den Datentyp VARCHAR(32K).

commitbereich
Dieser Parameter gibt an, dass ein Commit durchgeführt werden soll, nachdem jeweils *n* Datensätze geocodiert wurden. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein.

Falls der Parameter *commitbereich* den Wert NULL hat, ist das resultierende Verhalten davon abhängig, ob die Geocodierung für die (im Parameter *spaltenname* angegebene) Spalte definiert wurde, bevor die gespeicherte Prozedur ausgeführt wird. Hat der Parameter *commitbereich* den Wert NULL und trifft eine der folgenden Bedingungen zu, gilt Folgendes:

- Wurde beim Definieren der Geocodierung für die Spalte ein Wert angegeben, wird dieser Wert für den Parameter *commitbereich* verwendet.
- Wenn entweder die Geocodierung nicht definiert wurde oder hierbei kein Wert angegeben wurde, wird der Standardwert 0 (null) verwendet, und am Ende der Operation wird ein COMMIT durchgeführt.

Dieser Parameter hat den Datentyp INTEGER.

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_RUN-_GEOCODING über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird die Spalte LOCATION in der Tabelle namens CUSTOMER mit einem DB2-Befehl CALL geocodiert. In diesem Befehl CALL wird für den Parameter *geocodername* der Wert SAMPLEGC und für den Parameter *commitbereich* der Wert 10 verwendet. Nachdem jeweils 10 Datensätze geocodiert wurden, wird ein Commit durchgeführt.

```
call DB2GSE.ST_RUN_GEOCODING(NULL, 'CUSTOMERS', 'LOCATION',
     'SAMPLEGC',NULL,NULL,10,?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_SETUP_GEOCODING

Mit dieser gespeicherten Sicht können Sie einer Spalte, die geocodiert werden soll, einen Geocoder zuordnen und die entsprechenden Geocodierungsparameter definieren.

Informationen zu der Konfiguration können Sie in den Katalogsichten DB2GSE.ST_GEOCODING und DB2GSE.ST_GEOCODING_PARAMETERS abfragen.

Diese gespeicherte Prozedur ruft keine Geocodierungsoperation auf. Sie ist vielmehr eine schnelle Methode, um Parametereinstellungen für die Spalte anzugeben, die geocodiert werden soll. Mit diesen Einstellungen kann ein nachfolgender Aufruf der Geocodierung im Stapelbetrieb oder der automatischen Geocodierung über eine viel einfachere Schnittstelle erfolgen. Die Parametereinstellungen, die in diesem Konfigurationsschritt angegeben werden, setzen alle Standardparameterwerte für den Geocoder außer Kraft, die ggfs. bei der Registrierung des Geocoders angegeben wurden. Sie können diese Parametereinstellungen auch dadurch außer Kraft setzen, dass Sie die Prozedur ST_RUN_GEOCODING im Stapelmodus ausführen.

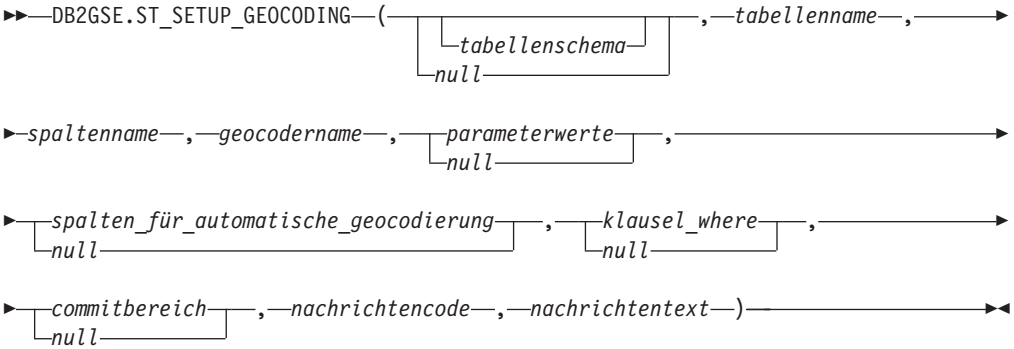
Dieser Schritt muss für die automatische Geocodierung unbedingt ausgeführt werden. Die automatische Geocodierung kann nicht aktiviert werden, ohne dass zuvor die Geocodierungsparameter definiert wurden. Für die Geocodierung im Stapelbetrieb ist dieser Schritt nicht unbedingt erforderlich. Sie können die Geocodierung im Stapelmodus unabhängig davon ausführen, ob zuvor der Konfigurationsschritt ausgeführt wurde oder nicht. Wird der Konfigurationsschritt vor der Geocodierung im Stapelbetrieb jedoch ausgeführt, werden die bei der Konfiguration angegebenen Parameterwerte verwendet, wenn sie zur Laufzeit nicht angegeben werden.

Berechtigung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte besitzen:

- Berechtigung DATAACCESS für die Datenbank mit der Tabelle, für die der angegebene Geocoder ausgeführt werden soll.
- Zugriffsrecht CONTROL für diese Tabelle
- Zugriffsrecht UPDATE für diese Tabelle

Syntax



Parameterbeschreibungen

tabellenschema
: Dieser Parameter gibt den Namen des Schemas an, zu dem die Tabelle bzw. Sicht, die im Parameter *tabellenname* angegeben ist, gehört. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Falls Sie für diesen Parameter den Wert NULL angeben, wird als Schemaname für die Tabelle bzw. Sicht der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

  Der Wert für *tabellenschema* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

  Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

tabellenname
: Dieser Parameter gibt den Namen ohne Qualifikationsmerkmal der Tabelle bzw. Sicht an, die die Spalte enthält, in der die geocodierten Daten eingefügt oder aktualisiert werden sollen. Wenn der Name einer Sicht angegeben wird, muss es sich um eine aktualisierbare Sicht handeln. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

  Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

spaltenname
Dieser Parameter gibt den Namen der Spalte an, in der die geocodierten Daten eingefügt oder aktualisiert werden sollen. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *spaltenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

geocodername
Dieser Parameter gibt den Namen des Geocoders an, der die Geocodierung vornehmen soll. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *geocodername* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

parameterwerte
Dieser Parameter gibt eine Liste der Geocodierungsparameterwerte für die Geocoderfunktion an. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn der Wert für den gesamten Parameter *parameterwerte* NULL lautet, werden die Standardparameterwerte verwendet, die bei der Registrierung des Geocoders angegeben wurden.

Bei der Angabe von Parameterwerten verwenden Sie die Reihenfolge, in der die Parameter durch die Funktion definiert sind, und grenzen Sie die einzelnen Werte durch ein Komma voneinander ab. Beispiel:

```
wert_für_parameter1,wert_für_parameter2,...
```

Jeder Parameterwert ist ein SQL-Ausdruck und kann ein Spaltenname, eine Zeichenfolge, ein numerischer Wert oder der Wert NULL sein. Beachten Sie die folgenden Richtlinien:

- Falls ein Parameterwert der Name einer Geocodierungsspalte ist, muss sich die Spalte in der gleichen Tabelle bzw. Sicht wie die geocodierte Spalte befinden.
- Handelt es sich bei dem Parameterwert um eine Zeichenfolge, muss sie in einfache Anführungszeichen gesetzt werden.
- Ein Zahlenwert darf nicht in einfache Anführungszeichen gesetzt werden.
- Wenn der Parameter den Wert NULL hat, muss er in den korrekten Typ umgesetzt werden. Statt lediglich NULL anzugeben, geben Sie beispielsweise Folgendes an:

  ```
  CAST(NULL AS INTEGER)
  ```

Falls ein Parameterwert nicht angegeben wird, Sie also zwei aufeinander folgende Kommas (`...,,...`) angeben, muss dieser Parameter entweder beim Definieren der Geocodierung oder bei der Ausführung der Geocodierung im Stapelmodus mit dem Parameter *parameterwerte* der jeweiligen gespeicherten Prozedur angegeben werden.

Dieser Parameter hat den Datentyp VARCHAR(32K).

spalten_für_automatische_geocodierung
Dieser Parameter gibt eine Liste der Spaltennamen an, für die der Trigger er-

stellt werden soll. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben und die automatische Geocodierung aktiviert ist, bewirkt die Aktualisierung einer beliebigen Spalte in der Tabelle, dass der Trigger aktiviert wird.

Wenn Sie für den Parameter *spalten_für_automatische_geocodierung* einen Wert definieren, können Sie die Spaltennamen in einer beliebigen Reihenfolge angeben, wobei die einzelnen Namen jeweils durch ein Komma voneinander abzugrenzen sind. Der Spaltenname muss in derselben Tabelle vorhanden sein, in der sich auch die geocodierte Spalte befindet.

Diese Parametereinstellung ist nur bei einer nachfolgenden automatischen Geocodierung wirksam.

Dieser Parameter hat den Datentyp VARCHAR(32K).

klausel_where
Dieser Parameter gibt den Hauptteil der Klausel WHERE an, die eine Einschränkung für die Gruppe der zu geocodierenden Datensätze definiert. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, werden keine Einschränkungen in der Klausel WHERE definiert.

Die Klausel kann sich auf eine beliebige Spalte in der Tabelle oder Sicht beziehen, für die der Geocoder ausgeführt werden soll. Das Schlüsselwort WHERE darf nicht angegeben werden.

Diese Parametereinstellung ist nur bei einer nachfolgenden Geocodierung im Stapelmodus wirksam.

Dieser Parameter hat den Datentyp VARCHAR(32K).

commitbereich
Dieser Parameter gibt an, dass ein Commit durchgeführt werden soll, nachdem jeweils *n* Datensätze geocodiert wurden. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Wenn Sie für diesen Parameter den Wert NULL angeben, wird ein Commit durchgeführt, nachdem alle Datensätze geocodiert wurden.

Diese Parametereinstellung ist nur bei einer nachfolgenden Geocodierung im Stapelmodus wirksam.

Dieser Parameter hat den Datentyp INTEGER.

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der

Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_SETUP-_GEOCODING über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird über einen DB2-Befehl CALL ein Geocodierungsprozess für die geocodierte Spalte namens LOCATION in der Tabelle CUSTOMER definiert. Dieser Befehl CALL verwendet für den Parameter *geocodername* den Wert SAMPLEGC:

```
call DB2GSE.ST_SETUP_GEOCODING(NULL, 'CUSTOMERS', 'LOCATION',
    'SAMPLEGC','ADDRESS,CITY,STATE,ZIP,1,100,80,,,,"$HOME/sqllib/
     gse/refdata/ky.edg","$HOME/sqllib/samples/extenders/spatial/EDGESample.loc"',
     'ADDRESS,CITY,STATE,ZIP',NULL,10,?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_UNREGISTER_GEOCODER

Mit dieser gespeicherten Prozedur können Sie die Registrierung eines Geocoders aufheben.

Einschränkung:
Die Registrierung eines Geocoders kann nicht zurückgenommen werden, wenn er in der Geocodierungskonfiguration einer Spalte angegeben ist.

In den Katalogsichten DB2GSE.ST_GEOCODING und DB2GSE.ST_GEOCODING_PARAMETERS können Sie ermitteln, ob ein Geocoder in der Geocodierungskonfiguration für eine Spalte angegeben ist. Informationen zum Geocoder, dessen Registrierung zurückgenommen werden soll, finden Sie in der Katalogsicht DB2GSE.ST_GEOCODERS.

Berechtigung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss die Berechtigung DBADM für die Datenbank besitzen, in der sich der Geocoder befindet, dessen Registrierung zurückgenommen werden soll.

Syntax

Parameterbeschreibungen

geocodername
Dieser Parameter kennzeichnet den Geocoder auf eindeutige Weise. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *geocodername* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_UNREGISTER_GEOCODER über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird die Registrierung des Geocoders SAMPLEGC mit einem DB2-Befehl CALL zurückgenommen:

```
call DB2GSE.ST_UNREGISTER_GEOCODER('SAMPLEGC',?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Prozedur ST_UNREGISTER_SPATIAL_COLUMN

Mit dieser gespeicherten Prozedur können Sie die Registrierung einer räumlichen Spalte entfernen.

Die gespeicherte Prozedur entfernt die Registrierung auf folgende Weise:

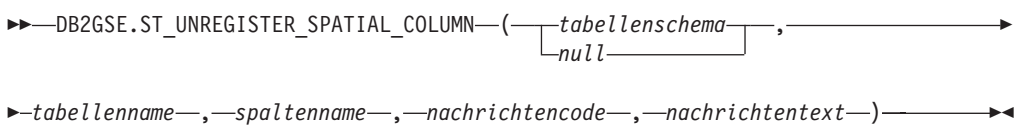
- Die Zuordnung des räumlichen Bezugssystems zur räumlichen Spalte wird entfernt. In der Katalogsicht ST_GEOMETRY_COLUMNS ist die räumliche Spalte zwar weiterhin angegeben, aber der Spalte ist kein räumliches Bezugssystem mehr zugeordnet.
- Bei einer Basistabelle wird die Integritätsbedingung gelöscht, die DB2 Spatial Extender für diese Tabelle eingerichtet hat, um sicherzustellen, dass alle Geometriewerte in dieser räumlichen Spalte dasselbe räumliche Bezugssystem verwenden.

Berechtigung

Die Benutzer-ID, unter der diese gespeicherte Prozedur aufgerufen wird, muss eine der folgenden Berechtigungen bzw. eines der folgenden Zugriffsrechte besitzen:

- Berechtigung DBADM
- Zugriffsrecht CONTROL für diese Tabelle
- Zugriffsrecht ALTER für diese Tabelle

Syntax



Parameterbeschreibungen

tabellenschema
Dieser Parameter gibt den Namen des Schemas an, zu dem die Tabelle, die im Parameter *tabellenname* angegeben ist, gehört. Sie müssen zwar einen Wert für diesen Parameter angeben, aber der Wert kann NULL sein. Falls Sie für diesen Parameter den Wert NULL angeben, wird als Schemaname für die Tabelle bzw. Sicht der Wert verwendet, der im Sonderregister CURRENT SCHEMA angegeben ist.

Der Wert für *tabellenschema* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

tabellenname
Dieser Parameter gibt den Namen ohne Qualifikationsmerkmal der Tabelle an, in der die Spalte enthalten ist, die im Parameter *spaltenname* angegeben ist. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *tabellenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

spaltenname
Dieser Parameter gibt den Namen der räumlichen Spalte an, deren Registrierung zurückgenommen werden soll. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Der Wert für *spaltenname* wird in Großbuchstaben umgewandelt, wenn Sie ihn nicht in doppelte Anführungszeichen setzen.

Dieser Parameter hat den Datentyp VARCHAR(128). Wenn er in doppelte Anführungszeichen gesetzt wird, lautet der Datentyp VARCHAR(130).

Ausgabeparameter

nachrichtencode
Dieser Parameter gibt den Nachrichtencode an, der von der gespeicherten Prozedur zurückgegeben wird. Der Wert dieses Ausgabeparameters gibt die Fehler-, Erfolgs- oder Warnungsbedingung an, die während der Verarbeitung der

Prozedur festgestellt wurde. Falls der Wert dieses Parameters eine Erfolgs- oder Warnungsbedingung ist, hat die Prozedur ihre Task vollständig ausgeführt. Handelt es sich bei dem Parameterwert um eine Fehlerbedingung, wurden keine Änderungen an der Datenbank vorgenommen.

Dieser Ausgabeparameter hat den Datentyp INTEGER.

nachrichtentext
Dieser Parameter gibt den eigentlichen Nachrichtentext an, der dem von der gespeicherten Prozedur zurückgegebenen Nachrichtencode zugeordnet ist. Der Nachrichtentext kann zusätzliche Informationen zur Erfolgs-, Warnungs- oder Fehlerbedingung enthalten, beispielsweise die Angabe, wo ein Fehler aufgetreten ist.

Dieser Ausgabeparameter hat den Datentyp VARCHAR(1024).

Beispiel

Das folgende Beispiel veranschaulicht, wie die gespeicherte Prozedur ST_UNREGISTER_SPATIAL_COLUMN über den DB2-Befehlszeilenprozessor aufgerufen werden kann. Im Beispiel wird die Registrierung der räumlichen Spalte namens LOCATION in der Tabelle CUSTOMERS mit einem DB2-Befehl CALL zurückgenommen:

```
call DB2GSE.ST_UNREGISTER_SPATIAL_COLUMN(NULL,'CUSTOMERS','LOCATION',?,?)
```

Die beiden Fragezeichen am Ende dieses Befehls CALL stellen die beiden Ausgabeparameter *nachrichtencode* und *nachrichtentext* dar. Die Werte für diese Ausgabeparameter werden im Anschluss an die Ausführung der gespeicherten Prozedur angezeigt.

Kapitel 18. Räumliche Funktionen

Mit räumlichen Funktionen können Sie Ihre Anwendungen programmieren. Erfahren Sie mehr über die Faktoren, die für alle oder die Mehrzahl der räumlichen Funktionen gelten, und über die Verwendung von Funktionen nach Kategorie.

Überlegungen und zugehörige Datentypen zu räumlichen Funktionen

Dieser Abschnitt bietet Informationen, die Sie für die Codierung räumlicher Funktionen benötigen.

Diese Informationen umfassen:

- Faktoren, die bedacht werden sollten: die Anforderung das Schema anzugeben, zu dem die räumlichen Funktionen gehören, und die Tatsache, dass einige Funktionen als Methoden aufgerufen werden können.
- Hinweise zur Vorgehensweise in Situationen, in denen eine räumliche Funktion den Typ von Geometrie nicht verarbeiten kann, der von einer anderen räumlichen Funktion zurückgegeben wurde.
- Eine Tabelle, die anzeigt, welche Funktionen Werte von welchem räumlichen Datentyp als Eingabe verwenden.

Wenn Sie räumliche Funktionen verwenden, sollten Sie stets folgende Faktoren bedenken:

- Vor dem Aufruf einer räumlichen Funktion muss deren Name durch den Namen des Schemas qualifiziert werden, zu dem die räumliche Funktion gehört: DB2GSE. Eine Möglichkeit dazu besteht darin, das Schema explizit in der SQL-Anweisung anzugeben, die auf diese Funktion verweist. Zum Beispiel:

  ```
  SELECT db2gse.ST_Relate (g1, g2, 'T*F**FFF2') EQUALS FROM relate_test
  ```

  Alternativ können Sie DB2GSE dem Sonderregister CURRENT FUNCTION PATH hinzufügen, um nicht bei jedem Aufruf einer Funktion das Schema angeben zu müssen. Um die aktuellen Einstellungen für dieses Sonderregister zu erhalten, geben Sie folgenden SQL-Befehl ein:

  ```
  VALUES CURRENT FUNCTION PATH
  ```

  Um das Sonderregister CURRENT FUNCTION PATH mit DB2GSE zu aktualisieren, geben Sie folgenden SQL-Befehl aus:

  ```
  set CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
  ```

- Einige räumliche Funktionen können als Methode aufgerufen werden. Im folgenden Beispiel wird ST_Area zuerst als Funktion und anschließend als Methode aufgerufen. In beiden Fällen wird ST_Area für die Verarbeitung eines Polygons codiert, das die ID 10 aufweist und in der Spalte SALES_ZONE der Tabelle STORES gespeichert ist. Nach dem Aufruf gibt ST_Area den Bereich des realen Objekts Sales Zone Nr. 10 zurück, das das Polygon darstellt.

  ST_Area wird als Funktion aufgerufen:

  ```
  SELECT ST_Area(sales_zone)
  FROM   stores
    WHERE id = 10
  ```

  ST_Area wird als Methode aufgerufen:

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

Die Funktionen ST_BuildMBRAggr und ST_BuildUnionAggr werden in "MBR Aggregate" und "Union Aggregate" beschrieben.

Werte vom Typ ST_Geometry als Werte eines untergeordneten Typs behandeln

Wenn eine räumliche Funktion eine Geometrie zurückgibt, deren statischer Typ ein übergeordneter Typ ist, und wenn die Geometrie an eine Funktion übergeben wird, die nur Geometrien eines Typs verwendet, der diesem übergeordneten Typ untergeordnet ist, wird bei der Kompilierung eine Ausnahmebedingung erzeugt.

Der statische Typ des Ausgabeparameters der Funktion ST_Union lautet zum Beispiel ST_Geometry, der übergeordnete Typ aller räumlichen Datentypen. Der statische Eingabeparameter für die Funktion ST_PointOnSurface kann entweder ST_Polygon oder ST_MultiPolygon sein. Beide sind untergeordnete Typen von ST_Geometry. Wenn DB2 Spatial Extender versucht, Geometrien zu übergeben, die von ST_Union an ST_PointOnSurface zurückgegeben wurden, erzeugt DB2 Spatial Extender beim Kompilieren die folgende Ausnahmebedingung:

```
SQL00440N No function by the name "ST_POINTONSURFACE"
having compatible arguments was found in the function
path.    SQLSTATE=42884
```

Diese Nachricht weist darauf hin, dass DB2 Spatial Extender keine Funktion finden konnte, die ST_PointOnSurface genannt wird und einen Eingabeparameter des Typs ST_Geometry verwendet.

Verwenden Sie den Operator TREAT, damit Geometrien eines übergeordneten Typs an Funktionen übergeben werden können, die nur untergeordnete Typen des übergeordneten Typs akzeptieren. Wie bereits zuvor erwähnt, gibt ST_Union Geometrien eines statischen Typs von ST_Geometry zurück. Diese Funktion kann ferner Geometrien eines dynamischen untergeordneten Typs von ST_Geometry zurückgeben. Nehmen wir zum Beispiel an, dass diese Funktion eine Geometrie des dynamischen Typs von ST_MultiPolygon zurückgibt. In diesem Fall erfordert der Operator TREAT, dass diese Geometrie mit dem statischen Typ ST_MultiPolygon verwendet wird. Dieser stimmt mit dem Datentyp des Eingabeparameters ST_PointOnSurface überein. Wenn ST_Union keinen Wert vom Typ ST_MultiPolygon zurückgibt, erzeugt DB2 Spatial Extender eine Laufzeitausnahmebedingung.

Wenn eine Funktion eine Geometrie eines übergeordneten Typs zurückgibt, kann der Operator TREAT in der Regel DB2 Spatial Extender veranlassen, diese Geometrie als untergeordneten Typ dieses übergeordneten Typs zu behandeln. Bedenken Sie jedoch, dass diese Operation nur dann erfolgreich sein kann, wenn der untergeordnete Typ einem statischen untergeordneten Typ untergeordnet ist, der als Eingabeparameter der Funktion definiert ist, an die die Geometrie übergeben wird, oder mit diesem übereinstimmt. Wenn diese Bedingung nicht zutrifft, erzeugt DB2 Spatial Extender eine Ausnahmebedingung zur Bearbeitungszeit.

Ein weiteres Beispiel: Nehmen wir an, dass Sie die winkeltreu projizierten Punkte für einen angegebenen Punkt auf der Grenze eines Polygons ermitteln möchten, das keine Löcher aufweist. Sie verwenden die Funktion ST_Boundary, um die Grenze des Polygons abzuleiten. Der statische Ausgabeparameter von ST_Boundary ist ST_Geometry. ST_PerpPoints verwendet jedoch nur Geometrien vom Typ ST_Curve. Da alle Polygone über eine Linienfolge (die ebenfalls eine Kurve ist) als Grenze verfügen, und da der Datentyp von Linienfolgen (ST_LineString) dem Typ

ST_Curve untergeordnet ist, kann mit der folgenden Operation ein Polygon vom Typ ST_Geometry, das von ST_Boundary zurückgegeben wurde, an ST_PerpPoints übergeben werden:

```
SELECT ST_AsText(ST_PerpPoints(TREAT(ST_Boundary(polygon) as ST_Curve)),
             ST_Point(30.5, 65.3, 1)))
FROM   polygon_table
```

Sie können ST_Boundary und ST_PerpPoints auch als Methoden aufrufen. Geben Sie dazu folgenden Code an:

```
SELECT TREAT(ST_Boundary(polygon) as ST_Curve)..
             ST_PerpPoints(St_Point(30.5, 65.3, ))..ST_AsText()
FROM   polygon_table
```

Räumliche Funktionen nach Eingabetyp

Liste räumlicher Funktionen nach dem von ihnen akzeptierten Eingabetyp sortiert.

Wichtig: Wie bereits an anderer Stelle erwähnt, bilden die räumlichen Datentypen eine Hierarchie mit ST_Geometry als Root. Wenn die Dokumentation für DB2 Spatial Extender darauf hinweist, dass ein Wert eines übergeordneten Typs in dieser Hierarchie als Eingabe für eine Funktion verwendet werden kann, kann alternativ ebenfalls ein Wert eines beliebigen untergeordneten Typs dieses übergeordneten Typs als Eingabe für die Funktion verwendet werden.

Die ersten Einträge in Tabelle 24 weisen zum Beispiel darauf hin, dass ST_Area und eine Anzahl anderer Funktionen Werte vom Typ ST_Geometry als Eingabe verwenden kann. Daher können Werte eines beliebigen untergeordneten Typs von ST_Geometry: ST_Point, ST_Curve, ST_LineString und so weiter, ebenfalls als Eingabe für diese Funktionen verwendet werden.

Tabelle 24. Liste räumlicher Funktionen nach Eingabetyp sortiert

Datentyp des Eingabeparameters	Funktion
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_DifferenceST_Dimension
	ST_DisjointST_Distance
	ST_Envelope
	ST_EnvIntersects
	ST_Equals
	ST_FindMeasure or ST_LocateAlong
	ST_Generalize
	ST_GeometryType

Tabelle 24. Liste räumlicher Funktionen nach Eingabetyp sortiert (Forts.)

Datentyp des Eingabeparameters	**Funktion**
ST_Geometry, Fortsetzung	
	ST_IntersectionST_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 oder ST_LocateBetween
	ST_MinM
	ST_MinX
	ST_MinY
	ST_MinZ
	ST_NumPoints
	ST_Overlaps
	ST_RelateST_SRID oder ST_SrsId
	ST_SrsName
	ST_SymDifference
	ST_ToGeomColl
	ST_ToLineString
	ST_ToMultiLine
	ST_ToMultiPoint
	ST_ToMultiPolygon
	ST_ToPoint
	ST_ToPolygon
	ST_Touches
	ST_Transform
	ST_Union
	ST_Within
ST_Point	
	ST_M
	ST_X
	ST_Y
	ST_Z
ST_Curve	
	ST_AppendPoint
	ST_ChangePoint
	ST_EndPoint
	ST_IsClosed
	ST_IsRing
	ST_Length
	ST_MidPoint
	ST_PerpPoints
	ST_RemovePoint
	ST_StartPoint
ST_LineString	
	ST_PointN
	ST_Polygon
ST_Surface	
	ST_Perimeter
	ST_PointOnSurface

Tabelle 24. Liste räumlicher Funktionen nach Eingabetyp sortiert (Forts.)

Datentyp des Eingabeparameters	Funktion
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

Kategorien und Verwendungsmöglichkeiten für räumliche Funktionen

Der vorliegende Abschnitt enthält Einführungen in alle räumlichen Funktionen, die nach Kategorien gegliedert sind.

DB2 Spatial Extender stellt die folgenden Funktionen zur Verfügung:

- Funktionen zum Umwandeln von Geometrien in verschiedene bzw. aus verschiedenen Datenaustauschformate(n). Diese Funktionen werden als Konstruktorfunktionen bezeichnet.
- Funktionen zum Vergleichen von Geometrien in Bezug auf Begrenzungen, Schnittpunkte und andere Informationen. Diese Funktionen werden als Vergleichsfunktionen bezeichnet.
- Funktionen für die Rückgabe von Informationen zu den Eigenschaften von Geometrien. Hierzu gehören z. B. die definierten Koordinaten und Bemaßungen der Geometrien, Angaben zu den zwischen den Geometrien geltenden Beziehungen sowie Informationen zu deren Begrenzungen und andere Informationen.
- Funktionen zum Generieren neuer Geometrien aus bereits bestehenden Geometrien.
- Funktionen zur Messung der kürzesten Distanz zwischen den Punkten von Geometrien.
- Funktionen zum Bereitstellen von Informationen zu Indexparametern.
- Funktionen zum Bereitstellen von Projektionen und Umwandlungen zwischen verschiedenen Koordinatensystemen.

Konstruktorfunktionen für die Umwandlung in und aus Datenaustauschformate(n)

DB2 Spatial Extender bietet räumliche Funktionen, mit denen Geometrien in Datenaustauschformate bzw. aus diesen Datenaustauschformaten umgewandelt werden können.

Folgende Datenaustauschformate werden unterstützt:

- WKT-Darstellung (WKT = Well-Known Text)
- WKB-Darstellung (WKB = Well-Known Binary)
- ESRI-Formdarstellung
- GML-Darstellung (GML = Geography Markup Language)

Die Funktionen für die Erstellung von Geometrien aus diesen Formaten sind unter der Bezeichnung *Konstruktorfunktionen* bekannt. Konstrukturfunktionen heißen genauso wie der Geometriedatentyp der Spalte, in die die Daten eingefügt werden. Diese Funktionen funktionieren für das jeweilige Austauschformat der Eingabedaten in konsistenter Weise. Im vorliegenden Abschnitt wird Folgendes beschrieben:

- SQL für den Aufruf von Funktionen, die mit Datenaustauschformaten arbeiten, und den Typ der Geometrie, die durch diese Funktionen zurückgegeben wird
- SQL für den Aufruf einer Funktion, die Punkte aus X- und Y-Koordinaten erstellt, und den Typ der Geometrie, der durch diese Funktion zurückgegeben wird
- Beispiele für Code und Ergebnismengen

Funktionen, die Datenaustauschformate in Geometrien umwandeln

Das Umwandeln von Geometriedarstellungen aus Datenaustauschformaten in Geometriewerte erlaubt das Austauschen von Darstellungen als Geometriewerte.

Funktionen für die Umwandlung von Geometrien in die WKT-Darstellung

Mithilfe der Umwandlung von Geometrien in die WKT-Darstellung können Geometrien im ASCII-Textformat ausgetauscht werden. Die WKT-Darstellungen sind Werte des Typs CLOB, die ASCII-Zeichenfolgen darstellen.

Die Funktion **ST_AsText** wandelt einen Geometriewert, der in einer Tabelle gespeichert ist, in eine WKT-Zeichenfolge um. Das folgende Beispiel wählt mit einer einfachen Befehlszeilenabfrage die Werte aus, die zuvor in die Tabelle SAMPLE_GEOMETRY eingefügt wurden.

```
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)
```

Im folgenden Beispiel werden die Werte, die zuvor in die Tabelle SAMPLE_GEOMETRY eingefügt wurden, mit eingebettetem SQL ausgewählt.

```
 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;
```

Alternativ können Sie beim Aufheben der Bindung von Geometrien auch die Umsetzungsgruppe ST_WellKnownText verwenden, um Geometrien implizit in die WKT-Darstellung umzuwandeln. Der folgende Mustercode veranschaulicht die Verwendung der Umsetzungsgruppe.

```
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;
```

In der Anweisung SELECT zur Umwandlung der Geometrie wird keine räumliche Funktion verwendet.

Neben den im vorliegenden Abschnitt beschriebenen Funktionen stellt DB2 Spatial Extender weitere Funktionen bereit, mit denen ebenfalls Geometrien in WKT-Darstellungen bzw. WKT-Darstellungen in Geometrien umgewandelt werden können. DB2 Spatial Extender stellt diese Funktionen zur Verfügung, um die OGC-Spezifikation „Simple Features for SQL" sowie den ISO-Standard "SQL/MM Part 3: Spatial" zu implementieren. Diese Funktionen lauten wie folgt:

- **ST_WKTToSQL**
- **ST_GeomFromText**
- **ST_GeomCollFromTxt**
- **ST_PointFromText**
- **ST_LineFromText**
- **ST_PolyFromText**
- **ST_MPointFromText**
- **ST_MLineFromText**
- **ST_MPolyFromText**

Funktionen für die Umwandlung von Geometrien in die WKB-Darstellung

Mithilfe der Umwandlung von Geometrien in die WKB-Darstellung können Geometrien im Binärformat ausgetauscht werden. Die WKB-Darstellung (WKB = Well-Known Binary Representation; bekannte Binärdarstellung) besteht aus binären Datenstrukturen, die BLOB-Werte sein müssen. Diese BLOB-Werte stellen binäre Datenstrukturen dar, die durch ein Anwendungsprogramm verwaltet werden müssen, das in einer von DB2 unterstützten Programmiersprache geschrieben wurde und für das DB2 eine Sprachenbindung aufweist.

Die Funktion **ST_AsBinary** wandelt einen Geometriewert, der in einer Tabelle gespeichert ist, in eine WKB-Darstellung um, die in eine BLOB-Variable im Programmspeicher abgerufen werden kann. Im folgenden Beispiel werden die Werte, die zuvor in die Tabelle SAMPLE_GEOMETRY eingefügt wurden, mit eingebettetem SQL ausgewählt.

```
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;
```

Alternativ können Sie beim Aufheben der Bindung von Geometrien auch die Umsetzungsgruppe ST_WellKnownBinary verwenden, um Geometrien implizit in die WKB-Darstellung umzuwandeln. Der folgende Mustercode veranschaulicht die Verwendung dieser Umsetzungsgruppe.

```
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;
```

In der Anweisung SELECT zur Umwandlung der Geometrie wird keine räumliche Funktion verwendet.

Neben den im vorliegenden Abschnitt beschriebenen Funktionen stellt DB2 Spatial Extender weitere Funktionen bereit, die ebenfalls Geometrien in WKB-Darstellungen bzw. aus diesen umwandeln. DB2 Spatial Extender stellt diese Funktionen zur Verfügung, um die OGC-Spezifikation „Simple Features for SQL" sowie den ISO-Standard "SQL/MM Part 3: Spatial" zu implementieren. Diese Funktionen lauten wie folgt:

- **ST_WKBToSQL**
- **ST_GeomFromWKB**
- **ST_GeomCollFromWKB**
- **ST_PointFromWKB**
- **ST_LineFromWKB**
- **ST_PolyFromWKB**
- **ST_MPointFromWKB**
- **ST_MLineFromWKB**
- **ST_MPolyFromWKB**

Funktionen für die Umwandlung von Geometrien in die ESRI-Formdarstellung

Mithilfe der Umwandlung von Geometrien in die ESRI-Formdarstellung können Geometrien im Binärformat ausgetauscht werden. Die ESRI-Formdarstellung besteht aus binären Datenstrukturen, die durch ein Anwendungsprogramm verwaltet werden müssen, das in einer unterstützten Sprache geschrieben ist.

Die Funktion **ST_AsShape** wandelt einen Geometriewert, der in einer Tabelle gespeichert ist, in eine ESRI-Formdarstellung um, die in eine BLOB-Variable im Programmspeicher abgerufen werden kann. Im folgenden Beispiel werden die Werte, die zuvor in die Tabelle SAMPLE_GEOMETRY eingefügt wurden, mit eingebettetem SQL ausgewählt.

```
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;
```

Alternativ können Sie beim Aufheben der Bindung von Geometrien auch die Umsetzungsgruppe ST_Shape verwenden, um Geometrien implizit in die Formdarstellung umzuwandeln. Der folgende Mustercode veranschaulicht die Verwendung der Umsetzungsgruppe.

```
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;
```

In der Anweisung SELECT zur Umwandlung der Geometrie wird keine räumliche Funktion verwendet.

Funktionen für die Umwandlung von Geometrien in die GML-Darstellung

Mithilfe der Umwandlung von Geometrien in die GML-Darstellung können Geometrien im ASCII-Textformat ausgetauscht werden. GML-Darstellungen sind ASCII-Zeichenfolgen.

Die Funktion **ST_AsGML** wandelt einen Geometriewert, der in einer Tabelle gespeichert ist, in eine GML-Textzeichenfolge um. Im folgenden Beispiel werden die Werte ausgewählt, die zuvor in die Tabelle SAMPLE_GEOMETRY eingefügt wurden. Im folgenden Beispiel wurden die Ergebnisse zur besseren Lesbarkeit erneut formatiert. Der Abstand bei Ihren Ergebnissen kann entsprechend der jeweiligen Onlineanzeige variieren.

```
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>
```

Alternativ können Sie beim Aufheben der Bindung von Geometrien auch die Umsetzungsgruppe ST_GML verwenden, um Geometrien implizit in die HTML-Darstellung umzuwandeln.

```
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>
```

In der Anweisung SELECT zur Umwandlung der Geometrie wird keine räumliche Funktion verwendet.

Funktion zur Erstellung von Geometrien aus Koordinaten

Die Funktion ST_Point erstellt Geometrien nicht nur aus Datenaustauschformaten, sondern auch aus numerischen Koordinatenwerten. Diese Funktion ist beispielsweise dann nützlich, wenn Standortdaten bereits in der Datenbank gespeichert sind.

Beispiele für den Aufruf von Konstruktorfunktionen

Prüfen Sie die Codebeispiele zum Aufruf von Konstruktorfunktionen, zur Erstellung von Tabellen für die Ausgabe von Konstruktorfunktionen und zum Abrufen der Ausgabe, damit Sie die Verwendung von Konstruktorfunktionen nachvollziehen können.

Im folgenden Beispiel wird eine Zeile in die Tabelle SAMPLE_GEOMETRY mit der ID 100 sowie ein Punktwert mit der X-Koordinate 30 und der Y-Koordinate 40 in das räumliche Bezugssystem 1 eingefügt, wobei die Koordinatendarstellung und die WKT-Darstellung verwendet wird. Anschließend wird eine weitere Zeile mit der ID 200 und ein Linienfolgenwert mit den angegebenen Koordinaten eingefügt.

```
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"
```

Wenn Ihnen bekannt ist, dass die räumliche Spalte nur Werte des Typs ST_Point enthalten darf, können Sie das folgende Beispiel verwenden, das zwei Punkte einfügt. Der Versuch, eine Linienfolge bzw. einen anderen Typ, der keinen Punkt darstellt, einzufügen, führt zu einem SQL-Fehler. Die erste Einfügeaktion erstellt aus der WKT-Darstellung eine Punktgeometrie. Die zweite Einfügeaktion erstellt aus numerischen Koordinatenwerten eine Punktgeometrie. Diese Eingabewerte könnten übrigens auch aus vorhandenen Tabellenspalten ausgewählt werden.

```
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"
```

Das folgende Beispiel verwendet eingebettetes SQL und geht davon aus, dass die Anwendung die Datenbereiche mit den entsprechenden Werten füllt.

```
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;

// * Hier wird die Anwendungslogik zum Lesen in Puffer */
// * platziert */

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));
```

Im folgenden Java™-Codebeispiel wird JDBC verwendet, um Punktgeometrien mithilfe numerischer Koordinatenwerte für X und Y einzufügen. Die Geometrien werden in der WKT-Darstellung angegeben.

```
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();
```

Vergleichsfunktionen für geografische Objekte

Bestimmte räumliche Funktionen geben Informationen über die Beziehung oder den Vergleich zwischen verschiedenen geografischen Objekten zurück. Andere räumliche Funktionen geben Informationen darüber zurück, ob zwei Definitionen von Koordinatensystemen oder zwei räumliche Bezugssysteme identisch sind.

Die zurückgegebenen Informationen sind immer das Ergebnis des Vergleichs von Geometrien, von Definitionen für Koordinatensysteme oder von räumlichen Be-

zugssystemen. Die Funktionen, die diese Informationen bereitstellen, werden als Vergleichsfunktionen bezeichnet. Die folgende Tabelle enthält Vergleichsfunktionen, die nach Zweck geordnet sind.

Tabelle 25. Vergleichsfunktionen nach Zweck

Zweck	Funktionen
Mit diesen Funktionen wird ermittelt, ob der Innenbereich einer Geometrie eine Überschneidung mit dem Innenbereich einer anderen Geometrie aufweist.	• ST_Contains • ST_Within
Mit diesen Funktionen können Informationen zu den Überschneidungen von Geometrien zurückgegeben werden.	• ST_Crosses • ST_Intersects • ST_Overlaps • ST_Touches
Mit diesen Funktionen kann ermittelt werden, ob das kleinste Rechteck, das eine der Geometrien einschließt, das kleinste Rechteck schneidet, das die andere Geometrie einschließt.	• ST_EnvIntersects • ST_MBRIntersects
Mit diesen Funktionen kann ermittelt werden, ob zwei Objekte identisch sind.	• ST_Equals • ST_EqualCoordsys • ST_EqualSRS
Mit dieser Funktion kann ermittelt werden, ob die verglichenen Geometrien die Bedingungen der Zeichenfolge der DE-9IM-Mustermatrix erfüllen.	• ST_Relate
Mit dieser Funktion kann überprüft werden, ob eine Überschneidung zwischen zwei Geometrien vorhanden ist.	• ST_Disjoint

Die Vergleichsfunktionen in DB2 Spatial Extender geben den Wert 1 (eins) zurück, wenn ein Vergleich bestimmte Bedingungen erfüllt, und den Wert 0 (null), wenn ein Vergleich die Bedingungen nicht erfüllt. Falls der Vergleich nicht ausgeführt werden konnte, wird kein Wert (Nullwert) zurückgegeben.

Vergleiche können nicht ausgeführt werden, wenn die Vergleichsoperation nicht für die Eingabeparameter definiert wurde oder wenn einer der Parameter null ist. Vergleiche können allerdings ausgeführt werden, wenn den Parametern Geometrien mit unterschiedlichen Datentypen oder Dimensionen zugeordnet sind.

Das Modell Dimensionally Extended 9 Intersection Model (DE-9IM) ist ein mathematischer Ansatz, der die paarweise räumliche Beziehung zwischen Geometrien mit unterschiedlichen Typen und Dimensionen definiert. Dieses Modell drückt die räumlichen Beziehungen zwischen allen Typen von Geometrien als paarweise Schnittmengen ihrer Innenbereiche, Begrenzungen und Außenbereiche unter Berücksichtigung der Dimension der resultierenden Schnittmengen aus.

Die angegebenen Geometrien sind *a* und *b*: I(*a*), B(*a*) und E(*a*) stellen Innenbereich, Begrenzung und Außenbereich von *a* dar. I(b), B(b) und E(b) stellen den Innenbereich, die Begrenzung und den Außenbereich von *b* dar. Die Schnittmengen von I(a), B(a) und E(a) mit I(b), B(b) und E(b) ergeben eine 3–mal–3-Matrix. Jede Schnittmenge kann Geometrien verschiedener Dimensionen ergeben. Die Schnitt-

menge der Begrenzungen zweier Polygone besteht beispielsweise aus einem Punkt und einer Linienfolge. In diesem Fall gibt die dim-Funktion die maximale Dimension 1 zurück.

Die dim-Funktion gibt den Wert - 1, 0, 1 oder 2 zurück. Der Wert - 1 entspricht einer Nullmenge oder dim(null). Dieses Ergebnis wird zurückgegeben, wenn keine Schnittmengen gefunden wurden.

Sie können die Ergebnisse, die von Vergleichsfunktionen zurückgegeben werden, nachvollziehen oder überprüfen, indem Sie die Ergebnisse der Vergleichsfunktion mit einer Mustermatrix vergleichen, die die akzeptablen Werte des Modells DE-9IM darstellt.

Die Mustermatrix enthält die zulässigen Werte für alle Schnittmengen-Matrixzellen. Folgende Musterwerte sind möglich:

T Es muss eine Schnittmenge vorhanden sein; dim = 0, 1 oder 2.

F Es darf keine Schnittmenge vorhanden sein; dim = -1.

***** Es spielt keine Rolle, ob eine Schnittmenge vorhanden ist; dim = -1, 0, 1 oder 2.

0 Es muss eine Schnittmenge vorhanden sein, und ihre Dimension muss exakt 0 sein; dim = 0.

1 Es muss eine Schnittmenge vorhanden sein, und ihre maximale Dimension muss 1 sein; dim = 1.

2 Es muss eine Schnittmenge vorhanden sein, und ihre maximale Dimension muss 2 sein; dim = 2.

Die Funktion ST_Within gibt den Wert 1 zurück, wenn sich die Innenbereiche der beiden Geometrien schneiden und wenn der Innenbereich oder die Begrenzung von *a* sich nicht mit dem Außenbereich von *b* schneidet. Alle weitere Bedingungen sind nicht von Bedeutung.

Jede Funktion hat mindestens eine Mustermatrix; einige erfordern jedoch auch mehrere Matrizen, um die Beziehungen zwischen verschiedenen Kombinationen von Geometrietypen zu beschreiben.

Das Modell DE-91M wurde von Clementini und Felice entwickelt, die das '9 Intersection Model' von Egenhofer und Herring dimensional erweiterten. DE-9IM ist eine Zusammenarbeit von vier Autoren (Clementini, Eliseo, Di Felice und van Osstrom). Diese Autoren haben das Modell in "A Small Set of Formal Topological Relationships Suitable for End-User Interaction," D. Abel and B.C. Ooi (Ed.), *Advances in Spatial Database—Third International Symposium. SSD '93*. LNCS 692. S. 277 - 295, veröffentlicht. Das Modell "9 Intersection" von M. J. Egenhofer und J. Herring (Springer-Verlag Singapore [1993]) wurde in "Categorizing binary topological relationships between regions, lines, and points in geographic databases", *Tech. Report, Department of Surveying Engineering*, University of Maine, Orono, ME 1991, veröffentlicht.

Funktionen, die bestimmen, ob eine Geometrie eine andere Geometrie enthält

ST_Contains und ST_Within verwenden jeweils zwei Geometrien als Eingabe und ermitteln, ob der Innenbereich der einen Geometrie den Innenbereich der anderen Geometrie schneidet.

Vereinfacht ausgedrückt bestimmt ST_Contains, ob die erste eingegebene Geometrie die zweite Geometrie einschließt, also die zweite Geometrie enthält. ST_Within ermittelt, ob sich die erste Geometrie vollständig innerhalb der zweiten Geometrie befindet.

Funktionen, die bestimmen, ob Geometrien sich schneiden

Mit den Funktionen ST_Intersects, ST_Crosses, ST_Overlaps und ST_Touches kann festgestellt werden, ob es Überschneidungen zwischen zwei Geometrien gibt.

Unterschiede zwischen diesen Funktionen bestehen hauptsächlich im Umfang der von ihnen untersuchten Schnittmenge:

- ST_Intersects ermittelt, ob die beiden eingegebenen Geometrien eine der folgenden vier Bedingungen erfüllen: 1. Die Innenbereiche der Geometrien schneiden sich. 2. Die Begrenzungen der Geometrien schneiden sich. 3. Die Begrenzung der ersten Geometrie schneidet den Innenbereich der zweiten Geometrie. 4. Der Innenbereich der ersten Geometrie schneidet die Begrenzung der zweiten Geometrie.
- ST_Crosses dient zur Analyse der Überschneidung von Geometrien mit unterschiedlichen Dimensionen. Hierbei gilt allerdings die Ausnahme, dass auch die Überschneidung von Linienfolgen analysiert werden kann. In allen Fällen wird die Position der Überschneidung selbst als Geometrie betrachtet. ST_Crosses setzt voraus, dass diese Geometrie eine kleinere Dimension als die größere der beiden sich schneidenden Geometrien aufweist (bzw. dass die Position der Überschneidung bei zwei Linienfolgen eine kleinere Dimension als eine der Linienfolgen aufweist). Die Dimensionen einer Linienfolge und eines Polygons lauten beispielsweise 1 bzw. 2. Wenn sich zwei solche Geometrien schneiden und der Schnittbereich linear ist (Verlauf der Linienfolge entlang des Polygons), kann diese Stelle selbst als Linienfolge betrachtet werden. Da die Dimension einer Linienfolge (1) kleiner ist als die Dimension eines Polygons (2), würde ST_Crosses nach der Analyse der Überschneidung in diesem Fall den Wert 1 zurückgeben.
- Die Geometrien, die als Eingabe für die Funktion ST_Overlaps verwendet werden, müssen dieselbe Dimension aufweisen. Bei ST_Overlaps müssen sich diese Geometrien teilweise überlappen und dabei eine neue Geometrie (den Überlappungsbereich) bilden, deren Dimension mit der Dimension der eingegebenen Geometrien identisch ist.
- Mit ST_Touches kann festgestellt werden, ob die Begrenzungen zweier Geometrien sich überschneiden.

Funktionen, die die Hüllen von Geometrien vergleichen

ST_EnvIntersects und ST_MBRIntersects sind ähnliche Funktionen, die ermitteln, ob das kleinste Rechteck, das eine der Geometrien einschließt, das kleinste Rechteck schneidet, das die andere Geometrie einschließt. Ein solches Rechteck wird für gewöhnlich als Hülle bezeichnet.

Multipolygone, Polygone, Mehrlinienfolgen und gekrümmte Linienfolgen stoßen an die Seiten ihrer Hüllen. Waagerechte Linienfolgen, vertikale Linienfolgen und Punkte sind etwas kleiner als ihre Hüllen. ST_EnvIntersects ermittelt, ob sich die Hüllen von Geometrien schneiden.

Der kleinste rechteckige Bereich, in den eine Geometrie passt, wird als minimal einschließendes Rechteck oder minimaler Begrenzungsrahmen (MBR = Minimal Bounding Rectangle) bezeichnet. Bei den Hüllen von Multipolygonen, Polygonen, Mehrlinienfolgen und gekrümmten Linienfolgen handelt es sich eigentlich um MBRs. Die Hüllen, die horizontale Linienfolgen, vertikale Linienfolgen und Punkte umgeben, sind keine MBRs, da sie nicht den Mindestbereich darstellen, in den die-

se Geometrien hineinpassen. Diese letzteren Geometrien belegen keinen definierbaren Bereich und können somit keine MBRs haben. Es wurde allerdings die Konvention übernommen, nach der sie als ihre eigenen MBRs gelten. Daher ermittelt ST_MBRIntersects bei Multipolygonen, Polygonen, Mehrlinienfolgen und gekrümmten Linienfolgen die Schnittmengen derselben umgebenden Rechtecke wie die Funktion ST_EnvIntersects. Bei horizontalen Linienfolgen, vertikalen Linienfolgen und Punkten ermittelt ST_MBRIntersects jedoch die Schnittmengen dieser Geometrien selbst.

Funktionen, die prüfen, ob zwei Elemente identisch sind

Diese Funktionen vergleichen räumliche Bezugssysteme, Koordinatensystemdefinitionen oder Geometrien.

- ST_EqualCoordsys
- ST_Equals
- ST_EqualSRS

Funktion, die bestimmt, dass Geometrien sich nicht schneiden

ST_Disjoint gibt den Wert 1 (eins) zurück, wenn die Schnittmenge der beiden Geometrien eine leere Menge ist. Diese Funktion gibt das genaue Gegenteil der Funktion ST_Intersects zurück.

Die Abbildung zeigt unterschiedliche Geometrien und dass die Begrenzungen an keinem Punkt Überschneidungen aufweisen.

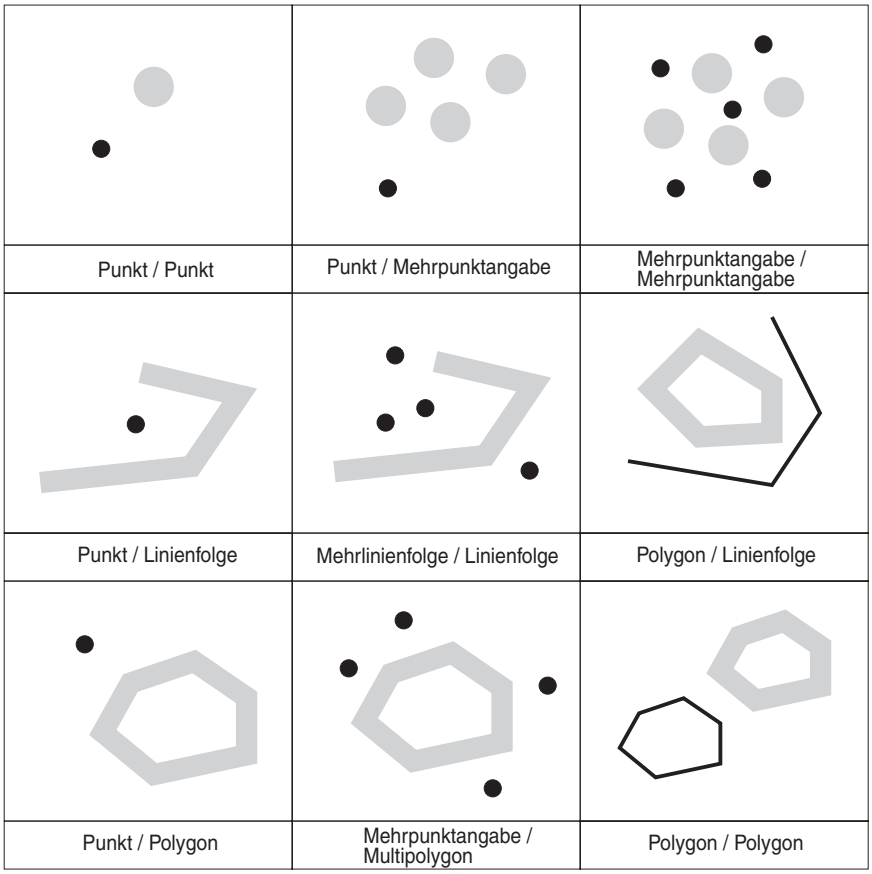


Abbildung 17. ST_Disjoint. Die dunklen Geometrien stellen die Geometrie a dar. Die grauen Geometrien stellen die Geometrie b dar. In allen Fällen bilden Geometrie a und Geometrie b keine Schnittmenge.

Die folgende Matrix gibt lediglich an, dass sich weder die Innenbereiche noch die Begrenzungen der Geometrien schneiden.

Tabelle 26. Matrix für ST_Disjoint

	Geometrie b Innenbereich	**Geometrie b Begrenzung**	**Geometrie b Außenbereich**
Geometrie a Begrenzung	F	F	*
Geometrie a Innenbereich	F	F	*
Geometrie a Außenbereich	*	*	*

Funktion, die zwei Geometrien mit der Zeichenfolge der DE-9IM-Mustermatrix vergleicht

Die Funktion ST_Relate vergleicht zwei Geometrien und gibt den Wert 1 (eins) zurück, wenn die Geometrien die Bedingungen erfüllen, die durch die Zeichenfolge der DE-9IM-Mustermatrix definiert sind. Andernfalls wird 0 (null) zurückgegeben.

Funktionen für das Abrufen von Informationen zu Geometrien und Indizes

DB2 Spatial Extender stellt Funktionen bereit, die Informationen zu den Eigenschaften von Geometrien und räumlichen Indizes zurückgeben.

Die zurückgegebenen Informationen betreffen:

- Datentypen von Geometrien
- Koordinaten und Bemaßungen in einer Geometrie
- Ringe, Begrenzungen, Hüllen und minimal einschließende Rechtecke (MBR)
- Dimensionen
- Angaben zu den Eigenschaften "Geschlossen", "Leer" oder "Einfach"
- Basisgeometrien in einer Geometriengruppe
- Räumliche Bezugssysteme
- Abstand zwischen Geometrien
- Parameter für die Definition eines räumlichen Index oder eines Index für eine räumliche Spalte

Einige Eigenschaften sind selbst Geometrien, beispielsweise der äußere und innere Ring einer Oberfläche oder der Start- und Endpunkt einer Kurve. Diese Geometrien werden durch einige der Funktionen in dieser Kategorie erzeugt. Funktionen, die andere Arten von Geometrien erzeugen (z. B. Geometrien für die Darstellung von Zonen, die einen bestimmten Standort umgeben), gehören zu der Kategorie „Räumliche Funktionen, die neue Geometrien generieren".

Funktion, die Datentypinformationen zurückgibt

ST_GeometryType verwendet eine Geometrie als Eingabeparameter und gibt den vollständigen Typnamen des dynamischen Typs dieser Geometrie zurück.

Funktionen, die Koordinaten- und Bemaßungsinformationen zurückgeben

Die folgenden Funktionen geben Informationen zu den Koordinaten und Bemaßungen innerhalb einer Geometrie zurück. Die Funktion ST_X kann beispielsweise die X-Koordinate in einem angegebenen Punkt zurückgeben. ST_MaxX gibt die

höchste X-Koordinate innerhalb einer Geometrie zurück, und ST_MinX gibt die niedrigste X-Koordinate innerhalb einer Geometrie zurück.

Diese Funktionen sind im Einzelnen:

- 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

Funktionen, die Informationen zu Geometrien innerhalb einer Geometrie zurückgeben

Die folgenden Funktionen geben Informationen zu Geometrien innerhalb einer Geometrie zurück. Einige Funktionen geben spezifische Punkte innerhalb einer Geometrie an. Andere Funktionen geben hingegen die Anzahl der Basisgeometrien innerhalb einer Gruppe zurück.

Diese Funktionen sind im Einzelnen:

- ST_Centroid
- ST_EndPoint
- ST_GeometryN
- ST_LineStringN
- ST_MidPoint
- ST_NumGeometries
- ST_NumLineStrings
- ST_NumPoints
- ST_NumPolygons
- ST_PointN
- ST_PolygonN
- ST_StartPoint

Funktionen, die Informationen zu Begrenzungen, Hüllen und Ringen zurückgeben

Die folgenden Funktionen geben Informationen zu den Grenzen zurück, die einen inneren Teil einer Geometrie von einem äußeren Teil trennen oder die die Geometrie selbst von ihrem externen Bereich trennen. Die Funktion ST_Boundary gibt beispielsweise die Begrenzung einer Geometrie in Form einer Kurve zurück.

Diese Funktionen sind im Einzelnen:

- ST_Boundary
- ST_Envelope
- ST_EnvIntersects
- ST_ExteriorRing
- ST_InteriorRingN
- ST_MBR
- ST_MBRIntersects
- ST_NumInteriorRing
- ST_Perimeter

Funktionen, die Informationen zu den Dimensionen einer Geometrie zurückgeben

Die folgenden Funktionen geben Informationen zu den Dimensionen einer Geometrie zurück, z. B. Fläche einer angegebenen Geometrie oder Länge einer angegebenen Kurve oder Mehrfachkurve.

Diese Funktionen sind im Einzelnen:

- ST_Area
- ST_Dimension
- ST_Length

Funktionen, die angeben, ob eine Geometrie geschlossen, leer oder einfach ist

DB2 Spatial Extender stellt Funktionen bereit, die angeben, ob eine Geometrie geschlossen, leer oder einfach ist.

Diese Funktionen zeigen an,

- ob eine angegebene Kurve oder Mehrfachkurve geschlossen ist (also der Anfangspunkt und der Endpunkt der Kurve bzw. Mehrfachkurve identisch sind).
- ob eine angegebene Geometrie leer ist (also keine Punkte enthält).
- ob eine Kurve, Mehrfachkurve oder Mehrpunktangabe einfach ist (also, ob solche Geometrien typische Konfigurationen haben).

Diese Funktionen lauten wie folgt:

- ST_IsClosed
- ST_IsEmpty
- ST_IsSimple

Funktionen, die das räumliche Bezugssystem angeben, das einer Geometrie zugeordnet ist

Die folgenden Funktionen geben Werte zurück, die das räumliche Bezugssystem kennzeichnen, das der Geometrie zugeordnet wurde. Außerdem kann die Funktion ST_SrsID das räumliche Bezugssystem der Geometrie ändern, ohne die Geometrie selbst zu ändern oder umzusetzen.

Diese Funktionen lauten wie folgt:

- ST_SrsId (wird auch als ST_SRID bezeichnet)
- ST_SrsName

Funktion, die Informationen zu Abständen zwischen Geometrien zurückgibt

Die Funktion ST_Distance verwendet zwei Geometrien und optional eine Einheit als Eingabeparameter und gibt den kürzesten Abstand zwischen einem beliebigen Punkt in der ersten Geometrie und einem beliebigen Punkt in der zweiten Geometrie zurück, der in der angegebenen Einheit gemessen wird.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist, wird sie in das andere räumliche Bezugssystem umgewandelt.

Falls eine der beiden eingegebenen Geometrien den Wert null aufweist oder leer ist, wird null zurückgegeben.

Mit ST_Distance kann z. B. die kürzeste Distanz ermittelt werden, die ein Flugzeug zwischen zwei Punkten zurücklegen muss. Abb. 18 zeigt diesen Sachverhalt.

Die Abbildung zeigt eine Karte der USA mit einer geraden Linie zwischen zwei Punkten, im vorliegenden Fall zwischen Los Angeles und Chicago.

Abbildung 18. Kürzeste Distanz zwischen zwei Städten. ST_Distance kann die Koordinaten der Position von Los Angeles und Chicago als Eingabe verwenden und einen Wert zurückgeben, der die kürzeste Distanz zwischen diesen beiden Positionen angibt.

Funktion, die Indexinformationen zurückgibt

Die Funktion ST_GetIndexParms verwendet entweder die Kennung für einen räumlichen Index oder für eine räumliche Spalte als Eingabeparameter und gibt entweder die Parameter, mit denen der Index definiert wurde, oder den Index für die räumliche Spalte zurück.

Wird die Nummer eines zusätzlichen Parameters angegeben, wird nur der durch diese Nummer gekennzeichnete Parameter zurückgegeben.

Funktionen für das Generieren neuer Geometrien aus vorhandenen Geometrien

Dieser Abschnitt stellt die Kategorie der Funktionen vor, die aus vorhandenen Geometrien neue Geometrien ableiten.

Diese Kategorie umfasst keine Funktionen zur Ableitung von Geometrien, die Eigenschaften anderer Geometrien darstellen. Sie ist vielmehr für Funktionen gedacht, die zur Ausführung folgender Operationen dienen:

- Geometrien in andere Geometrien umwandeln
- Geometrien erstellen, die Raumkonfigurationen darstellen

- einzelne Geometrien aus Mehrfachgeometrien ableiten
- Geometrien auf der Grundlage von Bemaßungen erstellen
- Änderungen von Geometrien erstellen

Funktionen, die eine Geometrie von einem übergeordneten Typ in den entsprechenden untergeordneten Typ umwandeln

Die folgenden Funktionen können Geometrien eines übergeordneten Typs in die entsprechenden Geometrien eines untergeordneten Typs umwandeln.

Die Funktion ST_ToLineString kann beispielsweise eine Linienfolge des Typs ST_Geometry in eine Linienfolge des Typs ST_LineString umwandeln. Einige dieser Funktionen können außerdem Basisgeometrien und Geometriengruppen in einer gemeinsamen Geometriengruppe kombinieren. Die Funktion ST_ToMultiLine beispielsweise kann eine Linienfolge und eine Mehrlinienfolge in eine gemeinsame Mehrlinienfolge umwandeln.

- ST_Polygon
- ST_ToGeomColl
- ST_ToLineString
- ST_ToMultiLine
- ST_ToMultiPoint
- ST_ToMultiPolygon
- ST_ToPoint
- ST_ToPolygon

Funktionen, die neue Geometrien mit unterschiedlichen Raumkonfigurationen erstellen

Die folgenden Funktionen verwenden vorhandene Geometrien als Ausgangspunkt und erstellen dann neue Geometrien, die kreisförmige Bereiche oder andere Raumkonfigurationen darstellen. Bei Eingabe eines bestimmten Punkts, der beispielsweise den Mittelpunkt eines geplanten Flughafens darstellt, kann die Funktion ST_Buffer eine Fläche erstellen, die die vorgeschlagene Ausdehnung des Flughafens in Kreisform zeigt.

Diese Funktionen sind im Einzelnen:

- ST_Buffer
- ST_ConvexHull
- ST_Difference
- ST_Intersection
- ST_SymDifference

Funktionen, die eine neue Geometrie aus mehreren Geometrien ableiten

Die folgenden Funktionen leiten einzelne Geometrien aus Mehrfachgeometrien ab. Beispielsweise durch die Kombination zweier Geometrien in einer gemeinsamen Geometrie.

- MBR Aggregate
- ST_Union
- Union-Gesamtverknüpfung

Funktionen, die eine neue Geometrie auf der Basis vorhandener Geometriebemaßungen erstellen

Diese Funktionen können eine neue Geometrie auf der Basis von Bemaßungen einer vorhandenen Geometrie erstellen. Sie können darüber hinaus den Abstand zu einer bestimmten Position entlang der Geometrie zurückgeben.

Diese Funktionen sind im Einzelnen:

- ST_DistanceToPoint
- ST_FindMeasure oder ST_LocateAlong
- ST_MeasureBetween oder ST_LocateBetween
- ST_PointAtDistance

Funktionen, die geänderte Formen bestehender Geometrien erstellen

Die folgenden Funktionen erstellen geänderte Formen bestehender Geometrien, beispielsweise erweiterte Versionen vorhandener Kurven. Jede Version enthält die Punkte in einer vorhandenen Kurve und einen zusätzlichen Punkt.

Diese Funktionen sind im Einzelnen:

- ST_AppendPoint
- ST_ChangePoint
- ST_Generalize
- ST_M
- ST_PerpPoints
- ST_RemovePoint
- ST_X
- ST_Y
- ST_Z

Funktionen, die Geometrien zwischen Koordinatensystemen umwandeln

Die Funktion ST_Transform verwendet eine Geometrie und die Kennung eines räumlichen Bezugssystems als Eingabeparameter und setzt die Geometrie für die Darstellung im angegebenen räumlichen Bezugssystem um.

Projektionen und Umwandlungen zwischen unterschiedlichen Koordinatensystemen werden ausgeführt, und die Koordinaten der Geometrien werden entsprechend angepasst.

Funktion EnvelopesIntersect

Mithilfe der Funktion EnvelopesIntersect können Sie bestimmen, ob zwei Geometrien sich schneiden oder ob eine Geometrie eine Hülle schneidet, die durch die vier Werte vom Typ DOUBLE definiert ist.

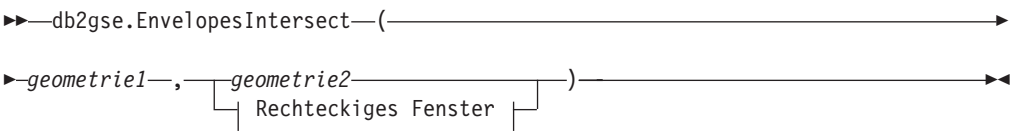
EnvelopesIntersect akzeptiert zwei Typen von Eingabeparametern:

- Zwei Geometrien

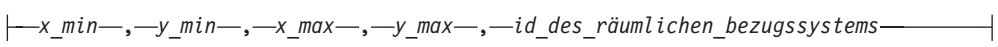
  EnvelopesIntersect gibt 1 zurück, wenn die Hülle der ersten Geometrie die Hülle der zweiten Geometrie schneidet. Andernfalls wird 0 (null) zurückgegeben.
- Eine Geometrie, vier Koordinatenwerte vom Typ DOUBLE, die die untere linke und obere rechte Ecke eines rechteckigen Fensters definieren, und die Kennung des räumlichen Bezugssystems (SRID).

EnvelopesIntersect gibt 1 zurück, wenn die Hülle der ersten Geometrie die durch die vier Werte vom Typ DOUBLE definierte Hülle schneidet. Andernfalls wird 0 (null) zurückgegeben.

Syntax



Rechteckiges Fenster:



Parameter

geometrie1
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, deren Hülle auf Schnittpunkte mit der Hülle der in *geometrie2* angegebenen Geometrie oder mit der Hülle des rechteckigen Fensters, das durch die vier Werte vom Typ DOUBLE definiert wird, getestet wird.

geometrie2
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, deren Hülle auf Schnittpunkte mit der Hülle der in *geometrie1* angegebenen Geometrie getestet wird.

x_min
Gibt den minimalen Wert der X-Koordinate für die Hülle an. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Dieser Parameter hat den Datentyp DOUBLE.

y_min
Gibt den minimalen Wert der Y-Koordinate für die Hülle an. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Dieser Parameter hat den Datentyp DOUBLE.

x_max
Gibt den maximalen Wert der X-Koordinate für die Hülle an. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Dieser Parameter hat den Datentyp DOUBLE.

y_max
Gibt den maximalen Wert der Y-Koordinate für die Hülle an. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Dieser Parameter hat den Datentyp DOUBLE.

id_des_räumlichen_bezugssystems
Dieser Parameter gibt die eindeutige Kennung des räumlichen Bezugssystems an. Die Kennung des räumlichen Bezugssystems muss mit der zum Geometrieparameter gehörenden Kennung des räumlichen Bezugssystems übereinstimmen. Für diesen Parameter müssen Sie einen anderen Wert als NULL angeben.

Dieser Parameter hat den Datentyp INTEGER.

Rückgabetyp

INTEGER

Beispiel

Dieses Beispiel erstellt zwei Polygone, die Bezirke darstellen, und ermittelt anschließend, ob einer der Bezirke ein geografisches Gebiet schneidet, das durch die vier Werte vom Typ DOUBLE angegeben wird.

```
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
```

Ergebnisse:

```
Name
--------------------
County_1
County_2
```

MBR Aggregate-Funktionen

Verwenden Sie eine Kombination der Funktionen ST_BuildMBRAggr und ST_GetAggrResult, um mehrere Geometrien in einer Spalte zu einer einzigen Geometrie zusammenzufassen. Die Kombination erzeugt ein Rechteck, das den minimalen Begrenzungsrahmen (MBR) darstellt, der alle Geometrien in der Spalte einschließt. Bei der Berechnung des Ergebnisses werden Z- und M-Koordinaten gelöscht.

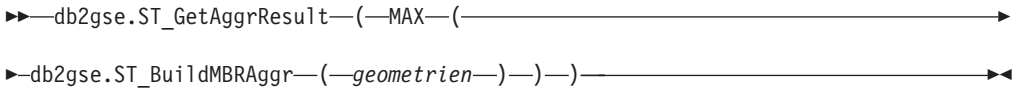
Beim folgenden Ausdruck handelt es sich um ein Beispiel für die Verwendung der Funktion MAX in Verbindung mit der räumlichen Funktion db2gse.ST_BuildMBRAggr zur Berechnung des MBR der Geometrien in der Spalte columnName sowie für die Verwendung der räumlichen Funktion db2gse.ST_GetAggrResult zur Ausgabe der daraus resultierenden Geometrie, die für das MBR berechnet wurde:

```
db2gse.ST_Get_AggrResult(MAX(db2gse.ST_BuildMBRAggr(columnName)))
```

Wenn alle zu kombinierenden Geometrien den Wert null aufweisen, wird null zurückgegeben. Wenn alle Geometrien entweder den Wert null aufweisen oder leer sind, wird eine leere Geometrie zurückgegeben. Wenn das minimal einschließende Rechteck (MBR) aller zu kombinierenden Geometrien einen Punkt als Ergebnis hat, wird dieser Punkt als ein Wert vom Typ ST_Point zurückgegeben. Wenn das minimal einschließende Rechteck aller zu kombinierenden Geometrien eine horizontale oder vertikale Linienfolge als Ergebnis hat, wird diese Linienfolge als ein Wert vom

Typ ST_LineString zurückgegeben. Andernfalls wird das minimal einschließende Rechteck als ein Wert vom Typ ST_Polygon zurückgegeben.

Syntax



Parameter

Geometrien
Eine ausgewählte Spalte, die den Typ ST_Geometry oder einen seiner untergeordneten Typen aufweist und alle Geometrien darstellt, für die das minimal einschließende Rechteck berechnet werden soll.

Rückgabetyp

db2gse.ST_Geometry

Einschränkungen

Sie können in den folgenden Situationen keine Union-Gesamtverknüpfung einer räumlichen Spalte in einem Fullselect erstellen:

- In einer Umgebung mit partitionierten Datenbanken.
- Wenn die Klausel GROUP BY für den Fullselect verwendet wird.
- Wenn Sie eine andere Funktion als die DB2-Spaltenfunktion MAX verwenden.

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel zeigt, wie die Funktion ST_BuildMBRAggr verwendet wird, um das maximal einschließende Rechteck aller Geometrien in einer Spalte zu erhalten. In diesem Beispiel werden der Spalte GEOMETRY in der Tabelle SAMPLE_POINTS mehrere Punkte hinzugefügt. Der SQL-Code ermittelt dann das maximal einschließende Rechteck aller zusammengefassten Punkte.

```
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
```

Ergebnisse:

```
Aggregate_of_Points
-------------------------------------------------------
POLYGON (( 2.00000000 2.00000000, 23.00000000 2.00000000,
23.00000000 15.00000000, 2.00000000 15.00000000, 2.00000000
2.00000000))
```

Funktion ST_AppendPoint

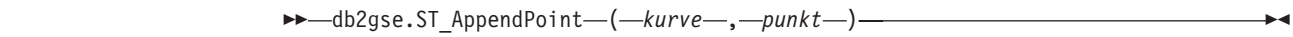
Die Funktion ST_AppendPoint verwendet eine Kurve und einen Punkt als Eingabeparameter und erweitert die Kurve um den angegebenen Punkt. Wenn die angegebene Kurve Z- oder M-Koordinaten aufweist, muss der Punkt ebenfalls Z- oder M-Koordinaten aufweisen. Die Ergebniskurve wird im räumlichen Bezugssystem der angegebenen Kurve dargestellt.

Wenn der hinzuzufügende Punkt nicht in demselben räumlichen Bezugssystem wie die Kurve dargestellt wird, wird der Punkt in das andere räumliche Bezugssystem umgewandelt.

Wenn die angegebene Kurve eine geschlossene oder einfache Kurve ist, ist die Ergebniskurve möglicherweise nicht mehr eine geschlossene oder einfache Kurve. Wenn die angegebene Kurve oder der angegebene Punkt den Wert null aufweist oder wenn die Kurve leer ist, wird null zurückgegeben. Wenn der hinzuzufügende Punkt leer ist, wird die angegebene Kurve unverändert zurückgegeben und eine Warnung (SQLSTATE 01HS3) wird erzeugt.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

kurve Ein Wert vom Typ ST_Curve oder einer seiner untergeordneten Typen, der die Kurve darstellt, zu der der in *punkt* angegebene Punkt hinzugefügt wird.

punkt Ein Wert vom Typ ST_Point, der den Punkt darstellt, der der in *kurve* angegebenen Kurve hinzugefügt wird.

Rückgabetyp

db2gse.ST_Curve

Beispiele

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Mit diesem Code werden zwei Linienfolgen mit jeweils drei Punkten erstellt.

```
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) )
```

Beispiel 1

In diesem Beispiel wird der Punkt (5, 5) dem Ende einer Linienfolge hinzugefügt.

```
SELECT CAST(ST_AsText(ST_AppendPoint(line, ST_Point(5, 5)))
    AS VARCHAR(120)) New
FROM   sample_lines
WHERE  id=1
```

Ergebnisse:

```
NEW
------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 10.00000000 0.00000000,
0.00000000 0.00000000, 5.00000000 5.00000000)
```

Beispiel 2

In diesem Beispiel wird der Punkt (15, 15, 7) dem Ende einer Linienfolge mit Z-Koordinaten hinzugefügt.

```
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
```

Ergebnisse:

```
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)
```

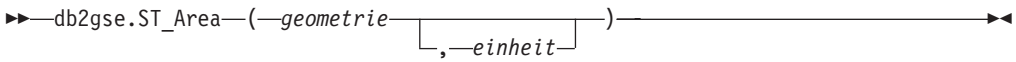
Funktion ST_Area

Die Funktion ST_Area verwendet eine Geometrie und optional eine Einheit als Eingabeparameter und gibt die von der angegebenen Geometrie bedeckte Fläche in der Standardmaßeinheit bzw. der angegebenen Maßeinheit zurück.

Wenn es sich bei der Geometrie um ein Polygon oder ein Multipolygon handelt, wird die von der Geometrie bedeckte Fläche zurückgegeben. Die bedeckte Fläche bei Punkten, Linienfolgen, Mehrpunktangaben oder Mehrlinienfolgen ist gleich 0 (null). Wenn die Geometrie den Wert null aufweist oder leer ist, wird 0 (null) zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die die Fläche bestimmt.

einheit

Ein Wert vom Typ VARCHAR(128), der die Einheiten kennzeichnet, in denen die Größe der Fläche gemessen wird. Die unterstützten Maßeinheiten sind in der Katalogsicht DB2GSE.ST_UNITS_OF_MEASURE aufgelistet.

Wenn der Parameter *einheit* ausgelassen wird, werden die folgenden Regeln verwendet, um die Einheit zu ermitteln, in der die Größe der Fläche gemessen wird.

- Wenn die *Geometrie* sich in einem projizierten oder geozentrischen Koordinatensystem befindet, wird die lineare Einheit verwendet, die diesem Koordinatensystem zugeordnet ist.
- Wenn die *Geometrie* sich in einem geografischen Koordinatensystem befindet, wird die Winkeleinheit verwendet, die diesem Koordinatensystem zugeordnet ist.

Einschränkungen bei Einheitenumsetzungen: Ein Fehler (SQLSTATE 38SU4) wird zurückgegeben, wenn eine der folgenden Bedingungen zutrifft:

- Die Geometrie befindet sich in einem nicht definierten Koordinatensystem, und der Parameter *einheit* wurde angegeben.
- Die Geometrie befindet sich in einem projizierten Koordinatensystem, und es wurde eine Winkeleinheit angegeben.
- Die Geometrie befindet sich in einem geografischen Koordinatensystem, und es wurde eine lineare Einheit angegeben.

Rückgabetyp

DOUBLE

Beispiele

Beispiel 1

Der Raumanalytiker benötigt eine Liste der Flächen, die von der jeweiligen Verkaufsregion bedeckt sind. Die Flächen der Verkaufsregionen sind in der Tabelle SAMPLE_POLYGONS gespeichert. Die Größe der Fläche wird durch Anwendung der Funktion ST_Area auf die Geometriespalte berechnet.

```
db2se create_srs se_bank  -srsId 4000 -srsName new_york1983 -xOffset 0
        -yOffset 0 -xScale 1 -yScale 1
    -coordsysName NAD_1983_StatePlane_New_York_East_FIPS_3101_Feet

CREATE TABLE sample_polygons (id INTEGER, geometry ST_POLYGON)

INSERT INTO sample_polygons (id, geometry)
  VALUES
    (1, ST_Polygon('polygon((0 0, 0 10, 10 10, 10 0, 0 0))', 4000) ),
    (2, ST_Polygon('polygon((20 0, 30 20, 40 0, 20 0 ))', 4000) ),
    (3, ST_Polygon('polygon((20 30, 25 35, 30 30, 20 30))', 4000))
```

Die folgende Anweisung SELECT ruft die Verkaufsregions-ID und die zugehörige Fläche ab:

```
SELECT id, ST_Area(geometry) AS area
FROM   sample_polygons
```

Ergebnisse:

```
ID       AREA
-------- -----------------------
       1   +1.00000000000000E+002
       2   +2.00000000000000E+002
       3   +2.50000000000000E+001
```

Beispiel 2

Die folgende Anweisung SELECT ruft die ID und Fläche der Verkaufsregion in verschiedenen Einheiten ab:

```
SELECT id,
             ST_Area(geometry) square_feet,
             ST_Area(geometry, 'METER') square_meters,
             ST_Area(geometry, 'STATUTE MILE') square_miles
FROM   sample_polygons
```

Ergebnisse:

```
  ID  SQUARE_FEET             SQUARE_METERS            SQUARE_MILES
  -- ----------------------   ----------------------   ----------------------
   1 +1.00000000000000E+002   +9.29034116132748E+000   +3.58702077598427E-006
   2 +2.00000000000000E+002   +1.85806823226550E+001   +7.17404155196855E-006
   3 +2.50000000000000E+001   +2.32258529033187E+000   +8.96755193996069E-007
```

Beispiel 3

In diesem Beispiel wird die Fläche gesucht, die von einem Polygon abgedeckt wird, das mit SPC-Koordinaten (SPC = State Plane Coordinates) definiert ist.

Das räumliche SPC-Bezugssystem (SPC = State Plane Coordinates) mit der ID 3 wird mit dem folgenden Befehl erstellt:

```
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
```

Mit den folgenden SQL-Anweisungen wird das Polygon im räumlichen Bezugssystem 3 der Tabelle hinzugefügt und dessen Größe in Quadratmetern, Quadratfuß und Quadratmeilen ermittelt.

```
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;
```

Ergebnisse:

```
  ID SQUARE FEET              SQUARE METERS            SQUARE MILES
  -- ----------------------   ----------------------   ----------------------
   1 +7.94698788000000E+008   +7.38302286101346E+007   +2.85060106320552E+001
```

Funktion ST_AsBinary

Die Funktion ST_AsBinary verwendet eine Geometrie als Eingabeparameter und gibt seine bekannte binäre Darstellung (WKB) zurück. Die Z- und M-Koordinaten werden gelöscht und in der WKT-Darstellung (WKT = Well-Known Text) nicht dargestellt.

Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der in die entsprechende, bekannte binäre Darstellung umgewandelt werden soll.

Rückgabetyp

BLOB(2G)

Beispiele

Der folgende Code zeigt, wie die Funktion ST_AsBinary zur Umwandlung von Punkten in den Geometriespalten der Tabelle SAMPLE_POINTS in die WKB-Darstellung (WKB = Well-Known Binary) in der Spalte BLOB verwendet wird.

```
CREATE TABLE SAMPLE_POINTS (id integer, geometry ST_POINT, wkb BLOB(32K))

INSERT INTO SAMPLE_POINTS (id, geometry)
  VALUES
       (1100, ST_Point(10, 20, 1))
```

Beispiel 1

In diesem Beispiel wird die Spalte WKB mit der ID 1111 aus der Spalte GEOMETRY mit der ID 1100 gefüllt.

```
INSERT INTO sample_points(id, wkb)
VALUES (1111,
  (SELECT ST_AsBinary(geometry)
   FROM sample_points
   WHERE   id = 1100))

SELECT id, cast(ST_Point(wkb)..ST_AsText AS varchar(35)) AS point
FROM sample_points
WHERE  id = 1111
```

Ergebnisse:

```
ID          Point
----------- -----------------------------------
             1111 POINT ( 10.00000000 20.00000000)
```

Beispiel 2

Dieses Beispiel zeigt die WKB-Darstellung.

```
SELECT id, substr(ST_AsBinary(geometry), 1, 21) AS point_wkb
FROM sample_points
WHERE  id = 1100
```

Ergebnisse:

```
ID     POINT_WKB
------ ---------------------------------------------
    1100 x'0101000000000000000000244000000000000034400'
```

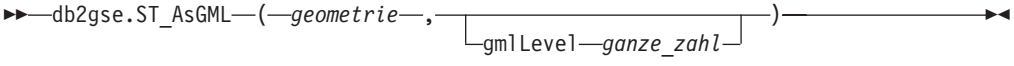
Funktion ST_AsGML

Die Funktion ST_AsGML verwendet eine Geometrie als Eingabeparameter und gibt deren GML-Darstellung (Geography Markup Language) zurück.

Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

gmlLevel
Dieser optionale Parameter gibt die GML-Spezifikationsstufe zum Formatieren der GML-Daten an, die zurückgegeben werden sollen. Gültige Werte:

- 2 - GML-Spezifikationsstufe 2 mit dem Tag <gml:coordinates> verwenden.
- 3 - GML-Spezifikationsstufe 3 mit dem Tag <gml:poslist> verwenden.

Wenn kein Parameter angegeben ist, wird die Ausgabe in der GML-Spezifikationsstufe 2 mit dem Tag <gml:coord> zurückgegeben.

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der in die entsprechende GML-Darstellung umgewandelt werden soll.

Rückgabetyp

CLOB(2G)

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Das folgende Codefragment verdeutlicht, wie die Funktion ST_AsGML zur Anzeige des GML-Fragments verwendet wird. In diesem Beispiel wird die Spalte GML aus der Geometriespalte mit der ID 2222 gefüllt.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE SAMPLE_POINTS (id integer, geometry ST_POINT, gml CLOB(32K))

INSERT INTO SAMPLE_POINTS (id, geometry)
  VALUES
        (1100, ST_Point(10, 20, 1))

INSERT INTO sample_points(id, gml)
VALUES (2222,
  (SELECT ST_AsGML(geometry)
   FROM sample_points
   WHERE   id = 1100))
```

Die folgende Anweisung SELECT listet die ID und die GML-Darstellungen der Geometrie auf.

```
SELECT id, cast(ST_AsGML(geometry) AS varchar(110)) AS gml_fragment
FROM sample_points
WHERE  id = 1100
```

Ergebnisse:

```
SELECT id,
       cast(ST_AsGML(geometry) AS varchar(110)) AS gml,
       cast(ST_AsGML(geometry,2) AS varchar(110)) AS gml2,
       cast(ST_AsGML(geometry,3) AS varchar(110)) AS gml3
FROM sample_points
WHERE  id = 1100
```

Die Anweisung SELECT gibt die folgende Ergebnisgruppe zurück:

```
ID          GML                               GML2                              GML3
----------- --------------------------------  --------------------------------  --------------------------------
       1100 <gml:Point srsName="EPSG:4269">   <gml:Point srsName="EPSG:4269">   <gml:Point srsName="EPSG:4269
                                                                                srsDimension="2">
            <gml:coord>                       <gml:coordinates>                 <gml:pos>
            <gml:X>10</gml:X><gml:Y>20</gml:Y> 10,20                            10,20
            </gml:coord></gml:Point>          </gml:coordinates></gml:Point>    </gml:pos></gml:Point>
```

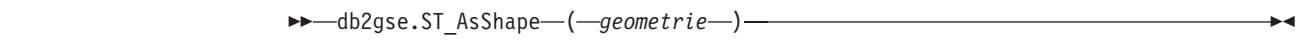
Funktion ST_AsShape

Die Funktion St_AsShape verwendet eine Geometrie als Eingabeparameter und gibt deren ESRI-Formdarstellung zurück.

Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der in die entsprechende ESRI-Formdarstellung umgewandelt werden soll.

Rückgabetyp

BLOB(2G)

Beispiel

Das folgende Codefragment verdeutlicht, wie die Funktion ST_AsShape zur Umwandlung von Punkten in der Geometriespalte der Tabelle SAMPLE_POINTS in die binäre Formdarstellung in der Formspalte BLOB verwendet wird. In diesem Beispiel wird die Formspalte aus der Geometriespalte gefüllt. Die binäre Formdarstellung wird verwendet, um die Geometrien in Geobrowsern anzuzeigen, die Geometrien erfordern, die dem ESRI-Formdateiformat entsprechen, oder um die Geometrien für die Datei *.SHP der Formdatei zu erstellen.

```
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
```

Ergebnisse:

```
ID     SHAPE

------ ----------------------------------------
  1100 x'0100000000000000000024400000000000003440'
```

Funktion ST_AsText

Die Funktion ST_AsText verwendet als Eingabeparameter eine Geometrie und gibt deren WKT-Darstellung (WKT = Well-Known Text) zurück.

Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der in die entsprechende WKT-Darstellung (WKT = Well-Known Text) umgewandelt werden soll.

Rückgabetyp

CLOB(2G)

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Nach der Erfassung und dem Einfügen der Daten in die Tabelle SAMPLE_GEOMETRIES möchte eine Analytiker prüfen, ob die eingefügten Werte richtig sind, indem er die WKT-Darstellung (WKT = Well-Known Text) der Geometrien überprüft.

```
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))
```

Die folgende Anweisung SELECT listet den räumlichen Typ und die WKT-Darstellung (WKT = Well-Known Text) der Geometrien auf. Die Geometrie wird mithilfe der Funktion ST_AsText in Text umgewandelt. Anschließend wird Sie in eine Angabe varchar(120) umgesetzt, da die Standardausgabe der Funktion ST_AsText CLOB(2G) lautet.

```
SELECT id, spatial_type, cast(geometry..ST_AsText
   AS varchar(150)) AS wkt
FROM   sample_geometries
```

Ergebnisse:

```
ID     SPATIAL_TYPE       WKT
------ ------------------ ------------------------------------------------

     1 st_point           POINT ( 50.00000000 50.00000000)

     2 st_linestring      LINESTRING ( 200.00000000 100.00000000,
                                       210.00000000 130.00000000, 220.00000000
                                       140.00000000)

     3 st_polygon         POLYGON (( 110.00000000 120.00000000,
                                     130.00000000 120.00000000, 130.00000000
                                     140.00000000, 110.00000000140.00000000,
                                     110.00000000 120.00000000))
```

Funktion ST_Boundary

Die Funktion ST_Boundary verwendet eine Geometrie als Eingabeparameter und gibt deren Grenze als neue Geometrie zurück. Die Ergebnisgeometrie wird im räumlichen Bezugssystem der angegebenen Geometrie dargestellt.

Wenn die angegebene Geometrie ein Punkt, eine Mehrpunktangabe, eine geschlossene Kurve oder eine geschlossene Mehrfachkurve oder leer ist, ist das Ergebnis eine leere Geometrie vom Typ ST_Point. Für Kurven und Mehrfachkurven, die nicht geschlossen sind, werden die Start- und Endpunkte der Kurven als ein Wert vom Typ ST_MultiPoint zurückgegeben. Es sei denn, ein solcher Punkt ist der Start- oder Endpunkt einer geraden Anzahl von Kurven. Für Oberflächen und Mehrfachoberflächen wird die Kurve zurückgegeben, die die Grenze der angegebenen Geometrie definiert. Die Rückgabe erfolgt entweder als ein Wert vom Typ ST_Curve oder als ein Wert vom Typ ST_MultiCurve. Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Nach Möglichkeit ist der spezifische Typ der zurückgegebenen Geometrie ST-_Point, ST_LineString oder ST_Polygon. Die Grenze eines Polygons ohne Löcher ist zum Beispiel eine einzige Linienfolge, die als ST_LineString dargestellt wird. Die Grenze eines Polygons mit mindestens einem Loch besteht aus mehreren Linienfolgen, die als ST_MultiLineString dargestellt werden.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen. Die Grenze dieser Geometrie wird zurückgegeben.

Rückgabetyp

db2gse.ST_Geometry

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

In diesem Beispiel werden mehrere Geometrien erstellt. Anschließend wird die Grenze jeder Geometrie ermittelt.

```
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

Ergebnisse:
ID      BOUNDARY
------- ------------------------------------------------------------------
      1 LINESTRING ( 40.00000000 120.00000000, 90.00000000 120.00000000,
        90.00000000 150.00000000, 40.00000000 150.00000000, 40.00000000
        120.00000000)

      2 MULTILINESTRING (( 40.00000000 120.00000000, 90.00000000 120.00000000,
        90.00000000 150.00000000, 40.00000000 150.00000000, 40.00000000
        120.00000000),( 70.00000000 130.00000000, 70.00000000 140.00000000,
        80.00000000 140.00000000, 80.00000000 130.00000000, 70.00000000
        130.00000000))

      3 MULTIPOINT ( 60.00000000 60.00000000, 70.00000000 70.00000000)
```

```
4 MULTIPOINT ( 50.00000000 50.00000000, 70.00000000 70.00000000,
               80.00000000 80.00000000, 90.00000000 90.00000000)

5 POINT EMPTY
```

Funktion ST_Buffer

Die Funktion ST_Buffer verwendet eine Geometrie, einen Abstand und optional eine Einheit als Eingabeparameter und gibt die Geometrie zurück, die die angegebene Geometrie im angegebenen Abstand gemessen in der angegebenen Einheit umgibt.

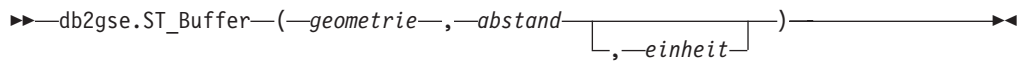
Jeder Punkt auf der Grenze der Ergebnisgeometrie ist im angegebenen Abstand von der angegebenen Geometrie entfernt. Die Ergebnisgeometrie wird im räumlichen Bezugssystem der angegebenen Geometrie dargestellt.

Jede kreisförmige Kurve der Grenze der Ergebnisgeometrie wird durch Linienfolgen angenähert. Der Puffer, der einen Punkt umgibt, und einen kreisförmigen Bereich bildet, wird z. B. durch ein Polygon näherungsweise bestimmt, dessen Begrenzung durch eine Linienfolge gebildet wird.

Wenn die angegebene Geometrie den Wert null aufweist oder leer ist, wird 0 (null) zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, um die herum der Puffer erstellt werden soll.

abstand
Ein Wert vom Typ DOUBLE PRECISION, der den Abstand angibt, der für den Puffer um die in *geometrie* angegebene Geometrie herum verwendet werden soll.

einheit
Ein Wert vom Typ VARCHAR(128), der die Einheit kennzeichnet, in der der in *abstand* angegebene Abstand gemessen wird. Die unterstützten Maßeinheiten sind in der Katalogsicht DB2GSE.ST_UNITS_OF_MEASURE aufgelistet.

Wenn der Parameter *einheit* ausgelassen wird, werden die folgenden Regeln verwendet, um die Maßeinheit für den Abstand zu ermitteln:

- Wenn die in *geometrie* angegebene Geometrie sich in einem projizierten oder geozentrischen Koordinatensystem befindet, wird als Standardwert die lineare Einheit verwendet, die diesem Koordinatensystem zugeordnet ist.
- Wenn die in *geometrie* angegebene Geometrie sich in einem geografischen Koordinatensystem befindet, wird standardmäßig die Winkeleinheit verwendet, die diesem Koordinatensystem zugeordnet ist.

Einschränkungen bei Einheitenumsetzungen: Ein Fehler (SQLSTATE 38SU4) wird zurückgegeben, wenn eine der folgenden Bedingungen zutrifft:

- Die Geometrie befindet sich in einem nicht definierten Koordinatensystem, und der Parameter *einheit* wurde angegeben.
- Die Geometrie befindet sich in einem projizierten Koordinatensystem, und es wurde eine Winkeleinheit angegeben.
- Die Geometrie befindet sich in einem geografischen Koordinatensystem, es handelt sich jedoch nicht um einen Wert für ST_Point und es wurde eine lineare Einheit angegeben.

Rückgabetyp

db2gse.ST_Geometry

Beispiele

In den folgenden Beispielen wurden die Ergebnisse zur besseren Lesbarkeit umformatiert. Der Abstand in den Ergebnissen variiert entsprechend der jeweiligen Anzeige.

Beispiel 1

Mit dem folgenden Code wird ein räumliches Bezugssystem erstellt. Ferner wird die Tabelle SAMPLE_GEOMETRIES erstellt und gefüllt.

```
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))
```

Beispiel 2

Die folgende Anweisung SELECT verwendet die Funktion ST_Buffer, um einen Puffer von 10 anzuwenden.

```
SELECT id, spatial_type,
           cast(geometry..ST_Buffer(10)..ST_AsText AS varchar(470)) AS buffer_10
FROM   sample_geometries
```

Ergebnisse:

```
ID          SPATIAL_TYPE       BUFFER_10
----------- ------------------ ----------------------------------------
1           st_point           POLYGON (( 60.00000000 50.00000000,
      59.00000000 55.00000000, 54.00000000 59.00000000, 49.00000000
      60.00000000, 44.00000000 58.00000000, 41.00000000 53.00000000,
```

```
      40.00000000 48.00000000,42.00000000 43.00000000, 47.00000000
      41.00000000, 52.00000000 40.00000000, 57.00000000 42.00000000,
      60.00000000 50.00000000))

2             st_linestring    POLYGON (( 230.00000000
      140.00000000, 229.00000000 145.00000000, 224.00000000
      149.00000000, 219.00000000 150.00000000, 213.00000000 147.00000000,
      203.00000000 137.00000000, 201.00000000 133.00000000, 191.00000000
      103.00000000, 191.00000000 99.00000000, 192.00000000  95.00000000,
      196.00000000 91.00000000, 200.00000000 91.00000000,204.00000000
      91.00000000, 209.00000000 97.00000000, 218.00000000 124.00000000,
      227.00000000 133.00000000, 230.00000000 140.00000000))

3              st_polygon         POLYGON (( 140.00000000 120.00000000,
      140.00000000 140.00000000, 139.00000000  145.00000000, 130.00000000
      150.00000000, 110.00000000 150.00000000, 105.00000000 149.00000000,
      100.00000000 140.00000000,100.00000000 120.00000000, 101.00000000
      115.00000000, 110.00000000 110.00000000,130.00000000 110.00000000,
      135.00000000 111.00000000, 140.00000000 120.00000000))

4              st_multipolygon    POLYGON (( 55.00000000 30.00000000,
      55.00000000 40.00000000, 54.00000000 45.00000000, 45.00000000
      50.00000000, 30.00000000 50.00000000, 25.00000000 49.00000000,
      20.00000000 40.00000000, 20.00000000 30.00000000, 21.00000000
      25.00000000, 30.00000000 20.00000000, 45.00000000 20.00000000,
      50.00000000 21.00000000, 55.00000000 30.00000000))
```

Beispiel 3

Die folgende Anweisung SELECT verwendet die Funktion ST_Buffer, um einen negativen Puffer von 5 anzuwenden.

```
SELECT id, spatial_type,
       cast(ST_AsText(ST_Buffer(geometry, -5)) AS varchar(150))
       AS buffer_negative_5
FROM   sample_geometries  WHERE  id = 3
```

Ergebnisse:

```
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))
```

Beispiel 4

Die folgende Anweisung SELECT verdeutlicht das Ergebnis der Anwendung eines Puffers unter Angabe des Parameters *einheit*.

```
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
```

Ergebnisse:

```
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))
```

MBR Aggregate-Funktionen

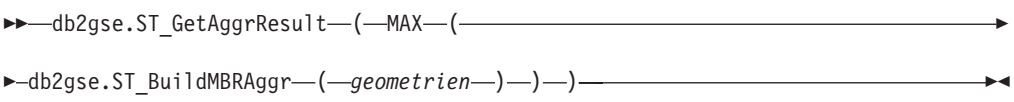
Verwenden Sie eine Kombination der Funktionen ST_BuildMBRAggr und ST_GetAggrResult, um mehrere Geometrien in einer Spalte zu einer einzigen Geometrie zusammenzufassen. Die Kombination erzeugt ein Rechteck, das den minimalen Begrenzungsrahmen (MBR) darstellt, der alle Geometrien in der Spalte einschließt. Bei der Berechnung des Ergebnisses werden Z- und M-Koordinaten gelöscht.

Beim folgenden Ausdruck handelt es sich um ein Beispiel für die Verwendung der Funktion MAX in Verbindung mit der räumlichen Funktion db2gse.ST_BuildMBRAggr zur Berechnung des MBR der Geometrien in der Spalte columnName sowie für die Verwendung der räumlichen Funktion db2gse.ST_GetAggrResult zur Ausgabe der daraus resultierenden Geometrie, die für das MBR berechnet wurde:

```
db2gse.ST_Get_AggrResult(MAX(db2gse.ST_BuildMBRAggr(columnName)))
```

Wenn alle zu kombinierenden Geometrien den Wert null aufweisen, wird null zurückgegeben. Wenn alle Geometrien entweder den Wert null aufweisen oder leer sind, wird eine leere Geometrie zurückgegeben. Wenn das minimal einschließende Rechteck (MBR) aller zu kombinierenden Geometrien einen Punkt als Ergebnis hat, wird dieser Punkt als ein Wert vom Typ ST_Point zurückgegeben. Wenn das minimal einschließende Rechteck aller zu kombinierenden Geometrien eine horizontale oder vertikale Linienfolge als Ergebnis hat, wird diese Linienfolge als ein Wert vom Typ ST_LineString zurückgegeben. Andernfalls wird das minimal einschließende Rechteck als ein Wert vom Typ ST_Polygon zurückgegeben.

Syntax



Parameter

Geometrien
Eine ausgewählte Spalte, die den Typ ST_Geometry oder einen seiner untergeordneten Typen aufweist und alle Geometrien darstellt, für die das minimal einschließende Rechteck berechnet werden soll.

Rückgabetyp

db2gse.ST_Geometry

Einschränkungen

Sie können in den folgenden Situationen keine Union-Gesamtverknüpfung einer räumlichen Spalte in einem Fullselect erstellen:

- In einer Umgebung mit partitionierten Datenbanken.
- Wenn die Klausel GROUP BY für den Fullselect verwendet wird.
- Wenn Sie eine andere Funktion als die DB2-Spaltenfunktion MAX verwenden.

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel zeigt, wie die Funktion ST_BuildMBRAggr verwendet wird, um das maximal einschließende Rechteck aller Geometrien in einer Spalte zu erhalten. In diesem Beispiel werden der Spalte GEOMETRY in der Tabelle SAMPLE_POINTS mehrere Punkte hinzugefügt. Der SQL-Code ermittelt dann das maximal einschließende Rechteck aller zusammengefassten Punkte.

```
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
```

Ergebnisse:

```
Aggregate_of_Points
------------------------------------------------------------
POLYGON (( 2.00000000 2.00000000, 23.00000000 2.00000000,
23.00000000 15.00000000, 2.00000000 15.00000000, 2.00000000
2.00000000))
```

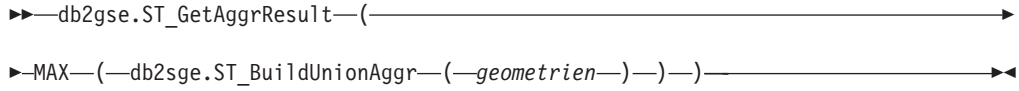
Union-Gesamtverknüpfungen

Eine Union-Gesamtverknüpfung von Geometrien ist die Kombination der Funktionen ST_BuildUnionAggr und ST_GetAggrResult. Mithilfe dieser Kombination können Sie eine Spalte mit Geometrien in einer Tabelle zu einer einzigen Geometrie zusammenfassen, indem die Union-Verknüpfung erstellt wird.

Wenn alle zu kombinierenden Geometrien in der Union-Verknüpfung den Wert null aufweisen, wird null zurückgegeben. Wenn alle der zu kombinierenden Geometrien in der Union-Verknüpfung entweder den Wert null aufweisen oder leer sind, wird eine leere Geometrie vom Typ ST_Point zurückgegeben.

Die Funktion ST_BuildUnionAggr kann auch als Methode aufgerufen werden.

Syntax



Parameter

Geometrien
Eine Spalte in einer Tabelle, die den Typ ST_Geometry oder einen seiner untergeordneten Typen aufweist und alle Geometrien darstellt, die in einer Union-Verknüpfung kombiniert werden sollen.

Rückgabetyp

db2gse.ST_Geometry

Einschränkungen

In den folgenden Situationen können Sie keine Union-Gesamtverknüpfung einer räumlichen Spalte in einer Tabelle erstellen:

- In einer Umgebungen mit partitionierten Datenbanken.
- Wenn eine Klausel GROUP BY in der Auswahlanweisung verwendet wird.
- Wenn Sie eine andere Funktion als die DB2-Spaltenfunktion MAX verwenden

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie eine Union-Gesamtverknüpfung zur Zusammenfassung einer Gruppe von Punkten zu einer Mehrpunktangabe verwendet werden kann. Der Tabelle SAMPLE_POINTS werden mehrere Punkte hinzugefügt. Die Funktionen ST_GetAggrResult und ST_BuildUnionAggr werden verwendet, um die Union-Verknüpfung der Punkte zu erstellen.

```
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
```

Ergebnisse:

```
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)
```

Funktion ST_Centroid

Die Funktion ST_Centroid verwendet eine Geometrie als Eingabeparameter und gibt deren geometrisches Zentrum, d. h. das Zentrum des minimal einschließenden Rechtecks (MBR) der angegebenen Geometrie, in Form eines Punktes zurück. Der Ergebnispunkt wird im räumlichen Bezugssystem der angegebenen Geometrie dargestellt.

Wenn die angegebene Geometrie den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, von der das geometrische Zentrum ermittelt werden soll.

Rückgabetyp

db2gse.ST_Point

Beispiel

In diesem Beispiel werden zwei Geometrien erstellt. Ferner wird der Mittelpunkt (Zentroid) dieser Geometrien gesucht.

```
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
```

Ergebnisse:

```
ID          CENTROID
----------- ----------------------------------------
                  1 POINT ( 65.00000000 135.00000000)
                  2 POINT ( 30.00000000 20.00000000)
```

Funktion ST_ChangePoint

Die Funktion ST_ChangePoint verwendet eine Kurve und zwei Punkte als Eingabeparameter. Diese Funktion ersetzt alle Vorkommen des ersten Punktes in der angegebenen Kurve durch den zweiten Punkt und gibt die Ergebniskurve zurück. Die Ergebnisgeometrie wird im räumlichen Bezugssystem der angegebenen Geometrie dargestellt.

Wenn die beiden Punkte nicht in demselben räumlichen Bezugssystem wie die Kurve dargestellt sind, werden sie in das räumliche Bezugssystem, das für die Kurve verwendet wird, umgewandelt.

Wenn die angegebenen Kurve leer ist, wird ein leerer Wert zurückgegeben. Wenn die angegebene Kurve den Wert null aufweist oder wenn einer der angegebenen Punkte den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

kurve Ein Wert vom Typ ST_Curve oder einer seiner untergeordneten Typen, der die Kurve darstellt, in der die durch *alter_Punkt* gekennzeichneten Punkte in *neuer_Punkt* geändert werden.

alter_Punkt
Ein Wert vom Typ ST_Point, der die Punkte in der Kurve kennzeichnet, die in *neuer_Punkt* geändert werden.

neuer_Punkt
Ein Wert vom Typ ST_Point, der die neuen Positionen der Punkte in der Kurve darstellt, die durch *alter_Punkt* gekennzeichnet sind.

Rückgabetyp

db2gse.ST_Curve

Einschränkungen

Der zu ändernde Punkt in der Kurve muss einer der Punkte sein, die zur Definition der Kurve verwendet wurden.

Wenn die Kurve Z- oder M-Koordinaten aufweist, müssen die angegebenen Punkte ebenfalls Z- oder M-Koordinaten aufweisen.

Beispiele

Beispiel 1

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Der folgende Code erstellt und füllt die Tabelle SAMPLE_LINES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines(id INTEGER, line ST_Linestring)


INSERT INTO sample_lines VALUES
       (1, ST_LineString('linestring (10 10, 5 5, 0 0, 10 0, 5 5, 0 10)', 0) )

INSERT INTO sample_lines VALUES
       (2, ST_LineString('linestring z (0 0 4, 5 5 5, 10 10 6, 5 5 7)', 0) )
```

Beispiel 2

In diesem Beispiel werden alle Vorkommen des Punktes (5, 5) in den Punkt (6, 6) in der Linienfolge geändert.

```
SELECT cast(ST_AsText(ST_ChangePoint(line, ST_Point(5, 5),
                                     ST_Point(6, 6))) as VARCHAR(160))
FROM   sample_lines
WHERE  id=1
```

Beispiel 3

In diesem Beispiel werden alle Vorkommen des Punktes (5, 5, 5) in den Punkt (6, 6, 6) in der Linienfolge geändert.

```
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
```

Ergebnisse:

```
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)
```

Funktion ST_Contains

Mit der Funktion ST_Contains können Sie ermitteln, ob eine Geometrie vollständig in einer anderen Geometrie enthalten ist.

Syntax

Parameter

geometrie1
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die daraufhin getestet wird, ob Sie die in *geometrie2* angegebene Geometrie vollständig enthält.

geometrie2
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die daraufhin getestet wird, ob sie vollständig in der in *geometrie1* angegebenen Geometrie enthalten ist.

Rückgabetyp

INTEGER

Verwendung

ST_Contains verwendet zwei Geometrien als Eingabeparameter und gibt 1 zurück, wenn die erste Geometrie die zweite Geometrie vollständig enthält. Andernfalls wird 0 (null) zurückgegeben, um anzuzeigen, dass die erste Geometrie die zweite Geometrie nicht vollständig enthält.

Die Funktion ST_Contains gibt genau das entgegengesetzte Ergebnis wie die Funktion ST_Within zurück.

Wenn eine der angegebenen Geometrien den Wert null aufweist oder leer ist, wird null zurückgegeben.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist und das gleiche zugrunde liegende Datum verwendet, wird sie in das andere räumliche Bezugssystem umgewandelt.

Die Mustermatrix der Funktion ST_Contains gibt an, dass sich die Innenbereiche der beiden Geometrien schneiden müssen und dass der Innenbereich oder die Begrenzung der zweiten Geometrie (Geometrie *b*) den Außenbereich der primären Geometrie (Geometrie *a*) nicht schneiden darf. Der Stern (*) gibt an, dass es nicht relevant ist, ob ein Schnittpunkt zwischen diesen Teilen der Geometrien vorhanden ist.

Tabelle 27. Matrix für ST_Contains

	Geometrie b Innenbereich	**Geometrie b Begrenzung**	**Geometrie b Außenbereich**
Geometrie a Innenbereich	T	*	*
Geometrie a Begrenzung	*	*	*
Geometrie a Außenbereich	F	F	*

Beispiele

Abb. 19 auf Seite 287 enthält Beispiele für ST_Contains:

- Eine Mehrpunktgeometrie enthält Punkt- oder Mehrpunktgeometrien, wenn alle Punkte innerhalb der ersten Geometrie liegen.
- Eine Polygongeometrie enthält eine Mehrpunktgeometrie, wenn alle vorhandenen Punkte entweder auf der Begrenzung des Polygons oder im Innenbereich des Polygons liegen.
- Eine Linienfolgengeometrie enthält Punkt-, Mehrpunkt- oder Linienfolgengeometrien, wenn alle Punkte innerhalb der ersten Geometrie liegen.
- Eine Polygongeometrie enthält Punkt-, Linienfolgen- oder Polygongeometrien, wenn die zweite Geometrie sich im Innenbereich des Polygons befindet.

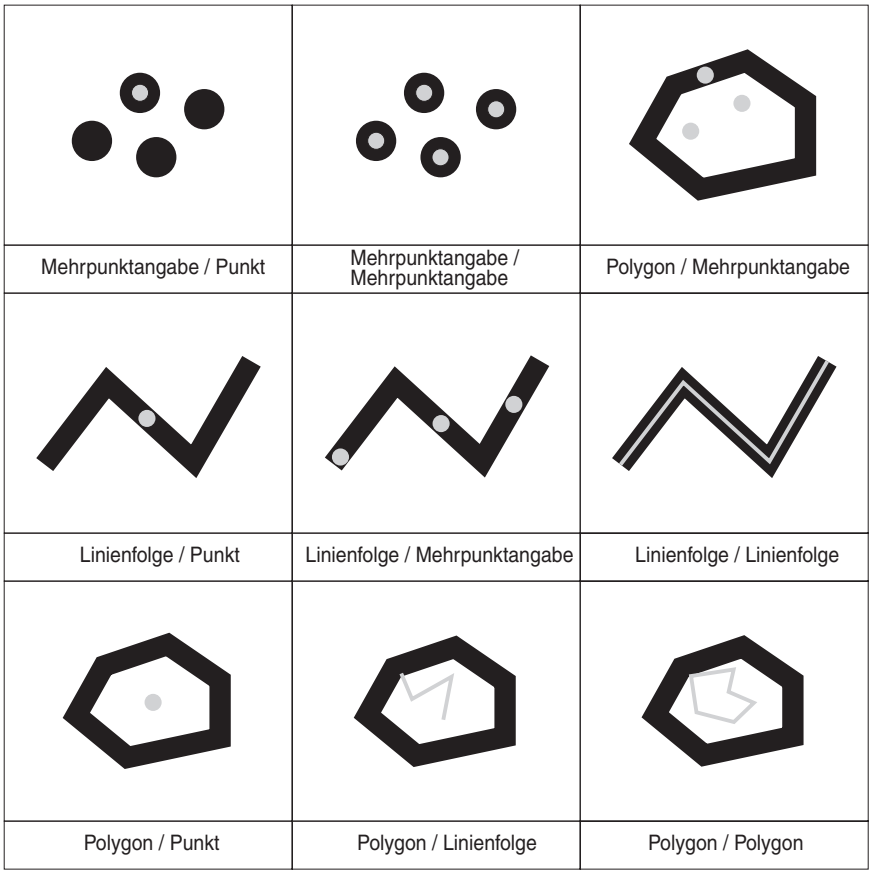


Abbildung 19. ST_Contains. Die dunklen Geometrien stellen die Geometrie a dar, die grauen Geometrien die Geometrie b. In allen Fällen ist Geometrie b vollständig in Geometrie a enthalten.

Beispiel 1

Mit dem folgenden Code werden diese Tabellen erstellt und gefüllt.

```
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) )
```

Beispiel 2

Das folgende Codefragment verwendet die Funktion ST_Contains, um zu ermitteln, welche Punkte in einem bestimmten Polygon enthalten sind.

```
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
```

Ergebnisse:

```
POLYGON_ID CONTAINS         POINT_ID
---------- ---------------- --------
              100 does contain            1
              100 does not contain        2
```

Beispiel 3

Das folgende Codefragment verwendet die Funktion ST_Contains um zu ermitteln, welche Linien in einem bestimmten Polygon enthalten sind.

```
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
```

Ergebnisse:

```
POLYGON_ID CONTAINS         LINE_ID
---------- ---------------- -------
              100 does contain          10
              100 does not contain      20
```

Funktion ST_ConvexHull

Die Funktion ST_ConvexHull verwendet eine Geometrie als Eingabeparameter und gibt die konvexe Hülle der Geometrie zurück.

Die Ergebnisgeometrie wird im räumlichen Bezugssystem der angegebenen Geometrie dargestellt.

Nach Möglichkeit ist der spezifische Typ der zurückgegebenen Geometrie ST-_Point, ST_LineString oder ST_Polygon. Die Grenze eines Polygons ohne Löcher ist zum Beispiel eine einzige Linienfolge, die als ST_LineString dargestellt wird. Die Grenze eines Polygons mit mindestens einem Loch besteht aus mehreren Linienfolgen, die als ST_MultiLineString dargestellt werden.

Wenn die angegebene Geometrie den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie zur Berechnung der konvexen Hülle darstellt.

Rückgabetyp

db2gse.ST_Geometry

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Mit dem folgenden Code wird die Tabelle SAMPLE_GEOMETRIES erstellt und gefüllt.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id INTEGER, spatial_type varchar(18),
       geometry ST_GEOMETRY)

INSERT INTO sample_geometries(id, spatial_type, geometry)
  VALUES
        (1, 'ST_LineString', ST_LineString
                ('linestring(20 20, 30 30, 20 40, 30 50)', 0)),
        (2, 'ST_Polygon', ST_Polygon('polygon
                ((110  120,  110 140, 120 130, 110 120))', 0) ),
        (3, 'ST_Polygon', ST_Polygon('polygon((30 30, 25 35, 15 50,
                35 80, 40 85, 80 90,70 75, 65 70, 55 50, 75 40, 60 30,
                30 30))', 0) ),
        (4, 'ST_MultiPoint', ST_MultiPoint('multipoint(20 20, 30 30,
                20 40, 30 50)', 1))
```

Die folgende Anweisung SELECT berechnet die konvexe Hülle für alle bereits konstruierten Geometrien und zeigt das Ergebnis an.

```
SELECT id, spatial_type, cast(geometry..ST_ConvexHull..ST_AsText
      AS varchar(300)) AS convexhull
FROM   sample_geometries
```

Ergebnisse:

```
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))
```

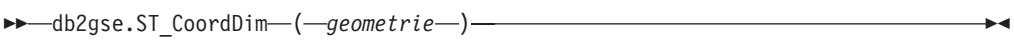
Funktion ST_CoordDim

Die Funktion ST_CoordDim verwendet eine Geometrie als Eingabeparameter und gibt die Dimensionalität ihrer Koordinaten zurück.

Wenn die angegebene Geometrie keine Z- oder M-Koordinaten aufweist, ist die Dimensionalität 2. Wenn die Geometrie Z-Koordinaten aber keine M-Koordinaten aufweist, ist die Dimensionalität 3. Wenn die Geometrie Z- und M-Koordinaten aufweist, ist die Dimensionalität 4. Wenn die Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, von der die Dimensionalität abgerufen werden soll.

Rückgabetyp

INTEGER

Beispiel

In diesem Beispiel werden mehrere Geometrien erstellt. Anschließend wird die Dimensionalität ihrer Koordinaten ermittelt.

```
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
```

Ergebnisse:

```
ID              COORDDIM
--------------- -----------
Empty Point               2
Linestring                2
Polygon                   2
Multipoint M              3
Multipoint Z              3
Point ZM                  4
```

Funktion ST_Crosses

Die Funktion ST_Crosses verwendet zwei Geometrien als Eingabeparameter und gibt 1 zurück, wenn die erste Geometrie die zweite Geometrie kreuzt. Andernfalls wird 0 (null) zurückgegeben.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist, wird sie in das andere räumliche Bezugssystem umgewandelt.

Wenn die erste Geometrie ein Polygon oder ein Multipolygon ist oder wenn die zweite Geometrie ein Punkt oder eine Mehrpunktangabe ist oder wenn eine der Geometrien den Wert null aufweist oder leer ist, wird null zurückgegeben. Wenn der Schnittpunkt der beiden Geometrien eine Geometrie zum Ergebnis hat, die eine Dimension weniger als die größte Dimension der beiden angegebenen Geometrien aufweist, und wenn die Ergebnisgeometrie nicht mit einer der beiden angegebenen Geometrien identisch ist, wird 1 zurückgegeben. Andernfalls wird 0 (null) zurückgegeben.

Syntax

Parameter

geometrie1
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die auf ein Kreuzen mit der in *geometrie2* angegebenen Geometrie getestet wird.

geometrie2
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die daraufhin getestet wird, ob sie von der in *geometrie1* angegebenen Geometrie gekreuzt wird.

Rückgabetyp

INTEGER

Beispiel

Mit diesem Code wird ermittelt, ob die konstruierten Geometrien einander kreuzen.

```
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
```

Ergebnisse:

```
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
```

Funktion ST_Difference

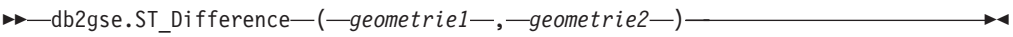
Die Funktion ST_Difference verwendet zwei Geometrien als Eingabeparameter und gibt den Teil der ersten Geometrie zurück, der keine Überschneidung mit der zweiten Geometrie aufweist.

Beide Geometrien müssen dieselbe Dimension aufweisen. Wenn eine der Geometrien den Wert null aufweist, wird null zurückgegeben. Wenn die erste Geometrie leer ist, wird eine leere Geometrie vom Typ ST_Point zurückgegeben. Wenn die zweite Geometrie leer ist, wird die erste Geometrie unverändert zurückgegeben.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist und das gleiche zugrunde liegende Datum verwendet, wird sie in das andere räumliche Bezugssystem umgewandelt.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie1
Ein Wert vom Typ ST_Geometry, der die erste Geometrie darstellt, die für die Berechnung der Differenz zu der in *geometrie2* angegebenen Geometrie verwendet wird.

geometrie2
Ein Wert vom Typ ST_Geometry, der die zweite Geometrie darstellt, die zur Berechnung der Differenz zu der in *geometrie1* angegebenen Geometrie verwendet wird.

Rückgabetyp

db2gse.ST_Geometry

Die Dimension der zurückgegebenen Geometrie stimmt mit der Dimension der Eingabegeometrien überein.

Beispiele

Im folgenden Beispiel wurden die Ergebnisse zur besseren Lesbarkeit erneut formatiert. Der Abstand in den Ergebnissen variiert entsprechend der jeweiligen Anzeige.

Mit dem folgenden Code wird die Tabelle SAMPLE_GEOMETRIES erstellt und gefüllt.

```
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))
```

Beispiel 1

In diesem Beispiel wird die Differenz zweier sich nicht schneidender Polygone gesucht.

```
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
```

Ergebnisse:

```
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))
```

Beispiel 2

In diesem Beispiel wird die Differenz zweier sich schneidender Polygone gesucht.

```
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
```

Ergebnisse:

```
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))
```

Beispiel 3

In diesem Beispiel wird die Differenz zweier sich überlappender Linienfolgen gesucht.

```
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
```

Ergebnisse:

```
ID       ID       DIFFERENCE
-------- -------- --------------------------------------------------
       4        5 LINESTRING ( 70.00000000 70.00000000, 75.00000000
                                75.00000000)
```

Funktion ST_Dimension

Die Funktion ST_Dimension verwendet eine Geometrie als Eingabeparameter und gibt ihre Dimension zurück.

Wenn die Geometrie leer ist, wird -1 zurückgegeben. Für Punkte und Mehrpunktangaben ist die Dimension 0 (Null). Für Kurven und Mehrfachkurven ist die Dimension 1. Für Polygone und Multipolygone ist die Dimension 2. Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry, der die Geometrie darstellt, für die die Dimension zurückgegeben wird.

Rückgabetyp

INTEGER

Beispiel

In diesem Beispiel werden mehrere unterschiedliche Geometrien erstellt und ihre Dimension ermittelt.

```
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
```

Ergebnisse:

```
ID              DIMENSION
--------------- -----------
Empty Point              -1
Point ZM                  0
MultiPoint M              0
LineString                1
Polygon                   2
```

Funktion ST_Disjoint

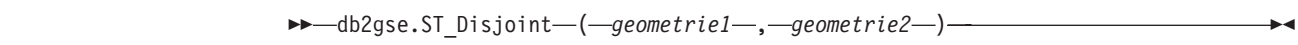
Die Funktion ST_Disjoint verwendet zwei Geometrien als Eingabeparameter und gibt 1 zurück, wenn die angegebenen Geometrien sich nicht schneiden. Wenn die Geometrien sich schneiden, wird 0 (null) zurückgegeben.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist, wird sie in das andere räumliche Bezugssystem umgewandelt.

Wenn eine der beiden Geometrien den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie1
Ein Wert vom Typ ST_Geometry, der die Geometrie darstellt, die auf Schnittmengen mit der in *geometrie2* angegebenen Geometrie getestet wird.

geometrie2
Ein Wert vom Typ ST_Geometry, der die Geometrie darstellt, die auf Schnittmengen mit der in *geometrie1* angegebenen Geometrie getestet wird.

Rückgabetyp

INTEGER

Beispiele

Beispiel 1

Mit diesem Code werden mehrere Geometrien in der Tabelle SAMPLE-_GEOMETRIES erstellt.

```
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))
```

Beispiel 2

In diesem Beispiel wird ermittelt, ob das erste Polygon eine Schnittmenge mit einer der Geometrien aufweist.

```
SELECT a.id, b.id, ST_Disjoint(a.geometry, b.geometry) DisJoint
FROM   sample_geoms a, sample_geoms b
WHERE a.id = 1
```

Ergebnisse:

```
ID          ID          DISJOINT
----------- ----------- -----------
          1           1           0
          1           2           0
          1           3           1
          1           4           1
          1           5           0
```

Beispiel 3

In diesem Beispiel wird ermittelt, ob das dritte Polygon eine Schnittmenge mit einer der Geometrien aufweist.

```
SELECT a.id, b.id, ST_Disjoint(a.geometry, b.geometry) DisJoint
FROM   sample_geoms a, sample_geoms b
WHERE a.id = 3
```

Ergebnisse:

```
ID          ID          DISJOINT
----------- ----------- -----------
          3           1           1
          3           2           0
          3           3           0
          3           4           0
          3           5           0
```

Beispiel 4

In diesem Beispiel wird ermittelt, ob die zweite Linienfolge eine Schnittmenge mit einer der Geometrien aufweist.

```
SELECT a.id, b.id, ST_Disjoint(a.geometry, b.geometry) DisJoint
FROM   sample_geoms a, sample_geoms b
WHERE a.id = 5
```

Ergebnisse:

```
ID          ID          DISJOINT
----------- ----------- -----------
          5           1           0
          5           2           0
          5           3           0
          5           4           1
          5           5           0
```

Funktion ST_Distance

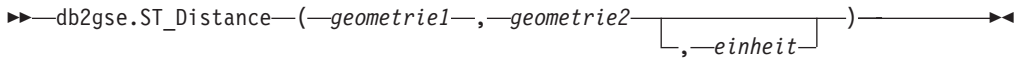
Die Funktion ST_Distance verwendet zwei Geometrien und optional eine Einheit als Eingabeparameter und gibt die kürzeste Distanz zwischen einem beliebigen Punkt in der ersten Geometrie und einem beliebigen Punkt in der zweiten Geometrie zurück, die in der Standardeinheit bzw. der angegebenen Einheit gemessen wird.

Wenn eine der beiden Geometrien den Wert null aufweist oder leer ist, wird null zurückgegeben.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist und das gleiche zugrunde liegende Datum verwendet, wird sie in das andere räumliche Bezugssystem umgewandelt.

Sie können diese Funktion auch als Methode aufrufen, wenn Sie eine Maßeinheit angeben.

Syntax



Parameter

geometrie1
Ein Wert vom Typ ST_Geometry, der die Geometrie darstellt, die zur Berechnung des Abstandes zu der in *geometrie2* angegebenen Geometrie verwendet wird.

geometrie2
Ein Wert vom Typ ST_Geometry, der die Geometrie darstellt, die zur Berechnung des Abstandes zu der in *geometrie1* angegebenen Geometrie verwendet wird.

einheit
Ein Wert vom Typ VARCHAR(128), der die Einheit kennzeichnet, in der das Ergebnis gemessen wird. Die unterstützten Maßeinheiten sind in der Katalogsicht DB2GSE.ST_UNITS_OF_MEASURE aufgelistet.

Wenn der Parameter *einheit* ausgelassen wird, werden die folgenden Regeln verwendet, um die Maßeinheit für das Ergebnis zu ermitteln:

- Wenn die in *geometrie1* angegebene Geometrie sich in einem projizierten oder geozentrischen Koordinatensystem befindet, wird als Standardwert die lineare Einheit verwendet, die diesem Koordinatensystem zugeordnet ist.
- Wenn die *geometrie1* sich in einem geografischen Koordinatensystem befindet, wird als Standardwert die Winkeleinheit verwendet, die diesem Koordinatensystem zugeordnet ist.

Einschränkungen bei Einheitenumsetzungen: Ein Fehler (SQLSTATE 38SU4) wird zurückgegeben, wenn eine der folgenden Bedingungen zutrifft:

- Die Geometrie befindet sich in einem nicht definierten Koordinatensystem, und der Parameter *einheit* wurde angegeben.
- Die Geometrie befindet sich in einem projizierten Koordinatensystem, und es wurde eine Winkeleinheit angegeben.

Rückgabetyp

DOUBLE

Beispiele

Beispiel 1

Mit den folgenden SQL-Anweisungen werden die Tabellen SAMPLE_GEOMETRIES1 und SAMPLE_GEOMETRIES2 erstellt und gefüllt.

```
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) )
```

Beispiel 2

Die folgende Anweisung SELECT berechnet den Abstand zwischen verschiedenen Geometrien in den Tabellen SAMPLE_GEOMTRIES1 und SAMPLE_GEOMTRIES2

```
SELECT   sg1.id AS sg1_id, sg1.spatial_type AS sg1_type,
                  sg2.id AS sg1_id, sg2.spatial_type AS sg2_type,
                  cast(ST_Distance(sg1.geometry, sg2.geometry)
                  AS Decimal(8, 4)) AS distance
FROM     sample_geometries1 sg1, sample_geometries2 sg2
ORDER BY sg1.id
```

Ergebnisse:

```
SG1_ID SG1_TYPE      SG1_ID SG2_TYPE      DISTANCE
------ ------------- ------ ------------- ----------
     1 ST_Point         101 ST_Point        141.4213
     1 ST_Point         102 ST_Point        223.6067
     1 ST_Point         103 ST_Point        141.4213
     1 ST_Point         110 ST_LineString   100.0000
     1 ST_Point         120 ST_Polygon      100.0000
    10 ST_LineString    101 ST_Point         79.0569
    10 ST_LineString    102 ST_Point        145.7737
    10 ST_LineString    103 ST_Point        145.7737
    10 ST_LineString    110 ST_LineString    75.0000
    10 ST_LineString    120 ST_Polygon       75.0000
    20 ST_Polygon       101 ST_Point         70.7106
    20 ST_Polygon       102 ST_Point        158.1138
```

```
    20 ST_Polygon       103 ST_Point         70.7106
    20 ST_Polygon       110 ST_LineString    50.0000
    20 ST_Polygon       120 ST_Polygon       50.0000
```

Beispiel 3

Die folgende Anweisung SELECT verdeutlicht, wie Sie alle Geometrien finden, die sich in einem Abstand von bis zu 100 voneinander befinden.

```
SELECT   sg1.id AS sg1_id, sg1.spatial_type AS sg1_type,
                 sg2.id AS sg1_id, sg2.spatial_type AS sg2_type,
                 cast(ST_Distance(sg1.geometry, sg2.geometry)
                 AS Decimal(8, 4)) AS distance
FROM     sample_geometries1 sg1, sample_geometries2 sg2
WHERE ST_Distance(sg1.geometry, sg2.geometry) <= 100
```

Ergebnisse:

```
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
```

Beispiel 4

Die folgende Anweisung SELECT berechnet den Abstand in Kilometern zwischen den verschiedenen Geometrien.

```
SAMPLE_GEOMTRIES1 and SAMPLE_GEOMTRIES2 tables.
SELECT   sg1.id AS sg1_id, sg1.spatial_type AS sg1_type,
                 sg2.id AS sg1_id, sg2.spatial_type AS sg2_type,
                 cast(ST_Distance(sg1.geometry, sg2.geometry, 'KILOMETER')
                 AS DECIMAL(10, 4)) AS distance
FROM     sample_geometries1 sg1, sample_geometries2 sg2
ORDER BY sg1.id
```

Ergebnisse:

```
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
```

Funktion ST_DistanceToPoint

Die Funktion ST_DistanceToPoint verwendet eine Kurvengeometrie oder eine Geometrie mit mehreren Kurven und eine Punktgeometrie als Eingabeparameter und gibt den Abstand zum angegebenen Punkt entlang der Kurvengeometrie zurück.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

kurvengeometrie
Ein Wert vom Typ ST_Curve oder ST_MultiCurve oder einer seiner untergeordneten Typen, der die zu verarbeitende Geometrie darstellt.

punktgeometrie
Ein Wert vom Typ ST_Point, der einen Punkt entlang der angegebenen Kurve darstellt.

Rückgabetyp

DOUBLE

Beispiel

Mit der folgenden SQL-Anweisung wird die Tabelle SAMPLE_GEOMETRIES mit zwei Spalten erstellt. Die Spalte 'ID' identifiziert jede Zeile eindeutig. In der Spalte 'GEOMETRY ST_LineString' werden Mustergeometrien gespeichert.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries(id INTEGER, geometry ST_LINESTRING)
```

Mit der folgenden SQL-Anweisung werden zwei Zeilen in die Tabelle SAMPLE_GEOMETRIES eingefügt.

```
INSERT INTO sample_geometries(id, geometry)
  VALUES
 (1,ST_LineString('LINESTRING ZM(0 0 0 0, 10 100 1000 10000)',1)),
 (2,ST_LineString('LINESTRING ZM(10 100 1000 10000, 0 0 0 0)',1))
```

Die folgende Anweisung SELECT und die entsprechende Ergebnismenge zeigen, wie die Funktion ST_DistanceToPoint dazu verwendet werden kann, den Abstand zum Punkt an der angegebenen Position zu ermitteln (1.5, 15.0).

```
SELECT ID, DECIMAL(ST_DistanceToPoint(geometry,ST_Point(1.5,15.0,1)),10,5) AS DISTANCE
FROM sample_geometries

ID          DISTANCE
----------- ------------
          1     15.07481
          2     85.42394

  2 record(s) selected.
```

Funktion ST_Endpoint

Die Funktion ST_Endpoint verwendet eine Kurve als Eingabeparameter und gibt den letzten Punkt (Endpunkt) der Kurve zurück. Der Ergebnispunkt wird im räumlichen Bezugssystem der angegebenen Kurve dargestellt.

Wenn die angegebene Kurve den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

kurve Ein Wert vom Typ ST_Curve, der die Geometrie darstellt, von der der letzte Punkt zurückgegeben wird.

Rückgabetyp

db2gse.ST_Point

Beispiel

Die Anweisung SELECT sucht den Endpunkt jeder Geometrie in der Tabelle SAMPLE_LINES.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines(id INTEGER, line ST_Linestring)


INSERT INTO sample_lines VALUES
        (1, ST_LineString('linestring (10 10, 5 5, 0 0, 10 0, 5 5, 0 10)', 0) )

INSERT INTO sample_lines VALUES
        (2, ST_LineString('linestring z (0 0 4, 5 5 5, 10 10 6, 5 5 7)', 0) )


SELECT id, CAST(ST_AsText(ST_EndPoint(line)) as VARCHAR(50)) Endpoint
FROM   sample_lines
```

Ergebnisse:

```
ID          ENDPOINT
----------- --------------------------------------------------
          1 POINT ( 0.00000000 10.00000000)
          2 POINT Z ( 5.00000000 5.00000000 7.00000000)
```

Funktion ST_Envelope

Die Funktion ST_Envelope verwendet eine Geometrie als Eingabeparameter und gibt eine Hülle um die Geometrie zurück. Die Hülle ist ein Rechteck, das als Polygon dargestellt wird.

Wenn die angegebene Geometrie ein Punkt, eine horizontale Linienfolge oder eine vertikale Linienfolge ist, wird ein Rechteck zurückgegeben, das etwas größer als die angegebene Geometrie ist. Andernfalls wird das minimal einschließende Recht-

eck der Geometrie als Hülle zurückgegeben. Wenn die angegebene Geometrie den Wert null aufweist oder leer ist, wird null zurückgegeben. Um das minimal einschließende Rechteck für alle Geometrien exakt zurückzugeben, verwenden Sie die Funktion ST_MBR.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry, der die Geometrie darstellt, für die die Hülle zurückgegeben wird.

Rückgabetyp

db2gse.ST_Polygon

Beispiel

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

In diesem Beispiel werden mehrere Geometrien erstellt. Anschließend werden die Umschläge dieser Geometrien ermittelt. Für den nicht leeren Punkt und die (horizontale) Linienfolge ist die Hülle ein Rechteck, das etwas größer als die Geometrie ist.

```
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
```

Ergebnisse:

```
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))
```

Funktion ST_EnvIntersects

Die Funktion ST_EnvIntersects verwendet zwei Geometrien als Eingabeparameter und gibt 1 zurück, wenn sich die Hüllen der beiden Geometrien schneiden. Andernfalls wird 0 (null) zurückgegeben.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist, wird sie in das andere räumliche Bezugssystem umgewandelt.

Wenn eine der angegebenen Geometrien den Wert null aufweist oder leer ist, wird der Wert null zurückgegeben.

Syntax

Parameter

geometrie1
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, deren Hülle auf Schnittpunkte mit der Hülle der in *geometrie2* angegebenen Geometrie getestet wird.

geometrie2
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, deren Hülle auf Schnittpunkte mit der Hülle der in *geometrie1* angegebenen Geometrie getestet wird.

Rückgabetyp

INTEGER

Beispiel

In diesem Beispiel werden zwei parallele Linienfolgen erstellt. Ferner wird geprüft, ob diese Linienfolgen sich schneiden. Die Linienfolgen selbst schneiden sich nicht, die Umschläge der Linienfolgen hingegen schneiden sich.

```
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
```

Ergebnisse:

```
ID          ID          INTERSECTS  ENVELOPE_INTERSECTS
----------- ----------- ----------- -------------------
          1           2           0                   1
```

Funktion ST_EqualCoordsys

Die Funktion ST_EqualCoordsys verwendet zwei Koordinatensystemdefinitionen als Eingabeparameter und gibt den Integerwert 1 (eins) zurück, wenn die angegebenen Definitionen identisch sind. Andernfalls wird der Integerwert 0 (null) zurückgegeben.

Die Koordinatensystemdefinitionen werden unabhängig von Differenzen bei Leerzeichen, runden Klammern, Großschreibung und Kleinschreibung und der Darstellung der Gleitkommazahlen verglichen.

Wenn eine der angegebenen Koordinatensystemdefinitionen den Wert null aufweist, wird null zurückgegeben.

Syntax

Parameter

koordinatensystem1
Ein Wert vom Typ VARCHAR(2048), der das erste Koordinatensystem definiert, das mit dem in *koordinatensystem2* angegebenen Koordinatensystem verglichen werden soll.

koordinatensystem2
Ein Wert vom Typ VARCHAR(2048), der das zweite Koordinatensystem definiert, das mit dem in *koordinatensystem1* angegebenen Koordinatensystem verglichen werden soll.

Rückgabetyp

INTEGER

Beispiel

In diesem Beispiel werden zwei australische Koordinatensysteme verglichen, um festzustellen, ob sie identisch sind.

```
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')
    )
```

Ergebnisse:

```
1
-----------
          0
```

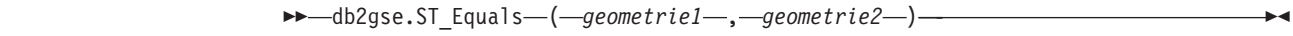
Funktion ST_Equals

Die Funktion ST_Equals verwendet zwei Geometrien als Eingabeparameter und gibt 1 zurück, wenn die Geometrien identisch sind. Andernfalls wird 0 (null) zurückgegeben. Die Reihenfolge der für die Definition der Geometrie verwendeten Punkte ist für die Überprüfung der Gleichheit beider Geometrien nicht von Bedeutung.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist, wird sie in das andere räumliche Bezugssystem umgewandelt.

Wenn eine der beiden angegebenen Geometrien den Wert null aufweist, wird null zurückgegeben.

Syntax



Parameter

geometrie1
Ein Wert vom Typ ST_Geometry, der die Geometrie darstellt, die mit der in *geometrie2* angegebenen Geometrie verglichen werden soll.

geometrie2
Ein Wert vom Typ ST_Geometry, der die Geometrie darstellt, die mit der in *geometrie1* angegebenen Geometrie verglichen werden soll.

Rückgabetyp

INTEGER

Beispiele

Beispiel 1

In diesem Beispiel werden zwei Polygone erstellt, deren Koordinaten sich in unterschiedlicher Reihenfolge befinden. ST_Equal wird verwendet, um zu zeigen, dass diese Polygone als identisch betrachtet werden.

```
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
```

Ergebnisse:

```
ID          ID          EQUALS
----------- ----------- -----------
          1           2           1
```

Beispiel 2

In diesem Beispiel werden zwei Geometrien mit denselben X- und Y-Koordinaten, jedoch mit unterschiedlichen M-Koordinaten (Bemaßungen) erstellt. Wenn die Geometrien mit der Funktion ST_Equal verglichen werden, wird eine 0 (null) zurückgegeben, um anzuzeigen, dass diese Geometrien nicht identisch sind.

```
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
```

Ergebnisse:

```
ID          ID          EQUALS
----------- ----------- -----------
          3           4           0
```

Beispiel 3

In diesem Beispiel werden zwei Geometrien mit einer unterschiedlichen Gruppe von Koordinaten erstellt, die jedoch beide dieselbe Geometrie darstellen. ST_Equal vergleicht die Geometrien und zeigt an, dass beide Geometrien tatsächlich identisch sind.

```
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
```

Ergebnisse:

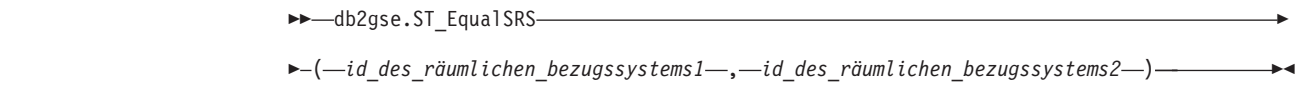
```
ID          ID          EQUALS
----------- ----------- -----------
          5           6           1
```

Funktion ST_EqualSRS

Die Funktion ST_EqualSRS verwendet zwei räumliche Bezugssystem-IDs als Eingabeparameter und gibt 1 zurück, wenn die angegebenen räumlichen Bezugssysteme identisch sind. Andernfalls wird 0 (null) zurückgegeben. Die Abstände, die Maßstabsfaktoren und die Koordinatensysteme werden verglichen.

Wenn eine der angegebenen Kennungen für das räumliche Bezugssystem den Wert null aufweist, wird null zurückgegeben.

Syntax



Parameter

id_des_räumlichen_bezugssystems1
Ein Wert vom Typ INTEGER, der das erste räumliche Bezugssystem kennzeichnet, das mit dem räumlichen Bezugssystem verglichen werden soll, das in *id_des_räumlichen_bezugssystems2* angegeben ist.

id_des_räumlichen_bezugssystems2
Ein Wert vom Typ INTEGER, der das zweite räumliche Bezugssystem kennzeichnet, das mit dem räumlichen Bezugssystem verglichen werden soll, das in *id_des_räumlichen_bezugssystems1* angegeben ist.

Rückgabetyp

INTEGER

Beispiel

Zwei ähnliche räumliche Bezugssysteme werden mit folgendem Aufruf von db2se erstellt.

```
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
```

Diese räumlichen Bezugssysteme weisen dieselben Offsetwerte und Maßstabsfaktoren und beziehen sich auf dasselbe Koordinatensystem. Der einzige Unterschied besteht im Namen und der Kennung des räumlichen Bezugssystems. Daher gibt der Vergleich 1 zurück, um anzuzeigen, dass die räumlichen Bezugssysteme identisch sind.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

VALUES ST_EqualSRS(12, 22)
```

Ergebnisse:

```
1
-----------
          1
```

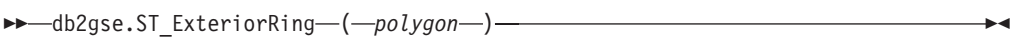
Funktion ST_ExteriorRing

Die Funktion ST_ExteriorRing verwendet als Eingabeparameter ein Polygon und gibt dessen äußeren Ring als Kurve zurück. Die Ergebniskurve wird im räumlichen Bezugssystem des angegebenen Polygons dargestellt.

Wenn das angegebene Polygon den Wert null aufweist oder leer ist, wird null zurückgegeben. Wenn das Polygon keine inneren Ringe aufweist, ist der zurückgegebene äußere Ring mit der Begrenzung des Polygons identisch.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

polygon
Ein Wert vom Typ ST_Polygon, der das Polygon darstellt, dessen äußerer Ring zurückgegeben werden soll.

Rückgabetyp

db2gse.ST_Curve

Beispiel

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

In diesem Beispiel werden zwei Polygone erstellt, wobei eines zwei innere Ringe und eines keinen inneren Ring aufweist. Anschließend werden die äußeren Ringe ermittelt.

```
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
```

Ergebnisse:

```
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)
```

Funktion ST_FindMeasure oder ST_LocateAlong

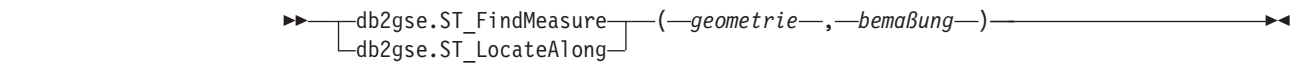
Die Funktion ST_FindMeasure oder ST_LocateAlong verwendet eine Geometrie und eine Bemaßung als Eingabeparameter und gibt eine Mehrpunktangabe oder Mehrfachkurve des Teils der angegebenen Geometrie zurück, der genau der angegebenen Bemaßung der angegebenen Geometrie entspricht, die die angegebene Bemaßung enthält.

Bei Punkten und Mehrpunktangaben werden alle Punkte mit der angegebenen Bemaßung zurückgegeben. Bei Kurven, Mehrfachkurven, Oberflächen und Mehrfachoberflächen wird zur Ergebnisberechnung eine Interpolation ausgeführt. Die Berechnung für Oberflächen und Mehrfachoberflächen wird für die Begrenzung der Geometrie ausgeführt.

Für Punkte und Mehrpunktangaben wird eine leere Geometrie zurückgegeben, wenn die angegebene Bemaßung nicht gefunden wurde. Für alle anderen Geometrien wird eine leere Geometrie zurückgegeben, wenn die angegebene Bemaßung kleiner als die kleinste Bemaßung oder größer als die größte Bemaßung in der Geometrie ist. Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, in der nach Teilen gesucht wird, deren M-Koordinaten (Bemaßungen) die in *bemaßung* angegebene Bemaßung enthalten.

bemaßung
Ein Wert vom Typ DOUBLE, der die Bemaßung angibt, in der die Teile der in *geometrie* angegebenen Geometrie im Ergebnis eingeschlossen sein müssen.

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Beispiel 1

Die folgende Anweisung CREATE TABLE erstellt die Tabelle SAMPLE_GEOMETRIES. Die Tabelle SAMPLE_GEOMETRIES verfügt über zwei Spalten: Die Spalte ID, die jede Zeile eindeutig kennzeichnet, und die Spalte GEOMETRY, die die Mustergeometrie speichert.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id SMALLINT, geometry ST_GEOMETRY)
```

Die folgenden INSERT-Anweisungen fügen zwei Zeilen ein. Die erste enthält eine Linienfolge die zweite eine Mehrpunktangabe.

```
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))
```

Beispiel 2

In der folgenden Anweisung SELECT und der entsprechenden Ergebnismenge wird die Funktion angewiesen, die Punkte zu suchen, deren Bemaßung 7 beträgt. Die erste Zeile gibt einen Punkt zurück. Die zweite Zeile gibt jedoch einen leeren Punkt zurück. Für lineare Funktionen (Geometrie mit einer Dimension größer 0) kann ST_FindMeasure den Punkt interpolieren: Bei Mehrpunktangaben muss das Zielmaß jedoch exakt übereinstimmen.

```
SELECT id, cast(ST_AsText(ST_FindMeasure(geometry, 7))
      AS varchar(45)) AS measure_7
FROM   sample_geometries
```

Ergebnisse:

```
ID     MEASURE_7
------ ---------------------------------------------
     1 POINT M ( 3.50000000 3.50000000 7.00000000)
     2 POINT EMPTY
```

Beispiel 3

In der folgenden Anweisung SELECT und der entsprechenden Ergebnismenge gibt die Funktion ST_FindMeasure einen Punkt und eine Mehrpunktangabe zurück. Die Zielbemaßung von 6 entspricht den Bemaßungen in den Quelldaten von ST_FindMeasure und der Mehrpunktangabe.

```
SELECT id, cast(ST_AsText(ST_FindMeasure(geometry, 6))
      AS varchar(120)) AS measure_6
FROM   sample_geometries
```

Ergebnisse:

```
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)
```

Funktion ST_Generalize

Die Funktion ST_Generalize verwendet eine Geometrie und einen Schwellenwert als Eingabeparameter und stellt die angegebene Geometrie mit einer reduzierten Anzahl an Punkten dar, wobei die allgemeinen Eigenschaften der Geometrie erhalten bleiben.

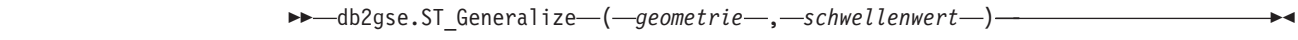
Dabei wird der Algorithmus zur Linienvereinfachung nach Douglas-Peucker verwendet, bei dem die Reihenfolge der Punkte, die die Geometrie definieren, rekursiv unterteilt wird, bis eine Folge von Punkten durch gerade Liniensegmente ersetzt werden kann. In diesem Liniensegment weicht keiner der definierenden Punkte um einen größeren als den angegebenen Schwellenwert von dem geraden

Liniensegment ab. Z- und M-Koordinaten werden bei der Vereinfachung nicht berücksichtigt. Die Ergebnisgeometrie befindet sich im räumlichen Bezugssystem der angegebenen Geometrie.

Wenn die angegebene Geometrie leer ist, wird eine leere Geometrie vom Typ ST_Point zurückgegeben. Wenn die angegebene Geometrie oder der Schwellenwert den Wert null aufweist, wird 0 (null) zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, für die die Linienvereinfachung angewendet wird.

schwellenwert
Ein Wert vom Typ DOUBLE, der den Schwellenwert angibt, der für den Algorithmus der Linienvereinfachung verwendet werden soll. Der Schwellenwert muss größer oder gleich 0 (null) sein. Je größer der Schwellenwert ist, desto kleiner ist die Anzahl der Punkte, die für die Darstellung der verallgemeinerten Geometrie verwendet wird.

Rückgabetyp

db2gse.ST_Geometry

Beispiele

In den folgenden Beispielen wurden die Ergebnisse zur besseren Lesbarkeit umformatiert. Der Abstand in den Ergebnissen variiert entsprechend der jeweiligen Anzeige.

Beispiel 1

Eine Linienfolge wird mit acht Punkten erstellt, die zwischen (10, 10) und (80, 80) liegen. Der Pfad ist nahezu eine gerade Linie, einige Punkte liegen jedoch etwas neben dieser Linie. Die Funktion ST_Generalize kann verwendet werden, um die Anzahl der Punkte in der Linie zu verringern.

```
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))
```

Beispiel 2

Wenn ein Generalisierungsfaktor von 3 verwendet wird, wird die Linienfolge auf 4 Koordinaten reduziert und sieht der ursprünglichen Darstellung der Linienfolge noch immer sehr ähnlich.

```
SELECT CAST(ST_AsText(ST_Generalize(geometry, 3)) as VARCHAR(115))
    Generalize_3
FROM   sample_lines
```

Ergebnisse:

```
GENERALIZE 3
------------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 34.00000000 26.00000000,
    59.00000000  63.00000000, 80.00000000 80.00000000)
```

Beispiel 3

Wenn ein Generalisierungsfaktor von 6 verwendet wird, wird die Linienfolge auf nur zwei Koordinaten verringert. Dadurch entsteht eine einfachere Linienfolge als im vorherigen Beispiel. Diese weicht jedoch mehr von der ursprünglichen Darstellung ab.

```
SELECT CAST(ST_AsText(ST_Generalize(geometry, 6)) as VARCHAR(65))
    Generalize_6
FROM   sample_lines
```

Ergebnisse:

```
GENERALIZE 6
------------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 80.00000000 80.00000000)
```

Funktion ST_GeomCollection

Mit der Funktion ST_GeomCollection können Sie eine Geometriengruppe erstellen.

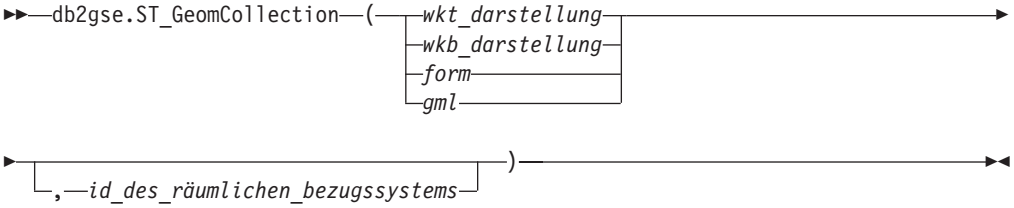
ST_GeomCollection konstruiert eine Geometriengruppe aus einer der folgenden Eingaben:

- WKT-Darstellung
- WKB-Darstellung
- ESRI-Formdarstellung
- Darstellung in GML (Geography Markup Language)

Eine Kennung für räumliche Bezugssysteme kann optional angegeben werden, um das räumliche Bezugssystem zu kennzeichnen, in dem sich die Ergebnisgeometriengruppe befindet.

Wenn die WKT-, die WKB-, die ESRI-Formdarstellung oder die GML-Darstellung den Wert null aufweisen, wird null zurückgegeben.

Syntax



Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung (WKT = Well-Known Text) der Ergebnisgeometriengruppe enthält.

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung (WKB = Well-Known Binary) der Ergebnisgeometriengruppe enthält.

form Ein Wert vom Typ BLOB(2G), der die ESRI-Formdarstellung der Ergebnisgeometriengruppe darstellt.

gml Ein Wert vom Typ CLOB(2G), der die Ergebnisgeometriengruppe unter Verwendung von GML darstellt.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für die Ergebnisgeometriengruppe darstellt.

Wenn der Parameter *id_des_räumlichen_bezugssystems* ausgelassen wird, verwendet das System implizit das räumliche Bezugssystem mit der numerischen Kennung 0 (null).

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben ist, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet wird, gibt das System einen Fehler (SQLSTATE 38SU1) zurück.

Rückgabetyp

db2gse.ST_GeomCollection

Hinweise

Wenn der Parameter *id_des_räumlichen_bezugssystems* ausgelassen wird, ist es möglicherweise notwendig, die Werte für *wkt_darstellung* und *GML* explizit in den Datentyp CLOB umzusetzen (Anweisung CAST). Andernfalls löst DB2 möglicherweise zur Funktion auf, die zur Umsetzung von Werten des Verweistyps REF(ST_GeomCollection) in den Typ ST_GeomCollection verwendet wird. Im folgenden Beispiel wird sichergestellt, dass DB2 zur richtigen Funktion auflöst:

Beispiel

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Der folgende Code verdeutlicht, wie die Funktion ST_GeomCollection verwendet werden kann, um eine Mehrpunktangabe, eine Mehrlinienfolge und ein Multipolygon aus einer WKT-Darstellung (WKT = Well-Known Text) und eine Mehrpunktangabe aus einer GML-Darstellung (GML = Geography Markup Language) zu erstellen und in die Spalte GeomCollection einzufügen.

```
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
```

Ergebnisse:

```
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)
```

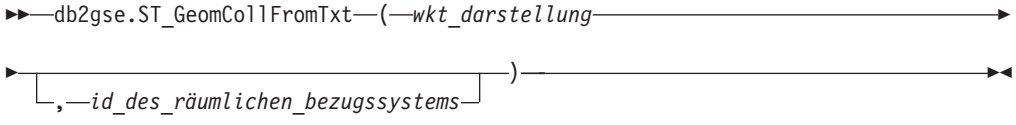
Funktion ST_GeomCollFromTxt

Die Funktion ST_GeomCollFromTxt verwendet eine WKT-Darstellung (WKT = Well-Known Text) einer Geometriengruppe und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt die entsprechende Geometriengruppe zurück.

Wenn die angegebene WKT-Darstellung den Wert null aufweist, wird null zurückgegeben.

Um dasselbe Ergebnis zu erreichen, wird die Funktion ST_GeomCollection empfohlen. Sie wird wegen ihrer Flexibilität empfohlen: ST_GeomCollection verwendet neben der WKB-Darstellung (WKB = Well-Known Binary) zusätzliche Formen der Eingabe.

Syntax



Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung (WKT = Well-Known Text) der Ergebnisgeometriengruppe enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für die Ergebnisgeometriengruppe darstellt.

Wenn der Parameter *id_des_räumlichen_bezugssystems* ausgelassen wird, verwendet das System implizit das räumliche Bezugssystem mit der numerischen Kennung 0 (null).

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben ist, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet wird, gibt das System einen Fehler (SQLSTATE 38SU1) zurück.

Rückgabetyp

db2gse.ST_GeomCollection

Beispiel

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Der folgende Code verdeutlicht, wie die Funktion ST_GeomCollFromTxt verwendet werden kann, um eine Mehrpunktangabe, eine Mehrlinienfolge und ein Multipolygon aus einer WKT-Darstellung (WKT = Well-Known Text) zu erstellen und in die Spalte GeomCollection einzufügen.

```
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
```

Ergebnisse:

```
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)))
```

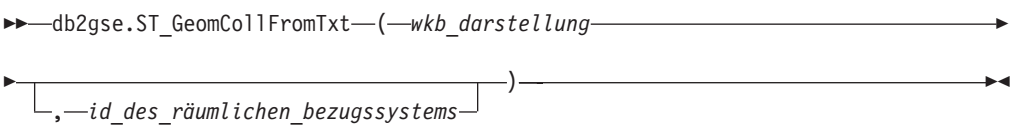
Funktion ST_GeomCollFromWKB

Die Funktion ST_GeomCollFromWKB verwendet eine WKB-Darstellung (WKB = Well-Known Binary) einer Geometriengruppe und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt die entsprechende Geometriengruppe zurück.

Wenn die angegebene WKB-Darstellung den Wert null aufweist, wird null zurückgegeben.

Die bevorzugte Version dieser Funktionalität ist ST_GeomCollection.

Syntax



Parameter

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung (WKB = Well-Known Binary) der Ergebnisgeometriengruppe enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für die Ergebnisgeometriengruppe darstellt.

Wenn der Parameter *id_des_räumlichen_bezugssystems* ausgelassen wird, verwendet das System implizit das räumliche Bezugssystem mit der numerischen Kennung 0 (null).

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben ist, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet wird, gibt das System einen Fehler (SQLSTATE 38SU1) zurück.

Rückgabetyp

db2gse.ST_GeomCollection

Beispiel

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Der folgende Code verdeutlicht, wie die Funktion ST_GeomCollFromWKB verwendet werden kann, um die Koordinaten einer Geometriengruppe in einer WKB-Darstellung (WKB = Well-Known Binary) zu erstellen und abzufragen. Die Zeilen wer-

den in die Tabelle SAMPLE_GEOMCOLLECTION mit den IDs 4021 und 4022 und Geometriengruppen werden in das räumliche Bezugssystem 1 eingefügt.

```
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
```

Ergebnisse:

```
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))
```

Funktion ST_Geometry

Die Funktion ST_Geometry konstruiert eine Geometrie aus einer angegebenen Darstellung.

ST_Geometry konstruiert aus einer der folgenden Eingaben eine Geometrie:

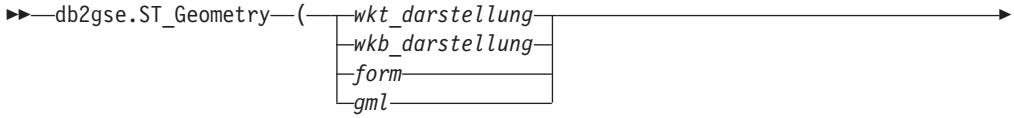
- WKT-Darstellung
- WKB-Darstellung
- ESRI-Formdarstellung
- Darstellung in GML (Geography Markup Language)

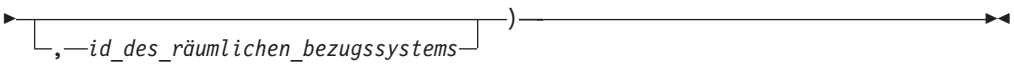
Eine Kennung für räumliche Bezugssysteme kann optional angegeben werden, um das räumliche Bezugssystem zu kennzeichnen, in dem sich die Ergebnisgeometrie befindet.

Der dynamische Typ der Ergebnisgeometrie ist einer der konkret belegbaren untergeordneten Typen von ST_Geometry.

Wenn die WKT-, die WKB-, die ESRI-Formdarstellung oder die GML-Darstellung den Wert null aufweisen, wird null zurückgegeben.

Syntax





Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung der Ergebnisgeometrie enthält.

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung der Ergebnisgeometrie enthält.

form Ein Wert vom Typ BLOB(2G), der die ESRI-Formdarstellung der Ergebnisgeometrie darstellt.

gml Ein Wert vom Typ CLOB(2G), der die Ergebnisgeometrie unter Verwendung von GML (Geography Markup Language) darstellt.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für die Ergebnisgeometrie kennzeichnet.

Wenn der Parameter *id_des_räumlichen_bezugssystems* ausgelassen wird, verwendet das System implizit das räumliche Bezugssystem mit der numerischen Kennung 0 (null).

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben ist, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet wird, gibt das System einen Fehler (SQLSTATE 38SU1) zurück.

Rückgabetyp

db2gse.ST_Geometry

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Der folgende Code verdeutlicht, wie die Funktion ST_Geometry verwendet werden kann, um einen Punkt aus einer WKT-Darstellung (WKT = Well-Known Text) des Punktes oder um eine Linie aus einer GML-Liniendarstellung (GML = Geography Markup Language) zu erstellen und einzufügen.

Die Funktion ST_Geometry ist die flexibelste Funktion der Konstruktionsfunktionen vom räumlichen Typ, da sie aus verschiedenen Geometriedarstellungen jeden räumlichen Typ erstellen kann. ST_LineFromText kann aus einer WKT-Darstellung (WKT = Well-Known Text) einer Linie nur eine Linie erstellen. ST_WKTToSql kann jeden Typ erstellen, jedoch nur aus einer WKT-Darstellung.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id INTEGER, geometry ST_GEOMETRY)

INSERT INTO sample_geometries(id, geometry)
  VALUES
        (7001, ST_Geometry('point(1 2)', 1) ),
        (7002, ST_Geometry('linestring(33 2, 34 3, 35 6)', 1) ),
```

```
        (7003, ST_Geometry('polygon((3 3, 4 6, 5 3, 3 3))', 1)),
        (7004, ST_Geometry('<gml:Point srsName=";EPSG:4269";><gml:coord>
         <gml:X>50</gml:X><gml:Y>60</gml:Y></gml:coord>
                     </gml:Point>', 1))

SELECT id, cast(geometry..ST_AsText AS varchar(120)) AS geometry
FROM   sample_geometries
```

Ergebnisse:

```
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)
```

Funktion ST_GeometryN

Die Funktion ST_GeometryN verwendet eine Geometriengruppe und einen Index als Eingabeparameter und gibt die Geometrie in der Gruppe zurück, die durch den Index angegeben ist. Die Ergebnisgeometrie wird im räumlichen Bezugssystem der angegebenen Geometriengruppe dargestellt.

Wenn die angegebene Geometriengruppe den Wert null aufweist oder leer ist, oder wenn der Index kleiner als 1 oder größer als die Anzahl der Geometrien in der Gruppe ist, wird null zurückgegeben und eine Warnung (01HS0) erzeugt.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

gruppe
Ein Wert vom Typ ST_GeomCollection oder einer seiner untergeordneten Typen, der die Geometriengruppe darstellt, in der die *n*. Geometrie gesucht werden soll.

index Ein Wert vom Typ INTEGER, der die *n*. Geometrie kennzeichnet, die von der *Gruppe* zurückgegeben werden soll.

Wenn der *Index* kleiner als 1 oder größer als die Anzahl der Geometrien in der Gruppe ist, wird null zurückgegeben und eine Warnung (SQLSTATE 01HS0) wird erzeugt.

Rückgabetyp

db2gse.ST_Geometry

Beispiel

Der folgende Code verdeutlicht, wie die zweite Geometrie innerhalb einer Geometriengruppe ausgewählt wird.

```
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
```

Ergebnisse:

```
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))
```

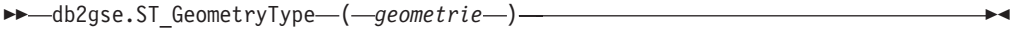
Funktion ST_GeometryType

Die Funktion ST_GeometryType verwendet einen Geometrietyp als Eingabeparameter und gibt den vollständigen Typnamen des dynamischen Typs dieser Geometrie zurück.

Die DB2-Funktionen TYPE_SCHEMA und TYPE_NAME haben die gleiche Wirkung.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry, für den der Geometrietyp zurückgegeben werden soll.

Rückgabetyp

VARCHAR(128)

Beispiele

Der folgende Code verdeutlicht, wie der Typ einer Geometrie ermittelt wird.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries (id INTEGER, geometry ST_GEOMETRY)

INSERT INTO sample_geometries(id, geometry)
  VALUES
        (7101, ST_Geometry('point(1 2)', 1) ),
        (7102, ST_Geometry('linestring(33 2, 34 3, 35 6)', 1) ),
        (7103, ST_Geometry('polygon((3 3, 4 6, 5 3, 3 3))', 1)),
        (7104, ST_Geometry('multipoint(1 2, 4 3)', 1) )

SELECT id, geometry..ST_GeometryType AS geometry_type
FROM   sample_geometries
```

Ergebnisse:

```
ID          GEOMETRY_TYPE
----------- ------------------------------
       7101 "DB2GSE  "."ST_POINT"
       7102 "DB2GSE  "."ST_LINESTRING"
       7103 "DB2GSE  "."ST_POLYGON"
       7104 "DB2GSE  "."ST_MULTIPOINT"
```

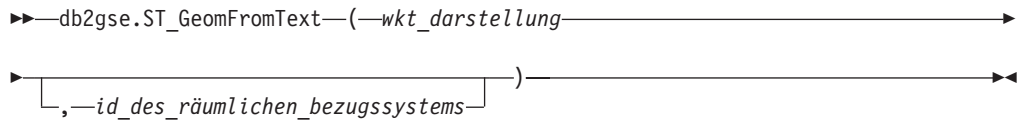
Funktion ST_GeomFromText

Die Funktion ST_GeomFromText verwendet eine WKT-Darstellung (WKT = Well-Known Text) einer Geometrie und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt die entsprechende Geometrie zurück.

Wenn die angegebene WKT-Darstellung den Wert null aufweist, wird null zurückgegeben.

Die bevorzugte Version dieser Funktionalität ist ST_Geometry.

Syntax



Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung der Ergebnisgeometrie enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für die Ergebnisgeometrie kennzeichnet.

Wenn der Parameter *id_des_räumlichen_bezugssystems* ausgelassen wird, verwendet das System implizit das räumliche Bezugssystem mit der numerischen Kennung 0 (null).

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben ist, das in der Katalogsicht

DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet wird, gibt das System einen Fehler (SQLSTATE 38SU1) zurück.

Rückgabetyp

db2gse.ST_Geometry

Beispiel

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

In diesem Beispiel wird die Funktion ST_GeomFromText zur Erstellung und zum Einfügen eines Punktes von der WKT-Darstellung (WKT = Well-Known Text) des Punktes verwendet.

Mit dem folgenden Code werden Zeilen in die Tabelle SAMPLE_POINTS mit IDs und Geometrien im räumlichen Bezugssystem 1 unter Verwendung der WKT-Darstellung eingefügt.

```
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))
```

Die folgenden Anweisung SELECT gibt die ID und die Geometrien aus der Tabelle SAMPLE_GEOMETRIES zurück.

```
SELECT id, cast(geometry..ST_AsText AS varchar(105))
      AS geometry
FROM   sample_geometries
```

Ergebnisse:

```
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))
```

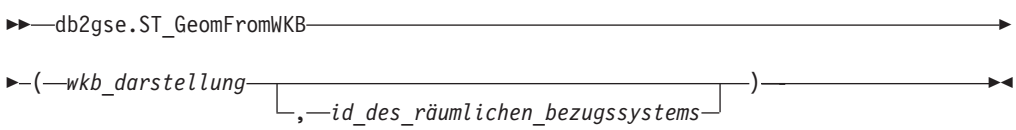
Funktion ST_GeomFromWKB

Die Funktion ST_GeomFromWKB verwendet eine WKB-Darstellung (WKB = Well-Known Binary) einer Geometrie und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt die entsprechende Geometrie zurück.

Wenn die angegebene WKB-Darstellung den Wert null aufweist, wird null zurückgegeben.

Die bevorzugte Version dieser Funktionalität ist ST_Geometry.

Syntax



Parameter

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung der Ergebnisgeometrie enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für die Ergebnisgeometrie kennzeichnet.

Wenn der Parameter *id_des_räumlichen_bezugssystems* ausgelassen wird, verwendet das System implizit das räumliche Bezugssystem mit der numerischen Kennung 0 (null).

Wenn der angegebene Parameter "id_des_räumlichen_bezugssystems" kein räumliches Bezugssystem kennzeichnet, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, wird ein Fehler (SQLSTATE 38SU1) zurückgegeben.

Rückgabetyp

db2gse.ST_Geometry

Beispiele

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Der folgende Code verdeutlicht, wie die Funktion ST_GeomFromWKB verwendet werden kann, um eine Linienfolge von der bekannten Binärdarstellung (WKB) der Linienfolge zu erstellen und einzufügen.

Im folgenden Beispiel wird ein Datensatz in die Tabelle SAMPLE_GEOMETRIES mit einer ID und einer Geometrie im räumlichen Bezugssystem 1 in eine WKB-Darstellung (WKB = Well-Known Binary) eingefügt.

```
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
```

Ergebnisse:

```
 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))
```

MBR Aggregate-Funktionen

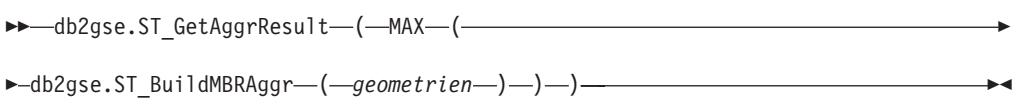
Verwenden Sie eine Kombination der Funktionen ST_BuildMBRAggr und ST_GetAggrResult, um mehrere Geometrien in einer Spalte zu einer einzigen Geometrie zusammenzufassen. Die Kombination erzeugt ein Rechteck, das den minimalen Begrenzungsrahmen (MBR) darstellt, der alle Geometrien in der Spalte einschließt. Bei der Berechnung des Ergebnisses werden Z- und M-Koordinaten gelöscht.

Beim folgenden Ausdruck handelt es sich um ein Beispiel für die Verwendung der Funktion MAX in Verbindung mit der räumlichen Funktion db2gse.ST_BuildMBRAggr zur Berechnung des MBR der Geometrien in der Spalte columnName sowie für die Verwendung der räumlichen Funktion db2gse.ST_GetAggrResult zur Ausgabe der daraus resultierenden Geometrie, die für das MBR berechnet wurde:

```
db2gse.ST_Get_AggrResult(MAX(db2gse.ST_BuildMBRAggr(columnName)))
```

Wenn alle zu kombinierenden Geometrien den Wert null aufweisen, wird null zurückgegeben. Wenn alle Geometrien entweder den Wert null aufweisen oder leer sind, wird eine leere Geometrie zurückgegeben. Wenn das minimal einschließende Rechteck (MBR) aller zu kombinierenden Geometrien einen Punkt als Ergebnis hat, wird dieser Punkt als ein Wert vom Typ ST_Point zurückgegeben. Wenn das minimal einschließende Rechteck aller zu kombinierenden Geometrien eine horizontale oder vertikale Linienfolge als Ergebnis hat, wird diese Linienfolge als ein Wert vom Typ ST_LineString zurückgegeben. Andernfalls wird das minimal einschließende Rechteck als ein Wert vom Typ ST_Polygon zurückgegeben.

Syntax



Parameter

Geometrien

Eine ausgewählte Spalte, die den Typ ST_Geometry oder einen seiner untergeordneten Typen aufweist und alle Geometrien darstellt, für die das minimal einschließende Rechteck berechnet werden soll.

Rückgabetyp

db2gse.ST_Geometry

Einschränkungen

Sie können in den folgenden Situationen keine Union-Gesamtverknüpfung einer räumlichen Spalte in einem Fullselect erstellen:

- In einer Umgebung mit partitionierten Datenbanken.
- Wenn die Klausel GROUP BY für den Fullselect verwendet wird.
- Wenn Sie eine andere Funktion als die DB2-Spaltenfunktion MAX verwenden.

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel zeigt, wie die Funktion ST_BuildMBRAggr verwendet wird, um das maximal einschließende Rechteck aller Geometrien in einer Spalte zu erhalten. In diesem Beispiel werden der Spalte GEOMETRY in der Tabelle SAMPLE_POINTS mehrere Punkte hinzugefügt. Der SQL-Code ermittelt dann das maximal einschließende Rechteck aller zusammengefassten Punkte.

```
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
```

Ergebnisse:

```
Aggregate_of_Points
-----------------------------------------------------------
POLYGON (( 2.00000000 2.00000000, 23.00000000 2.00000000,
23.00000000 15.00000000, 2.00000000 15.00000000, 2.00000000
2.00000000))
```

Union-Gesamtverknüpfungen

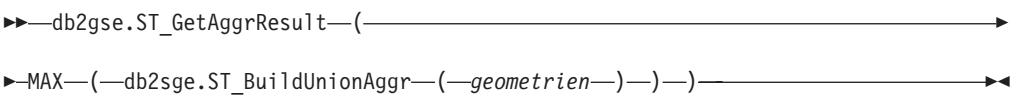
Eine Union-Gesamtverknüpfung von Geometrien ist die Kombination der Funktionen ST_BuildUnionAggr und ST_GetAggrResult. Mithilfe dieser Kombination können Sie eine Spalte mit Geometrien in einer Tabelle zu einer einzigen Geometrie zusammenfassen, indem die Union-Verknüpfung erstellt wird.

Wenn alle zu kombinierenden Geometrien in der Union-Verknüpfung den Wert null aufweisen, wird null zurückgegeben. Wenn alle der zu kombinierenden Geo-

metrien in der Union-Verknüpfung entweder den Wert null aufweisen oder leer sind, wird eine leere Geometrie vom Typ ST_Point zurückgegeben.

Die Funktion ST_BuildUnionAggr kann auch als Methode aufgerufen werden.

Syntax



Parameter

Geometrien
Eine Spalte in einer Tabelle, die den Typ ST_Geometry oder einen seiner untergeordneten Typen aufweist und alle Geometrien darstellt, die in einer Union-Verknüpfung kombiniert werden sollen.

Rückgabetyp

db2gse.ST_Geometry

Einschränkungen

In den folgenden Situationen können Sie keine Union-Gesamtverknüpfung einer räumlichen Spalte in einer Tabelle erstellen:

- In einer Umgebungen mit partitionierten Datenbanken.
- Wenn eine Klausel GROUP BY in der Auswahlanweisung verwendet wird.
- Wenn Sie eine andere Funktion als die DB2-Spaltenfunktion MAX verwenden

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie eine Union-Gesamtverknüpfung zur Zusammenfassung einer Gruppe von Punkten zu einer Mehrpunktangabe verwendet werden kann. Der Tabelle SAMPLE_POINTS werden mehrere Punkte hinzugefügt. Die Funktionen ST_GetAggrResult und ST_BuildUnionAggr werden verwendet, um die Union-Verknüpfung der Punkte zu erstellen.

```
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
```

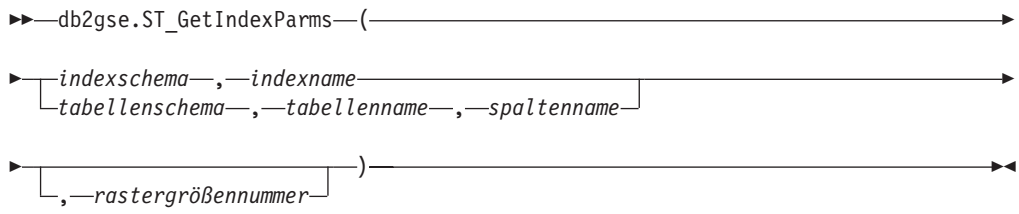
Ergebnisse:

```
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)
```

Funktion ST_GetIndexParms

Die Funktion ST_GetIndexParms verwendet die Kennung für einen räumlichen Index oder für eine räumliche Spalte als Eingabeparameter und gibt entweder die Parameter, mit denen der Index definiert wurde, oder den Index für die räumliche Spalte zurück. Wenn eine zusätzliche Parameternummer angegeben ist, wird nur die Rastergröße zurückgegeben, die durch die Nummer gekennzeichnet wird.

Syntax



Parameter

indexschema
Ein Wert vom Typ VARCHAR(128), der das Schema kennzeichnet, in dem sich der räumliche Index mit dem unqualifizierten Namen *indexname* befindet. Beim Schemanamen wird zwischen Groß- und Kleinschreibung unterschieden. Ferner muss der Schemaname sich in der Katalogsicht SYSCAT.SCHEMATA befinden.

Wenn dieser Parameter den Wert null aufweist, wird der Wert für das Sonderregister CURRENT SCHEMA als Schemaname für den räumlichen Index verwendet.

indexname
Ein Wert vom Typ VARCHAR(128), der den unqualifizierten Namen des räumlichen Indexes enthält, für den die Indexparameter zurückgegeben werden. Beim Indexnamen wird zwischen Groß- und Kleinschreibung unterschieden. Ferner muss der Indexname sich in der Katalogsicht SYSCAT.INDEXES für das Schema *indexschema* befinden.

tabellenschema
Ein Wert vom Typ VARCHAR(128), der das Schema kennzeichnet, in dem sich die Tabelle mit dem unqualifizierten Namen *tabellenname* befindet. Beim Schemanamen wird zwischen Groß- und Kleinschreibung unterschieden. Ferner muss der Schemaname sich in der Katalogsicht SYSCAT.SCHEMATA befinden.

Wenn dieser Parameter den Wert null aufweist, wird der Wert des Sonderregisters CURRENT SCHEMA als Schemaname für den räumlichen Index verwendet.

tabellenname
Ein Wert vom Typ VARCHAR(128), der den unqualifizierten Namen der Tabelle mit der räumlichen Spalte *spaltenname* enthält. Beim Tabellennamen wird zwischen Groß- und Kleinschreibung unterschieden. Ferner muss der Tabellenname in der Katalogsicht SYSCAT.TABLES für das Schema *tabellenschema* aufgelistet sein.

spaltenname
Ein Wert vom Typ VARCHAR(128), der die Spalte in der Tabelle *tabellenschema.tabellenname* kennzeichnet, für die die Indexparameter des räumlichen Indexes für diese Spalte zurückgegeben werden. Beim Spaltennamen wird zwischen Groß- und Kleinschreibung unterschieden. Ferner muss der Spaltenname in der Katalogsicht SYSCAT.COLUMNS für die Tabelle *tabellenschema.tabellenname* aufgelistet sein.

Wenn in der Spalte kein räumlicher Index definiert ist, wird ein Fehler (SQLSTATE 38SQ0) erzeugt.

rastergrößennummer
Ein Wert vom Typ DOUBLE, der den Parameter kennzeichnet, dessen Wert oder Werte zurückgegeben werden sollen.

Wenn dieser Wert kleiner als 1 oder größer als 3 ist, wird ein Fehler (SQLSTATE 38SQ1) erzeugt.

Rückgabetyp

DOUBLE (wenn *rastergrößennummer* angegeben ist)

Wenn *rastergrößennummer* nicht angegeben ist, wird eine Tabelle mit zwei Spalten ORDINAL und VALUE zurückgegeben. Die Spalte ORDINAL weist den Typ INTEGER auf und die Spalte VALUE den Typ DOUBLE.

Wenn die Parameter für ein Rasterindex zurückgegeben werden, enthält die Spalte ORDINAL die Werte 1, 2 und 3 jeweils für die erste, zweite und dritte Rastergröße. Die Spalte VALUE enthält die Rastergrößen.

Die Spalte VALUE enthält die entsprechenden Werte für jeden der Parameter.

Beispiele

Beispiel 1

Mit diesem Code wird eine Tabelle mit einer räumlichen Spalte und einem räumlichen Index erstellt.

```
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)
```

Die Funktion ST_GetIndexParms kann verwendet werden, um die Werte für die Parameter abzurufen, die bei der Erstellung des räumlichen Indexes verwendet wurden.

Beispiel 2

Dieses Beispiel zeigt, wie die drei Rastergrößen für einen räumlichen Rasterindex getrennt abgerufen werden, indem explizit angegeben wird, welcher Parameter zurückgegeben werden soll. Die Parameter werden dabei durch ihre Nummer gekennzeichnet.

```
VALUES ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION', 1)
```

Ergebnisse:

```
1
-----------------------
    +1.00000000000000E+000
```

```
VALUES ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION', 2)
```

Ergebnisse:

```
1
-----------------------
    +1.00000000000000E+001
```

```
VALUES ST_GetIndexParms('SCH', 'IDX', 3)
```

Ergebnisse:

```
1
-----------------------
    +1.00000000000000E+003
```

Beispiel 3

Dieses Beispiel zeigt, wie alle Parameter eines räumlichen Rasterindexes abgerufen werden. Die Funktion ST_GetIndexParms gibt eine Tabelle zurück, die die Parameternummer und die entsprechende Rastergröße anzeigt.

```
SELECT * FROM TABLE ( ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION') ) AS t
```

Ergebnisse:

```
ORDINAL     VALUE
----------- -----------------------
          1   +1.00000000000000E+000
          2   +1.00000000000000E+001
          3   +1.00000000000000E+003
```

```
SELECT * FROM TABLE ( ST_GetIndexParms('SCH', 'IDX') ) AS t
```

Ergebnisse:

```
ORDINAL     VALUE
----------- -----------------------
          1   +1.00000000000000E+000
          2   +1.00000000000000E+001
          3   +1.00000000000000E+003
```

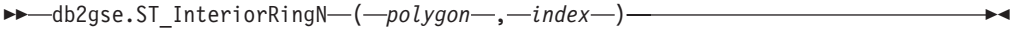
N

Die Funktion ST_InteriorRingN verwendet ein Polygon und einen Index als Eingabeparameter und gibt den inneren Ring, der durch den angegebenen Index definiert ist, als Linienfolge zurück. Die inneren Ringe werden entsprechend den Regeln angeordnet, die durch die internen Geometrienprüfroutinen definiert sind.

Wenn das angegebene Polygon den Wert null aufweist oder leer ist, oder wenn es nicht über innere Ringe verfügt, wird null zurückgegeben. Wenn der Index kleiner als 1 oder größer als die Anzahl der inneren Ringe im Polygon ist, wird null zurückgegeben und eine Warnungsbedingung (1HS1) erzeugt.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

polygon
Ein Wert vom Typ ST_Polygon, der die Geometrie darstellt, von der der innere Ringe zurückgegeben wird, der durch den unter *index* angegebenen Index gekennzeichnet ist.

index Ein Wert vom Typ INTEGER, der den *n*. inneren Ring kennzeichnet, der zurückgegeben wird. Wenn kein innerer Ring durch den *Index* gekennzeichnet ist, wird eine Warnungsbedingung (01HS1) erzeugt.

Rückgabetyp

db2gse.ST_Curve

Beispiel

In diesem Beispiel wird ein Polygon mit zwei inneren Ringen erstellt. Der Aufruf von ST_InteriorRingN wird anschließend verwendet, um den zweiten inneren Ring abzurufen.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys VALUES
             (1, ST_Polygon('polygon((40 120, 90 120, 90 150, 40 150, 40 120),
                            (50 130, 60 130, 60 140, 50 140, 50 130),
                            (70 130, 80 130, 80 140, 70 140, 70 130))' ,0))


SELECT id, CAST(ST_AsText(ST_InteriorRingN(geometry, 2)) as VARCHAR(180))
             Interior_Ring
FROM sample_polys
```

Ergebnisse:

```
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)
```

Funktion ST_Intersection

Die Funktion ST_Intersection verwendet zwei Geometrien als Eingabeparameter und gibt die Geometrie zurück, die die Schnittmenge der beiden angegebenen Geometrien darstellt. Die Schnittmenge ist der Teil der ersten Geometrie, der ebenfalls Teil der zweiten Geometrie ist. Die Ergebnisgeometrie wird im räumlichen Bezugssystem der ersten Geometrie dargestellt.

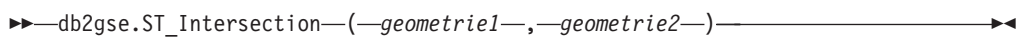
Nach Möglichkeit ist der spezifische Typ der zurückgegebenen Geometrie ST-_Point, ST_LineString oder ST_Polygon. Die Schnittmenge eines Punktes und eines Polygons ist entweder leer oder ein einzelner Punkt, der mit ST_MultiPoint dargestellt wird.

Wenn eine der beiden angegebenen Geometrien den Wert null aufweist, wird 0 (null) zurückgegeben.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist und das gleiche zugrunde liegende Datum verwendet, wird sie in das andere räumliche Bezugssystem umgewandelt.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie1
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die erste Geometrie darstellt, zu der die Schnittmenge mit der in *geometrie2* angegebenen Geometrie berechnet werden soll.

geometrie2
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die zweite Geometrie darstellt, zu der die Schnittmenge mit der in *geometrie1* angegebenen Geometrie berechnet werden soll.

Rückgabetyp

db2gse.ST_Geometry

Die Dimension der zurückgegebenen Geometrie stimmt mit der Dimension der Eingabe mit der niedrigeren Dimension überein.

Beispiel

In den folgenden Beispielen wurden die Ergebnisse zur besseren Lesbarkeit umformatiert. Der Abstand in den Ergebnissen variiert entsprechend der jeweiligen Anzeige.

In diesem Beispiel werden mehrere unterschiedliche Geometrien erstellt und anschließend wird die Schnittmenge (falls vorhanden) mit der ersten Geometrie ermittelt.

```
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
```

Ergebnisse:

```
ID          ID          INTERSECTION
----------- ----------- ------------------------------------------------
          1           1 POLYGON (( 30.00000000 30.00000000, 50.00000000
30.00000000, 50.00000000 50.00000000, 30.00000000 50.00000000, 30.00000000
30.00000000))

          1           2 LINESTRING ( 30.00000000 40.00000000, 30.00000000
30.00000000)

          1           3 POLYGON (( 40.00000000 40.00000000, 50.00000000
40.00000000, 50.00000000 50.00000000, 40.00000000 50.00000000, 40.00000000
40.00000000))

          1           4 POINT EMPTY

          1           5 LINESTRING ( 30.00000000 30.00000000, 50.00000000
50.00000000)

    5 record(s) selected.
```

Funktion ST_Intersects

Mit der Funktion ST_Intersects können Sie ermitteln, ob zwei Geometrien sich schneiden.

Syntax

Parameter

geometrie1
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die auf eine Überschneidung mit der in *geometrie2* angegebenen Geometrie überprüft werden soll.

geometrie2
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die auf eine Überschneidung mit der in *geometrie1* angegebenen Geometrie überprüft werden soll.

Rückgabetyp

INTEGER

Verwendung

ST_Intersects verwendet zwei Geometrien als Eingabeparameter und gibt 1 zurück, wenn die angegebenen Geometrien sich schneiden. Wenn die Geometrien sich nicht schneiden, wird 0 (null) zurückgegeben.

Wenn eine der beiden Geometrien den Wert null aufweist oder leer ist, wird null zurückgegeben.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist und das gleiche zugrunde liegende Datum verwendet, wird sie in das andere räumliche Bezugssystem umgewandelt.

ST_Intersects gibt genau das entgegengesetzte Ergebnis von ST_Disjoint zurück.

Die Funktion ST_Intersects gibt den Wert 1 (eins) zurück, wenn die Bedingung einer der folgenden Mustermatrizen TRUE zurückgibt.

Tabelle 28. Matrix für ST_Intersects (1). Die Funktion ST_Intersects gibt den Wert 1 (eins) zurück, wenn sich die Innenbereiche der beiden Geometrien schneiden.

	Geometrie b Innenbereich	**Geometrie b Begrenzung**	**Geometrie b Außenbereich**
Geometrie a Begrenzung	*	*	*
Geometrie a Innenbereich	T	*	*
Geometrie a Außenbereich	*	*	*

Tabelle 29. Matrix für ST_Intersects (2). Die Funktion ST_Intersects gibt den Wert 1 (eins) zurück, wenn die Begrenzung der ersten Geometrie die Begrenzung der zweiten Geometrie schneidet.

	Geometrie b Innenbereich	**Geometrie b Begrenzung**	**Geometrie b Außenbereich**
Geometrie a Begrenzung	*	*	*
Geometrie a Innenbereich	*	T	*
Geometrie a Außenbereich	*	*	*

Tabelle 30. Matrix für ST_Intersects (3). Die Funktion ST_Intersects gibt den Wert 1 (eins) zurück, wenn die Begrenzung der ersten Geometrie den Innenbereich der zweiten Geometrie schneidet.

	Geometrie b Innenbereich	**Geometrie b Begrenzung**	**Geometrie b Außenbereich**
Geometrie a Begrenzung	T	*	*
Geometrie a Innenbereich	*	*	*
Geometrie a Außenbereich	*	*	*

Tabelle 31. Matrix für ST_Intersects (4). Die Funktion ST_Intersects gibt den Wert 1 (eins) zurück, wenn sich die Begrenzungen einer der beiden Geometrien schneiden.

	Geometrie b Innenbereich	**Geometrie b Begrenzung**	**Geometrie b Außenbereich**
Geometrie a Begrenzung	*	T	*
Geometrie a Innenbereich	*	*	*
Geometrie a Außenbereich	*	*	*

Verwendung

Beispiel

Mit den folgenden Anweisungen werden die Tabellen SAMPLE_GEOMETRIES1 und SAMPLE_GEOMETRIES2 erstellt und gefüllt.

```
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) )
```

Die folgende Anweisung SELECT ermittelt, ob die verschiedenen Geometrien in den Tabellen SAMPLE_GEOMTRIES1 und SAMPLE_GEOMTRIES2 sich schneiden.

```
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
```

Ergebnisse:

```
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
```

Funktion ST_Is3d

Die Funktion ST_Is3d verwendet eine Geometrie als Eingabeparameter und gibt den Wert 1 zurück, wenn die angegebene Geometrie Z-Koordinaten aufweist. Andernfalls wird 0 (null) zurückgegeben.

Wenn die angegebene Geometrie den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die auf das Vorhandensein von Z-Koordinaten getestet werden soll.

Rückgabetyp

INTEGER

Beispiel

In diesem Beispiel werden mehrere Geometrien mit und ohne Z- und M-Koordinaten (Bemaßungen) erstellt. Anschließend wird ST_Is3d verwendet, um zu ermitteln, welche Geometrien Z-Koordinaten enthalten.

```
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
```

Ergebnisse:

```
ID          IS_3D
----------- -----------
          1           0
          2           0
```

```
3           0
4           1
5           1
```

Funktion ST_IsClosed

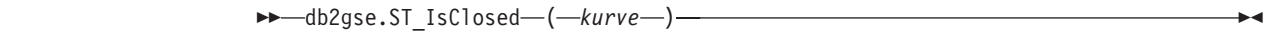
Die Funktion ST_IsClosed verwendet eine Kurve oder Mehrfachkurve als Eingabeparameter und gibt den Wert 1 zurück, wenn die angegebene Kurve oder Mehrfachkurve geschlossen ist. Andernfalls wird 0 (null) zurückgegeben.

Eine Kurve ist geschlossen, wenn ihr Startpunkt mit dem Endpunkt identisch ist. Wenn die Kurve Z-Koordinaten aufweist, müssen die Z-Koordinaten des Start- und Endpunkts identisch sein. Andernfalls werden die Punkte nicht als gleich betrachtet, und die Kurve ist nicht geschlossen. Eine Mehrfachkurve ist geschlossen, wenn jede ihrer Kurven geschlossen ist.

Wenn die angegebene Kurve oder Mehrfachkurve leer ist, wird 0 (null) zurückgegeben. Wenn die Kurve den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

kurve Ein Wert vom Typ ST_Curve oder ST_MultiCurve oder einer seiner untergeordneten Typen, der die Kurve oder Mehrfachkurve darstellt, die getestet werden soll.

Rückgabetyp

INTEGER

Beispiele

Beispiel 1

In diesem Beispiel werden mehrere Linienfolgen erstellt. Die letzten beiden Linienfolgen weisen dieselben X- und Y-Koordinaten auf. Eine Linienfolge enthält jedoch andere Z-Koordinaten, wodurch die Linienfolge nicht geschlossen ist. Die andere Linienfolge enthält andere M-Koordinaten (Bemaßungen), die keinen Einfluss darauf haben, ob die Linienfolge geschlossen ist.

```
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
```

Ergebnisse:

```
ID          IS_CLOSED
----------- -----------
          1           0
          2           0
          3           1
          4           1
          5           0
```

Beispiel 2

In diesem Beispiel werden zwei Mehrlinienfolgen erstellt. Anschließend wird ST_IsClosed verwendet, um zu ermitteln, ob die Mehrlinienfolgen geschlossen sind. Die erste Mehrlinienfolge ist nicht geschlossen, obwohl alle Kurven zusammen eine vollständig geschlossene Schleife bilden. Die Mehrlinienfolge ist nicht geschlossen, weil die einzelnen Kurven selbst nicht geschlossen sind.

Die zweite Mehrlinienfolge ist geschlossen, weil die einzelnen Kurven selbst geschlossen sind.

```
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),



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
```

Ergebnisse:

```
ID          IS_CLOSED
----------- -----------
          6           0
          7           1
```

Funktion ST_IsEmpty

Die Funktion ST_IsEmpty verwendet eine Geometrie als Eingabeparameter und gibt den Wert 1 zurück, wenn die angegebene Geometrie leer ist. Andernfalls wird 0 (null) zurückgegeben.

Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die getestet werden soll.

Rückgabetyp

INTEGER

Beispiel

Mit dem folgenden Code werden drei Geometrien erstellt. Anschließend wird ermittelt, ob sie leer sind.

```
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
```

Ergebnisse:

```
ID IS_EMPTY
---------- -----------
         1           1
         2           0
         3           0
         4           0
         5           0
```

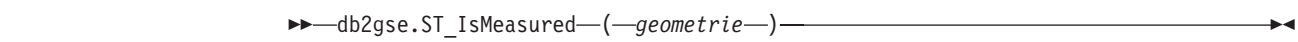
Funktion ST_IsMeasured

Die Funktion ST_IsMeasured verwendet eine Geometrie als Eingabeparameter. Falls die angegebene Geometrie M-Koordinaten (Bemaßungen) aufweist, gibt sie 1 zurück. Andernfalls wird 0 (null) zurückgegeben.

Wenn die angegebene Geometrie den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die auf das Vorhandensein von M-Koordinaten (Bemaßungen) getestet werden soll.

Rückgabetyp

INTEGER

Beispiel

In diesem Beispiel werden mehrere Geometrien mit und ohne Z- und M-Koordinaten (Bemaßungen) erstellt. Anschließend wird ST_IsMeasured verwendet, um zu ermitteln, welche der Geometrien Bemaßungen enthalten.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geoms (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geoms VALUES
             (1, ST_Geometry('point EMPTY',0))

INSERT INTO sample_geoms VALUES
             (2, ST_Geometry('polygon((40 120, 90 120, 90 150, 40 150, 40 120))' ,0))

INSERT INTO sample_geoms VALUES
          (3, ST_Geometry('multipoint m (10 10 5, 50 10 6, 10 30 8)' ,0))

INSERT INTO sample_geoms VALUES
             (4, ST_Geometry('linestring z (10 10 166, 20 10 168)',0))

INSERT INTO sample_geoms VALUES
             (5, ST_Geometry('point zm (10 10 16 30)' ,0))


SELECT id, ST_IsMeasured(geometry)  Is_Measured
FROM sample_geoms
```

Ergebnisse:

```
ID          IS_MEASURED
----------- -----------
          1           0
          2           0
          3           1
          4           0
          5           1
```

Funktion ST_IsRing

Die Funktion ST_IsRing verwendet eine Kurve als Eingabeparameter und gibt 1 zurück, wenn es sich um einen Ring handelt. Andernfalls wird 0 (null) zurückgegeben. Eine Kurve ist ein Ring, wenn sie einfach und geschlossen ist.

Wenn die angegebene Kurve leer ist, wird 0 (null) zurückgegeben. Wenn die Kurve den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

kurve Ein Wert vom Typ ST_Curve oder einer seiner untergeordneten Typen, der die Kurve darstellt, die getestet werden soll.

Rückgabetyp

INTEGER

Beispiele

In diesem Beispiel werden vier Linienfolgen erstellt. ST_IsRing wird verwendet, um zu überprüfen, ob es sich um Ringe handelt. Die letzte Linienfolge wird nicht als Ring betrachtet. Diese Linienfolge ist zwar geschlossen, kreuzt sich jedoch selbst.

```
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
```

Ergebnisse:

```
ID          IS_CLOSED   IS_RING
----------- ----------- -----------
          1           1           0
          2           0           0
          3           1           1
          4           1           0
```

Funktion ST_IsSimple

Die Funktion ST_IsSimple verwendet eine Geometrie als Eingabeparameter und gibt den Wert 1 zurück, wenn die angegebene Geometrie einfach ist. Andernfalls wird 0 (null) zurückgegeben.

Punkte, Oberflächen und Mehrfachoberflächen sind immer einfach. Eine Kurve ist einfach, wenn sie keinen Punkt zweimal schneidet; eine Mehrpunktangabe ist einfach, wenn sie keine zwei gleichen Punkte enthält; eine Mehrfachkurve ist einfach,

wenn alle ihrer Kurven einfach sind und die einzigen Schnittpunkte an Punkten auftreten, die sich an der Grenze der Kurven in der Mehrfachkurve befinden.

Wenn die angegebene Geometrie leer ist, wird 1 zurückgegeben. Wenn die Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die getestet werden soll.

Rückgabetyp

INTEGER

Beispiele

In diesem Beispiel werden mehrere Geometrien erstellt und daraufhin überprüft, ob sie einfach sind. Die Geometrie mit der ID 4 wird nicht als einfach angesehen, da sie mehr als einen identischen Punkt enthält. Die Geometrie mit der ID 6 wird nicht als einfach angesehen, da die Linienfolge sich selbst kreuzt.

```
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
```

Ergebnisse:

```
ID          IS_SIMPLE
----------- -----------
          1           1
          2           1
```

```
3           1
4           0
5           1
6           0
7           1
```

Funktion ST_IsValid

Die Funktion ST_IsValid verwendet eine Geometrie als Eingabeparameter und gibt den Wert 1 zurück, wenn diese gültig ist. Andernfalls wird 0 (null) zurückgegeben.

Eine Geometrie ist nur dann gültig, wenn alle Attribute im strukturierten Typ mit der internen Darstellung von Geometriedaten konsistent sind und wenn die interne Darstellung nicht beschädigt ist.

Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen.

Rückgabetyp

INTEGER

Beispiel

In diesem Beispiel werden mehrere Geometrien erstellt. ST_IsValid wird verwendet, um zu prüfen, ob die Geometrien gültig sind. Alle Geometrien sind gültig, da die Konstruktorroutinen wie ST_Geometry nicht zulassen, dass ungültige Geometrien konstruiert werden.

```
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
```

Ergebnisse:

```
ID          IS_VALID
----------- -----------
          1           1
          2           1
          3           1
          4           1
          5           1
```

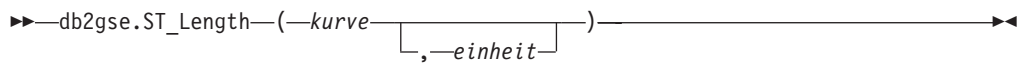
Funktion ST_Length

Die Funktion ST_Length verwendet eine Kurve oder Mehrfachkurve und optional eine Einheit als Eingabeparameter und gibt die Länge der angegebenen Kurve bzw. Mehrfachkurve in der Standardeinheit oder der angegebenen Maßeinheit zurück.

Wenn die angegebene Kurve oder Mehrfachkurve den Wert null aufweist oder leer ist, wird 0 (null) zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

kurve Ein Wert vom Typ ST_Curve oder ST_MultiCurve, der die Kurven darstellt, für die die Länge zurückgegeben wird.

einheit
Ein Wert vom Typ VARCHAR(128), der die Einheiten kennzeichnet, in der die Länge der Kurve gemessen wird. Die unterstützten Maßeinheiten sind in der Katalogsicht DB2GSE.ST_UNITS_OF_MEASURE aufgelistet.

Wenn der Parameter *einheit* ausgelassen wird, werden die folgenden Regeln verwendet, um zu ermitteln, in welcher Einheit die Länge gemessen wird:

- Wenn die in *kurve* angegebene Kurve sich in einem projizierten oder geozentrischen Koordinatensystem befindet, wird als Standardwert die lineare Einheit verwendet, die diesem Koordinatensystem zugeordnet ist.
- Wenn die *kurve* sich in einem geografischen Koordinatensystem befindet, wird als Standardwert die Winkeleinheit verwendet, die diesem Koordinatensystem zugeordnet ist.

Einschränkungen bei Einheitenumsetzungen: Ein Fehler (SQLSTATE 38SU4) wird zurückgegeben, wenn eine der folgenden Bedingungen zutrifft:

- Die in *kurve* angegebene Kurve befindet sich in einem nicht definierten Koordinatensystem, und der Parameter *einheit* wurde angegeben.
- Die in *kurve* angegebene Kurve befindet sich in einem projizierten Koordinatensystem, und es wurde eine Winkeleinheit angegeben.

- Die *kurve* befindet sich in einem geografischen Koordinatensystem, und es wurde eine lineare Einheit angegeben.

Rückgabetyp

DOUBLE

Beispiele

Beispiel 1

Mit den folgenden SQL-Anweisungen wird eine Tabelle SAMPLE_GEOMETRIES erstellt und eine Linienfolge sowie eine Mehrlinienfolge in die Tabelle eingefügt.

```
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))
```

Beispiel 2

Die folgende Anweisung SELECT berechnet die Länge der Linienfolge in der Tabelle SAMPLE_GEOMTRIES.

```
SELECT id, spatial_type, cast(ST_Length(geometry..ST_ToLineString)
      AS DECIMAL(7, 2)) AS "Line Length"
FROM   sample_geometries  WHERE  id = 1110
```

Ergebnisse:

```
ID     SPATIAL_TYPE         Line Length
------ -------------------- -----------
  1110 ST_LineString              10.00
```

Beispiel 3

Die folgende Anweisung SELECT berechnet die Länge der Mehrlinienfolge in der Tabelle SAMPLE_GEOMTRIES.

```
SELECT id, spatial_type, ST_Length(ST_ToMultiLine(geometry))
      AS multiline_length
FROM   sample_geometries  WHERE  id = 1111
```

Ergebnisse:

```
ID     SPATIAL_TYPE         MULTILINE_LENGTH
 ------ -------------------- ------------------------
   1111 ST_MultiLineString     +2.76437123387202E+001
```

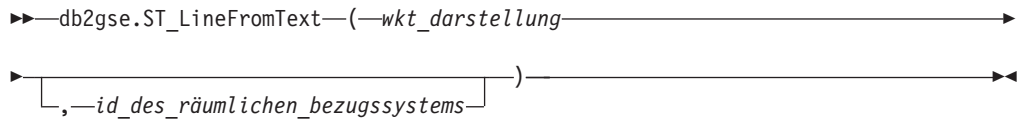
Funktion ST_LineFromText

Die Funktion ST_LineFromText verwendet eine WKT-Darstellung (WKT = Well-Known Text) einer Linienfolge und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt die entsprechende Linienfolge zurück.

Wenn die angegebene WKT-Darstellung den Wert null aufweist, wird null zurückgegeben.

Die bevorzugte Version dieser Funktionalität ist ST_LineString.

Syntax



Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung (WKT = Well-Known Text) der Ergebnislinienfolge enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für die Ergebnislinienfolge kennzeichnet.

Wenn der Paramater *id_des_räumlichen_bezugssystems* ausgelassen wird, wird das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben ist, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet wird, gibt das System einen Fehler (SQLSTATE 38SU1) zurück.

Rückgabetyp

db2gse.ST_LineString

Beispiel

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Der folgende Code verwendet die Funktion ST_LineFromText zur Erstellung und zum Einfügen einer Linie aus einer WKT-Darstellung (WKT = Well-Known Text) der Linie. Die Zeilen werden in die Tabelle SAMPLE_LINES mit einer ID und einem Linienwert im räumlichen Bezugssystem 1 in der WKT-Darstellung eingefügt.

```
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
```

Ergebnisse:

```
ID     LINESTRING
------ ---------------------------------------------------------------
  1110 LINESTRING ( 850.00000000 250.00000000, 850.00000000 850.00000000)
  1111 LINESTRING EMPTY
```

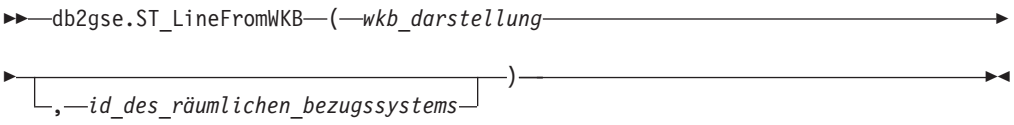
Funktion ST_LineFromWKB

Die Funktion ST_LineFromWKB verwendet eine WKB-Darstellung (WKB = Well-Known Binary) einer Linienfolge und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt die entsprechende Linienfolge zurück.

Wenn die angegebene WKB-Darstellung den Wert null aufweist, wird null zurückgegeben.

Die bevorzugte Version dieser Funktionalität ist ST_LineString.

Syntax



Parameter

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung der Ergebnislinienfolge enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für die Ergebnislinienfolge kennzeichnet.

Wenn der Paramater ***id_des_räumlichen_bezugssystems*** ausgelassen wird, wird das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben ist, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet wird, gibt das System einen Fehler (SQLSTATE 38SU1) zurück.

Rückgabetyp

db2gse.ST_LineString

Beispiel

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Der folgende Code verwendet die Funktion ST_LineFromWKB zur Erstellung und zum Einfügen einer Linie aus einer WKB-Darstellung (WKB = Well-Known Binary). Die Zeile wird in die Tabelle SAMPLE_LINES mit einer ID und einer Linie im räumlichen Bezugssystem 1 in der WKB-Darstellung eingefügt.

```
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
```

Ergebnisse:

```
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)
```

Funktion ST_LineString

Die Funktion ST_LineString konstruiert eine Linienfolge aus einer Eingabe.

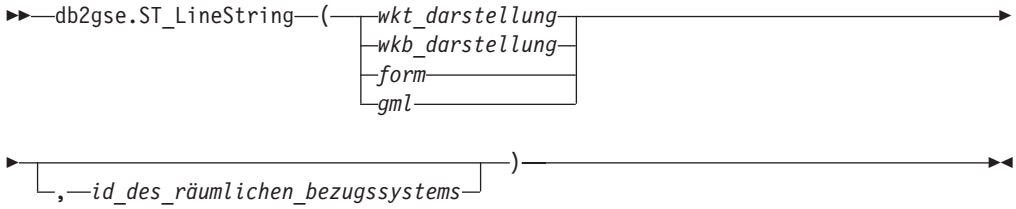
Eingaben können in einem der folgenden Formate erfolgen:

- WKT-Darstellung
- WKB-Darstellung
- ESRI-Formdarstellung
- Darstellung in GML (Geography Markup Language)

Die Kennung eines räumlichen Bezugssystems kann optional angegeben werden, um das räumliche Bezugssystem zu kennzeichnen, in dem sich die Ergebnislinienfolge befindet.

Wenn die WKT-, die WKB-, die ESRI-Formdarstellung oder die GML-Darstellung den Wert null aufweisen, wird null zurückgegeben.

Syntax



Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung (WKT = Well-Known Text) des Ergebnispolygons enthält.

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung des Ergebnispolygons enthält.

form Ein Wert vom Typ BLOB(2G), der die ESRI-Formdarstellung der Ergebnisfläche darstellt.

gml Ein Wert vom Typ CLOB(2G), der die Ergebnisfläche unter Verwendung von GML (Geography Markup Language) darstellt.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem des Ergebnispolygons kennzeichnet.

Falls der Parameter *id_des_räumlichen_bezugssystems* nicht angegeben wird, verwendet das System das räumliche Bezugssystem mit der numerischen Kennung 0 (null).

Wenn *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem kennzeichnet, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, wird ein Fehler (SQLSTATE 38SU1) zurückgegeben.

Rückgabetyp

db2gse.ST_LineString

Beispiele

Der folgende Code verwendet die Funktion ST_LineString, um eine Linie aus einer WKT-Darstellung (WKT = Well-Known Text) oder einer WKB-Darstellung (WKB = Well-Known Binary) der Linie zu erstellen und einzufügen.

Im folgenden Beispiel wird eine Zeile in die Tabelle SAMPLE_LINES mit einer ID und eine Linie im räumlichen Bezugssystem 1 in der WKT-Darstellung (WKT = Well-Known Text) und in der GML-Darstellung eingefügt.

```
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
```

Ergebnisse:

```
ID     LINESTRING
------ ---------------------------------------------------------------
  1110 LINESTRING ( 850.00000000 250.00000000, 850.00000000 850.00000000)
  1111 LINESTRING ( 90.00000000 90.00000000, 100.00000000 100.00000000)
```

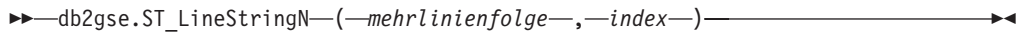
Funktion ST_LineStringN

Die Funktion ST_LineStringN verwendet eine Mehrlinienfolge und einen Index als Eingabeparameter und gibt die Linienfolge zurück, die durch den Index gekennzeichnet ist. Die Ergebnislinienfolge wird im räumlichen Bezugssystem der angegebenen Mehrlinienfolge dargestellt.

Wenn die angegebene Mehrlinienfolge den Wert null aufweist oder leer ist oder wenn der Index kleiner als 1 oder größer als die Anzahl der Linienfolgen ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

mehrlinienfolge
Ein Wert vom Typ ST_MultiLineString, der die Mehrlinienfolge darstellt, von der die Linienfolge, die durch den in *index* angegebenen Index gekennzeichnet ist, zurückgegeben wird.

index Ein Wert vom Typ INTEGER, der die *n*. Linienfolge kennzeichnet, die von der in *mehrlinienfolge* angegebenen Mehrlinienfolge zurückgegeben wird.

Wenn der *Index* kleiner als 1 oder größer als die Anzahl der Linienfolgen in der *Mehrlinienfolge* ist, wird null und eine Warnungsbedingung (SQLSTATE 01HS0) zurückgegeben.

Rückgabetyp

db2gse.ST_LineString

Beispiel

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Die folgende Anweisung SELECT verdeutlicht, wie die zweite Geometrie innerhalb einer Mehrlinienfolge in der Tabelle SAMPLE_MLINES ausgewählt wird.

```
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
```

Ergebnisse:

```
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)
```

Funktion ST_M

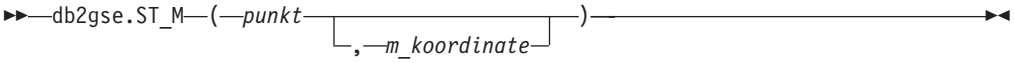
Die Funktion ST_M verwendet einen Punkt als Eingabeparameter und gibt die zugehörige Bemaßung (M-Koordinate) zurück. Sie können optional eine M-Koordinate als Eingabeparameter zusätzlich zu dem Punkt angeben, sodass die Funktion den Punkt selbst mit der M-Koordinate zurückgibt, für die ein bestimmter Wert gesetzt ist.

Wenn die angegebene M-Koordinate null ist, wird die M-Koordinate des Punktes entfernt.

Wenn der angegebene Punkt den Wert null aufweist oder leer ist, wird nullzurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

punkt Ein Wert vom Typ ST_Point, für den die M-Koordinate zurückgegeben oder geändert wird.

m_koordinate
Ein Wert vom Typ DOUBLE, der die neue M-Koordinate für den in *punkt* angegebenen Punkt darstellt.

Wenn *m_koordinate* null ist, wird die M-Koordinate von dem in *punkt* angegebenen Punkt entfernt.

Rückgabetypen

- DOUBLE, wenn die *M-Koordinate* nicht angegeben ist.
- db2gse.ST_Point, wenn die *M-Koordinate* angegeben ist.

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_M. Drei Punkte werden erstellt und in die Tabelle SAMPLE_POINTS eingefügt. Alle drei Punkte befinden sich im räumlichen Bezugssystem mit der ID 1.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)

INSERT INTO sample_points
    VALUES (1, ST_Point (2, 3, 32, 5, 1))

INSERT INTO sample_points
    VALUES (2, ST_Point (4, 5, 20, 4, 1))

INSERT INTO sample_points
    VALUES (3, ST_Point (3, 8, 23, 7, 1))
```

Beispiel 2

In diesem Beispiel werden die M-Koordinaten der Punkte in der Tabelle SAMPLE_POINTS gesucht.

```
SELECT id, ST_M (geometry) M_COORD
  FROM sample_points
```

Ergebnisse:

```
ID          M_COORD
----------- ------------------------
          1 +5.00000000000000E+000
          2 +4.00000000000000E+000
          3 +7.00000000000000E+000
```

Beispiel 3

Dieses Beispiel gibt einen der Punkte mit seiner M-Koordinate zurück, die auf 40 gesetzt wurde.

```
SELECT id, CAST (ST_AsText (ST_M (geometry, 40) )
    AS VARCHAR(60) ) M_COORD_40
  FROM sample_points
  WHERE id=3
```

Ergebnisse:

```
ID         M_COORD_40
---------- ------------------------------------------------------------
         3 POINT ZM (3.00000000 8.00000000 23.00000000 40.00000000)
```

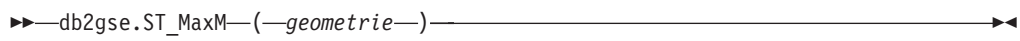
Funktion ST_MaxM

Die Funktion ST_MaxM verwendet eine Geometrie als Eingabeparameter und gibt deren maximale M-Koordinate zurück.

Wenn die angegebene Geometrie den Wert null aufweist oder leer ist oder wenn sie keine M-Koordinaten aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, für den die maximale M-Koordinate zurückgegeben wird.

Rückgabetyp

DOUBLE

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_MaxM. Drei Polygone werden erstellt und in die Tabelle SAMPLE_POLYS eingefügt.

```
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) )
```

Beispiel 2

In diesem Beispiel wird die größte M-Koordinate jedes einzelnen Polygons in SAMPLE_POLYS gesucht.

```
SELECT id, CAST ( ST_MaxM(geometry) AS INTEGER) MAX_M
  FROM sample_polys
```

Ergebnisse:

```
ID          MAX_M
----------- ------------
          1            4
          2           12
          3           16
```

Beispiel 3

In diesem Beispiel wird die größte M-Koordinate aller Polygone in der Spalte GEOMETRY gesucht.

```
SELECT CAST ( MAX ( ST_MaxM(geometry) ) AS INTEGER) OVERALL_MAX_M
  FROM sample_polys
```

Ergebnisse:

```
OVERALL_MAX_M
-------------
           16
```

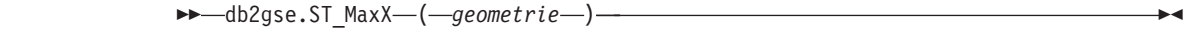
Funktion ST_MaxX

Die Funktion ST_MaxX verwendet eine Geometrie als Eingabeparameter und gibt ihre maximale X-Koordinate zurück.

Wenn die angegebene Geometrie den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, für den die maximale X-Koordinate zurückgegeben wird.

Rückgabetyp

DOUBLE

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_MaxX. Drei Polygone werden erstellt und in die Tabelle SAMPLE_POLYS eingefügt. Das dritte Beispiel verdeutlicht, wie Sie alle Funktionen verwenden können, die die größten und kleinsten Koordinatenwerte zurückgeben, um den räumlichen Bereich der Geometrien, die in einer bestimmten räumlichen Spalte gespeichert sind, abzuschätzen.

```
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) )
```

Beispiel 2

In diesem Beispiel wird die größte X-Koordinate jedes einzelnen Polygons in SAMPLE_POLYS gesucht.

```
SELECT id, CAST ( ST_MaxX(geometry) AS INTEGER) MAX_X_COORD
  FROM sample_polys
```

Ergebnisse:

```
ID          MAX_X_COORD
----------- ------------
          1          120
          2            5
          3           12
```

Beispiel 3

In diesem Beispiel wird die größte X-Koordinate aller Polygone in der Spalte GEOMETRY gesucht.

```
SELECT CAST ( MAX ( ST_MaxX(geometry) ) AS INTEGER) OVERALL_MAX_X
  FROM sample_polys
```

Ergebnisse:

```
OVERALL_MAX_X
-------------
          120
```

Beispiel 4

In diesem Beispiel wird der räumliche Bereich (allgemeines Minimum und allgemeines Maximum) aller Polygone in der Tabelle SAMPLE_POLYS gesucht. Diese Berechnung wird in der Regel verwendet, um den tatsächlichen räumlichen Bereich von Geometrien mit dem räumlichen Bereich des räumlichen Bezugssystems zu vergleichen, das den Daten zugeordnet ist. Ziel ist es zu ermitteln, ob die Daten Raum zum Wachsen haben.

```
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
```

Ergebnisse:

```
MIN_X    MIN_Y    MIN_Z    MIN_M    MAX_X    MAX_Y    MAX_Z   MAX_M
-------- -------- -------- -------- -------- -------- -------- --------
       0        0       10        3      120      140       40       16
```

Funktion ST_MaxY

Die Funktion ST_MaxY verwendet eine Geometrie als Eingabeparameter und gibt ihre maximale Y-Koordinate zurück.

Wenn die angegebene Geometrie den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, für den die maximale Y-Koordinate zurückgegeben wird.

Rückgabetyp

DOUBLE

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_MaxY. Drei Polygone werden erstellt und in die Tabelle SAMPLE_POLYS eingefügt.

```
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) )
```

Beispiel 2

In diesem Beispiel wird die größte Y-Koordinate jedes einzelnen Polygons in SAMPLE_POLYS gesucht.

```
SELECT id, CAST ( ST_MaxY(geometry) AS INTEGER) MAX_Y
  FROM sample_polys
```

Ergebnisse:

```
ID          MAX_Y
----------- ------------
          1          140
          2            4
          3           13
```

Beispiel 3

In diesem Beispiel wird die größte Y-Koordinate aller Polygone in der Spalte GEOMETRY gesucht.

```
SELECT CAST ( MAX ( ST_MaxY(geometry) ) AS INTEGER) OVERALL_MAX_Y
  FROM sample_polys
```

Ergebnisse:

```
OVERALL_MAX_Y
--------------
          140
```

Funktion ST_MaxZ

Die Funktion ST_MaxZ verwendet eine Geometrie als Eingabeparameter und gibt deren maximale Z-Koordinate zurück.

Wenn die angegebene Geometrie den Wert null aufweist oder leer ist oder wenn sie keine Z-Koordinaten aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie

Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, für den die maximale Z-Koordinate zurückgegeben wird.

Rückgabetyp

DOUBLE

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_MaxZ. Drei Polygone werden erstellt und in die Tabelle SAMPLE_POLYS eingefügt.

```
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) )
```

Beispiel 2

In diesem Beispiel wird die größte Z-Koordinate jedes einzelnen Polygons in SAMPLE_POLYS gesucht.

```
SELECT id, CAST ( ST_MaxZ(geometry) AS INTEGER) MAX_Z
  FROM sample_polys
```

Ergebnisse:

```
ID          MAX_Z
----------- ------------
          1           26
          2           40
          3           12
```

Beispiel 3

In diesem Beispiel wird die größte Z-Koordinate aller Polygone in der Spalte GEOMETRY gesucht.

```
SELECT CAST ( MAX ( ST_MaxZ(geometry) ) AS INTEGER) OVERALL_MAX_Z
  FROM sample_polys
```

Ergebnisse:

```
OVERALL_MAX_Z
-------------
           40
```

Funktion ST_MBR

Die Funktion ST_MBR verwendet eine Geometrie als Eingabeparameter und gibt ihr minimal einschließendes Rechteck (MBR) zurück.

Wenn die angegebene Geometrie ein Punkt ist, dann wird der Punkt selbst zurückgegeben. Wenn die Geometrie eine horizontale oder vertikale Linienfolge ist, wird die horizontale oder vertikale Linienfolge selbst zurückgegeben. Wenn die angegebene Geometrie den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, für die das minimal einschließende Rechteck zurückgegeben wird.

Rückgabetyp

db2gse.ST_Geometry

Beispiel

Dieses Beispiel verdeutlicht, wie die Funktion ST_MBR verwendet werden kann, um das minimal einschließende Rechteck (MBR) eines Polygons zurückzugeben. Da die angegebene Geometrie ein Polygon ist, wird das minimal einschließende Rechteck als Polygon zurückgegeben.

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

```
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
```

Ergebnisse:

```
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 ))
```

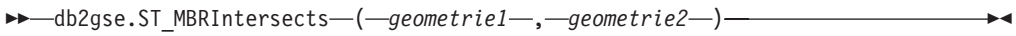
Funktion ST_MBRIntersects

Die Funktion ST_MBRIntersects verwendet zwei Geometrien als Eingabeparameter und gibt 1 zurück, wenn die minimal einschließenden Rechtecke der beiden Geometrien sich schneiden. Andernfalls wird 0 (null) zurückgegeben. Das minimal einschließende Rechteck eines Punktes und einer horizontalen oder vertikalen Linienfolge ist die Geometrie selbst.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist, wird sie in das andere räumliche Bezugssystem umgewandelt.

Wenn eine der angegebenen Geometrien den Wert null aufweist oder leer ist, wird null zurückgegeben.

Syntax



Parameter

geometrie1
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, deren minimal einschließendes Rechteck (MBR) auf Schnittpunkte mit dem minimal einschließenden Rechteck der in *geometrie2* angegebenen Geometrie getestet werden soll.

geometrie2
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, deren minimal einschließendes Rechteck auf Schnittpunkte mit dem minimal einschließenden Rechteck der in *geometrie1* angegebenen Geometrie getestet werden soll.

Rückgabetyp

INTEGER

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung von ST_MBRIntersects zur Ermittlung einer Annäherung dazu, ob zwei sich nicht schneidende Polygone nahe beieinander liegen. Hierzu wird überprüft, ob ihre minimal einschließenden Rechtecke sich schneiden. Das erste Beispiel verwendet den SQL-Ausdruck CASE. Das zweite Beispiel verwendet eine einzelne Anweisung SELECT, um die Polygone zu finden, die das minimal einschließende Rechteck des Polygons mit der ID 2 schneiden.

```
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) )
```

Beispiel 2

Die folgende Anweisung SELECT verwendet einen Ausdruck CASE, um die IDs der Polygone zu bestimmen, deren minimal einschließende Rechtecke sich schneiden.

```
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
```

Ergebnisse:

```
ID         ID       MBR_INTERSECTS
---------- --------- --------------
         1         1 MBRs intersect
         1         2 MBRs intersect
         2         2 MBRs intersect
         1         3 MBRs do not intersect
         2         3 MBRs do not intersect
         3         3 MBRs intersect
```

Beispiel 3

Die folgende Anweisung SELECT ermittelt, ob die minimal einschließenden Rechtecke für die Geometrien das minimal einschließende Rechteck für das Polygon mit der ID 2 schneiden.

```
SELECT a.id, b.id, ST_MBRIntersects (a.geometry, b.geometry) MBR_INTERSECTS
  FROM sample_polys a, sample_polys b
    WHERE a.id = 2
```

Ergebnisse:

```
ID         ID           MBR_INTERSECTS
---------- ------------ ---------------
         2            1               1
         2            2               1
         2            3               0
```

Funktion ST_LocateBetween oder ST_MeasureBetween

Die Funktionen ST_LocateBetween oder ST_MeasureBetween verwenden eine Geometrie und zwei M-Koordinaten (Bemaßungen) als Eingabeparameter und geben den Teil der angegebenen Geometrie zurück, der die Gruppe nicht verbundener Pfade oder Punkte zwischen den beiden M-Koordinaten darstellt.

Bei Kurven, Mehrfachkurven, Oberflächen und Mehrfachoberflächen wird zur Ergebnisberechnung eine Interpolation ausgeführt. Die Ergebnisgeometrie wird im räumlichen Bezugssystem der angegebenen Geometrie dargestellt.

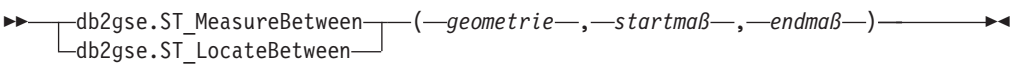
Wenn die angegebene Geometrie eine Oberfläche oder eine Mehrfachoberfläche ist, wird ST_MeasureBetween oder ST_LocateBetween auf den äußeren oder inneren Ring der Geometrie angewendet. Wenn kein Teil der angegebenen Geometrie sich

im durch die angegebenen M-Koordinaten definierten Intervall befindet, wird eine leere Geometrie zurückgegeben. Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Wenn die resultierende Geometrie nicht leer ist, wird ein Mehrpunkt- oder ein Mehrlinienfolgetyp zurückgegeben.

Beide Funktionen können auch als Methoden aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, in der sich die Teile mit Maßwerten zwischen dem Wert für *startmaß* und *endmaß* befinden.

startmaß
Ein Wert vom Typ DOUBLE, der die untere Grenze des Maßintervalls darstellt. Wenn dieser Wert gleich null ist, wird keine untere Grenze angewendet.

endmaß
Ein Wert vom Typ DOUBLE, der die obere Grenze für das Maßintervall darstellt. Wenn dieser Wert gleich null ist, wird keine obere Grenze angewendet.

Rückgabetyp

db2gse.ST_Geometry

Beispiel

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Die M-Koordinate (Bemaßung) einer Geometrie wird durch den Benutzer definiert. Die Grenze ist sehr vielseitig, da sie alles darstellen kann, was Sie messen möchten. Zum Beispiel Messungen des Abstands zu einer Autobahn, der Temperatur, des Drucks oder des pH-Werts.

Dieses Beispiel verdeutlicht die Verwendung der M-Koordinate zur Aufzeichnung gesammelter Daten von Messungen des pH-Wertes. Ein Forscher misst den pH-Wert des Bodens entlang einer Autobahn an bestimmten Stellen. Nach seiner Standardvorgehensweise schreibt er die Werte, die er benötigt, an jeder Stelle auf, an der er eine Messung durchführt: die X- und Y-Koordinaten des Ortes und den gemessenen pH-Wert.

```
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 ) )
```

Um den Bereich zu finden, in dem der pH-Wert zwischen 4 und 6 liegt, verwendet der Forscher die Anweisung SELECT:

```
SELECT id, CAST( ST_AsText( ST_MeasureBetween( 4, 6) )
    AS VARCHAR(150) ) MEAS_BETWEEN_4_AND_6
  FROM   sample_lines
```

Ergebnisse:

```
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)
```

Funktion ST_LocateBetween oder ST_MeasureBetween

Die Funktionen ST_LocateBetween oder ST_MeasureBetween verwenden eine Geometrie und zwei M-Koordinaten (Bemaßungen) als Eingabeparameter und geben den Teil der angegebenen Geometrie zurück, der die Gruppe nicht verbundener Pfade oder Punkte zwischen den beiden M-Koordinaten darstellt.

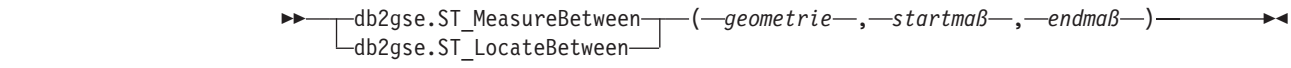
Bei Kurven, Mehrfachkurven, Oberflächen und Mehrfachoberflächen wird zur Ergebnisberechnung eine Interpolation ausgeführt. Die Ergebnisgeometrie wird im räumlichen Bezugssystem der angegebenen Geometrie dargestellt.

Wenn die angegebene Geometrie eine Oberfläche oder eine Mehrfachoberfläche ist, wird ST_MeasureBetween oder ST_LocateBetween auf den äußeren oder inneren Ring der Geometrie angewendet. Wenn kein Teil der angegebenen Geometrie sich im durch die angegebenen M-Koordinaten definierten Intervall befindet, wird eine leere Geometrie zurückgegeben. Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Wenn die resultierende Geometrie nicht leer ist, wird ein Mehrpunkt- oder ein Mehrlinienfolgetyp zurückgegeben.

Beide Funktionen können auch als Methoden aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, in der sich die Teile mit Maßwerten zwischen dem Wert für *startmaß* und *endmaß* befinden.

startmaß
Ein Wert vom Typ DOUBLE, der die untere Grenze des Maßintervalls darstellt. Wenn dieser Wert gleich null ist, wird keine untere Grenze angewendet.

endmaß
Ein Wert vom Typ DOUBLE, der die obere Grenze für das Maßintervall darstellt. Wenn dieser Wert gleich null ist, wird keine obere Grenze angewendet.

Rückgabetyp

db2gse.ST_Geometry

Beispiel

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Die M-Koordinate (Bemaßung) einer Geometrie wird durch den Benutzer definiert. Die Grenze ist sehr vielseitig, da sie alles darstellen kann, was Sie messen möchten. Zum Beispiel Messungen des Abstands zu einer Autobahn, der Temperatur, des Drucks oder des pH-Werts.

Dieses Beispiel verdeutlicht die Verwendung der M-Koordinate zur Aufzeichnung gesammelter Daten von Messungen des pH-Wertes. Ein Forscher misst den pH-Wert des Bodens entlang einer Autobahn an bestimmten Stellen. Nach seiner Standardvorgehensweise schreibt er die Werte, die er benötigt, an jeder Stelle auf, an der er eine Messung durchführt: die X- und Y-Koordinaten des Ortes und den gemessenen pH-Wert.

```
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 ) )
```

Um den Bereich zu finden, in dem der pH-Wert zwischen 4 und 6 liegt, verwendet der Forscher die Anweisung SELECT:

```
SELECT id, CAST( ST_AsText( ST_MeasureBetween( 4, 6) )
    AS VARCHAR(150) ) MEAS_BETWEEN_4_AND_6
  FROM   sample_lines
```

Ergebnisse:

```
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)
```

Funktion ST_MidPoint

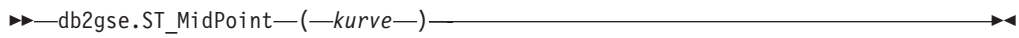
Die Funktion ST_MidPoint verwendet eine Kurve als Eingabeparameter und gibt den Punkt der Kurve zurück, der entlang der Kurve gemessen von beiden Endpunkten der Kurve gleich weit entfernt ist. Der Ergebnispunkt wird im räumlichen Bezugssystem der angegebenen Kurve dargestellt.

Wenn die angegebene Kurve leer ist, wird ein leerer Punkt zurückgegeben. Wenn die angegebene Kurve den Wert null aufweist, wird null zurückgegeben.

Wenn die Kurve Z- oder M-Koordinaten (Bemaßungen) enthält, wird der Mittelpunkt allein durch die Wert der X- und Y-Koordinaten in der Kurve ermittelt. Die Z-Koordinate und die Bemaßung im zurückgegebenen Punkt sind interpoliert.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

kurve Ein Wert vom Typ ST_Curve oder einer seiner untergeordneten Typen, der die Kurve darstellt, für die der Mittelpunkt zurückgegeben wird.

Rückgabetyp

db2gse.ST_Point

Beispiel

Dieses Beispiel verdeutlicht die Verwendung von ST_MidPoint, um den Mittelpunkt von Kurven zurückzugeben.

```
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
```

Ergebnisse:

```
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)
```

Funktion ST_MinM

Die Funktion ST_MinM verwendet eine Geometrie als Eingabeparameter und gibt deren kleinste M-Koordinate zurück.

Wenn die angegebene Geometrie den Wert null aufweist oder leer ist oder wenn sie keine M-Koordinaten aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, für den die kleinste M-Koordinate zurückgegeben wird.

Rückgabetyp

DOUBLE

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_MinM. Drei Polygone werden erstellt und in die Tabelle SAMPLE_POLYS eingefügt.

```
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) )
```

Beispiel 2

In diesem Beispiel wird die kleinste M-Koordinate jedes Polygons in SAMPLE_POLYS gesucht.

```
SELECT id, CAST ( ST_MinM(geometry) AS INTEGER) MIN_M
  FROM sample_polys
```

Ergebnisse:

```
ID          MIN_M
----------- ------------
          1          3
          2          5
          3         11
```

Beispiel 3

In diesem Beispiel wird die kleinste M-Koordinate aller Polygone in der Spalte GEOMETRY gesucht.

```
SELECT CAST ( MIN ( ST_MinM(geometry) ) AS INTEGER) OVERALL_MIN_M
  FROM sample_polys
```

Ergebnisse:

```
OVERALL_MIN_M
-------------
            3
```

Funktion ST_MinX

Die Funktion ST_MinX verwendet eine Geometrie als Eingabeparameter und gibt ihre kleinste X-Koordinate zurück.

Wenn die angegebene Geometrie den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, für den die kleinste X-Koordinate zurückgegeben wird.

Rückgabetyp

DOUBLE

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_MinX. Drei Polygone werden erstellt und in die Tabelle SAMPLE_POLYS eingefügt.

```
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) )
```

Beispiel 2

In diesem Beispiel wird die kleinste X-Koordinate jedes Polygons in SAMPLE_POLYS gesucht.

```
SELECT id, CAST ( ST_MinX(geometry) AS INTEGER) MIN_X
  FROM sample_polys
```

Ergebnisse:

```
ID          MIN_X
----------- -----------
          1         110
          2           0
          3           8
```

Beispiel 3

In diesem Beispiel wird die kleinste X-Koordinate aller Polygone in der Spalte GEOMETRY gesucht.

```
SELECT CAST ( MIN ( ST_MinX(geometry) ) AS INTEGER) OVERALL_MIN_X
  FROM sample_polys
```

Ergebnisse:

```
OVERALL_MIN_X
-------------
            0
```

Funktion ST_MinY

Die Funktion ST_MinY verwendet eine Geometrie als Eingabeparameter und gibt deren kleinste Y-Koordinate zurück.

Wenn die angegebene Geometrie den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, für den die kleinste Y-Koordinate zurückgegeben wird.

Rückgabetyp

DOUBLE

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_MinY. Drei Polygone werden erstellt und in die Tabelle SAMPLE_POLYS eingefügt.

```
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) )
```

Beispiel 2

In diesem Beispiel wird die kleinste Y-Koordinate jedes Polygons in SAMPLE_POLYS gesucht.

```
SELECT id, CAST ( ST_MinY(geometry) AS INTEGER) MIN_Y
  FROM sample_polys
```

Ergebnisse:

```
ID          MIN_Y
----------- ------------
          1          120
          2            0
          3            4
```

Beispiel 3

In diesem Beispiel wird die kleinste Y-Koordinate für alle Polygone in der Spalte GEOMETRY gesucht.

```
SELECT CAST ( MIN ( ST_MinY(geometry) ) AS INTEGER) OVERALL_MIN_Y
  FROM sample_polys
```

Ergebnisse:

```
OVERALL_MIN_Y
--------------
            0
```

Funktion ST_MinZ

Die Funktion ST_MinZ verwendet eine Geometrie als Eingabeparameter und gibt ihre kleinste Z-Koordinate zurück.

Wenn die angegebene Geometrie den Wert null aufweist oder leer ist oder wenn sie keine Z-Koordinaten aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, für den die kleinste Z-Koordinate zurückgegeben wird.

Rückgabetyp

DOUBLE

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_MinZ. Drei Polygone werden erstellt und in die Tabelle SAMPLE_POLYS eingefügt.

```
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) )
```

Beispiel 2

In diesem Beispiel wird die kleinste Z-Koordinate jedes einzelnen Polygons in SAMPLE_POLYS gesucht.

```
SELECT id, CAST ( ST_MinZ(geometry) AS INTEGER) MIN_Z
  FROM sample_polys
```

Ergebnisse:

```
ID          MIN_Z
----------- ------------
          1           20
          2           31
          3           10
```

Beispiel 3

In diesem Beispiel wird die kleinste Z-Koordinate aller Polygone in der Spalte GEOMETRY gesucht.

```
SELECT CAST ( MIN ( ST_MinZ(geometry) ) AS INTEGER) OVERALL_MIN_Z
  FROM sample_polys
```

Ergebnisse:

```
OVERALL_MIN_Z
--------------
            10
```

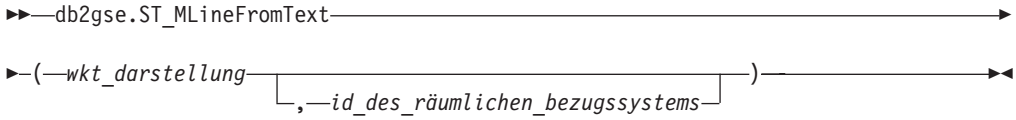
Funktion ST_MLineFromText

Die Funktion ST_MLineFromText verwendet eine WKT-Darstellung (WKT = Well-Known Text) einer Mehrlinienfolge und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt die entsprechende Mehrlinienfolge zurück.

Wenn die angegebene WKT-Darstellung den Wert null aufweist, wird null zurückgegeben.

Um dasselbe Ergebnis zu erreichen, wird die Funktion ST_MultiLineString empfohlen. Sie wird wegen ihrer Flexibilität empfohlen: ST_MultiLineString verwendet neben der WKT-Darstellung zusätzliche Formen der Eingabe.

Syntax



Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung der Ergebnismehrlinienfolge enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für die Ergebnismehrlinienfolge kennzeichnet.

Wenn der Paramater *id_des_räumlichen_bezugssystems* ausgelassen wird, wird das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem kennzeichnet, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, wird eine Ausnahmebedingung (SQLSTATE 38SU1) erzeugt.

Rückgabetyp

db2gse.ST_MultiLineString

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie ST_MLineFromText zur Erstellung und zum Einfügen einer Mehrlinienfolge aus der zugehörigen WKT-Darstellung (WKT = Well-Known Text) verwendet werden kann. Der eingefügte Datensatz weist die ID 1110 auf, und die Geometrie ist eine Mehrlinienfolge im räumlichen Bezugssystem 1. Die Mehrlinienfolge ist in der WKT-Darstellung einer Mehrlinienfolge definiert. Die X- und Y-Koordinaten für diese Geometrie lauten:

- Zeile 1: (33, 2) (34, 3) (35, 6)

- Zeile 2: (28, 4) (29, 5) (31, 8) (43, 12)
- Zeile 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) )
```

Die folgende Anweisung SELECT gibt die Mehrlinienfolge zurück, die in der Tabelle aufgezeichnet wurde:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(280) ) MULTI_LINE_STRING
  FROM   sample_mlines
  WHERE  id = 1110
```

Ergebnisse:

```
ID         MULTI_LINE_STRING
---------- ------------------------------------------------------------------------
      1110 MULTILINESTRING (( 33.00000000 2.00000000, 34.00000000 3.00000000,
                               35.00000000 6.00000000),
                             ( 28.00000000 4.00000000, 29.00000000 5.00000000,
                            31.00000000 8.0000000, 43.00000000 12.00000000),
                           ( 39.00000000 3.00000000, 37.00000000 4.00000000,
                             36.00000000 7.00000000 ))
```

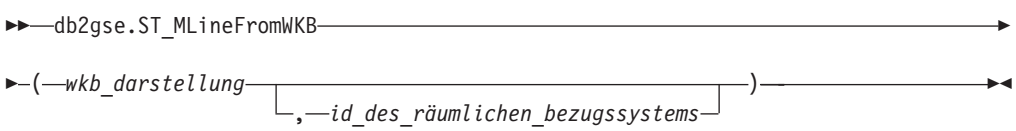
Funktion ST_MLineFromWKB

Die Funktion ST_MLineFromWKB verwendet eine WKB-Darstellung (WKB = Well-Known Binary) einer Mehrlinienfolge und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt die entsprechende Mehrlinienfolge zurück.

Wenn die angegebene WKB-Darstellung den Wert null aufweist, wird null zurückgegeben.

Um dasselbe Ergebnis zu erreichen, wird die Funktion ST_MultiLineString empfohlen. Sie wird wegen ihrer Flexibilität empfohlen: ST_MultiLineString verwendet neben der WKB-Darstellung zusätzliche Formen der Eingabe.

Syntax



Parameter

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung der Ergebnismehrlinienfolge enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für die Ergebnismehrlinienfolge kennzeichnet.

Wenn der Paramater *id_des_räumlichen_bezugssystems* ausgelassen wird, wird das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem kennzeichnet, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, wird eine Ausnahmebedingung (SQLSTATE 38SU1) erzeugt.

Rückgabetyp

db2gse.ST_MultiLineString

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie ST_MLineFromWKB zur Erstellung einer Mehrlinienfolge aus der zugehörigen WKB-Darstellung verwendet werden kann. Die Geometrie ist eine Mehrlinienfolge im räumlichen Bezugssystem 1. In diesem Beispiel wird die Mehrlinienfolge mit der ID 10 in der Spalte GEOMETRY der Tabelle SAMPLE_MLINES gespeichert. Anschließend wird die Spalte WKB unter Verwendung der Funktion ST_AsBinary mit der WKB-Darstellung aktualisiert. Schließlich wird die Funktion ST_MLineFromWKB verwendet, um die Mehrlinienfolge aus der Spalte WKB zurückzugeben. Die X- und Y-Koordinaten für diese Geometrie lauten:

- Linie 1: (61, 2) (64, 3) (65, 6)
- Linie 2: (58, 4) (59, 5) (61, 8)
- Linie 3: (69, 3) (67, 4) (66, 7) (68, 9)

Die Tabelle SAMPLE_MLINES verfügt über eine Spalte GEOMETRY, in der die Mehrlinienfolge gespeichert ist, und eine Spalte WKB, in der die WKB-Darstellung der Mehrlinienfolge gespeichert ist.

```
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
```

In der folgenden Anweisung SELECT wird die Funktion ST_MLineFromWKB verwendet, um die Mehrlinienfolge aus der Spalte WKB abzurufen.

```
SELECT id, CAST( ST_AsText( ST_MLineFromWKB (wkb) )
              AS VARCHAR(280) ) MULTI_LINE_STRING
  FROM   sample_mlines
    WHERE id = 10
```

Ergebnisse:

```
ID         MULTI_LINE_STRING
---------- ------------------------------------------------------------------
        10 MULTILINESTRING (( 61.00000000 2.00000000, 64.00000000 3.00000000,
                              65.00000000 6.00000000),
                              ( 58.00000000 4.00000000, 59.00000000 5.00000000,
                              61.00000000 8.0000000),
                              ( 69.00000000 3.00000000, 67.00000000 4.00000000,
                              66.00000000 7.00000000, 68.00000000 9.00000000 ))
```

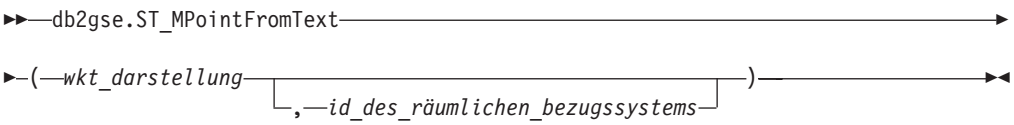
Funktion ST_MPointFromText

Die Funktion ST_MPointFromText verwendet eine WKT-Darstellung (WKT = Well-Known Text) einer Mehrpunktangabe und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt die entsprechende Mehrpunktangabe zurück.

Wenn die angegebene WKT-Darstellung den Wert null aufweist, wird null zurückgegeben.

Um dasselbe Ergebnis zu erreichen, wird die Funktion ST_MultiPoint empfohlen. Sie wird wegen ihrer Flexibilität empfohlen: ST_MultiPoint verwendet neben der WKT-Darstellung zusätzliche Formen der Eingabe.

Syntax



Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung (WKT = Well-Known Text) der Ergebnismehrpunktangabe enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für die Ergebnismehrpunktangabe kennzeichnet.

Wenn der Paramater *id_des_räumlichen_bezugssystems* ausgelassen wird, wird das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem kennzeichnet, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, wird eine Ausnahmebedingung (SQLSTATE 38SU1) erzeugt.

Rückgabetyp

db2gse.ST_MultiPoint

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie ST_MPointFromText zur Erstellung und zum Einfügen einer Mehrpunktangabe aus der zugehörigen WKT-Darstellung (WKT = Well-Known Text) verwendet werden kann. Der eingefügte Datensatz weist die ID 1110 auf, und die Geometrie ist eine Mehrpunktangabe im räumlichen Bezugssystem 1. Die Mehrpunktangabe ist in der WKT-Darstellung einer Mehrpunktangabe definiert. Die X- und Y-Koordinaten für diese Geometrie lauten: (1, 2) (4, 3) (5, 6).

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpoints (id INTEGER, geometry ST_MultiPoint)

INSERT INTO sample_mpoints
    VALUES (1110, ST_MPointFromText ('multipoint (1 2, 4 3, 5 6) )', 1) )
```

Die folgende Anweisung SELECT gibt die Mehrpunktangabe zurück, die in der Tabelle aufgezeichnet wurde:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(280) ) MULTIPOINT
    FROM sample_mpoints
  WHERE  id = 1110
```

Ergebnisse:

```
ID         MULTIPOINT
---------- ----------------------------------------------------------------------
      1110 MULTIPOINT (1.00000000 2.00000000, 4.00000000 3.00000000,
                       5.00000000 6.00000000)
```

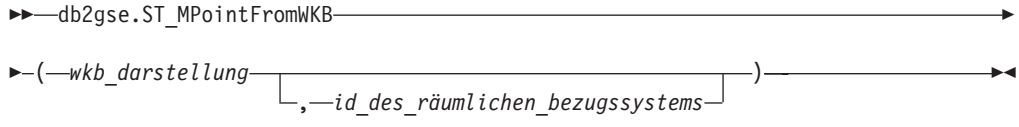
Funktion ST_MPointFromWKB

Die Funktion ST_MPointFromWKB verwendet eine WKB-Darstellung (WKB = Well-Known Binary) einer Mehrpunktangabe und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt die entsprechende Mehrpunktangabe zurück.

Wenn die angegebene WKB-Darstellung den Wert null aufweist, wird null zurückgegeben.

Um dasselbe Ergebnis zu erreichen, wird die Funktion ST_MultiPoint empfohlen. Sie wird wegen ihrer Flexibilität empfohlen: ST_MultiPoint verwendet neben der WKB-Darstellung zusätzliche Formen der Eingabe.

Syntax



Parameter

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung der Ergebnismehrpunktangabe enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für die Ergebnismehrpunktangabe kennzeichnet.

Wenn der Paramater *id_des_räumlichen_bezugssystems* ausgelassen wird, wird das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem kennzeichnet, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, wird eine Ausnahmebedingung (SQLSTATE 38SU1) erzeugt.

Rückgabetyp

db2gse.ST_MultiPoint

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie ST_MPointFromWKB zur Erstellung einer Mehrpunktangabe aus der zugehörigen WKB-Darstellung verwendet werden kann. Die Geometrie ist eine Mehrpunktangabe im räumlichen Bezugssystem 1. In diesem Beispiel wird die Mehrpunktangabe mit der ID 10 in der Spalte GEOMETRY der Tabelle SAMPLE_MPOINTS gespeichert. Anschließend wird die Spalte WKB unter Verwendung der Funktion ST_AsBinary mit der WKB-Darstellung aktualisiert. Schließlich wird die Funktion ST_MPointFromWKB verwendet, um die Mehrpunktangabe aus der Spalte WKB zurückzugeben. Die X- und Y-Koordinaten für diese Geometrie lauten: (44, 14) (35, 16) (24, 13).

Die Tabelle SAMPLE_MPOINTS verfügt über eine Spalte GEOMETRY, in der die Mehrpunktangabe gespeichert wird, und eine Spalte WKB, in der die WKB-Darstellung der Mehrpunktangabe gespeichert wird.

```
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
```

In der folgenden Anweisung SELECT wird die Funktion ST_MPointFromWKB verwendet, um die Mehrpunktangabe aus der Spalte WKB abzurufen.

```
SELECT id, CAST( ST_AsText( ST_MLineFromWKB (wkb)) AS VARCHAR(100)) MULTIPOINT
    FROM sample_mpoints
    WHERE id = 10
```

Ergebnisse:

```
ID         MULTIPOINT
---------- ----------------------------------------------------------------
        10 MULTIPOINT (44.00000000 14.00000000, 35.00000000
                       16.00000000 24.00000000 13.00000000)
```

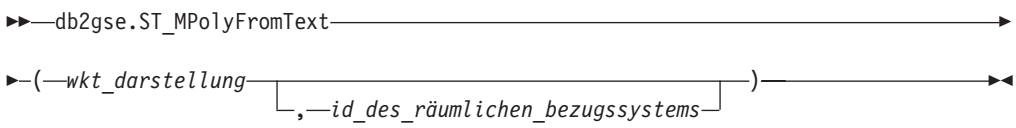
Funktion ST_MPolyFromText

Die Funktion ST_MPolyFromText verwendet eine WKT-Darstellung (WKT = Well-Known Text) eines Multipolygons und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt das entsprechende Multipolygon zurück.

Wenn die angegebene WKT-Darstellung den Wert null aufweist, wird null zurückgegeben.

Um dasselbe Ergebnis zu erreichen, wird die Funktion ST_MultiPolygon empfohlen. Sie wird wegen ihrer Flexibilität empfohlen: ST_MultiPolygon verwendet neben der WKT-Darstellung zusätzliche Formen der Eingabe.

Syntax



Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung des resultierenden Multipolygons enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem des resultierenden Multipolygons angibt.

Wenn der Paramater *id_des_räumlichen_bezugssystems* ausgelassen wird, wird das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem kennzeichnet, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, wird eine Ausnahmebedingung (SQLSTATE 38SU1) erzeugt.

Rückgabetyp

db2gse.ST_MultiPolygon

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie ST_MPolyFromText zur Erstellung und zum Einfügen eines Multipolygons aus der WKT-Darstellung (WKT = Well-Known Text) verwendet werden kann. Der eingefügte Datensatz weist die ID 1110 auf, und die Geometrie ist ein Multipolygon im räumlichen Bezugssystem 1. Das Multipolygon ist in der WKT-Darstellung eines Multipolygons definiert. Die X- und Y-Koordinaten für diese Geometrie lauten:

- Polygon 1: (3, 3) (4, 6) (5, 3) (3, 3)
- Polygon 2: (8, 24) (9, 25) (1, 28) (8, 24)
- Polygon 3: (13, 33) (7, 36) (1, 40) (10, 43) (13, 33)

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpolys (id INTEGER, geometry ST_MultiPolygon)

INSERT INTO sample_mpolys
    VALUES (1110,
```

```
                    ST_MPolyFromText ('multipolygon (( (3 3, 4 6, 5 3, 3 3),
                                                       (8 24, 9 25, 1 28, 8 24),
                                                       (13 33, 7 36, 1 40, 10 43 13 33) ))', 1) )
```

Die folgende Anweisung SELECT gibt das Multipolygon zurück, das in der Tabelle aufgezeichnet wurde.

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(350) ) MULTI_POLYGON
  FROM sample_mpolys
  WHERE  id = 1110
```

Ergebnisse:

```
ID      MULTI_POLYGON
------- ------------------------------------------------------------------
   1110 MULTIPOLYGON ((( 13.00000000 33.00000000, 10.00000000 43.00000000,
                         1.00000000 40.00000000, 7.00000000 36.00000000,
                        13.00000000 33.00000000)),
                        (( 8.00000000 24.00000000, 9.00000000 25.00000000,
                         1.00000000 28.0000000, 8.00000000 24.00000000)),
                       ( 3.00000000 3.00000000, 5.00000000 3.00000000,
                         4.00000000 6.00000000, 3.00000000 3.00000000)))
```

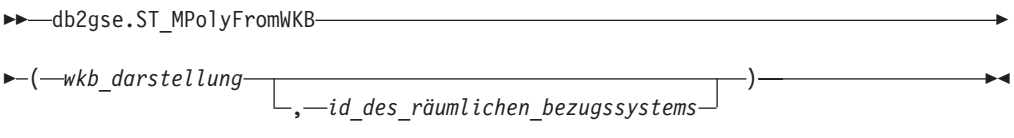
Funktion ST_MPolyFromWKB

Die Funktion ST_MPolyFromWKB verwendet eine WKB-Darstellung (WKB = Well-Known Binary) eines Multipolygons und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt das entsprechende Multipolygon zurück.

Wenn die angegebene WKB-Darstellung den Wert null aufweist, wird null zurückgegeben.

Um dasselbe Ergebnis zu erreichen, wird die Funktion ST_MultiPolygon empfohlen. Sie wird wegen ihrer Flexibilität empfohlen: ST_MultiPolygon verwendet neben der WKB-Darstellung (WKB = Well-Known Binary) zusätzliche Formen der Eingabe.

Syntax



Parameter

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung des resultierenden Multipolygons enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem des resultierenden Multipolygons angibt.

Wenn der Paramater *id_des_räumlichen_bezugssystems* ausgelassen wird, wird das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem kennzeichnet, das in der Katalogsicht

DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, wird eine Ausnahmebedingung (SQLSTATE 38SU1) erzeugt.

Rückgabetyp

db2gse.ST_MultiPolygon

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie ST_MPolyFromWKB zur Erstellung eines Multipolygons aus der WKB-Darstellung (WKB = Well-Known Binary) verwendet werden kann. Die Geometrie ist ein Multipolygon im räumlichen Bezugssystem 1. In diesem Beispiel wird das Multipolygon mit der ID 10 in der Spalte GEOMETRY der Tabelle SAMPLE_MPOLYS gespeichert. Anschließend wird die Spalte WKB unter Verwendung der Funktion ST_AsBinary mit den Daten der WKB-Darstellung (WKB = Well-Known Binary) aktualisiert. Schließlich wird die Funktion ST_MPolyFromWKB verwendet, um das Multipolygon aus der Spalte WKB zurückzugeben. Die X- und Y-Koordinaten für diese Geometrie lauten:

- 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)

Die Tabelle SAMPLE_MPOLYS verfügt über eine Spalte GEOMETRY, in der das Multipolygon gespeichert ist, und eine Spalte WKB, in der die WKB-Darstellung (WKB = Well-Known Binary) des Multipolygons gespeichert ist.

```
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
```

In der folgenden Anweisung SELECT wird die Funktion ST_MPolyFromWKB verwendet, um das Multipolygon aus der Spalte WKB abzurufen.

```
SELECT id, CAST( ST_AsText( ST_MPolyFromWKB (wkb) )
        AS VARCHAR(320) ) MULTIPOLYGON
  FROM sample_mpolys
    WHERE id = 10
```

Ergebnisse:

```
ID         MULTIPOLYGON
---------- ------------------------------------------------------------------
        10 MULTIPOLYGON ((( 10.00000000 20.00000000, 30.00000000
                            41.00000000, 10.00000000 40.00000000, 10.00000000
                            20.00000000)),
                          ( 1.00000000 72.00000000, 5.00000000
                            76.00000000, 4.00000000 79.0000000, 1.00000000
```

```
 72,00000000)),
( 9.00000000 43.00000000, 6.00000000
 47.00000000, 7.00000000 44.00000000, 9.00000000
 43.00000000 )))
```

Funktion ST_MultiLineString

Die Funktion ST_MultiLineString konstruiert eine Mehrlinienfolge aus einer Eingabe.

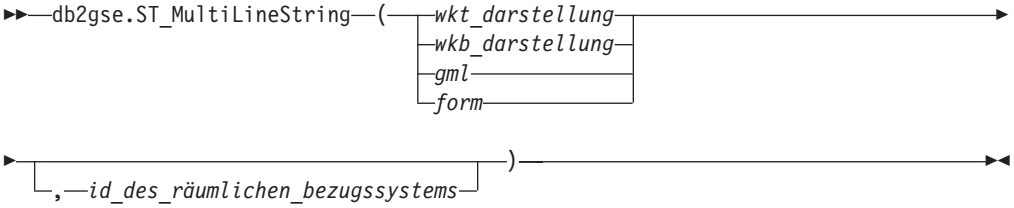
Die Eingabe kann in einem der folgenden Formate erfolgen:

- WKT-Darstellung
- WKB-Darstellung
- Formdarstellung
- Darstellung in GML (Geography Markup Language)

Eine Kennung für räumliche Bezugssysteme kann optional angegeben werden, um das räumliche Bezugssystem zu kennzeichnen, in dem sich die Ergebnismehrlinienfolge befindet.

Wenn die WKT-Darstellung (WKT = Well-Known Text), die WKB-Darstellung (WKB = Well-Known Binary), die Formdarstellung oder die GML-Darstellung null ist, wird null zurückgegeben.

Syntax



Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung der Ergebnismehrlinienfolge enthält.

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung der Ergebnismehrlinienfolge enthält.

gml Ein Wert vom Typ CLOB(2G), der die Ergebnismehrlinienfolge unter Verwendung von GML (Geography Markup Language) darstellt.

form Ein Wert vom Typ BLOB(2G), der die Formdarstellung der Ergebnismehrlinienfolge darstellt.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für die Ergebnismehrlinienfolge kennzeichnet.

Falls der Parameter *id_des_räumlichen_bezugssystems* nicht angegeben wird, wird das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben wird, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, gibt das System eine Ausnahmebedingung (SQLSTATE 38SU1) aus.

Rückgabetyp

db2gse.ST_MultiLineString

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie ST_MultiLineString zur Erstellung und zum Einfügen einer Mehrlinienfolge aus der zugehörigen WKT-Darstellung (WKT = Well-Known Text) verwendet werden kann. Der eingefügte Datensatz weist die ID 1110 auf, und die Geometrie ist eine Mehrlinienfolge im räumlichen Bezugssystem 1. Die Mehrlinienfolge ist in der WKT-Darstellung einer Mehrlinienfolge definiert. Die X- und Y-Koordinaten für diese Geometrie lauten:

- Zeile 1: (33, 2) (34, 3) (35, 6)
- Zeile 2: (28, 4) (29, 5) (31, 8) (43, 12)
- Zeile 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) )
```

Die folgende Anweisung SELECT gibt die Mehrlinienfolge zurück, die in der Tabelle aufgezeichnet wurde:

```
SELECT id,
              CAST( ST_AsText( geometry ) AS VARCHAR(280) )
    MULTI_LINE_STRING
  FROM   sample_mlines
  WHERE  id = 1110
```

Ergebnisse:

```
ID         MULTI_LINE_STRING
------- ------------------------------------------------------------------
   1110 MULTILINESTRING (( 33.00000000 2.00000000, 34.00000000 3.00000000,
                           35.00000000 6.00000000),
                        ( 28.00000000 4.00000000, 29.00000000 5.00000000,
                           31.00000000 8.0000000, 43.00000000 12.00000000),
                        ( 39.00000000 3.00000000, 37.00000000 4.00000000,
                          36.00000000 7.00000000 ))
```

Funktion ST_MultiPoint

Die Funktion ST_MultiPoint konstruiert eine Mehrpunktangabe aus einer Eingabe.

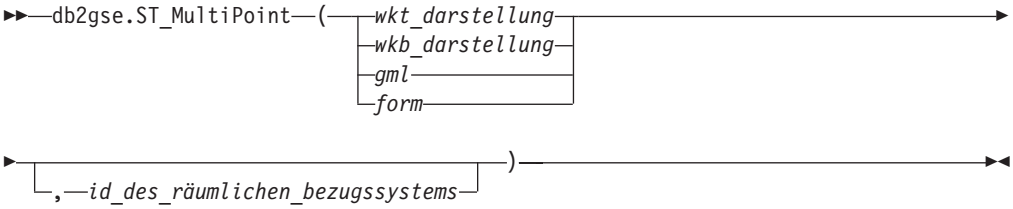
Die Eingabe kann in einem der folgenden Formate erfolgen:

- WKT-Darstellung
- WKB-Darstellung
- Formdarstellung
- Darstellung in GML (Geography Markup Language)

Eine Kennung für räumliche Bezugssysteme kann optional angegeben werden, um das räumliche Bezugssystem zu kennzeichnen, in dem sich die Ergebnismehrpunktangabe befindet.

Wenn die WKT-Darstellung (WKT = Well-Known Text), die WKB-Darstellung (WKB = Well-Known Binary), die Formdarstellung oder die GML-Darstellung null ist, wird null zurückgegeben.

Syntax



Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung (WKT = Well-Known Text) der Ergebnismehrpunktangabe enthält.

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung der Ergebnismehrpunktangabe enthält.

gml Ein Wert vom Typ CLOB(2G), der die Ergebnismehrpunktangabe unter Verwendung von GML (Geography Markup Language) darstellt.

form Ein Wert vom Typ BLOB(2G), der die Formdarstellung der Ergebnismehrpunktangabe darstellt.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für die Ergebnismehrpunktangabe kennzeichnet.

Wenn der Paramater *id_des_räumlichen_bezugssystems* ausgelassen wird, wird das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben wird, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, gibt das System eine Ausnahmebedingung (SQLSTATE 38SU1) aus.

Rückgabetyp

db2gse.ST_Point

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie ST_MultiPoint zur Erstellung und zum Einfügen einer Mehrpunktangabe aus der zugehörigen WKT-Darstellung (WKT = Well-Known Text) verwendet werden kann. Der eingefügte Datensatz weist die ID 1110 auf, und die Geometrie ist eine Mehrpunktangabe im räumlichen Bezugssystem 1. Die Mehrpunktangabe ist in der WKT-Darstellung einer Mehrpunktangabe definiert. Die X- und Y-Koordinaten für diese Geometrie lauten: (1, 2) (4, 3) (5, 6).

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpoints (id INTEGER, geometry ST_MultiPoint)

INSERT INTO sample_mpoints
    VALUES (1110, ST_MultiPoint ('multipoint (1 2, 4 3, 5 6) )', 1))
```

Die folgende Anweisung SELECT gibt die Mehrpunktangabe zurück, die in der Tabelle aufgezeichnet wurde:

```
SELECT id, CAST( ST_AsText(geometry) AS VARCHAR(90)) MULTIPOINT
    FROM sample_mpoints
  WHERE  id = 1110
```

Ergebnisse:

```
ID         MULTIPOINT
---------- ----------------------------------------------------
      1110 MULTIPOINT (1.00000000 2.00000000, 4.00000000
           3.00000000, 5.00000000 6.00000000)
```

Funktion ST_MultiPolygon

Die Funktion ST_MultiPolygon konstruiert ein Multipolygon aus einer Eingabe.

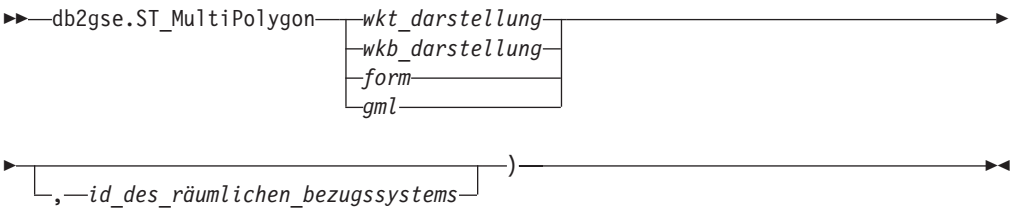
Die Eingabe kann in einem der folgenden Formate erfolgen:

- WKT-Darstellung
- WKB-Darstellung
- Formdarstellung
- Darstellung in GML (Geography Markup Language)

Eine Kennung für räumliche Bezugssysteme kann optional angegeben werden, um das räumliche Bezugssystem zu kennzeichnen, in dem sich das resultierende Multipolygon befindet.

Wenn die WKT-Darstellung (WKT = Well-Known Text), die WKB-Darstellung (WKB = Well-Known Binary), die Formdarstellung oder die GML-Darstellung null ist, wird null zurückgegeben.

Syntax



Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung des resultierenden Multipolygons enthält.

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung des resultierenden Multipolygons enthält.

gml Ein Wert vom Typ CLOB(2G), der das resultierende Multipolygon im GML-Format (GML = Geography Markup Language) darstellt.

form Ein Wert vom Typ BLOB(2G), der die Formdarstellung des resultierenden Multipolygons darstellt.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem des resultierenden Multipolygons angibt.

Wenn der Paramater *id_des_räumlichen_bezugssystems* ausgelassen wird, wird das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben wird, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, gibt das System eine Ausnahmebedingung (SQLSTATE 38SU1) aus.

Rückgabetyp

db2gse.ST_MultiPolygon

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie ST_MultiPolygon zur Erstellung und zum Einfügen eines Multipolygons aus der WKT-Darstellung (WKT = Well-Known Text) verwendet werden kann. Der eingefügte Datensatz weist die ID 1110 auf, und die Geometrie ist ein Multipolygon im räumlichen Bezugssystem 1. Das Multipolygon ist in der WKT-Darstellung eines Multipolygons definiert. Die X- und Y-Koordinaten für diese Geometrie lauten:

- 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) )
```

Die folgende Anweisung SELECT gibt das Multipolygon zurück, das in der Tabelle aufgezeichnet wurde.

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(350) ) MULTI_POLYGON
  FROM sample_mpolys
  WHERE  id = 1110
```

Ergebnisse:

```
ID         MULTI_POLYGON
------- ------------------------------------------------------------------
   1110 MULTIPOLYGON ((( 13.00000000 33.00000000, 10.00000000 43.00000000,
                         1.00000000 40.00000000, 7.00000000 36.00000000,
                        13.00000000 33.00000000)),
                         (( 8.00000000 24.00000000, 9.00000000 25.00000000,
                          1.00000000 28.0000000, 8.00000000 24.00000000)),
                         (( 3.00000000 3.00000000, 5.00000000 3.00000000,
                          4.00000000 6.00000000, 3.00000000 3.00000000)))
```

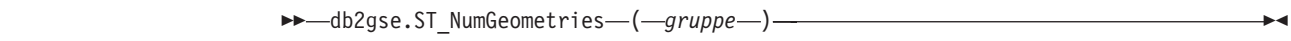
Funktion ST_NumGeometries

Die Funktion ST_NumGeometries verwendet eine Geometriengruppe als Eingabeparameter und gibt die Anzahl der Geometrien in der Gruppe zurück.

Wenn die angegebene Geometriengruppe den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

gruppe
Ein Wert vom Typ ST_GeomCollection oder einer seiner untergeordneten Typen, der die Geometriengruppe darstellt, für die die Anzahl der Geometrien zurückgegeben wird.

Rückgabetyp

INTEGER

Beispiel

Die Geometriengruppen werden in der Tabelle SAMPLE_GEOMCOLL gespeichert. Bei einer handelt es sich um ein Multipolygon, bei der anderen um eine Mehrpunktangabe. Die Funktion ST_NumGeometries ermittelt, wie viele einzelne Geometrien sich in jeder Geometriengruppe befinden.

```
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
```

Ergebnisse:

```
ID          NUM_GEOMS_IN_COLL
----------- -----------------
          1                 3
          2                 5
```

Funktion ST_NumInteriorRing

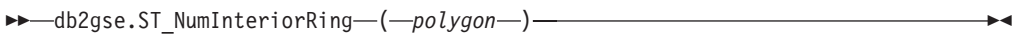
Die Funktion ST_NumInteriorRing verwendet als Eingabeparameter ein Polygon und gibt die Anzahl der inneren Ringe dieses Polygons zurück.

Wenn das angegebene Polygon den Wert null aufweist oder leer ist, wird null zurückgegeben.

Wenn das Polygon nicht über innere Ringe verfügt, wird 0 (null) zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

polygon
Ein Wert vom Typ ST_Polygon, der das Polygon darstellt, für das die Anzahl der inneren Ringe zurückgegeben wird.

Rückgabetyp

INTEGER

Beispiel

Im folgenden Beispiel werden zwei Polygone erstellt:

- Ein Polygon mit zwei inneren Ringen
- Ein Polygon ohne innere Ringe

```
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) )
```

Die Funktion ST_NumInteriorRing wird verwendet, um die Anzahl der Ringe in den Geometrien in der Tabelle zurückzugeben:

```
SELECT id, ST_NumInteriorRing(geometry) NUM_RINGS
FROM sample_polys
```

Ergebnisse:

```
ID         NUM_RINGS
---------- --------------
         1              2
         2              0
```

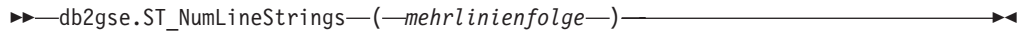
Funktion ST_NumLineStrings

Die Funktion ST_NumLineStrings verwendet eine Mehrlinienfolge als Eingabeparameter und gibt die Anzahl der darin enthaltenen Linienfolgen zurück.

Wenn die angegebene Mehrlinienfolge den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

mehrlinienfolge
Ein Wert vom Typ ST_MultiLineString, der die Mehrlinienfolge darstellt, für die die Anzahl der Linienfolgen zurückgegeben wird.

Rückgabetyp

INTEGER

Beispiel

Mehrlinienfolgen werden in der Tabelle SAMPLE_MLINES gespeichert. Die Funktion ST_NumLineStrings ermittelt, wie viele einzelne Geometrien sich in jeder Mehrlinienfolge befinden.

```
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
```

Ergebnisse:

```
ID          NUM_WITHIN
----------- ----------
        110          3
        111          2
```

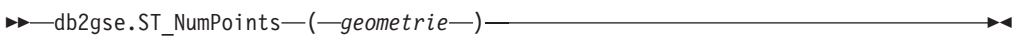
Funktion ST_NumPoints

Die Funktion ST_NumPoints verwendet eine Geometrie als Eingabeparameter und gibt die Anzahl der Punkte zurück, die zur Definition dieser Geometrie verwendet wurden. Wenn die Geometrie zum Beispiel ein Polygon ist und fünf Punkte für die Definition dieses Polygons verwendet wurden, wird die Zahl 5 zurückgegeben.

Wenn die angegebene Geometrie den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, für die die Anzahl der Punkte zurückgegeben wird.

Rückgabetyp

INTEGER

Beispiel

Mehrere Geometrien werden in der Tabelle gespeichert. Die Funktion ST_NumPoints ermittelt, wie viele Punkte sich in jeder Geometrie in der Tabelle SAMPLE_GEOMETRIES befinden.

```
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
```

Ergebnisse:

```
SPATIAL_TYPE    NUM_POINTS
--------------- ----------
st_point                 1
st_linestring            3
st_polygon               4
```

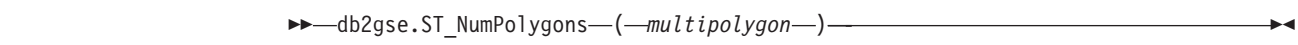
Funktion ST_NumPolygons

Die Funktion ST_NumPolygons verwendet ein Multipolygon als Eingabeparameter und gibt die Anzahl der Polygone zurück, die dieses Multipolygon enthält.

Wenn das angegebene Multipolygon den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

multipolygon
Ein Wert vom Typ ST_MultiPolygon, der das Multipolygon darstellt, für das die Anzahl der Polygone zurückgegeben wird.

Rückgabetyp

INTEGER

Beispiel

Multipolygone werden in der Tabelle SAMPLE_MPOLYS gespeichert. Die Funktion ST_NumPolygons ermittelt, wie viele einzelne Geometrien sich in jedem Multipolygon befinden.

```
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
```

Ergebnisse:

```
ID          NUM_WITHIN
----------- ----------
          1          3
          2          0
          3          2
```

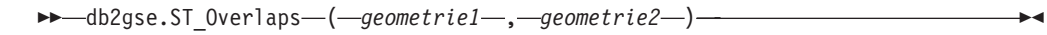
Funktion ST_Overlaps

Die Funktion ST_Overlaps verwendet zwei Geometrien als Eingabeparameter. Wenn die Schnittmenge der Geometrien eine Geometrie derselben Dimension zum Ergebnis hat, aber nicht mit einer der beiden angegebenen Geometrien identisch ist, wird 1 zurückgegeben. Andernfalls wird 0 (null) zurückgegeben.

Wenn eine der beiden Geometrien den Wert null aufweist oder leer ist, wird null zurückgegeben.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist, wird sie in das andere räumliche Bezugssystem umgewandelt.

Syntax



Parameter

geometrie1
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die auf Überlappungen mit der in *geometrie2* angegebenen Geometrie getestet wird.

geometrie2
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die auf Überlappungen mit der in *geometrie1* angegebenen Geometrie getestet wird.

Rückgabetyp

INTEGER

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung von ST_Overlaps. Verschiedene Geometrien werden erstellt und in die Tabelle SAMPLE_GEOMETRIES eingefügt.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
    VALUES
       (1, ST_Point(10, 20, 1)),
```

```
        (2, ST_Point ('point (41 41)', 1) ),
        (10, ST_LineString ('linestring (1 10, 3 12, 10 10)', 1) ),
        (20, ST_LineString ('linestring (50 10, 50 12, 45 10)', 1) ),
        (30, ST_LineString ('linestring (50 12, 50 10, 60 8)', 1) ),
        (100, ST_Polygon ('polygon ((0 0, 0 40, 40 40, 40 0, 0 0))', 1) ),
        (110, ST_Polygon ('polygon ((30 10, 30 30, 50 30, 50 10, 30 10))', 1) ),
        (120, ST_Polygon ('polygon ((0 50, 0 60, 40 60, 40 60, 0 50))', 1) )
```

Beispiel 2

In diesem Beispiel werden die IDs der überlappenden Punkte gesucht.

```
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
```

Ergebnisse:

```
ID          ID         OVERLAP
----------- ---------- ------------------------
          1          1 Points_do_not_overlap
          2          1 Points_do_not_overlap
          2          2 Points_do_not_overlap
```

Beispiel 3

In diesem Beispiel werden die IDs der überlappenden Linien gesucht.

```
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
```

Ergebnisse:

```
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
```

Beispiel 4

In diesem Beispiel werden die IDs der überlappenden Polygone gesucht.

```
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
```

Ergebnisse:

```
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
```

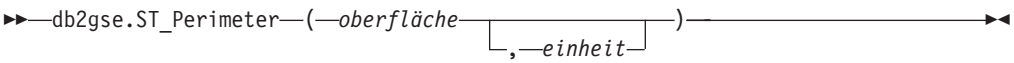
Funktion ST_Perimeter

Die Funktion ST_Perimeter verwendet eine Oberfläche oder Mehrfachoberfläche und optional eine Einheit als Eingabeparameter, und gibt den Umfang der Oberfläche oder Mehrfachoberfläche, d. h. die Länge ihrer Begrenzung, gemessen in den Standardeinheiten oder den angegebenen Einheiten zurück.

Wenn die angegebene Oberfläche oder Mehrfachoberfläche den Wert null aufweist oder leer ist, wird 0 (null) zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

oberfläche
Ein Wert vom Typ ST_Surface, ST_MultiSurface oder einer der zugehörigen untergeordneten Typen, für den der Umfang zurückgegeben wird.

einheit
Ein Wert vom Typ VARCHAR(128), der die Einheiten kennzeichnet, in denen der Umfang gemessen wird. Die unterstützten Maßeinheiten sind in der Katalogsicht DB2GSE.ST_UNITS_OF_MEASURE aufgelistet.

Wenn der Parameter *einheit* ausgelassen wird, werden die folgenden Regeln verwendet, um zu ermitteln, in welcher Einheit der Umfang gemessen werden soll:

- Wenn die in *oberfläche* angegebene Oberfläche sich in einem projizierten oder geozentrischen Koordinatensystem befindet, wird als Standardwert die lineare Einheit verwendet, die diesem Koordinatensystem zugeordnet ist.
- Wenn die *oberfläche* sich in einem geografischen Koordinatensystem befindet, wird als Standardwert die Winkeleinheit verwendet, die diesem Koordinatensystem zugeordnet ist.

Einschränkungen bei Einheitenumsetzungen: Ein Fehler (SQLSTATE 38SU4) wird zurückgegeben, wenn eine der folgenden Bedingungen zutrifft:

- Die Geometrie befindet sich in einem nicht definierten Koordinatensystem, und der Parameter *einheit* wurde angegeben.
- Die Geometrie befindet sich in einem projizierten Koordinatensystem, und es wurde eine Winkeleinheit angegeben.
- Die Geometrie befindet sich in einem geografischen Koordinatensystem, und es wurde eine lineare Einheit angegeben.

Rückgabetyp

DOUBLE

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_Perimeter. Ein räumliches Bezugssystem mit der ID 4000 wird mithilfe eines Aufrufs von db2se erstellt. In diesem räumlichen Bezugssystem wird ein Polygon erstellt.

```
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
```

Die Tabelle SAMPLE_POLYS wird erstellt, um eine Geometrie mit einem Umfang von 18 aufzunehmen.

```
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))
```

Beispiel 2

In diesem Beispiel wird die ID und der Umfang des Polygons aufgelistet.

```
SELECT id, ST_Perimeter (geometry) AS PERIMETER
  FROM sample_polys
```

Ergebnisse:

```
ID          PERIMETER
----------- ------------------------
          1 +1.80000000000000E+001
```

Beispiel 3

In diesem Beispiel wird die ID und der Umfang des Polygons mit dem in Metern gemessenen Umfang aufgelistet.

```
SELECT id, ST_Perimeter (geometry, 'METER') AS PERIMETER_METER
  FROM sample_polys
```

Ergebnisse:

```
ID          PERIMETER_METER
----------- ------------------------
          1 +5.48641097282195E+000
```

Funktion ST_PerpPoints

Die Funktion ST_PerpPoints verwendet eine Kurve oder Mehrfachkurve und einen Punkt als Eingabeparameter und gibt die winkeltreue Projektion des angegebenen Punktes auf der Kurve oder Mehrfachkurve zurück.

Der Punkt mit dem kleinsten Abstand zwischen dem angegebenen Punkt und dem winkeltreu projizierten Punkt wird zurückgegeben. Wenn zwei oder mehr dieser winkeltreu projizierten Punkte den gleichen Abstand zum angegebenen Punkt aufweisen, werden alle diese Punkte zurückgegeben. Wenn kein Punkt der winkeltreuen Projektion konstruiert werden kann, wird ein leerer Punkt zurückgegeben.

Wenn die angegebene Kurve oder Mehrfachkurve Z- oder M-Koordinaten aufweist, werden die Z- oder M-Koordinaten der Ergebnispunkte durch Interpolation auf der angegebenen Kurve oder Mehrfachkurve berechnet.

Wenn die angegebene Kurve oder der angegebene Punkt leer ist, wird ein leerer Punkt zurückgegeben. Wenn die angegebene Kurve oder der angegebene Punkt den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

kurve Ein Wert vom Typ ST_Curve, ST_MultiCurve oder einer der zugehörigen untergeordneten Typen, der die Kurve oder Mehrfachkurve darstellt, in der die winkeltreue Projektion des in *punkt* angegebenen Punktes zurückgegeben wird.

punkt Ein Wert vom Typ ST_Point, der den Punkt darstellt, der auf der in *kurve* angegebenen Kurve winkeltreu projiziert wird.

Rückgabetyp

db2gse.ST_MultiPoint

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_PerpPoints zum Suchen der Punkte, die orthogonal zu der Linienfolge liegen, die in der folgenden Tabelle gespeichert ist. Die Funktion ST_LineString wird in der Anweisung INSERT verwendet, um die Linienfolge zu erstellen.

```
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) )
```

Beispiel 2

In diesem Beispiel wird die winkeltreue Projektion auf der Linienfolge eines Punktes mit den Koordinaten (5, 0) gesucht. Die Funktion ST_AsText wird verwendet, um den zurückgegebenen Wert (eine Mehrpunktangabe) in die zugehörige WKT-Darstellung (WKT = Well-Known Text) umzuwandeln.

```
SELECT CAST(ST_AsText(ST_PerpPoints(line,ST_Point(5,0)))
    AS VARCHAR(50) ) PERP
FROM   sample_lines
```

Ergebnisse:

```
PERP
----------------------------------------
MULTIPOINT(5.00000000 0.00000000)
```

Beispiel 3

In diesem Beispiel wird die winkeltreue Projektion auf der Linienfolge eines Punktes mit den Koordinaten (5, 5) gesucht. In diesem Fall gibt es drei Punkte auf der Linienfolge, die denselben Abstand zur angegebenen Position aufweisen. Daher wird eine Mehrpunktangabe mit allen drei Punkten zurückgegeben.

```
SELECT  CAST(ST_AsText(ST_PerpPoints(line, ST_Point(5,5)))
  AS VARCHAR160)) PERP
  FROM   sample_lines
```

Ergebnisse:

```
PERP
--------------------------------------------------------------------------------
MULTIPOINT(0.00000000 5.00000000,5.00000000 0.00000000,10.00000000 5.00000000)
```

Beispiel 4

In diesem Beispiel werden die winkeltreuen Projektionen auf der Linienfolge eines Punktes mit den Koordinaten (5, 10) gesucht. In diesem Fall können drei unterschiedliche Punkte der winkeltreuen Projektion gefunden werden. Die Funktion ST_PerpPoints gibt jedoch nur die Punkte zurück, die den geringsten Abstand zum angegebenen Punkt aufweisen. Daher wird eine Mehrpunktangabe zurückgegeben, die nur die beiden Punkte mit dem geringsten Abstand enthält. Der dritte Punkt ist nicht enthalten.

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(5, 10) ) )
  AS VARCHAR(80) ) PERP
  FROM   sample_lines
```

Ergebnisse:

```
PERP
------------------------------------------------------------
MULTIPOINT(0.00000000 10.00000000, 10.00000000 10.00000000 )
```

Beispiel 5

In diesem Beispiel wird die winkeltreue Projektion der Linienfolge eines Punktes mit den Koordinaten (5, 15) gesucht.

```
SELECT CAST(ST_AsText(ST_PerpPoints(line,ST_Point('point(5 15)',0)))
    AS VARCHAR(80) ) PERP
FROM   sample_lines
```

Ergebnisse:

```
PERP
-----------------------------------
MULTIPOINT ( 5.00000000 0.00000000)
```

Beispiel 6

In diesem Beispiel weist der angegebene Punkt mit den Koordinaten (15, 15) keine winkeltreue Projektion auf der Linienfolge auf. Daher wird eine leere Geometrie zurückgegeben.

```
SELECT CAST(ST_AsText(ST_PerpPoints(line,ST_Point(15,15)))
  AS VARCHAR(80)) PERP
  FROM   sample_lines
```

Ergebnisse:

```
PERP
--------------------------------------------
MULTIPOINT EMPTY
```

Funktion ST_Point

Die Funktion ST_Point konstruiert einen Punkt aus Koordinateninformationen oder einen resultierenden Punkt einer Darstellung.

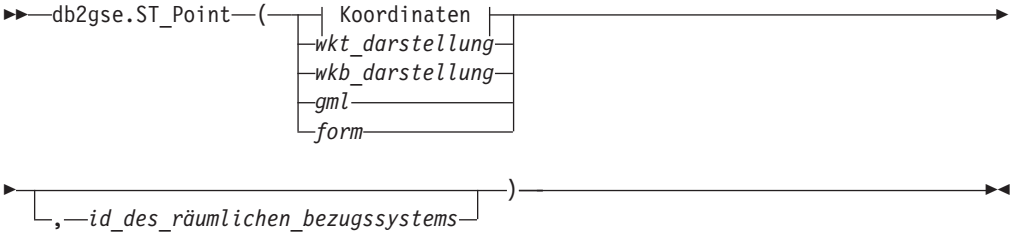
ST_Point konstruiert einen Punkt aus einer der folgenden Eingabegruppen:

- Nur X- und Y-Koordinaten
- X-, Y- und Z-Koordinaten
- X-, Y-, Z- und M-Koordinaten
- WKT-Darstellung
- WKB-Darstellung
- Formdarstellung
- Darstellung in GML (Geography Markup Language)

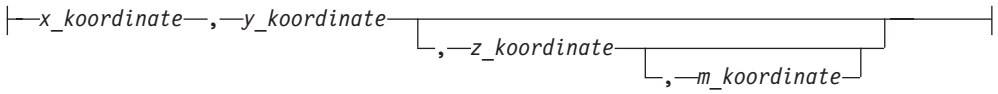
Eine Kennung für räumliche Bezugssysteme kann optional angegeben werden, um das räumliche Bezugssystem zu kennzeichnen, in dem sich der Ergebnispunkt befindet.

Wenn der Punkt aus Koordinaten konstruiert wurde und wenn die X- oder Y-Koordinate den Wert null aufweist, wird eine Ausnahmebedingung (SQLSTATE 38SUP) erzeugt. Wenn die Z- oder M-Koordinate den Wert null aufweist, verfügt der Ergebnispunkt nicht über eine Z- bzw. M-Koordinate. Wenn der Punkt auf der Basis seiner WKT-, WKB-, Form- oder GML-Darstellung konstruiert wurde und die Darstellung den Wert null aufweist, dann wird null zurückgegeben.

Syntax



Koordinaten:



Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung des Ergebnispunktes enthält.

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung des Ergebnispunktes enthält.

gml Ein Wert vom Typ CLOB(2G), der den Ergebnispunkt unter Verwendung von GML (Geography Markup Language) darstellt.

form Ein Wert vom Typ BLOB(2G), der die Formdarstellung des Ergebnispunktes darstellt.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für den Ergebnispunkt darstellt.

Wenn der Paramater *id_des_räumlichen_bezugssystems* ausgelassen wird, wird das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben wird, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, gibt das System eine Ausnahmebedingung (SQLSTATE 38SU1) aus.

x_koordinate
Ein Wert vom Typ DOUBLE, der die X-Koordinate für den Ergebnispunkt angibt.

y_koordinate
Ein Wert vom Typ DOUBLE, der die Y-Koordinate für den Ergebnispunkt angibt.

z_koordinate
Ein Wert vom Typ DOUBLE, der die Z-Koordinate für den Ergebnispunkt angibt.

Wenn der Parameter *z_koordinate* ausgelassen wird, weist der Ergebnispunkt keine Z-Koordinate auf. Das Ergebnis der Funktion ST_Is3D für einen solchen Punkt ist 0 (null).

m_koordinate
Ein Wert vom Typ DOUBLE, der die M-Koordinate für den Ergebnispunkt angibt.

Wenn der Parameter *m_koordinate* ausgelassen wird, weist der Ergebnispunkt keine Bemaßung auf. Das Ergebnis der Funktion ST_IsMeasured für einen solchen Punkt ist 0 (null).

Rückgabetyp

db2gse.ST_Point

Beispiel

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Beispiel 1

Dieses Beispiel verdeutlicht, wie ST_Point zur Erstellung und zum Einfügen von Punkten verwendet werden kann. Der erste Punkt wird unter Verwendung einer Gruppe von X- und Y-Koordinaten erstellt. Der zweite Punkt wird auf der Basis der zugehörigen WKT-Darstellung erstellt. Beide Punkte sind Geometrien im räumlichen Bezugssystem 1.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)

INSERT INTO sample_points
    VALUES (1100, ST_Point (10, 20, 1) )
```

```
INSERT INTO sample_points
    VALUES (1101, ST_Point ('point (30 40)', 1) )
```

Die folgende Anweisung SELECT gibt die Punkte zurück, die in der Tabelle aufgezeichnet wurden:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(90)) POINTS
  FROM sample_points
```

Ergebnisse:

```
ID         POINTS
---------- ----------------------------------
      1110 POINT ( 10.00000000 20.00000000)
      1101 POINT ( 30.00000000 40.00000000)
```

Beispiel 2

In diesem Beispiel wird ein Datensatz in die Tabelle SAMPLE_POINTS mit der ID 1103 und ein Punktwert mit einer X-Koordinate von 120 und einer Y-Koordinate von 358 und einer M-Koordinate von 34 ohne Z-Koordinate eingefügt.

```
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
```

Ergebnisse:

```
 ID         POINTS
 ---------- -----------------------------------------------
       1103 POINT M ( 120.0000000 358.0000000 34.00000000)
```

Beispiel 3

In diesem Beispiel wird eine Zeile in die Tabelle SAMPLE_POINTS mit der ID 1104 und ein Punktwert mit einer X-Koordinate von 1003, einer Y-Koordinate von 9876 und einer Z-Koordinate von 20 im räumlichen Bezugssystem 0 unter Verwendung von GML für die Darstellung eingefügt.

```
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
```

Ergebnisse:

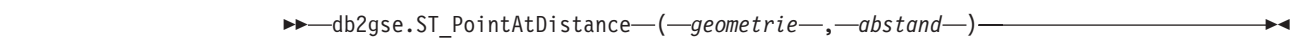
```
 ID         POINTS
 ---------- -----------------------------------------------------
       1104 POINT Z ( 1003.000000 9876.000000 20.00000000)
```

Funktion ST_PointAtDistance

Die Funktion ST_PointAtDistance verwendet eine Kurvengeometrie oder eine Geometrie mit mehreren Kurven und einen Abstand als Eingabeparameter und gibt die Punktgeometrie mit dem angegebenen Abstand entlang der Kurvengeometrie zurück.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Curve oder ST_MultiCurve oder einer seiner untergeordneten Typen, der die zu verarbeitende Geometrie darstellt.

abstand
Ein Wert vom Typ DOUBLE, der den Abstand entlang der Geometrie zur Lokalisierung des Punkts angibt.

Rückgabetyp

db2gse.ST_Point

Beispiel

Mit der folgenden SQL-Anweisung wird die Tabelle SAMPLE_GEOMETRIES mit zwei Spalten erstellt. Die Spalte 'ID' identifiziert jede Zeile eindeutig. In der Spalte 'GEOMETRY ST_LineString' werden Mustergeometrien gespeichert.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries(id INTEGER, geometry ST_LINESTRING)
```

Mit der folgenden SQL-Anweisung werden zwei Zeilen in die Tabelle SAMPLE_GEOMETRIES eingefügt.

```
INSERT INTO sample_geometries(id, geometry)
  VALUES
 (1,ST_LineString('LINESTRING ZM(0 0 0 0, 10 100 1000 10000)',1)),
 (2,ST_LineString('LINESTRING ZM(10 100 1000 10000, 0 0 0 0)',1))
```

Die folgende Anweisung SELECT und die entsprechende Ergebnismenge zeigen, wie die Funktion ST_PointAtDistance dazu verwendet werden kann, Punkte mit einem Abstand von 15 Koordinateneinheiten ab Beginn der Linienfolge zu ermitteln.

```
SELECT ID, VARCHAR(ST_AsText(ST_PointAtDistance(geometry, 15)), 50) AS POINTAT
FROM   sample_geometries
ID          POINTAT
----------- --------------------------------------------------
          1 POINT ZM(1.492556 14.925558 149 1493)
          2 POINT ZM(8.507444 85.074442 851 8507)

  2 record(s) selected.
```

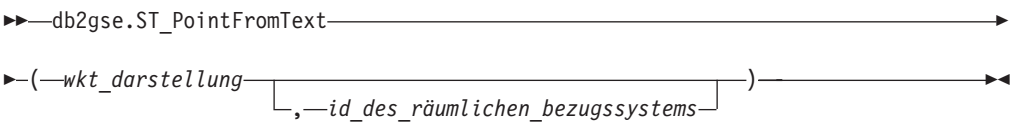
Funktion ST_PointFromText

Die Funktion ST_PointFromText verwendet eine WKT-Darstellung (WKT = Well-Known Text) eines Punktes und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt den entsprechenden Punkt zurück.

Wenn die angegebene WKT-Darstellung den Wert null aufweist, wird null zurückgegeben.

Um dasselbe Ergebnis zu erreichen wird die Funktion ST_Point empfohlen. Sie wird wegen ihrer Flexibilität empfohlen: ST_Point verwendet neben der WKT-Darstellung zusätzliche Formen der Eingabe.

Syntax



Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung des Ergebnispunktes enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für den Ergebnispunkt darstellt.

Wenn der Paramater *id_des_räumlichen_bezugssystems* ausgelassen wird, wird das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben wird, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, gibt das System eine Ausnahmebedingung (SQLSTATE 38SU1) aus.

Rückgabetyp

db2gse.ST_Point

Beispiel

Dieses Beispiel verdeutlicht, wie ST_PointFromText zur Erstellung und zum Einfügen eines Punktes aus der zugehörigen WKT-Darstellung (WKT = Well-Known Text) verwendet werden kann. Der eingefügte Datensatz weist die ID 1110 auf, und die Geometrie ist ein Punkt im räumlichen Bezugssystem 1. Der Punkt ist in der WKT-Darstellung eines Punktes definiert. Die X- und Y-Koordinaten für diese Geometrie lauten: (10, 20).

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)

INSERT INTO sample_points
    VALUES (1110, ST_PointFromText ('point (30 40)', 1) )
```

Die folgende Anweisung SELECT gibt das Polygon zurück, das in der Tabelle aufgezeichnet wurde:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(35) ) POINTS
  FROM sample_points
  WHERE  id = 1110
```

Ergebnisse:

```
ID         POINTS
---------- -----------------------------------
      1110 POINTS ( 30.00000000 40.00000000)
```

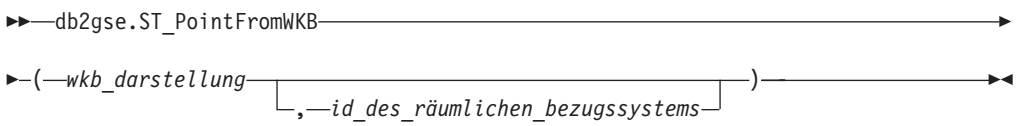
Funktion ST_PointFromWKB

Die Funktion ST_PointFromWKB verwendet eine WKB-Darstellung (WKB = Well-Known Binary) eines Punktes und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt den entsprechenden Punkt zurück.

Wenn die angegebene WKB-Darstellung den Wert null aufweist, wird null zurückgegeben.

Um dasselbe Ergebnis zu erreichen wird die Funktion ST_Point empfohlen. Sie wird wegen ihrer Flexibilität empfohlen: ST_Point verwendet neben der WKB-Darstellung zusätzliche Formen der Eingabe.

Syntax



Parameter

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung des Ergebnispunktes enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für den Ergebnispunkt darstellt.

Wenn der Parameter *id_des_räumlichen_bezugssystems* ausgelassen wird, verwendet das System implizit das räumliche Bezugssystem mit der numerischen Kennung 0 (null).

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben wird, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, gibt das System eine Ausnahmebedingung (SQLSTATE 38SU1) aus.

Rückgabetyp

db2gse.ST_Point

Beispiel

Dieses Beispiel verdeutlicht, wie ST_PointFromWKB zur Erstellung eines Punktes aus der zugehörigen WKB-Darstellung (WKB = Well-Known Binary) verwendet werden kann. Die Geometrien sind Punkte im räumlichen Bezugssystem 1. In diesem Beispiel werden die Punkte in der Spalte GEOMETRY der Tabelle SAMPLE_POLYS gespeichert. Die Spalte WKB wird dann unter Verwendung der Funktion ST_AsBinary mit der zugehörigen WKB-Darstellung aktualisiert. Schließlich wird die Funktion ST_PointFromWKB verwendet, um die Punkte aus der Spalte WKB zurückzugeben.

Die Tabelle SAMPLE_POINTS verfügt über eine Spalte GEOMETRY, in der die Punkte gespeichert werden, und eine Spalte WKB, in der die WKB-Darstellungen gespeichert werden.

```
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
```

In der folgenden Anweisung SELECT wird die Funktion ST_PointFromWKB verwendet, um die Punkte aus der Spalte WKB abzurufen.

```
SELECT id, CAST( ST_AsText( ST_PolyFromWKB (wkb) ) AS VARCHAR(35) ) POINTS
  FROM sample_points
```

Ergebnisse:

```
ID         POINTS
---------- -----------------------------------
        10 POINT ( 44.00000000 14.00000000)
        11 POINT ( 24.00000000 13.00000000)
```

Funktion ST_PointN

Die Funktion ST_PointN verwendet eine Linienfolge bzw. eine Mehrpunktangabe und einen Index als Eingabeparameter und gibt den Punkt in der Linienfolge bzw. Mehrpunktangabe zurück, der durch den Index angegeben wird. Der Ergebnispunkt wird im räumlichen Bezugssystem der angegebenen Linienfolge oder Mehrpunktangabe dargestellt.

Wenn die angegebene Linienfolge oder Mehrpunktangabe den Wert null aufweist oder leer ist, wird null zurückgegeben. Wenn der Index kleiner als 1 oder größer als die Anzahl der Punkte in der Linienfolge oder Mehrpunktangabe ist, wird null zurückgegeben und eine Warnungsbedingung (SQLSTATE 01HS2) zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_LineString oder ST_MultiPoint, der die Geometrie darstellt, von der der Punkt zurückgegeben wird, der durch den in *index* angegebenen Index gekennzeichnet ist.

index Ein Wert vom Typ INTEGER, der den *n*. Punkt kennzeichnet, der von der in *geometrie* angegebenen Geometrie zurückgegeben werden soll.

Rückgabetyp

db2gse.ST_Point

Beispiel

Das folgende Beispiel verdeutlicht die Verwendung von ST_PointN.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_lines (id INTEGER, line ST_LineString)

INSERT INTO sample_lines
    VALUES (1, ST_LineString ('linestring (10 10, 5 5, 0 0, 10 0, 5 5, 0 10)', 0) )

SELECT id, CAST ( ST_AsText (ST_PointN (line, 2) ) AS VARCHAR(60) ) SECOND_INDEX
  FROM   sample_lines
```

Ergebnisse:

```
ID        SECOND_INDEX
--------- --------------------------------
        1 POINT (5.00000000 5.00000000)
```

Funktion ST_PointOnSurface

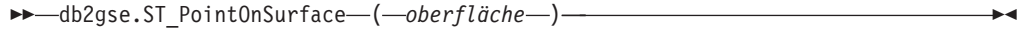
Die Funktion ST_PointOnSurface verwendet eine Oberfläche oder eine Mehrfachoberfläche als Eingabeparameter und gibt einen Punkt zurück, der sich mit Sicherheit im Inneren der Oberfläche oder Mehrfachoberfläche befindet. Dieser Punkt ist der parazentrische Punkt der Oberfläche.

Der Ergebnispunkt wird im räumlichen Bezugssystem der angegebenen Oberfläche oder Mehrfachoberfläche dargestellt.

Wenn die angegebene Oberfläche oder Mehrfachoberfläche den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

oberfläche
Ein Wert vom Typ ST_Surface, ST_MultiSurface oder einer der zugehörigen untergeordneten Typen, der die Geometrie darstellt, für die ein Punkt auf der Oberfläche zurückgegeben wird.

Rückgabetyp

db2gse.ST_Point

Beispiel

Im folgenden Beispiel werden zwei Polygone erstellt. Anschließend wird die Funktion ST_PointOnSurface verwendet. Eines der Polygone weist ein Loch in der Mitte auf. Die zurückgegebenen Punkte befinden sich auf der Oberfläche der Polygone. Sie befinden sich nicht notwendigerweise genau in der Mitte der Polygone.

```
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
```

Ergebnisse:

```
ID          POINT_ON_SURFACE
----------- ------------------------------------
          1 POINT ( 65.00000000 125.00000000)
          2 POINT ( 30.00000000 15.00000000)
```

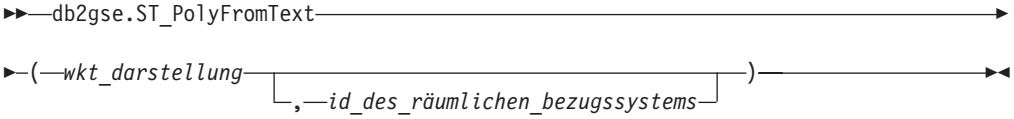
Funktion ST_PolyFromText

Die Funktion ST_PolyFromText verwendet eine WKT-Darstellung (WKT = Well-Known Text) eines Polygons und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt das entsprechende Polygon zurück.

Wenn die angegebene WKT-Darstellung den Wert null aufweist, wird null zurückgegeben.

Um dasselbe Ergebnis zu erreichen, wird die Funktion ST_Polygon empfohlen. Sie wird wegen ihrer Flexibilität empfohlen: ST_Polygon verwendet neben der WKT-Darstellung zusätzliche Formen der Eingabe.

Syntax



Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung (WKT = Well-Known Text) des Ergebnispolygons enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem des Ergebnispolygons kennzeichnet.

Wenn der Paramater *id_des_räumlichen_bezugssystems* ausgelassen wird, wird das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben wird, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, gibt das System eine Ausnahmebedingung (SQLSTATE 38SU1) aus.

Rückgabetyp

db2gse.ST_Polygon

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie ST_PolyFromText zur Erstellung und zum Einfügen eines Polygons aus der zugehörigen WKT-Darstellung (WKT = Well-Known Text) verwendet werden kann. Der eingefügte Datensatz weist die ID 1110 auf, und die Geometrie ist ein Polygon im räumlichen Bezugssystem 1. Das Polygon ist in der WKT-Darstellung eines Polygons definiert. Die X- und Y-Koordinaten für diese Geometrie lauten: (50, 20) (50, 40) (70, 30).

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
    VALUES (1110, ST_PolyFromText ('polygon ((50 20, 50 40, 70 30, 50 20))', 1) )
```

Die folgende Anweisung SELECT gibt das Polygon zurück, das in der Tabelle aufgezeichnet wurde:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(120) ) POLYGON
  FROM sample_polys
  WHERE  id = 1110
```

Ergebnisse:

```
ID         POLYGON
---------- ------------------------------------------------------------------
      1110 POLYGON (( 50.00000000 20.00000000, 70.00000000 30.00000000,
                      50.00000000 40.00000000, 50.00000000 20.00000000))
```

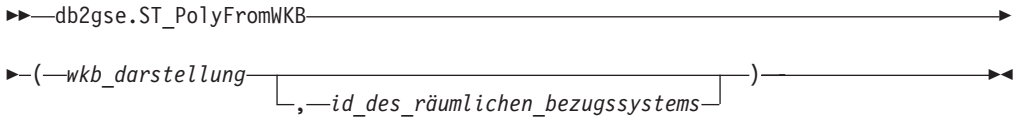
Funktion ST_PolyFromWKB

Die Funktion ST_PolyFromWKB verwendet eine WKB-Darstellung (WKB = Well-Known Binary) eines Polygons und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter und gibt das entsprechende Polygon zurück.

Wenn die angegebene WKB-Darstellung den Wert null aufweist, wird null zurückgegeben.

Um dasselbe Ergebnis zu erreichen, wird die Funktion ST_Polygon empfohlen. Sie wird wegen ihrer Flexibilität empfohlen: ST_Polygon verwendet neben der WKB-Darstellung zusätzliche Formen der Eingabe.

Syntax



Parameter

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung des Ergebnispolygons enthält.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem des Ergebnispolygons kennzeichnet.

Wenn der Paramater *id_des_räumlichen_bezugssystems* ausgelassen wird, wird das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben wird, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, gibt das System eine Ausnahmebedingung (SQLSTATE 38SU1) aus.

Rückgabetyp

db2gse.ST_Polygon

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie ST_PolyFromWKB zur Erstellung eines Polygons aus der zugehörigen WKB-Darstellung (WKB = Well-Known Binary) verwendet werden kann. Die Geometrie ist ein Polygon im räumlichen Bezugssystem 1. In diesem Beispiel wird das Polygon mit der ID 1115 in der Spalte GEOMETRY der Tabelle SAMPLE_POLYS gespeichert. Anschließend wird die Spalte WKB unter Verwendung der Funktion ST_AsBinary mit der WKB-Darstellung aktualisiert. Schließlich wird die Funktion ST_PolyFromWKB verwendet, um das Multipolygon aus der Spalte WKB zurückzugeben. Die X- und Y-Koordinaten für diese Geometrie lauten: (50, 20) (50, 40) (70, 30).

Die Tabelle SAMPLE_POLYS verfügt über eine Spalte GEOMETRY, in der das Polygon gespeichert wird, und eine Spalte WKB, in der die WKB-Darstellung (WKB = Well-Known Binary) des Polygons gespeichert wird.

```
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
```

In der folgenden Anweisung SELECT wird die Funktion ST_PolyFromWKB verwendet, um das Polygon aus der Spalte WKB abzurufen.

```
SELECT id, CAST( ST_AsText( ST_PolyFromWKB (wkb) )
    AS VARCHAR(120) ) POLYGON
  FROM sample_polys
  WHERE id = 1115
```

Ergebnisse:

```
ID         POLYGON
---------- ------------------------------------------------------------
      1115 POLYGON (( 50.00000000 20.00000000, 70.00000000
                      30.00000000,50.00000000 40.00000000, 50.00000000
                      20.00000000))
```

Funktion ST_Polygon

Die Funktion ST_Polygon konstruiert ein Polygon aus einer Eingabe.

Die Eingabe kann in einem der folgenden Formate erfolgen:

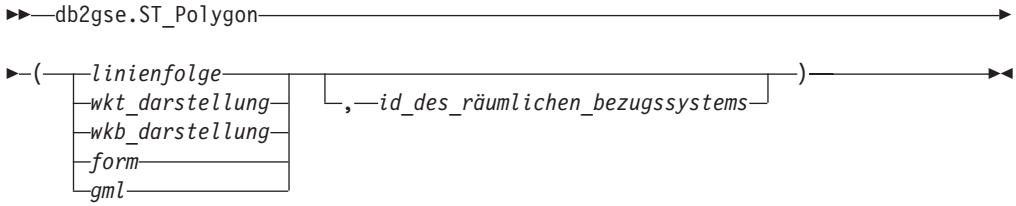
- Geschlossene Linienfolge, die den äußeren Ring des Ergebnispolygons definiert
- WKT-Darstellung
- WKB-Darstellung
- Formdarstellung
- Darstellung in GML (Geography Markup Language)

Eine Kennung für räumliche Bezugssysteme kann optional angegeben werden, um das räumliche Bezugssystem anzugeben, in dem sich das Ergebnispolygon befindet.

Wenn das Polygon aus einer Linienfolge konstruiert wurde und die angegebene Linienfolge den Wert null aufweist, wird null zurückgegeben. Wenn die angegebene Linienfolge leer ist, wird ein leeres Polygon zurückgegeben. Wenn das Polygon aus der zugehörigen WKT-, WKB-, Form- oder GML-Darstellung konstruiert wurde und wenn die Darstellung den Wert null aufweist, dann wird null zurückgegeben.

Diese Funktion kann in den folgenden Fällen auch als Methode aufgerufen werden: ST_Polygon(*linienfolge*) und ST_Polygon(*linienfolge*, *id_des_räumlichen_bezugssystems*).

Syntax



Parameter

linienfolge
Ein Wert vom Typ ST_LineString, der die Linienfolge darstellt, die den äußeren Ring für die äußere Begrenzung definiert. Wenn die in *linienfolge* angegebene Linienfolge nicht geschlossen und einfach ist, wird eine Ausnahmebedingung (SQLSTATE 38SSL) erzeugt.

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung (WKT = Well-Known Text) des Ergebnispolygons enthält.

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung des Ergebnispolygons enthält.

form Ein Wert vom Typ BLOB(2G), der die Formdarstellung des Ergebnispolygons darstellt.

gml Ein Wert vom Typ CLOB(2G), der das Ergebnispolygon im GML-Format (GML = Geography Markup Language) darstellt.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem des Ergebnispolygons kennzeichnet.

Wenn das Polygon aus einem angegebenen Parameter *linienfolge* konstruiert wurde und der Parameter *id_des_räumlichen_bezugssystems* ausgelassen wurde, wird implizit das räumliche Bezugssystem von *linienfolge* verwendet. Andernfalls wird, wenn der Parameter *id_des_räumlichen_bezugssystems* ausgelassen wird, das räumliche Bezugssystem mit der numerischen Kennung 0 (null) verwendet.

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben wird, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, gibt das System eine Ausnahmebedingung (SQLSTATE 38SU1) aus.

Rückgabetyp

db2gse.ST_Polygon

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie ST_Polygon zur Erstellung und zum Einfügen von Polygonen verwendet werden kann. Drei Polygone werden erstellt und eingefügt. Alle diese Polygone sind als Geometrien im räumlichen Bezugssystem 1 definiert.

- Das erste Polygon wird aus einem Ring (einer geschlossenen, einfachen Linienfolge) erstellt. Die X- und Y-Koordinaten für dieses Polygon lauten: (10, 20) (10, 40) (20, 30).
- Das zweite Polygon wird auf der Basis der zugehörigen WKT-Darstellung erstellt. Die X- und Y-Koordinaten für dieses Polygon lauten: (110, 120) (110, 140) (120, 130).
- Das dritte Polygon ist ein sog. Donut-Polygon. Ein Donut-Polygon besteht aus einem inneren und einem äußeren Polygon. Dieses Donut-Polygon wird auf der Basis der zugehörigen WKT-Darstellung erstellt. Die X- und Y-Koordinaten für das äußere Polygon lauten: (110, 120) (110, 140) (130, 140) (130, 120) (110, 120). Die X- und Y-Koordinaten für das innere Polygon lauten: (115, 125) (115, 135) (125, 135) (125, 135) (115, 125).

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
    VALUES (1100,
                    ST_Polygon (ST_LineString ('linestring
                                (10 20, 10 40, 20 30, 10 20)',1), 1))

INSERT INTO sample_polys
    VALUES (1101,
                    ST_Polygon ('polygon
                                ((110 120, 110 140, 120 130, 110 120))', 1))
```

```
INSERT INTO sample_polys
    VALUES (1102,
                    ST_Polygon ('polygon
                                ((110 120, 110 140, 130 140, 130 120, 110 120),
                                (115 125, 115 135, 125 135, 125 135, 115 125))', 1))
```

Die folgende Anweisung SELECT gibt die Polygone zurück, die in der Tabelle aufgezeichnet wurden:

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(120) ) POLYGONS
  FROM sample_polys
```

Ergebnisse:

```
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))
```

Funktion ST_PolygonN

Die Funktion ST_PolygonN verwendet ein Multipolygon und einen Index als Eingabeparameter und gibt das Polygon zurück, das durch den Index definiert wird. Das resultierende Polygon wird im räumlichen Bezugssystem des angegebenen Multipolygons dargestellt.

Wenn das angegebene Multipolygon den Wert null aufweist oder leer ist, oder wenn der Index kleiner als 1 (eins) oder größer als die Anzahl der Polygone ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

multipolygon
Ein Wert vom Typ ST_MultiPolygon, der das Multipolygon darstellt, von dem das in *index* angegebene Polygon zurückgegeben wird.

index Ein Wert vom Typ INTEGER, der das *n*. Polygon kennzeichnet, das von dem in *multipolygon* angegebenen Multipolygon zurückgegeben werden soll.

Rückgabetyp

db2gse.ST_Polygon

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht die Verwendung von ST_PolygonN.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mpolys (id INTEGER, geometry ST_MultiPolygon)

INSERT INTO sample_mpolys
    VALUES (1, ST_Polygon ('multipolygon (((3 3, 4 6, 5 3, 3 3),
                                           (8 24, 9 25, 1 28, 8 24)
                                           (13 33, 7 36, 1 40, 10 43,
                                            13 33)))', 1))

SELECT id, CAST ( ST_AsText (ST_PolygonN (geometry, 2) )
        AS VARCHAR(120) ) SECOND_INDEX
FROM sample_mpolys
```

Ergebnisse:

```
ID        SECOND_INDEX
--------- ------------------------------------------------------------
        1 POLYGON (( 8.00000000 24.00000000, 9.00000000 25.00000000,
                     1.00000000 28.00000000, 8.00000000 24.00000000))
```

Funktion ST_Relate

Die Funktion ST_Relate verwendet zwei Geometrien und eine DE-9IM-Matrix (Dimensionally Extended 9 Intersection Model) als Eingabeparameter. Wenn die angegebenen Geometrien die Bedingungen erfüllen, die durch die Matrix angegeben sind, gibt sie 1 zurück. Andernfalls wird 0 (null) zurückgegeben.

Wenn eine der angegebenen Geometrien den Wert null aufweist oder leer ist, wird null zurückgegeben.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist, wird sie in das andere räumliche Bezugssystem umgewandelt.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie1
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die in Bezug auf die in *geometrie2* angegebene Geometrie getestet wird.

geometrie2
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die in Bezug auf die in *geometrie1* angegebene Geometrie getestet wird.

matrix Ein Wert vom Typ CHAR(9), der die DE-9IM-Matrix darstellt, die für den Test von *geometrie1* und *geometrie2* verwendet wird.

Rückgabetyp

INTEGER

Beispiel

Der folgende Code erstellt zwei separate Polygone. Anschließend wird die Funktion ST_Relate verwendet, um die verschiedenen Beziehungen zwischen den beiden Polygonen zu ermitteln. Hierbei wird z. B. festgestellt, ob die beiden Polygone sich überlappen.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES( 1,
          ST_Polygon('polygon ( (40 120, 90 120, 90 150, 40 150, 40 120) )', 0))
INSERT INTO sample_polys
  VALUES(2,
          ST_Polygon('polygon ( (30 110, 50 110, 50 130, 30 130, 30 110) )', 0))

SELECT ST_Relate(a.geometry, b.geometry, 'T*T***T**')  "Overlaps  ",
       ST_Relate(a.geometry, b.geometry, 'T*T***FF*')  "Contains  ",
       ST_Relate(a.geometry, b.geometry, 'T*F**F***')  "Within    "
       ST_Relate(a.geometry, b.geometry, 'T********')  "Intersects",
       ST_Relate(a.geometry, b.geometry, 'T*F**FFF2')  "Equals    "
  FROM sample_polys a, sample_polys b
    WHERE a.id = 1 AND b.id = 2
```

Ergebnisse:

```
Overlaps    Contains    Within      Intersects  Equals
----------- ----------- ----------- ----------- -----------
          1           0           0           1           0
```

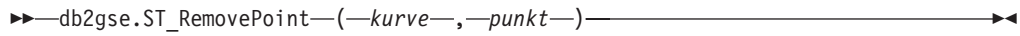
Funktion ST_RemovePoint

Die Funktion ST_RemovePoint verwendet eine Kurve und einen Punkt als Eingabeparameter und gibt die angegebene Kurve mit allen Punkten zurück, die gleich dem angegebenen Punkt sind, der von der Kurve entfernt wurde. Wenn die angegebene Kurve Z- oder M-Koordinaten aufweist, muss der Punkt ebenfalls Z- oder M-Koordinaten aufweisen. Die Ergebnisgeometrie wird im räumlichen Bezugssystem der angegebenen Geometrie dargestellt.

Wenn die angegebene Kurve leer ist, wird eine leere Kurve zurückgegeben. Wenn die angegebene Kurve den Wert null aufweist oder wenn der angegebene Punkt den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

kurve Ein Wert vom Typ ST_Curve oder einer seiner untergeordneten Typen, der die Kurve darstellt, aus der der in *punkt* angegebene Punkt entfernt wird.

punkt Ein Wert vom Typ ST_Point, der die Punkte kennzeichnet, die aus der in *kurve* angegebenen Kurve entfernt werden.

Rückgabetyp

db2gse.ST_Curve

Beispiele

Beispiel 1

Im folgenden Beispiel werden der Tabelle SAMPLE_LINES zwei Linienfolgen hinzugefügt. Diese Linienfolgen werden in den folgenden Beispielen verwendet.

```
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))
```

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Beispiel 2

Im folgenden Beispiel wird der Punkt (5, 5) aus der Linienfolge mit der ID 1 entfernt. Dieser Punkt kommt in der Linienfolge zwei Mal vor. Daher werden beide Vorkommen entfernt.

```
SELECT CAST(ST_AsText (ST_RemovePoint (line, ST_Point(5, 5) ) )
    AS VARCHAR(120) ) RESULT
  FROM   sample_lines
  WHERE  id = 1
```

Ergebnisse:

```
RESULT
------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 0.00000000 0.00000000,
             10.00000000 0.00000000, 0.00000000 10.00000000)
```

Beispiel 3

Im folgenden Beispiel wird der Punkt (5, 5, 5) aus der Linienfolge mit der ID 2 entfernt. Dieser Punkt kommt nur einmal vor, sodass nur ein Vorkommen entfernt wird.

```
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
```

Ergebnisse:

```
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)
```

Funktion ST_SrsId oder ST_SRID

Die Funktion ST_SrsId oder ST_SRID verwendet eine Geometrie und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter.

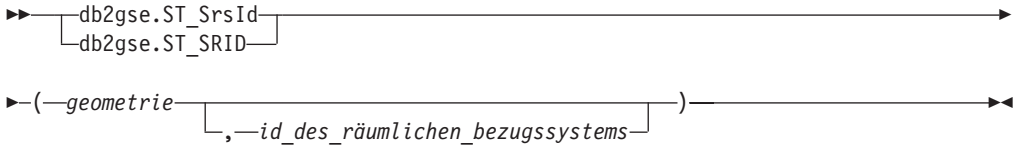
Die Rückgabe hängt davon ab, welche Eingabeparameter angegeben wurden:

- Wenn die Kennung eines räumlichen Bezugssystems angegeben wurde, wird eine Geometrie zurückgegeben, deren räumliches Bezugssystem in das angegebene räumliche Bezugssystem geändert wurde. Es wird keine Umsetzung der Geometrie ausgeführt.
- Wenn keine Kennung eines räumlichen Bezugssystems als Eingabeparameter angegeben wurde, wird die aktuelle Kennung des räumlichen Bezugssystems der angegebenen Geometrie zurückgegeben.

Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktionen können auch als Methoden aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, für die die Kennung des räumlichen Bezugssystems festgelegt oder zurückgegeben werden soll.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem kennzeichnet, das für die Ergebnisgeometrie verwendet werden soll.

Achtung: Wenn dieser Parameter angegeben ist, wird die Geometrie nicht umgesetzt, sondern mit geändertem räumlichen Bezugssystem zurückgegeben. Das räumliche Bezugssystem der Geometrie wird in das angegebene räumliche Bezugssystem geändert. Als Folge der Änderung in das neue räumliche Bezugssystem können die Daten möglicherweise beschädigt werden. Verwenden Sie für Umsetzungen stattdessen die Funktion ST_Transform.

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben wird, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, gibt das System eine Ausnahmebedingung (SQLSTATE 38SU1) aus.

Rückgabetypen

- INTEGER, wenn keine ID des räumlichen Bezugssystems angegeben wurde.
- db2gse.ST_Geometry, wenn eine ID des räumlichen Bezugssystems angegeben wurde.

Beispiel

Es werden zwei Punkte in zwei unterschiedlichen räumlichen Bezugssystemen erstellt. Die Kennung des räumlichen Bezugssystems, die dem jeweiligen Punkt zugeordnet ist, kann mithilfe der Funktion ST_SrsId ermittelt werden.

```
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
```

Ergebnisse:

```
ID          SRSID
----------- -----------
          1           0
          2           1
```

Funktion ST_SrsId oder ST_SRID

Die Funktion ST_SrsId oder ST_SRID verwendet eine Geometrie und optional die Kennung eines räumlichen Bezugssystems als Eingabeparameter.

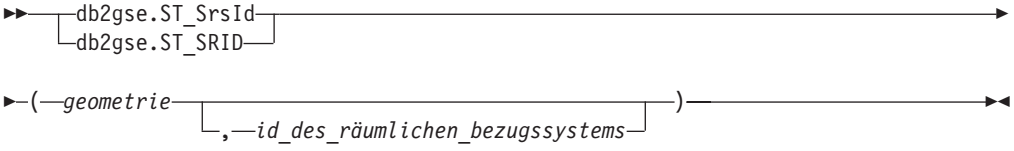
Die Rückgabe hängt davon ab, welche Eingabeparameter angegeben wurden:

- Wenn die Kennung eines räumlichen Bezugssystems angegeben wurde, wird eine Geometrie zurückgegeben, deren räumliches Bezugssystem in das angegebene räumliche Bezugssystem geändert wurde. Es wird keine Umsetzung der Geometrie ausgeführt.
- Wenn keine Kennung eines räumlichen Bezugssystems als Eingabeparameter angegeben wurde, wird die aktuelle Kennung des räumlichen Bezugssystems der angegebenen Geometrie zurückgegeben.

Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktionen können auch als Methoden aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, für die die Kennung des räumlichen Bezugssystems festgelegt oder zurückgegeben werden soll.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem kennzeichnet, das für die Ergebnisgeometrie verwendet werden soll.

Achtung: Wenn dieser Parameter angegeben ist, wird die Geometrie nicht umgesetzt, sondern mit geändertem räumlichen Bezugssystem zurückgegeben. Das räumliche Bezugssystem der Geometrie wird in das angegebene räumliche Bezugssystem geändert. Als Folge der Änderung in das neue räumliche Bezugssystem können die Daten möglicherweise beschädigt werden. Verwenden Sie für Umsetzungen stattdessen die Funktion ST_Transform.

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben wird, das in der Katalogsicht DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, gibt das System eine Ausnahmebedingung (SQLSTATE 38SU1) aus.

Rückgabetypen

- INTEGER, wenn keine ID des räumlichen Bezugssystems angegeben wurde.
- db2gse.ST_Geometry, wenn eine ID des räumlichen Bezugssystems angegeben wurde.

Beispiel

Es werden zwei Punkte in zwei unterschiedlichen räumlichen Bezugssystemen erstellt. Die Kennung des räumlichen Bezugssystems, die dem jeweiligen Punkt zugeordnet ist, kann mithilfe der Funktion ST_SrsId ermittelt werden.

```
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
```

Ergebnisse:

```
ID          SRSID
----------- ------------
          1            0
          2            1
```

Funktion ST_SrsName

Die Funktion ST_SrsName verwendet eine Geometrie als Eingabeparameter und gibt den Namen des räumlichen Bezugssystems zurück, in dem die angegebene Geometrie dargestellt ist.

Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, für die der Name des räumlichen Bezugssystems zurückgegeben wird.

Rückgabetyp

VARCHAR(128)

Beispiel

Es werden zwei Punkte in unterschiedlichen räumlichen Bezugssystemen erstellt. Die Funktion ST_SrsName wird verwendet, um den Namen des räumlichen Bezugssystems zu ermitteln, das dem jeweiligen Punkt zugeordnet ist.

```
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
```

Ergebnisse:

```
ID          SRSNAME
----------- ------------------------------
          1 DEFAULT_SRS
          2 NAD83_SRS_1
```

Funktion ST_StartPoint

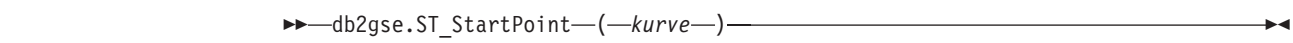
Die Funktion ST_StartPoint verwendet eine Kurve als Eingabeparameter und gibt den ersten Punkt (Anfangspunkt) der Kurve zurück.

Der Ergebnispunkt wird im räumlichen Bezugssystem der angegebenen Kurve dargestellt. Dieses Ergebnis entspricht dem Aufruf der Funktion ST_PointN(*kurve*, 1)

Wenn die angegebene Kurve den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

kurve Ein Wert vom Typ ST_Curve oder einer seiner untergeordneten Typen, der die Geometrie darstellt, von der der erste Punkt zurückgegeben wird.

Rückgabetyp

db2gse.ST_Point

Beispiel

Im folgenden Beispiel werden der Tabelle SAMPLE_LINES zwei Linienfolgen hinzugefügt. Die erste Linienfolge enthält X- und Y-Koordinaten. Die zweite Linienfolge enthält X-, Y- und Z-Koordinaten. Die Funktion ST_StartPoint wird verwendet, um den ersten Punkt in jeder Linienfolge zurückzugeben.

```
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
```

Ergebnisse:

```
ID          START_POINT
----------- ----------------------------------------
          1 POINT ( 10.00000000 10.00000000)
          2 POINT Z ( 0.00000000 0.00000000 4.00000000)
```

Funktion ST_SymDifference

Die Funktion ST_SymDifference verwendet zwei Geometrien als Eingabeparameter und gibt die Geometrie zurück, die die symmetrische Differenz der beiden angegebenen Geometrien darstellt. Die symmetrische Differenz ist der sich nicht schneidende Teil der beiden angegebenen Geometrien.

Die Ergebnisgeometrie wird im räumlichen Bezugssystem der ersten Geometrie dargestellt. Die Dimension der zurückgegebenen Geometrie stimmt mit der Dimension der Eingabegeometrien überein. Beide Geometrien müssen dieselbe Dimension aufweisen.

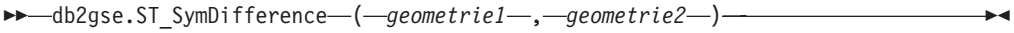
Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist und das gleiche zugrunde liegende Datum verwendet, wird sie in das andere räumliche Bezugssystem umgewandelt.

Wenn die Geometrien identisch sind, wird eine leere Geometrie vom Typ ST_Point zurückgegeben. Wenn eine der Geometrien den Wert null aufweist, wird 0 (null) zurückgegeben.

Die Ergebnisgeometrie wird in dem räumlichen Bezugssystem dargestellt, das am besten geeignet ist. Wenn die Ergebnisgeometrie als Punkt, Linienfolge oder Polygon dargestellt werden kann, wird einer dieser Typen verwendet. Andernfalls wird der Typ Mehrpunktangabe, Mehrlinienfolge oder Multipolygon verwendet.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie1
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die erste Geometrie darstellt, deren symmetrische Differenz zu der in *geometrie2* angegebenen Geometrie berechnet werden soll.

geometrie2
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die zweite Geometrie darstellt, deren symmetrische Differenz zu der in *geometrie1* angegebenen Geometrie berechnet werden soll.

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_SymDifference. Die Geometrien werden in der Tabelle SAMPLE_GEOMS gespeichert.

```
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));
```

In den folgenden Beispielen wurden die Ergebnisse zur besseren Lesbarkeit umformatiert. Die Ergebnisse variieren abhängig von der jeweiligen Anzeige.

Beispiel 2

In diesem Beispiel wird ST_SymDifference verwendet, um die symmetrische Differenz zweier sich nicht schneidender Polygone in der Tabelle SAMPLE_GEOMS zurückzugeben.

```
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
```

Ergebnisse:

```
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)))
```

Beispiel 3

In diesem Beispiel wird ST_SymDifference verwendet, um die symmetrische Differenz zweier sich schneidender Polygone in der Tabelle SAMPLE_GEOMS zurückzugeben.

```
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
```

Ergebnisse:

```
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)))
```

Beispiel 4

In diesem Beispiel wird ST_SymDifference verwendet, um die symmetrische Differenz zweier sich schneidender Linienfolgen in der Tabelle SAMPLE_GEOMS zurückzugeben.

```
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
```

Ergebnisse:

```
ID   ID  SYM_DIFF
-- -- ---------------------------------------------------------------------
 4  5 MULTILINESTRING (( 70.00000000 70.00000000, 75.00000000 75.00000000),
                      ( 80.00000000 80.00000000, 90.00000000 90.00000000))
```

Funktion ST_ToGeomColl

Die Funktion ST_ToGeomColl verwendet eine Geometrie als Eingabeparameter und wandelt sie in eine Geometriengruppe um. Die Ergebnisgeometriengruppe wird im räumlichen Bezugssystem der angegebenen Geometrie dargestellt.

Wenn die angegebene Geometrie leer ist, kann sie einen beliebigen Typ aufweisen. Sie wird jedoch anschließend entsprechend in ST_Multipoint, ST_MultiLineString oder ST_MultiPolygon umgewandelt.

Wenn die angegebene Geometrie nicht leer ist, muss Sie den Typ ST_Point, ST_LineString oder ST_Polygon aufweisen. Diese Typen werden anschließend in ST_Multipoint, ST_MultiLineString bzw. ST_MultiPolygon umgewandelt.

Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die in eine Geometriengruppe umgewandelt wird.

Rückgabetyp

db2gse.ST_GeomCollection

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_ToGeomColl.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
    VALUES (1, ST_Polygon ('polygon ((3 3, 4 6, 5 3, 3 3))', 1)),
               (2, ST_Point ('point (1 2)', 1))
```

In der folgenden Anweisung SELECT wird die Funktion ST_ToGeomColl verwendet, um Geometrien als ihre entsprechenden Geometriengruppensubtypen zurückzugeben.

```
SELECT id, CAST( ST_AsText( ST_ToGeomColl(geometry) )
        AS VARCHAR(120) ) GEOM_COLL
FROM   sample_geometries
```

Ergebnisse:

```
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)
```

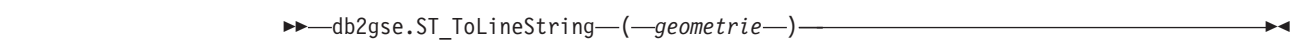
Funktion ST_ToLineString

Die Funktion ST_ToLineString verwendet eine Geometrie als Eingabeparameter und wandelt sie in eine Linienfolge um. Die Ergebnislinienfolge wird im räumlichen Bezugssystem der angegebenen Geometrie dargestellt.

Die angegebene Geometrie muss leer oder eine Zeilenfolge sein. Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie

Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die in eine Linienfolge umgewandelt wird.

Eine Geometrie kann in eine Linienfolge umgewandelt werden, wenn sie leer ist oder wenn sie eine Linienfolge ist. Wenn die Umwandlung nicht ausgeführt werden kann, wird eine Ausnahmebedingung (SQLSTATE 38SUD) erzeugt.

Rückgabetyp

db2gse.ST_LineString

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_ToLineString.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
    VALUES (1, ST_Geometry ('linestring z (0 10 1, 0  0 3, 10 0 5)', 0)),
                  (2, ST_Geometry ('point empty', 1) ),
                  (3, ST_Geometry ('multipolygon empty', 1) )
```

In der folgenden Anweisung SELECT wird die Funktion ST_ToLineString verwendet, um die Linienfolgen zurückzugeben, die vom statischen Typ ST_Geometry in den Typ ST_LineString umgewandelt wurden.

```
SELECT CAST( ST_AsText( ST_ToLineString(geometry) )
       AS VARCHAR(130) ) LINES
FROM   sample_geometries
```

Ergebnisse:

```
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
```

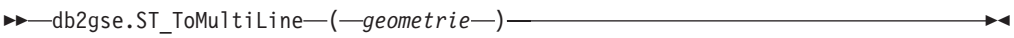
Funktion ST_ToMultiLine

Die Funktion ST_ToMultiLine verwendet eine Geometrie als Eingabeparameter und wandelt diese in eine Mehrlinienfolge um. Die Ergebnismehrlinienfolge wird im räumlichen Bezugssystem der angegebenen Geometrie dargestellt.

Die angegebene Geometrie muss leer, eine Mehrlinienfolge oder eine Linienfolge sein. Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die in eine Mehrlinienfolge umgewandelt wird.

Eine Geometrie kann in eine Mehrlinienfolge umgewandelt werden, wenn sie leer, eine Linienfolge oder eine Mehrlinienfolge ist. Wenn die Umwandlung nicht ausgeführt werden kann, wird eine Ausnahmebedingung (SQLSTATE 38SUD) erzeugt.

Rückgabetyp

db2gse.ST_MultiLineString

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_ToMultiLine.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)
```

```
INSERT INTO sample_geometries
    VALUES (1, ST_Geometry ('multilinestring ((0 10 1, 0 0 3, 10 0 5),
                                             (23 43, 27 34, 35 12))', 0) ),
           (2, ST_Geometry ('linestring z (0 10 1, 0 0 3, 10 0 5)', 0) ),
           (3, ST_Geometry ('point empty', 1) ),
           (4, ST_Geometry ('multipolygon empty', 1) )
```

In der folgenden Anweisung SELECT wird die Funktion ST_ToMultiLine verwendet, um die Mehrlinienfolgen zurückzugeben, die vom statischen Typ ST_Geometry in den Typ ST_MultiLineString umgewandelt wurden.

```
SELECT CAST( ST_AsText( ST_ToMultiLine(geometry) )
    AS VARCHAR(130) ) LINES
  FROM   sample_geometries
```

Ergebnisse:

```
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
```

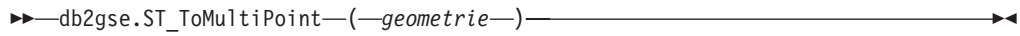
Funktion ST_ToMultiPoint

Die Funktion ST_ToMultiPoint verwendet eine Geometrie als Eingabeparameter und wandelt diese in eine Mehrpunktangabe um. Die Ergebnismehrpunktangabe wird im räumlichen Bezugssystem der angegebenen Geometrie dargestellt.

Die angegebene Geometrie muss leer, ein Punkt oder eine Mehrpunktangabe sein. Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie

Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die in eine Mehrpunktangabe umgewandelt wird.

Eine Geometrie kann in eine Mehrpunktangabe umgewandelt werden, wenn sie leer, ein Punkt oder eine Mehrpunktangabe ist. Wenn die Umwandlung nicht ausgeführt werden kann, wird eine Ausnahmebedingung (SQLSTATE 38SUD) erzeugt.

Rückgabetyp

db2gse.ST_MultiPoint

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_ToMultiPoint.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
    VALUES (1, ST_Geometry ('multipoint (0 0, 0 4)', 1) ),
                (2, ST_Geometry ('point (30 40)', 1) ),
                (3, ST_Geometry ('multipolygon empty', 1) )
```

In der folgenden Anweisung SELECT wird die Funktion ST_ToMultiPoint verwendet, um die Mehrpunktangaben zurückzugeben, die vom statischen Typ ST_Geometry in den Typ ST_MultiPoint umgewandelt wurden.

```
SELECT CAST( ST_AsText( ST_ToMultiPoint(geometry))
    AS VARCHAR(62) ) MULTIPOINTS
FROM   sample_geometries
```

Ergebnisse:

```
MULTIPOINTS
------------------------------------------------------------
MULTIPOINT ( 0.00000000 0.00000000, 0.00000000 4.00000000)
MULTIPOINT ( 30.00000000 40.00000000)
MULTIPOINT EMPTY
```

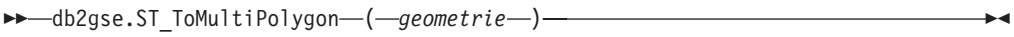
Funktion ST_ToMultiPolygon

Die Funktion ST_ToMultiPolygon verwendet eine Geometrie als Eingabeparameter und wandelt diese in ein Multipolygon um. Das resultierende Multipolygon wird im räumlichen Bezugssystem der angegebenen Geometrie dargestellt.

Die angegebene Geometrie muss leer, ein Polygon oder ein Multipolygon sein. Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

geometrie

Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die in ein Multipolygon umgewandelt wird.

Eine Geometrie kann in ein Multipolygon umgewandelt werden, wenn sie leer, ein Polygon oder ein Multipolygon ist. Wenn die Umwandlung nicht ausgeführt werden kann, wird eine Ausnahmebedingung (SQLSTATE 38SUD) erzeugt.

Rückgabetyp

db2gse.ST_MultiPolygon

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

In diesem Beispiel werden mehrere Geometrien erstellt. Anschließend wird ST_ToMultiPolygon verwendet, um die Multipolygone zurückzugeben.

```
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))
```

In der folgenden Anweisung SELECT wird die Funktion ST_ToMultiPolygon verwendet, um die Multipolygone zurückzugeben, die vom statischen Typ ST_Geometry in den Typ ST_MultiPolygon umgewandelt wurden.

```
SELECT CAST( ST_AsText( ST_ToMultiPolygon(geometry) )
    AS VARCHAR(130) ) POLYGONS
  FROM   sample_geometries
```

Ergebnisse:

```
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
```

Funktion ST_ToPoint

Die Funktion ST_ToPoint verwendet eine Geometrie als Eingabeparameter und wandelt diese in einen Punkt um. Der Ergebnispunkt wird im räumlichen Bezugssystem der angegebenen Geometrie dargestellt.

Die angegebene Geometrie muss leer oder ein Punkt sein. Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die in einen Punkt umgewandelt wird.

Eine Geometrie kann in einen Punkt umgewandelt werden, wenn sie leer oder ein Punkt ist. Wenn die Umwandlung nicht ausgeführt werden kann, wird eine Ausnahmebedingung (SQLSTATE 38SUD) erzeugt.

Rückgabetyp

db2gse.ST_Point

Beispiel

In diesem Beispiel werden drei Geometrien in SAMPLE_GEOMETRIES erstellt und jeweils in einen Punkt umgewandelt.

```
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) )
```

In der folgenden Anweisung SELECT wird die Funktion ST_ToPoint verwendet, um die Punkte zurückzugeben, die vom statischen Typ ST_Geometry in den Typ ST_Point umgewandelt wurden.

```
SELECT CAST( ST_AsText( ST_ToPoint(geometry) ) AS VARCHAR(35) ) POINTS
  FROM   sample_geometries
```

Ergebnisse:

```
POINTS
---------------------------------
POINT ( 30.00000000 40.00000000)
POINT EMPTY
POINT EMPTY
```

Funktion ST_ToPolygon

Die Funktion ST_ToPolygon verwendet als Eingabeparameter eine Geometrie und wandelt diese in ein Polygon um. Das Ergebnispolygon wird im räumlichen Bezugssystem der angegebenen Geometrie dargestellt.

Die angegebene Geometrie muss leer oder ein Polygon sein. Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie

Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die in ein Polygon umgewandelt wird.

Eine Geometrie kann in ein Polygon umgewandelt werden, wenn sie leer oder ein Polygon ist. Wenn die Umwandlung nicht ausgeführt werden kann, wird eine Ausnahmebedingung (SQLSTATE 38SUD) erzeugt.

Rückgabetyp

db2gse.ST_Polygon

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

In diesem Beispiel werden drei Geometrien in SAMPLE_GEOMETRIES erstellt und jeweils in ein Polygon umgewandelt.

```
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) )
```

In der folgenden Anweisung SELECT wird die Funktion ST_ToPolygon verwendet, um Polygone zurückzugeben, die vom statischen Typ ST_Geometry in den Typ ST_Polygon umgewandelt wurden.

```
SELECT CAST( ST_AsText( ST_ToPolygon(geometry) ) AS VARCHAR(130) ) POLYGONS
  FROM   sample_geometries
```

Ergebnisse:

```
POLYGONS
--------------------------------------------------------------------------------
POLYGON (( 0.00000000 0.00000000, 5.00000000 0.00000000,
           5.00000000 4.00000000,0.00000000 4.00000000,
           0.00000000 0.00000000))

POLYGON EMPTY

POLYGON EMPTY
```

Funktion ST_Touches

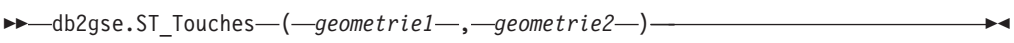
Die Funktion ST_Touches verwendet zwei Geometrien als Eingabeparameter und gibt 1 zurück, wenn die angegebenen Geometrien sich räumlich berühren. Andernfalls wird 0 (null) zurückgegeben.

Zwei Geometrien berühren sich, wenn das Innere beider Geometrien sich nicht überschneidet, die Grenze der einen Geometrie jedoch entweder die Grenze oder das Innere der anderen Geometrie schneidet.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist, wird sie in das andere räumliche Bezugssystem umgewandelt.

Wenn beide der angegebenen Geometrien Punkte oder Mehrpunktangaben sind, oder wenn eine der angegebenen Geometrien den Wert null aufweist oder leer ist, wird null zurückgegeben.

Syntax



Parameter

geometrie1
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die auf eine Berührung mit der in *geometrie2* angegebenen Geometrie getestet wird.

geometrie2
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die auf eine Berührung mit der in *geometrie1* angegebenen Geometrie getestet wird.

Rückgabetyp

INTEGER

Beispiel

Der Tabelle SAMPLE_GEOMS werden mehrere Geometrien hinzugefügt. Die Funktion ST_Touches wird anschließend verwendet, um zu ermitteln, welche Geometrien einander berühren.

```
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
```

Ergebnisse:

```
     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
```

Funktion ST_Transform

Die Funktion ST_Transform verwendet eine Geometrie und die Kennung eines räumlichen Bezugssystems als Eingabeparameter und setzt die Geometrie für die Darstellung im angegebenen räumlichen Bezugssystem um. Projektionen und Umwandlungen zwischen unterschiedlichen Koordinatensystemen werden ausgeführt, und die Koordinaten der Geometrien werden entsprechend angepasst.

Die Geometrie kann nur in das angegebene räumliche Bezugssystem umgewandelt werden, wenn das aktuelle räumliche Bezugssystem der Geometrie auf demselben geografischen Koordinatensystem beruht wie das angegebene räumliche Bezugssystem. Wenn das aktuelle räumliche Bezugssystem der Geometrie oder das angegebene räumliche Bezugssystem auf einem projizierten Koordinatensystem beruht, wird eine Umkehrprojektion ausgeführt, um das geografische Koordinatensystem zu ermitteln, das dem projizierten Koordinatensystem zugrunde liegt.

Wenn die angegebene Geometrie den Wert null aufweist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der die Geometrie darstellt, die in das räumliche Bezugssystem umgesetzt wird, das in *id_des_räumlichen_bezugssystems* angegeben ist.

id_des_räumlichen_bezugssystems
Ein Wert vom Typ INTEGER, der das räumliche Bezugssystem für die Ergebnisgeometrie kennzeichnet.

Wenn die Umsetzung in das angegebene räumliche Bezugssystem nicht ausgeführt werden kann, da das aktuelle räumliche Bezugssystem für die in *geometrie* angegebene Geometrie nicht mit dem räumlichen Bezugssystem kompatibel ist, das in *id_des_räumlichen_bezugssystems* angegeben wurde, wird eine Ausnahmebedingung (SQLSTATE 38SUC) erzeugt.

Wenn in *id_des_räumlichen_bezugssystems* kein räumliches Bezugssystem angegeben wird, das in der Katalogsicht

DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS aufgelistet ist, gibt das System eine Ausnahmebedingung (SQLSTATE 38SU1) aus.

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Beispiel 1

Das folgende Beispiel verdeutlicht die Verwendung von ST_Transform zur Umsetzung einer Geometrie von einem räumlichen Bezugssystem in ein anderes.

Zuerst wird das räumliche Bezugssystem (SPC-Georeferenzsystem) mit der ID 3 unter Verwendung eines Aufrufs von db2se erstellt.

```
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
```

Anschließend werden Punkte hinzugefügt:

- Die Tabelle SAMPLE_POINTS_SP im SPC-Georeferenzsystem koordiniert unter Verwendung dieses räumlichen Bezugssystems.
- Die Tabelle SAMPLE_POINTS_LL verwendet Koordinaten, die in Breiten- und Längengraden angegeben sind.

```
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) )
```

Anschließend wird die Funktion ST_Transform verwendet, um diese Geometrien umzusetzen.

Beispiel 2

In diesem Beispiel werden Punkte von Koordinaten in Form von Längen- und Breitengraden in SPC-Koordinaten umgewandelt.

```
SELECT id, CAST( ST_AsText( ST_Transform( geometry, 3) )
        AS VARCHAR(100) ) STATE_PLANE
    FROM sample_points_ll
```

Ergebnisse:

```
ID          STATE_PLANE
----------- ----------------------------------------
      12457 POINT ( 567176.00000000 1166411.00000000)
      12477 POINT ( 637948.00000000 1177640.00000000)
```

Beispiel 3

In diesem Beispiel werden Punkte in Form von SPC-Koordinaten in Koordinaten in Form von Längen- und Breitengraden umgewandelt.

```
SELECT id, CAST( ST_AsText( ST_Transform( geometry, 1) )
        AS VARCHAR(100) ) LAT_LONG
    FROM sample_points_sp
```

Ergebnisse:

```
ID          LAT_LONG
----------- ---------------------------------
      12457 POINT ( -74.22371500 42.03498800)
      12477 POINT ( -73.96293100 42.06488000)
```

Funktion ST_Union

Die Funktion ST_Union verwendet zwei Geometrien als Eingabeparameter und gibt die Geometrie zurück, die die Union-Verknüpfung der angegebenen Geometrien darstellt. Die Ergebnisgeometrie wird im räumlichen Bezugssystem der ersten Geometrie dargestellt.

Beide Geometrien müssen dieselbe Dimension aufweisen. Wenn eine der beiden angegebenen Geometrien den Wert null aufweist, wird null zurückgegeben.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist und das gleiche zugrunde liegende Datum verwendet, wird sie in das andere räumliche Bezugssystem umgewandelt.

Die Ergebnisgeometrie wird in dem räumlichen Bezugssystem dargestellt, das am besten geeignet ist. Wenn die Ergebnisgeometrie als Punkt, Linienfolge oder Polygon dargestellt werden kann, wird einer dieser Typen verwendet. Andernfalls wird der Typ Mehrpunktangabe, Mehrlinienfolge oder Multipolygon verwendet.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax

Parameter

geometrie1
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der mit der in *geometrie2* angegebenen Geometrie kombiniert wird.

geometrie2
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der mit der in *geometrie1* angegebenen Geometrie kombiniert wird.

Rückgabetyp

db2gse.ST_Geometry

Beispiele

Beispiel 1

Mit den folgenden SQL-Anweisungen wird die Tabelle SAMPLE_GEOMS erstellt und gefüllt.

```
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))
```

In den folgenden Beispielen wurden die Ergebnisse zur besseren Lesbarkeit umformatiert. Die Ergebnisse variieren abhängig von der jeweiligen Anzeige.

Beispiel 2

In diesem Beispiel wird die Union-Verknüpfung zweier sich nicht schneidender Polygone gesucht.

```
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
```

Ergebnisse:

```
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)))
```

Beispiel 3

In diesem Beispiel wird die Union-Verknüpfung zweier sich schneidender Polygone gesucht.

```
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
```

Ergebnisse:

```
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))
```

Beispiel 4

In diesem Beispiel wird die Union-Verknüpfung zweier Linienfolgen gesucht.

```
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
```

Ergebnisse:

```
ID    ID    UNION
--- --- ------------------------------------------------------------
  4   5 MULTILINESTRING((70.00000000 70.00000000,80.00000000 80.00000000),
                        (80.00000000 80.00000000,100.00000000 70.00000000))
```

Funktion ST_Within

Mit der Funktion ST_Within können Sie ermitteln, ob eine Geometrie vollständig in einer anderen Geometrie enthalten ist.

Syntax

Parameter

geometrie1
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der daraufhin getestet wird, ob er sich vollkommen innerhalb der in *geometrie2* angegebenen Geometrie befindet.

geometrie2
Ein Wert vom Typ ST_Geometry oder einer seiner untergeordneten Typen, der daraufhin getestet wird, ob er sich vollkommen innerhalb der in *geometrie1* angegebenen Geometrie befindet.

Rückgabetyp

INTEGER

Verwendung

ST_Within verwendet zwei Geometrien als Eingabeparameter und gibt 1 zurück, wenn die erste Geometrie sich vollkommen innerhalb der zweiten Geometrie befindet. Andernfalls wird 0 (null) zurückgegeben.

Wenn eine der angegebenen Geometrien den Wert null aufweist oder leer ist, wird 0 (null) zurückgegeben.

Wenn die zweite Geometrie nicht in demselben räumlichen Bezugssystem wie die erste Geometrie dargestellt ist und das gleiche zugrunde liegende Datum verwendet, wird sie in das andere räumliche Bezugssystem umgewandelt.

ST_Within führt dieselbe logische Operation wie ST_Contains mit umgekehrten Parametern aus. ST_Within gibt genau das entgegengesetzte Ergebnis von ST_Contains zurück.

Die Mustermatrix der Funktion ST_Within gibt an, dass sich die Innenbereiche der beiden Geometrien schneiden müssen und dass der Innenbereich oder die Begren-

zung der primären Geometrie (Geometrie *a*) den Außenbereich der sekundären Geometrie (Geometrie *b*) nicht schneiden darf. Der Stern (*) gibt an, dass alle anderen Schnittpunkte nicht relevant sind.

Tabelle 32. Matrix für ST_Within

	Geometrie b Innenbereich	**Geometrie b Begrenzung**	**Geometrie b Außenbereich**
Geometrie a Innenbereich	T	*	F
Geometrie a Begrenzung	*	*	F
Geometrie a Außenbereich	*	*	*

Beispiele

Abb. 20 auf Seite 433 enthält Beispiele für ST_Within:

- Eine Punktgeometrie befindet sich innerhalb einer Mehrpunktgeometrie, wenn ihr Innenbereich eine Überschneidung mit einem der Punkte in der zweiten Geometrie aufweist.
- Eine Mehrpunktgeometrie befindet sich innerhalb einer Mehrpunktgeometrie, wenn die Innenbereiche aller Punkte eine Überschneidung mit der zweiten Geometrie aufweisen.
- Eine Mehrpunktgeometrie befindet sich innerhalb einer Polygongeometrie, wenn alle vorhandenen Punkte entweder auf der Begrenzung des Polygons oder im Innenbereich des Polygons liegen.
- Eine Punktgeometrie befindet sich innerhalb einer Linienfolgengeometrie, wenn sich alle Punkte innerhalb der zweiten Geometrie befinden. In Abb. 20 auf Seite 433 befindet sich der Punkt nicht innerhalb der Linienfolge, da sein Innenbereich keine Überschneidung mit der Linienfolge aufweist. Allerdings befindet sich die Mehrpunktgeometrie innerhalb der Linienfolge, weil alle zugehörigen Punkte den Innenbereich der Linienfolge schneiden.
- Eine Linienfolgengeometrie befindet sich innerhalb einer anderen Linienfolgengeometrie, wenn alle zugehörigen Punkte die zweite Geometrie schneiden.
- Eine Punktgeometrie befindet sich nicht innerhalb einer Polygongeometrie, weil ihr Innenbereich keine Überschneidung mit der Begrenzung oder dem Innenbereich des Polygons aufweist.
- Eine Linienfolgengeometrie befindet sich innerhalb einer Polygongeometrie, wenn alle zugehörigen Punkte eine Überschneidung mit der Begrenzung oder dem Innenbereich des Polygons aufweisen.
- Eine Polygongeometrie befindet sich innerhalb einer Polygongeometrie, wenn alle zugehörigen Punkte eine Überschneidung mit der Begrenzung oder dem Innenbereich des Polygons aufweisen.

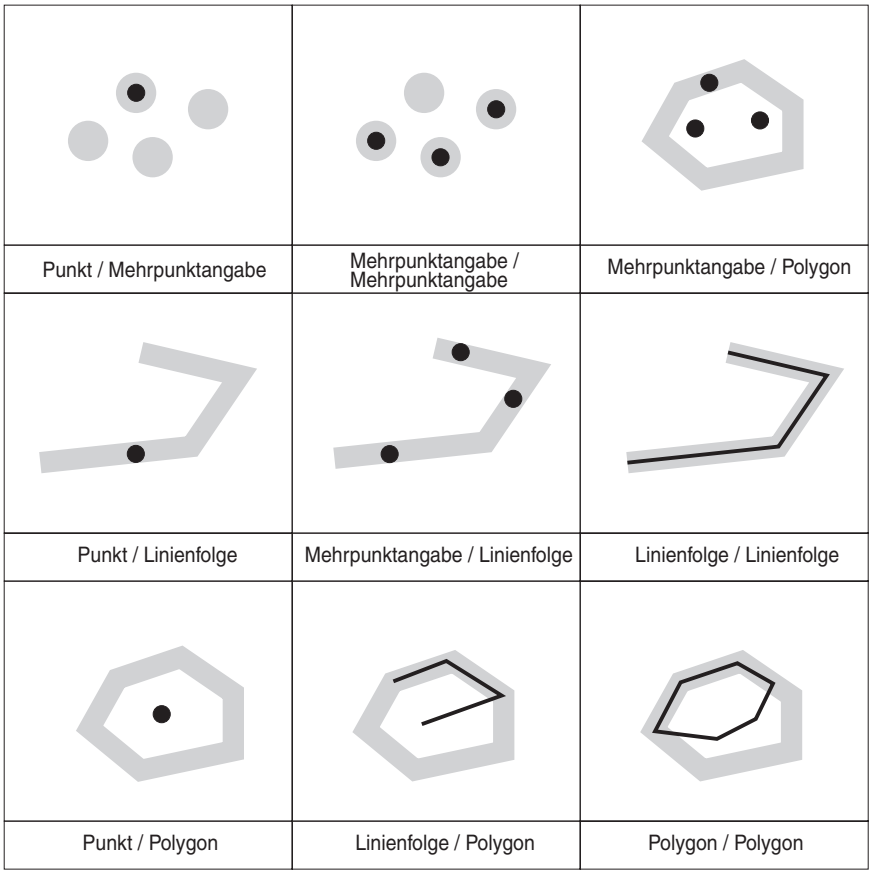


Abbildung 20. Funktion ST_Within

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_Within. Geometrien werden erstellt und in die drei Tabellen SAMPLE_POINTS, SAMPLE_LINES und SAMPLE_POLYGONS eingefügt.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_points (id INTEGER, geometry ST_Point)
CREATE TABLE sample_lines (id INTEGER, line ST_LineString)
CREATE TABLE sample_polygons (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_points (id, geometry)
    VALUES (1, ST_Point (10, 20, 1) ),
           (2, ST_Point ('point (41 41)', 1) )

INSERT INTO sample_lines (id, line)
    VALUES (10, ST_LineString ('linestring (1 10, 3 12, 10 10)', 1) ),
           (20, ST_LineString ('linestring (50 10, 50 12, 45 10)', 1) )

INSERT INTO sample_polygons (id, geometry)
    VALUES (100, ST_Polygon ('polygon (( 0 0, 0 40, 40 40, 40 0, 0 0))', 1) )
```

Beispiel 2

In diesem Beispiel werden die Punkte aus der Tabelle SAMPLE_POINTS gesucht, die sich in den Polygonen in der Tabelle SAMPLE_POLYGONS befinden.

```
SELECT a.id POINT_ID_WITHIN_POLYGONS
    FROM sample_points a, sample_polygons b
    WHERE ST_Within( b.geometry, a.geometry) = 0
```

Ergebnisse:

```
POINT_ID_WITHIN_POLYGONS
----------------------
                     2
```

Beispiel 3

In diesem Beispiel werden die Linienfolgen aus der Tabelle SAMPLE_LINES gesucht, die sich in den Polygonen der Tabelle SAMPLE_POLYGONS befinden.

```
SELECT a.id LINE_ID_WITHIN_POLYGONS
    FROM sample_lines a, sample_polygons b
    WHERE ST_Within( b.geometry, a.geometry) = 0
```

Ergebnisse:

```
LINE_ID_WITHIN_POLYGONS
----------------------
                     1
```

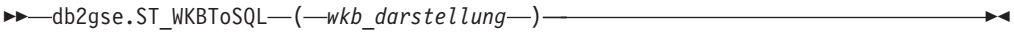
Funktion ST_WKBToSQL

Die Funktion ST_WKBToSQL verwendet eine WKB-Darstellung (WKB = Well-Known Binary) einer Geometrie als Eingabeparameter und gibt die entsprechende Geometrie zurück. Das räumliche Bezugssystem mit der ID 0 (null) wird für die Ergebnisgeometrie verwendet.

Wenn die angegebene WKB-Darstellung den Wert null aufweist, wird null zurückgegeben.

ST_WKBToSQL(*wkb_darstellung*) führt zu demselben Ergebnis wie ST_Geometry(*wkb_darstellung*,0). Die Verwendung von ST_Geometry wird gegenüber der Verwendung von ST_WKBToSQL wegen ihrer Flexibilität empfohlen: ST_Geometry verwendet neben der WKB-Darstellung zusätzliche Formen der Eingabe.

Syntax



Parameter

wkb_darstellung
Ein Wert vom Typ BLOB(2G), der die WKB-Darstellung der Ergebnisgeometrie enthält.

Rückgabetyp

db2gse.ST_Geometry

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_WKBToSQL. Zuerst werden Geometrien in der Spalte GEOMETRY der Tabelle SAMPLE_GEOMETRIES gespeichert. Anschließend werden die WKB-Darstellungen unter Verwendung der

Funktion ST_AsBinary in der Anweisung UPDATE in der Spalte WKB gespeichert. Schließlich wird die Funktion ST_WKBToSQL verwendet, um die Koordinaten der Geometrien in der Spalte WKB zurückzugeben.

```
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
```

Verwenden Sie diese Anweisung SELECT, um die Geometrien in der Spalte WKB anzuzeigen.

```
SELECT id, CAST( ST_AsText( ST_WKBToSQL(wkb) ) AS VARCHAR(120) ) GEOMETRIES
  FROM   sample_geometries
```

Ergebnisse:

```
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))
```

Funktion ST_WKTToSQL

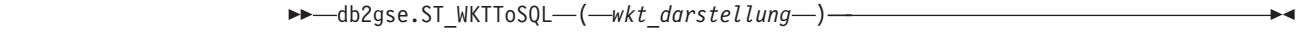
Die Funktion ST_WKTToSQL verwendet eine WKT-Darstellung (WKT = Well-Known Text) einer Geometrie als Eingabeparameter und gibt die entsprechende Geometrie zurück.

Das räumliche Bezugssystem mit der ID 0 (null) wird für die Ergebnisgeometrie verwendet.

Wenn die angegebene WKT-Darstellung den Wert null aufweist, wird null zurückgegeben.

ST_WKTToSQL(*wkt_darstellung*) führt zu demselben Ergebnis wie ST_Geometry(*wkt_darstellung*,0). Die Verwendung der Funktion ST_Geometry wird aufgrund ihrer Flexibilität der Funktion ST_WKTToSQL vorgezogen: ST_Geometry verwendet neben der WKT-Darstellung zusätzliche Formen der Eingabe.

Syntax



Parameter

wkt_darstellung
Ein Wert vom Typ CLOB(2G), der die WKT-Darstellung der Ergebnisgeometrie enthält.

Rückgabetyp

db2gse.ST_Geometry

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie mit ST_WKTToSQL Geometrien unter Verwendung ihrer WKT-Darstellung (WKT = Well-Known Text) erstellt und eingefügt werden können.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_geometries (id INTEGER, geometry ST_Geometry)

INSERT INTO sample_geometries
    VALUES (10, ST_WKTToSQL( 'point (44 14)' ) ),
                (11, ST_WKTTSQL ( 'point (24 13)' ) ),
                (12, ST_WKTToSQL ('polygon ((50 20, 50 40, 70 30, 50 20))' ) )
```

Diese Anweisung SELECT gibt die Geometrien zurück, die eingefügt wurden.

```
SELECT id, CAST( ST_AsText(geometry) AS VARCHAR(120) ) GEOMETRIES
  FROM   sample_geometries
```

Ergebnisse:

```
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))
```

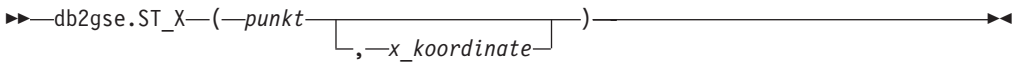
Funktion ST_X

Die Funktion ST_X verwendet einen Punkt als Eingabeparameter und gibt die zugehörige X-Koordinate zurück. Sie können optional eine X-Koordinate als Eingabeparameter zusätzlich zu dem Punkt angeben, sodass die Funktion den Punkt selbst mit der X-Koordinate zurückgibt, für die ein bestimmter Wert gesetzt ist.

Wenn der angegebene Punkt den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

punkt Ein Wert vom Typ ST_Point, für den die X-Koordinate zurückgegeben oder geändert wird.

x_koordinate
Ein Wert vom Typ DOUBLE, der die neue X-Koordinate für den in *punkt* angegebenen Punkt darstellt.

Rückgabetypen

- DOUBLE, wenn *x_koordinate* nicht angegeben ist.
- db2gse.ST_Point, wenn *x_koordinate* angegeben ist.

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_X. Geometrien werden erstellt und in die Tabelle SAMPLE_POINTS eingefügt.

```
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) )
```

Beispiel 2

In diesem Beispiel werden die X-Koordinaten der Punkte in der Tabelle gesucht.

```
SELECT id, ST_X (geometry) X_COORD
  FROM sample_points
```

Ergebnisse:

```
ID         X_COORD
---------- ----------------------
         1 +2.00000000000000E+000
         2 +4.00000000000000E+000
         3 +3.00000000000000E+000
```

Beispiel 3

In diesem Beispiel wird ein Punkt mit seiner X-Koordinate zurückgegeben, die auf 40 gesetzt wurde.

```
SELECT id, CAST( ST_AsText( ST_X (geometry, 40)) AS VARCHAR(60) )
  X_40
  FROM sample_points
  WHERE id=3
```

Ergebnisse:

```
ID     X_40
------ ------------------------------------------------------------
     3 POINT ZM ( 40.00000000 8.00000000 23.00000000 7.00000000)
```

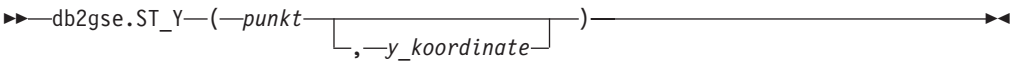
Funktion ST_Y

Die Funktion ST_Y verwendet einen Punkt als Eingabeparameter und gibt die zugehörige Y-Koordinate zurück. Sie können optional eine Y-Koordinate als Eingabeparameter zusätzlich zu dem Punkt angeben, sodass die Funktion den Punkt selbst mit der Y-Koordinate zurückgibt, für die ein bestimmter Wert gesetzt ist.

Wenn der angegebene Punkt den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

punkt Ein Wert vom Typ ST_Point, für den die Y-Koordinate zurückgegeben oder geändert wird.

y_koordinate
Ein Wert vom Typ DOUBLE, der die neue Y-Koordinate für den in *punkt* angegebenen Punkt darstellt.

Rückgabetypen

- DOUBLE, wenn *y_koordinate* nicht angegeben ist.
- db2gse.ST_Point, wenn *y_koordinate* angegeben ist.

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_Y. Geometrien werden erstellt und in die Tabelle SAMPLE_POINTS eingefügt.

```
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) )
```

Beispiel 2

In diesem Beispiel werden die Y-Koordinaten der Punkte in der Tabelle gesucht.

```
SELECT id, ST_Y (geometry) Y_COORD
  FROM sample_points
```

Ergebnisse:

```
ID         Y_COORD
---------- ----------------------
         1 +3.00000000000000E+000
         2 +5.00000000000000E+000
         3 +8.00000000000000E+000
```

Beispiel 3

In diesem Beispiel wird ein Punkt mit seiner Y-Koordinate zurückgegeben, die auf 40 gesetzt wurde.

```
SELECT id, CAST( ST_AsText( ST_Y (geometry, 40)) AS VARCHAR(60) )
  Y_40
  FROM sample_points
  WHERE id=3
```

Ergebnisse:

```
ID          Y_40
--------- ----------------------------------------------------
        3 POINT ZM ( 3.00000000 40.00000000 23.00000000 7.00000000)
```

Funktion ST_Z

Die Funktion ST_Z verwendet einen Punkt als Eingabeparameter und gibt die zugehörige Y-Koordinate zurück. Sie können optional eine Z-Koordinate als Eingabeparameter zusätzlich zu dem Punkt angeben, sodass die Funktion den Punkt selbst mit der Y-Koordinate zurückgibt, für die ein bestimmter Wert gesetzt ist.

ST_Z verwendet die folgenden Komponenten:

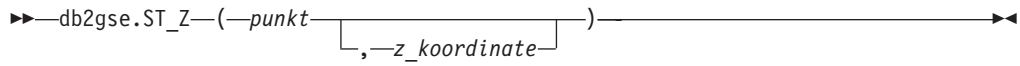
- Einen Punkt als Eingabeparameter und gibt dessen Z-Koordinate zurück.
- Einen Punkt und eine Z-Koordinate und gibt den Punkt selbst mit seiner Z-Koordinate zurück, die auf den angegebenen Wert gesetzt wurde. Die Rückgabe erfolgt auch dann, wenn der angegebene Punkt keine Z-Koordinate aufweist.

Wenn die angegebene Z-Koordinate den Wert null aufweist, wird die Z-Koordinate vom Punkt entfernt.

Wenn der angegebene Punkt den Wert null aufweist oder leer ist, wird null zurückgegeben.

Diese Funktion kann auch als Methode aufgerufen werden.

Syntax



Parameter

punkt Ein Wert vom Typ ST_Point, für den die Z-Koordinate zurückgegeben oder geändert wird.

z_koordinate
Ein Wert vom Typ DOUBLE, der die neue Z-Koordinate für den in *punkt* angegebenen Punkt darstellt.

Wenn *z_koordinate* den Wert null aufweist, wird die Z-Koordinate von dem in *punkt* angegebenen Punkt entfernt.

Rückgabetypen

- DOUBLE, wenn *z_koordinate* nicht angegeben ist.
- db2gse.ST_Point, wenn *z_koordinate* angegeben ist.

Beispiele

Beispiel 1

Dieses Beispiel verdeutlicht die Verwendung der Funktion ST_Z. Geometrien werden erstellt und in die Tabelle SAMPLE_POINTS eingefügt.

```
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) )
```

Beispiel 2

In diesem Beispiel werden die Z-Koordinaten der Punkte in der Tabelle gesucht.

```
SELECT id, ST_Z (geometry) Z_COORD
  FROM sample_points
```

Ergebnisse:

```
ID         Z_COORD
---------- ----------------------
         1 +3.20000000000000E+001
         2 +2.00000000000000E+001
         3 +2.30000000000000E+001
```

Beispiel 3

In diesem Beispiel wird ein Punkt mit seiner Z-Koordinate zurückgegeben, die auf den Wert 40 gesetzt wurde.

```
SELECT id, CAST( ST_AsText( ST_Z (geometry, 40)) AS VARCHAR(60) )
  Z_40
  FROM sample_points
  WHERE id=3
```

Ergebnisse:

```
ID         Z_40
------ ------------------------------------------------------
     3 POINT ZM ( 3.00000000 8.00000000 40.00000000 7.00000000)
```

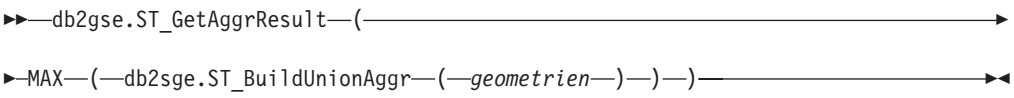
Union-Gesamtverknüpfungen

Eine Union-Gesamtverknüpfung von Geometrien ist die Kombination der Funktionen ST_BuildUnionAggr und ST_GetAggrResult. Mithilfe dieser Kombination können Sie eine Spalte mit Geometrien in einer Tabelle zu einer einzigen Geometrie zusammenfassen, indem die Union-Verknüpfung erstellt wird.

Wenn alle zu kombinierenden Geometrien in der Union-Verknüpfung den Wert null aufweisen, wird null zurückgegeben. Wenn alle der zu kombinierenden Geometrien in der Union-Verknüpfung entweder den Wert null aufweisen oder leer sind, wird eine leere Geometrie vom Typ ST_Point zurückgegeben.

Die Funktion ST_BuildUnionAggr kann auch als Methode aufgerufen werden.

Syntax



Parameter

Geometrien
Eine Spalte in einer Tabelle, die den Typ ST_Geometry oder einen seiner untergeordneten Typen aufweist und alle Geometrien darstellt, die in einer Union-Verknüpfung kombiniert werden sollen.

Rückgabetyp

db2gse.ST_Geometry

Einschränkungen

In den folgenden Situationen können Sie keine Union-Gesamtverknüpfung einer räumlichen Spalte in einer Tabelle erstellen:

- In einer Umgebungen mit partitionierten Datenbanken.
- Wenn eine Klausel GROUP BY in der Auswahlanweisung verwendet wird.
- Wenn Sie eine andere Funktion als die DB2-Spaltenfunktion MAX verwenden

Beispiel

Im folgenden Beispiel wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen variiert entsprechend der jeweiligen Onlineanzeige.

Dieses Beispiel verdeutlicht, wie eine Union-Gesamtverknüpfung zur Zusammenfassung einer Gruppe von Punkten zu einer Mehrpunktangabe verwendet werden kann. Der Tabelle SAMPLE_POINTS werden mehrere Punkte hinzugefügt. Die Funktionen ST_GetAggrResult und ST_BuildUnionAggr werden verwendet, um die Union-Verknüpfung der Punkte zu erstellen.

```
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
```

Ergebnisse:

```
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)
```

Kapitel 19. Umsetzungsgruppen

Spatial Extender stellt vier Umsetzungsgruppen bereit, die für die Übertragung von Geometrien zwischen dem DB2-Server und einer Clientanwendung verwendet werden.

Diese Umsetzungsgruppen verarbeiten die folgenden Datenaustauschformate:

- WKT-Darstellung (WKT = Well-Known Text)
- WKB-Darstellung (WKB = Well-Known Binary)
- ESRI-Formdarstellung
- GML (Geography Markup Language)

Beim Abrufen von Daten aus einer Tabelle, die eine räumliche Spalte enthält, werden die Daten aus der räumlichen Spalte entweder in den Typ CLOB(2G) oder den Typ BLOB(2G) umgesetzt. Dies richtet sich danach, ob die Binär- oder die Textdarstellung gewählt wurde. Sie können die Umsetzungsgruppen darüber hinaus dazu verwenden, um räumliche Daten an die Datenbank zu übertragen.

Die Auswahl der Umsetzungsgruppe, die beim Übertragen der Daten verwendet werden muss, erfolgt über die Anweisung SET CURRENT DEFAULT TRANSFORM GROUP, mit der das DB2-Sonderregister CURRENT DEFAULT TRANSFORM GROUP geändert wird. DB2 ermittelt anhand des Wertes für dieses Sonderregister, welche Umsetzungsfunktionen aufgerufen werden müssen, um die erforderlichen Umwandlungen vorzunehmen.

Umsetzungsgruppen können die Anwendungsprogrammierung vereinfachen. Statt in den SQL-Anweisungen explizit Umwandlungsfunktionen zu verwenden, können Sie eine Umsetzungsgruppe angeben, und diese Task auf diesem Weg an DB2 übergeben.

Umsetzungsgruppe ST_WellKnownText

Mit der Umsetzungsgruppe ST_WellKnownText können Sie Daten mithilfe der WKT-Darstellung von und an DB2 übertragen.

Beim Aufheben der Bindung eines Werts vom Datenbankserver an den Client wird eine Geometrie mit derselben Funktion in die WKT-Darstellung umgewandelt, die auch durch die Funktion ST_AsText() bereitgestellt wird. Wird die WKT-Darstellung einer Geometrie an den Datenbankserver übertragen, werden die Umwandlungen in einen Wert des Typs ST_Geometry implizit mit der Funktion ST_Geometry(CLOB) ausgeführt. Durch das Binden von Werten an DB2 über Umsetzungsgruppen werden die Geometrien im räumlichen Bezugssystem mit der numerischen Kennung 0 (null) dargestellt.

Beispiele

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen kann entsprechend der jeweiligen Onlineanzeige variieren.

Beispiel 1

Das folgende SQL-Script veranschaulicht, wie Sie mit der Umsetzungsgruppe ST_WellKnownText eine Geometrie in ihrer WKT-Darstellung abrufen können, ohne dass die explizite Funktion ST_AsText verwendet wird.

```
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
```

Ergebnisse:

```
ID   GEOM
---  ------------------------------------------------------------------
  1  LINESTRING ( 100.00000000 100.00000000, 200.00000000 100.00000000)
```

Beispiel 2

Der folgende C-Code illustriert, wie Sie mit der Umsetzungsgruppe ST_WellKnownText Geometrien einfügen können, wobei die explizite Funktion ST_Geometry für die Hostvariable `wkt_buffer` verwendet wird. Diese Variable hat den Typ CLOB und enthält die WKT-Darstellung des Punktes (10 10), der eingefügt werden soll.

```
EXEC SQL BEGIN DECLARE SECTION;
      sqlint32 id = 0;
      SQL TYPE IS db2gse.ST_Geometry AS CLOB(1000) wkt_buffer;
              EXEC SQL END DECLARE SECTION;

// Umsetzungsgruppe für alle nachfolgenden SQL-Anweisungen definieren
 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);

// Punkt mit WKT in Spalte des Typs ST_Geometry einfügen
 EXEC SQL
        INSERT
    INTO   transforms_sample(id, geom)
    VALUES (:id, :wkt_buffer);
```

Umsetzungsgruppe ST_WellKnownBinary

Mit der Umsetzungsgruppe ST_WellKnownBinary können Sie Daten, die in WKB-Darstellung vorliegen, von und an DB2 übertragen.

Beim Aufheben der Bindung eines Werts vom Datenbankserver an den Client wird eine Geometrie mit derselben Funktion in die WKB-Darstellung umgewandelt, die auch durch die Funktion ST_AsBinary() bereitgestellt wird. Wird die WKB-Darstellung einer Geometrie an den Datenbankserver übertragen, werden die Umwandlungen in einen Wert des Typs ST_Geometry implizit mit der Funktion ST_Geometry(BLOB) ausgeführt. Durch das Binden von Werten an DB2 über Umsetzungsgruppen können die Geometrien im räumlichen Bezugssystem mit der numerischen Kennung 0 (null) dargestellt werden.

Beispiel

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen kann entsprechend der jeweiligen Onlineanzeige variieren.

Beispiel 1

Das folgende SQL-Script veranschaulicht, wie Sie mit der Umsetzungsgruppe ST_WellKnownBinary eine Geometrie in ihrer WKB-Darstellung abrufen können, ohne dass die explizite Funktion ST_AsBinary verwendet wird.

```
CREATE TABLE transforms_sample (
      id INTEGER,
      geom db2gse.ST_Geometry)

     INSERT
      INTO transforms_sample
      VALUES ( 1, db2gse.ST_Polygon('polygon ((10 10, 20 10, 20 20,
                          10 20, 10 10))', 0))

SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownBinary

   SELECT id, geom
      FROM transforms_sample
   WHERE  id = 1
```

Ergebnisse:

```
ID    GEOM
----   ------------------------------------------------
   1   x'0103000000010000000500000000000000000000244000
       0000000000244000000000000024400000000000003440
       0000000000003440000000000000344000000000000034
       4000000000000024400000000000002440000000000000
       2440'
```

Beispiel 2

Der folgende C-Code illustriert, wie Sie mit der Umsetzungsgruppe ST_WellKnownBinary Geometrien einfügen können, wobei die explizite Funktion ST_Geometry für die Hostvariable `wkb_buffer` verwendet wird. Diese Variable hat den Typ BLOB und enthält die WKB-Darstellung einer Geometrie, die eingefügt werden soll.

```
EXEC SQL BEGIN DECLARE SECTION;
      sqlint32 id = 0;
      SQL TYPE IS db2gse.ST_Geometry AS BLOB(1000) wkb_buffer;
              EXEC SQL END DECLARE SECTION;

// Umsetzungsgruppe für alle nachfolgenden SQL-Anweisungen definieren
 EXEC SQL
      SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownBinary;

// Hostvariablen initialisieren
...
// Geometrie mit WKB in Spalte des Typs ST_Geometry einfügen

 EXEC SQL
        INSERT
    INTO   transforms_sample(id, geom)
    VALUES ( :id, :wkb_buffer );
```

Umsetzungsgruppe ST_Shape

Mit der Umsetzungsgruppe ST_Shape können Sie Daten, die in ESRI-Formdarstellung vorliegen, von und an DB2 übertragen.

Beim Aufheben der Bindung eines Werts vom Datenbankserver an den Client wird eine Geometrie mit derselben Funktion in ihre Formdarstellung umgewandelt, die auch durch die Funktion ST_AsShape() bereitgestellt wird. Wird die Formdarstellung einer Geometrie an den Datenbankserver übertragen, werden die Umwandlungen in einen Wert des Typs ST_Geometry implizit mit der Funktion ST_Geometry(BLOB) ausgeführt. Durch das Binden von Werten an DB2 über Umsetzungsgruppen können die Geometrien im räumlichen Bezugssystem mit der numerischen Kennung 0 (null) dargestellt werden.

Beispiele

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen kann entsprechend der jeweiligen Onlineanzeige variieren.

Beispiel 1

Das folgende SQL-Script veranschaulicht, wie Sie mit der Umsetzungsgruppe ST_Shape eine Geometrie in ihrer Formdarstellung abrufen können, ohne dass die explizite Funktion ST_AsShape verwendet wird.

```
CREATE TABLE transforms_sample(
      id INTEGER,
      geom db2gse.ST_Geometry)

     INSERT
      INTO transforms_sample
      VALUES ( 1, db2gse.ST_Point(20.0, 30.0, 0) )

SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape

   SELECT id, geom
      FROM transforms_sample
   WHERE  id = 1
```

Ergebnisse:

```
ID    GEOM
----   ------------------------------------------------
   1   x'010000000000000000003440000000000003E40'
```

Beispiel 2

Der folgende C-Code demonstriert, wie Sie mit der Umsetzungsgruppe ST_Shape Geometrien einfügen können, wobei die explizite Funktion ST_Geometry für die Hostvariable `shape_buffer` verwendet wird. Diese Variable hat den Typ BLOB und enthält die Formdarstellung einer Geometrie, die eingefügt werden soll.

```
EXEC SQL BEGIN DECLARE SECTION;
      sqlint32 id = 0;
      SQL TYPE IS db2gse.ST_Geometry AS BLOB(1000) shape_buffer;
              EXEC SQL END DECLARE SECTION;

// Umsetzungsgruppe für alle nachfolgenden SQL-Anweisungen definieren
 EXEC SQL
   SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape;

// Hostvariablen initialisieren
```

```
...

SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape;

// Geometrie mit Formdarstellung
// in Spalte des Typs ST_Geometry einfügen
 EXEC SQL
          INSERT
    INTO   transforms_sample(id, geom)
    VALUES ( :id, :shape_buffer );
```

Umsetzungsgruppe ST_GML

Mit der Umsetzungsgruppe ST_GML können Sie Daten, die GML (Geography Markup Language) verwenden, von und an DB2 übertragen.

Beim Aufheben der Bindung eines Werts vom Datenbankserver an den Client wird eine Geometrie mit derselben Funktion in ihre GML-Darstellung umgewandelt, die auch durch die Funktion ST_AsGML() bereitgestellt wird. Wird die GML-Darstellung einer Geometrie an den Datenbankserver übertragen, werden die Umwandlungen in einen Wert des Typs ST_Geometry implizit mit der Funktion ST_Geometry(CLOB) ausgeführt. Durch das Binden von Werten an DB2 über Umsetzungsgruppen können die Geometrien im räumlichen Bezugssystem mit der numerischen Kennung 0 (null) dargestellt werden.

Beispiele

In den folgenden Beispielen wurden die Zeilen mit den Ergebnissen zur besseren Lesbarkeit umformatiert. Der Abstand bei Ihren Ergebnissen kann entsprechend der jeweiligen Onlineanzeige variieren.

Beispiel 1

Das folgende SQL-Script veranschaulicht, wie Sie mit der Umsetzungsgruppe ST_GML eine Geometrie in ihrer GML-Darstellung abrufen können, ohne dass die explizite Funktion ST_AsGML verwendet wird.

```
CREATE TABLE transforms_sample (
      id INTEGER,
      geom db2gse.ST_Geometry)

     INSERT
    INTO  transforms_sample
    VALUES ( 1, db2gse.ST_Geometry('multipoint z (10 10
                      3, 20 20 4, 15 20 30)', 0) )
  SET CURRENT DEFAULT TRANSFORM GROUP = ST_GML

   SELECT id, geom
FROM  transforms_sample
WHERE  id = 1
```

Ergebnisse:

```
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>
```

Beispiel 2

Der folgende C-Code demonstriert, wie Sie mit der Umsetzungsgruppe ST_GML Geometrien einfügen können, ohne dass die explizite Funktion ST_Geometry für die Hostvariable gml_buffer zu verwenden. Diese Variable hat den Typ CLOB und enthält die GML-Darstellung des einzufügenden Punktes (20 ,20).

```
EXEC SQL BEGIN DECLARE SECTION;
      sqlint32 id = 0;
      SQL TYPE IS db2gse.ST_Geometry AS CLOB(1000) gml_buffer;
              EXEC SQL END DECLARE SECTION;

// Umsetzungsgruppe für alle nachfolgenden SQL-Anweisungen definieren
 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>");

// Hostvariablen initialisieren
wkt_buffer.length = strlen(gml_buffer.data);

// Punkt mit WKT in Spalte des Typs ST_Geometry einfügen
 EXEC SQL
        INSERT
    INTO   transforms_sample(id, geom)
   VALUES ( :id, :gml_buffer );
```

Kapitel 20. Unterstützte Datenformate

DB2 Spatial Extender stellt Branchenstandardformate für räumliche Daten bereit, die verwendet werden können.

Die folgenden vier Formate für räumliche Daten werden beschrieben:

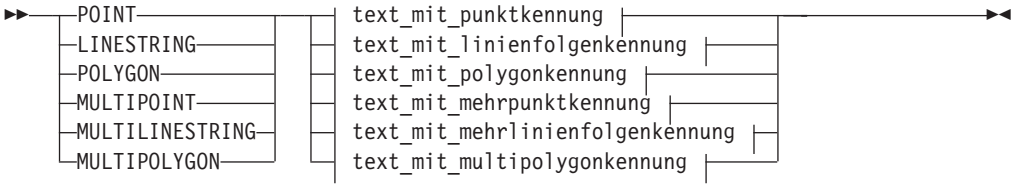
- WKT-Darstellung (WKT = Well-Known Text)
- WKB-Darstellung (WKB = Well-Known Binary)
- Formdarstellung
- GML-Darstellung (GML = Geography Markup Language)

WKT-Darstellung (WKT = Well-Known Text)

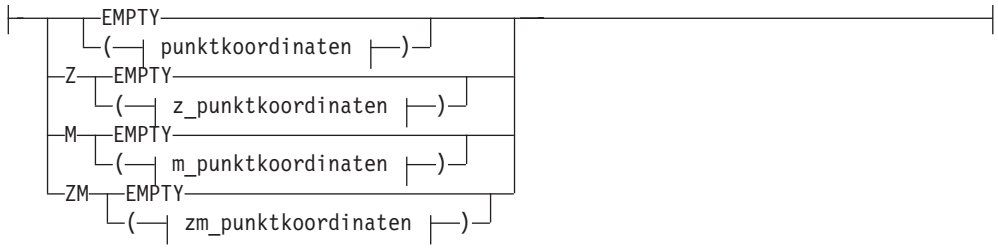
Die OpenGIS Consortium-Spezifikation "Simple Features for SQL" definiert die WKT-Darstellung (WKT = Well-Known Text) für den Austausch von Geometriedaten im ASCII-Format. Diese Darstellung wird auch durch den ISO-Standard "SQL/MM Part: 3 Spatial" angegeben.

Informationen zu Funktionen, die WKT-Daten verwenden und erzeugen, finden Sie unter "Räumliche Funktionen, die Geometrien in Datenaustauschformate umwandeln und umgekehrt".

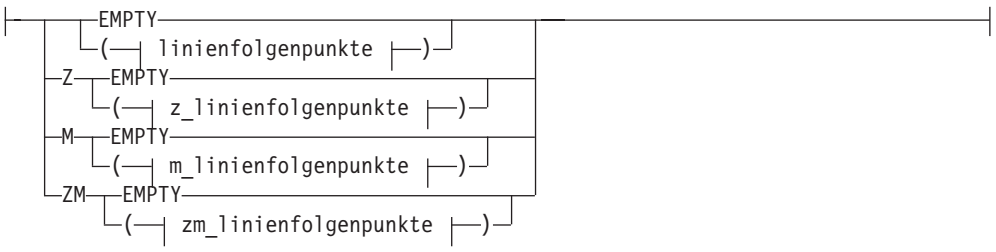
Die WKT-Darstellung einer Geometrie ist wie folgt definiert:



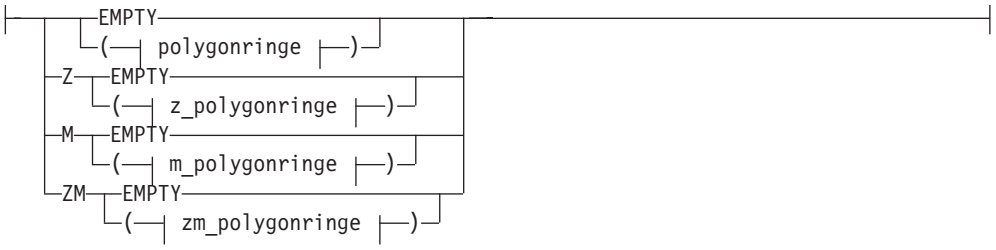
text_mit_punktkennung:



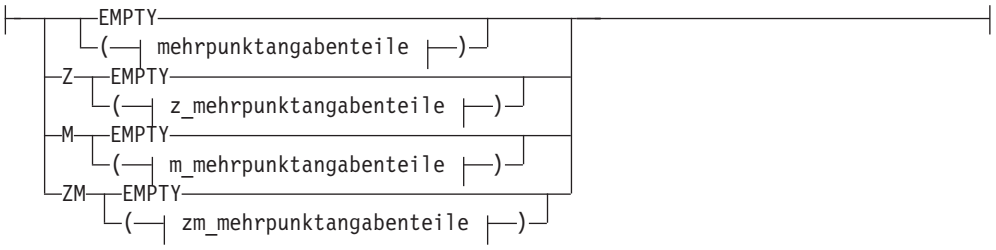
text_mit_linenfolgenkennung:



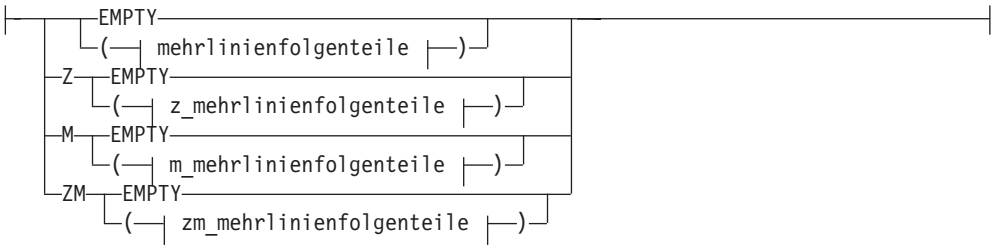
text_mit_polygonkennung:



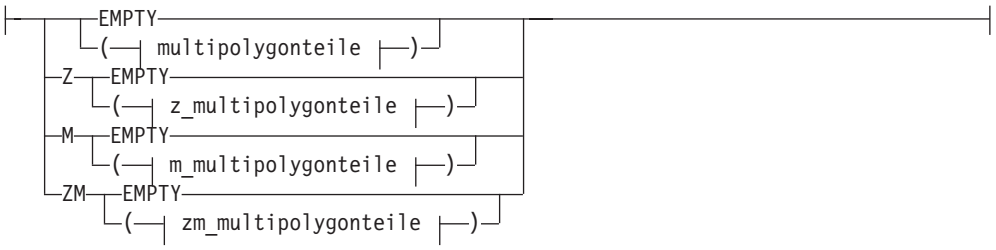
text_mit_mehrpunktkennung:



text_mit_mehrlinienfolgenkennung:

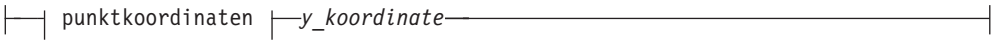


text_mit_multipolygonkennung:



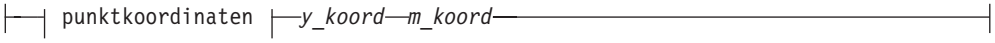
punktkoordinaten:

z_punktkoordinaten:



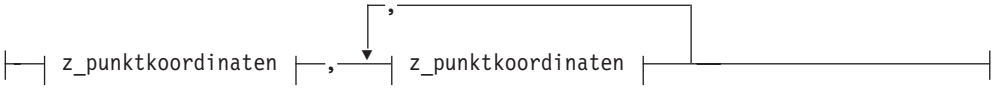
m_punktkoordinaten:

zm_punktkoordinaten:



linienfolgenpunkte:

z_linienfolgenpunkte:

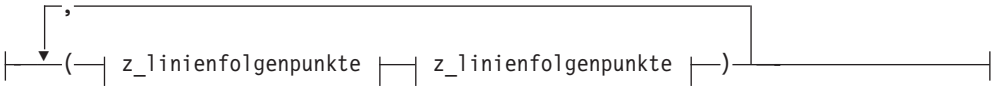


m_linienfolgenpunkte:

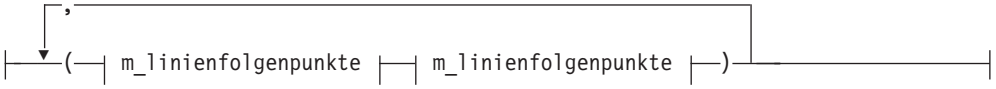
zm_linienfolgenpunkte:

polygonringe:

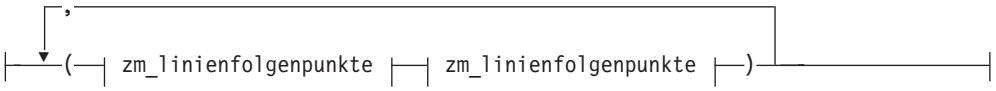
z_polygonringe:



m_polygonringe:



zm_polygonringe:



mehrpunktangabenteile:

z_mehrpunktangabenteile:

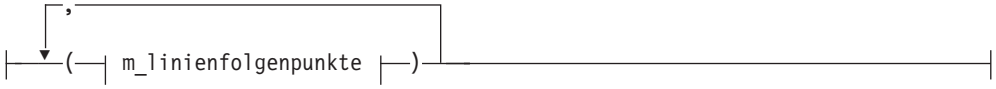
m_mehrpunktangabenteile:

zm_mehrpunktangabenteile:

mehrlinienfolgenteile:

z_mehrlinienfolgenteile:

m_mehrlinienfolgenteile:



zm_mehrlinienfolgenteile:

multipolygonteile:

z_multipolygonteile:

m_multipolygonteile:

zm_multipolygonteile:

Parameter

x_koordinate

Ein numerischer Wert (fester Wert, ganze Zahl oder Gleitkommazahl), der die X-Koordinate eines Punktes angibt.

y_koordinate

Ein numerischer Wert (fester Wert, ganze Zahl oder Gleitkommazahl), der die Y-Koordinate eines Punktes angibt.

z_koordinate

Ein numerischer Wert (fester Wert, ganze Zahl oder Gleitkommazahl), der die Z-Koordinate eines Punktes angibt.

m_koordinate

Ein numerischer Wert (fester Wert, ganze Zahl oder Gleitkommazahl), der die M-Koordinate (Bemaßung) eines Punktes angibt.

Wenn die Geometrie leer ist, muss anstelle der Koordinatenliste das Schlüsselwort EMPTY angegeben werden. Das Schlüsselwort EMPTY darf nicht in die Koordinatenliste eingebettet werden.

Die folgende Tabelle enthält einige Beispiele für gültige Textdarstellungen.

Tabelle 33. Geometrietypen und ihre Textdarstellung

Geometrietyp	WKT-Darstellung	Kommentar
Punkt	POINT EMPTY	Leerer Punkt
Punkt	POINT (10.05 10.28)	Punkt
Punkt	POINT Z(10.05 10.28 2.51)	Punkt mit Z-Koordinate
Punkt	POINT M(10.05 10.28 4.72)	Punkt mit M-Koordinate
Punkt	POINT ZM(10.05 10.28 2.51 4.72)	Punkt mit Z-Koordinate und M-Koordinate
Linienfolge	LINESTRING EMPTY	Leere Linienfolge
Polygon	POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))	Polygon
Mehrpunktangabe	MULTIPOINT Z(10 10 2, 20 20 3)	Mehrpunktangabe mit Z-Koordinaten
Mehrlinienfolge	MULTILINESTRING M((310 30 1, 40 30 20, 50 20 10)(10 10 0, 20 20 1))	Mehrlinienfolge mit M-Koordinaten
Multipolygon	MULTIPOLYGON ZM(((1 1 1 1, 1 2 3 4, 2 2 5 6, 2 1 7 8, 1 1 1 1)))	Multipolygon mit Z-Koordinaten und M-Koordinaten

WKB-Darstellung (WKB = Well-Known Binary)

Der folgende Abschnitt beschreibt die WKB-Darstellung (WKB = Well-Known Binary) für Geometrien.

Die OpenGIS Consortium-Spezifikation "Simple Features for SQL" definiert die WKB-Darstellung. Diese Darstellung wird auch durch den ISO-Standard "SQL/MM Part: 3 Spatial" definiert. Informationen zu Funktionen, die WKB verwenden und erzeugen, finden Sie am Ende dieses Abschnitts unter "Zugehörige Referenzen".

Der Grundbaustein für WKB-Darstellungen ist der Bytestrom für einen Punkt, der aus zwei Doppelwerten besteht. Die Byteströme für andere Geometrien werden unter Verwendung der Byteströme von bereits definierten Geometrien erstellt.

Das folgende Beispiel zeigt den Grundbaustein für WKB-Darstellungen.

```
// Basistypdefinitionen
// byte : 1 Byte
// uint32 : 32-Bit-Integer ohne Vorzeichen (4 Byte)
// double : Zahl doppelter Genauigkeit (8 Byte)

// Bausteine : 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;
  }
};
```

Die folgende Abbildung ist ein Beispiel für eine Geometrie in WKB-Darstellung, bei der die NDR-Codierung verwendet wird.

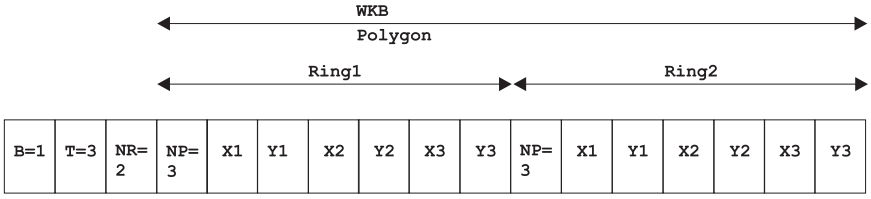


Abbildung 21. Geometriedarstellung im NDR-Format. (B=1) des Typs "Polygon" (T=3) mit 2 Linearen (NR=2), wobei jeder Ring drei Punkte hat (NP=3).

Formdarstellung

Die Formdarstellung ist ein weit verbreiteter Branchenstandard, der durch ESRI definiert wurde.

Eine vollständige Beschreibung der Formdarstellung finden Sie auf der ESRI-Website unter der Adresse http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

GML-Darstellung (GML = Geography Markup Language)

GML (Geography Markup Language) ist eine XML-Codierung für geografische Informationen, die in der OpenGIS Consortium-Spezifikation "Geography Markup Language V2" definiert ist.

DB2 Spatial Extender bietet verschiedene Funktionen, die Geometrien aus GML-Darstellungen (Geography Markup Language) generieren.

Weitere Details zu dieser OpenGIS Consortium-Spezifikation finden Sie unter „OpenGIS Geography Markup Language (GML) Encoding Standard” auf der Website http://www.opengeospatial.org/standards/gml.

Kapitel 21. Unterstützte Koordinatensysteme

DB2 Spatial Extender verwendet eine spezielle Syntax von Koordinatensystemen und unterstützte Koordinatensystemwerte, um eine Standardtextdarstellung für Informationen zum Koordinatensystem bereitzustellen.

Syntax von Koordinatensystemen

Die Syntax von Koordinatensystemen ist eine Zeichenfolgedarstellung eines solchen Koordinatensystems.

Die WKT-Darstellung (WKT = Well-Known Text) von räumlichen Bezugssystemen bietet eine Standardtextdarstellung für Informationen zum Koordinatensystem. Die Definitionen der WKT-Darstellung (WKT = Well-Known Text Representation; bekannte Textdarstellung) werden in der OGC-Spezifikation "Simple Features for SQL" und im ISO-Standard "SQL/MM Part 3: Spatial" definiert.

Ein Koordinatensystem ist ein (auf Breiten- und Längengradwerten basierendes) geografisches Koordinatensystem, ein (auf X- und Y-Werten basierendes) projiziertes Koordinatensystem oder ein (auf X-, Y- und Z-Werten basierendes) geozentrisches Koordinatensystem. Das Koordinatensystem besteht aus mehreren Objekten. Jedes Objekt verfügt über ein in Großbuchstaben angegebenes Schlüsselwort (z. B. DATUM oder UNIT), dem die durch Kommas abgetrennten Definitionsparameter des Objekts in eckigen Klammern folgen. Manche Objekte sind aus anderen Objekten zusammengesetzt, sodass das Ergebnis eine verschachtelte Struktur darstellt.

Anmerkung: Implementierungen können statt der eckigen Klammern [ ] auch runde Klammern () verwenden und sollten nach Möglichkeit beide Arten von Klammern lesen können.

Die EBNF-Definition (Extended Backus Naur Form) für die Zeichenfolgedarstellung eines Koordinatensystems mit eckigen Klammern lautet wie folgt (siehe vorherige Anmerkung zur Verwendung der eckigen Klammern):

```
<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>
```

Der Typ des Koordinatensystems wird durch das verwendete Schlüsselwort kenntlich gemacht:

PROJCS
Das Koordinatensystem eines Datensatzes wird durch das Schlüsselwort PROJCS angegeben, wenn die Daten in projizierten Koordinaten stehen.

GEOGCS
Das Koordinatensystem eines Datensatzes wird durch das Schlüsselwort GEOGCS angegeben, wenn die Daten in geografischen Koordinaten stehen.

GEOCCS

Das Koordinatensystem eines Datensatzes wird durch das Schlüsselwort GEOCCS angegeben, wenn die Daten in geozentrischen Koordinaten stehen.

Das Schlüsselwort PROJCS wird gefolgt von allen "Bestandteilen", die das projizierte Koordinatensystem definieren. Das erste Bestandteil jedes Objekts ist immer der Name. Auf den Namen des projizierten Koordinatensystems folgen mehrere Objekte: das geografische Koordinatensystem, die Kartenprojektion, einer oder mehrere Parameter und die lineare Maßeinheit. Alle projizierten Koordinatensysteme basieren auf einem geografischen Koordinatensystem; in diesem Abschnitt werden daher zunächst die Bestandteile beschrieben, die für ein projiziertes Koordinatensystem spezifisch sind. Das System "UTM zone 10N" für das Datum NAD83 ist beispielsweise wie folgt definiert:

```
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]]
```

Der Name und verschiedene Objekte definieren wiederum das geografische Koordinatensystemobjekt: das Datum, der Nullmeridian und die Winkelmaßeinheit.

```
<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>
```

Die große Halbachse wird in Metern gemessen und muss größer als null sein.

Die Zeichenfolge für das geografische Koordinatensystem "UTM zone 10" für NAD83 lautet:

```
GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",
SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],
UNIT["Degree",0.0174532925199433]]
```

Das Objekt UNIT kann eine Winkeleinheit oder eine lineare Maßeinheit sein:

```
<angular unit> = <unit>
<linear unit> = <unit>
<unit> = UNIT["<name>", <conversion factor>]
<conversion factor> = <number>
```

Der Umwandlungsfaktor gibt die Anzahl der Meter (bei einer linearen Einheit) oder die Anzahl der Bogenmaße (bei einer Winkeleinheit) pro Einheit an und muss größer als null sein.

Die vollständige Zeichenfolgendarstellung für "UTM Zone 10N" lautet also wie folgt:

```
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]]
```

Ein geozentrisches Koordinatensystem ähnelt einem geografischen Koordinatensystem:

```
<geocentric cs> = GEOCCS["<name>", <datum>, <prime meridian>, <linear unit>]
```

Unterstützte lineare Einheiten

Verwenden Sie lineare Einheiten, die von DB2 Spatial Extender unterstützt werden.

Tabelle 34. Unterstützte lineare Einheiten

Einheit	Umwandlungsfaktor
Meter	1,0
Fuß (international)	0,3048
Fuß (US)	12/39,37
Modifizierter amerikanischer Fuß	12,0004584/39,37
Clarkes Fuß	12/39,370432
Indischer Fuß	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 (indisch)	36/39,370141
Yard (Sears)	36/39,370147
Fathom	1,8288
Nautische Meile	1852,0

Unterstützte Winkeleinheiten

Verwenden Sie Winkeleinheiten, die von DB2 Spatial Extender unterstützt werden.

Tabelle 35. Unterstützte Winkeleinheiten

Einheit	Gültiger Bereich für Breitengrad	Gültiger Bereich für Längengrad	Umwandlungsfaktor
Bogenmaß	–pi/2 und pi/2 Radiane (inklusive)	–pi und pi Radiane (inklusive)	1,0
Dezimalgrad	–90 und 90 Grad (inklusive)	–180 und 180 Grad (inklusive)	pi/180

Tabelle 35. Unterstützte Winkeleinheiten (Forts.)

Einheit	Gültiger Bereich für Breitengrad	Gültiger Bereich für Längengrad	Umwandlungsfaktor
Dezimale Minute	–5400 und 5400 Minuten (inklusive)	–10800 und 10800 Minuten (inklusive)	(pi/180)/60
Dezimale Sekunde	–324000 und 324000 Sekunden (inklusive)	–648000 und 648000 Sekunden (inklusive)	(pi/180)*3600
Gon	–100 und 100 Gon (inklusive)	–200 und 200 Gon (inklusive)	pi/200
Grad	–100 und 100 Gon (inklusive)	–200 und 200 Gon (inklusive)	pi/200

Unterstützte Sphäroide

Verwenden Sie Sphäroide, die von DB2 Spatial Extender unterstützt werden.

Tabelle 36. Unterstützte Sphäroide

Name	Große Halbachse	Inversionsabflachung
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

Tabelle 36. Unterstützte Sphäroide (Forts.)

Name	Große Halbachse	Inversionsabflachung
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
GRS 1980	6378137,0	298,257222101
Helmert 1906	6378200,0	298,3
Hough 1960	6378270,0	297,0
Indonesian National Spheroid	6378160,0	298,247
International 1924	6378388,0	297,0
International 1967	6378160,0	298,25
Krassowsky 1940	6378245,0	298,3
NWL 9D	6378145,0	298,25
NWL 10D	6378135,0	298,26
OSU 86F	6378136,2	298,25722
OSU 91A	6378136,3	298,25722
Plessis 1817	6376523,0	308,64
Sphere	6371000,0	0,0
Sphere (ArcInfo)	6370997,0	0,0
Struve 1860	6378298,3	294,73
Walbeck	6376896,0	302,78
War Office	6378300,0	296,0
WGS 1966	6378145,0	298,25
WGS 1972	6378135,0	298,26

Tabelle 36. Unterstützte Sphäroide (Forts.)

Name	Große Halbachse	Inversionsabflachung
WGS 1984	6378137,0	298,257223563

Unterstützte Nullmeridiane

Verwenden Sie Nullmeridiane, die von DB2 Spatial Extender unterstützt werden.

Tabelle 37. Unterstützte Nullmeridiane

Standort	Koordinaten
Greenwich	0° 0' 0"
Bern	7° 26' 22.5" E
Bogota	74° 4' 51.3" W
Brüssel	4° 22' 4.71" E
Ferro	17° 40' 0" W
Jakarta	106° 48' 27.79" E
Lissabon	9° 7' 54.862" W
Madrid	3° 41' 16.58" W
Paris	2° 20' 14.025" E
Rom	12° 27' 8.4" E
Stockholm	18° 3' 29" E

Unterstützte Kartenprojektionen

Verwenden Sie Kartenprojektionen, die von DB2 Spatial Extender unterstützt werden.

Tabelle 38. Zylinderprojektionen

Zylinderprojektionen	Pseudozylinderprojektionen
Behrmann	Parabolische Projektion nach Craster
Cassini	Eckert I
Flächentreue Zylinderprojektion	Eckert II
Winkeltreue Projektion	Eckert III
Stereografische Projektion nach Gall	Eckert IV
Gauß-Krüger	Eckert V
Merkatorprojektion	Eckert VI

Tabelle 38. Zylinderprojektionen (Forts.)

Zylinderprojektionen	Pseudozylinderprojektionen
Zylinderprojektion nach Miller	Biquadratische polare Flachprojektion nach McBryde-Thomas
Schrägprojektion	Mercator (Hotine) Mollweide
Plate-Carée	Robinson
Times	Sinusoidal (Sansom-Flamsteed)
Transversale Merkatorprojektion	Winkel I

Tabelle 39. Kegelprojektionen

Name	Kegelprojektion
Flächentreue Kegelprojektion nach Albers	Trimetrische Projektion nach Chamberlin
Bipolare winkeltreue konische Schrägprojektion	Abstandstreue 2-Punkt-Projektion
Bonne	Flächentreue Projektion nach Hammer-Aitoff
Abstandstreue Kegelprojektion	Van der Grinten I
Konforme Kegelprojektion nach Lambert	Verschiedene
Polykonische Projektion	Alaska series E
Einfache Kegelprojektion	Alaska Grid (modifizierte stereografische Projektion nach Snyder)

Tabelle 40. Kartenprojektionsparameter

Parameter	Beschreibung
central_meridian	Der als Ursprung der X-Koordinaten ausgewählte Längengrad.
scale_factor	Wird im Allgemeinen verwendet, um die Verzerrung in Kartenprojektionen zu verringern.
standard_parallel_1	Ein Breitengrad, der normalerweise keine Verzerrung aufweist. Er wird auch für "maßstabsgerechter Breitengrad" verwendet.
standard_parallel_2	Ein Längengrad, der normalerweise keine Verzerrung aufweist.
longitude_of_center	Der Längengrad, der den Mittelpunkt der Kartenprojektion definiert.
latitude_of_center	Der Breitengrad, der den Mittelpunkt der Kartenprojektion definiert.
longitude_of_origin	Der Längengrad, der als Ursprung der X-Koordinaten ausgewählt wurde.

Tabelle 40. Kartenprojektionsparameter (Forts.)

Parameter	Beschreibung
latitude_of_origin	Der Breitengrad, der als Ursprung der Y-Koordinaten ausgewählt wurde.
false_easting	Ein Wert, der zu X-Koordinaten hinzugefügt wird, damit alle X-Koordinaten einen positiven Wert aufweisen.
false_northing	Ein Wert, der zu Y-Koordinaten hinzugefügt wird, damit alle Y-Koordinaten einen positiven Wert aufweisen.
azimuth	Der Winkel östlich von Nord, der die Mittellinie einer schiefen Projektion definiert.
longitude_of_point_1	Der Längengrad des ersten für eine Kartenprojektion erforderlichen Punkts.
latitude_of_point_1	Der Breitengrad des ersten für eine Kartenprojektion erforderlichen Punkts.
longitude_of_point_2	Der Längengrad des zweiten für eine Kartenprojektion erforderlichen Punkts.
latitude_of_point_2	Der Breitengrad des zweiten für eine Kartenprojektion erforderlichen Punkts.
longitude_of_point_3	Der Längengrad des dritten für eine Kartenprojektion erforderlichen Punkts.
latitude_of_point_3	Der Breitengrad des dritten für eine Kartenprojektion erforderlichen Punkts.
landsat_number	Die Nummer eines Landsat-Satelliten.
path_number	Die Umlaufbahnnummer für einen bestimmten Satelliten.
perspective_point_height	Die Höhe des perspektivischen Punkts der Kartenprojektion über der Erde.
fipszone	Zonennummer des SPC-Koordinatensystems (SPC = State Plane Coordinate).
zone	UTM-Zonennummer.

Anhang A. Übersicht über technische Informationen zu DB2

Technische Informationen zu DB2 liegen in verschiedenen Formaten vor, die auf unterschiedliche Weise abgerufen werden können.

Die technischen Informationen zu DB2 stehen über die folgenden Tools und Methoden zur Verfügung:

- DB2 Information Center
  - Themen (zu Tasks, Konzepten und Referenzinformationen)
  - Beispielprogramme
  - Lernprogramme
- DB2-Bücher
  - PDF-Dateien (für den Download verfügbar)
  - PDF-Dateien (auf der DB2-PDF-DVD)
  - Gedruckte Bücher
- Hilfe für Befehlszeile
  - Hilfe für Befehle
  - Hilfe für Nachrichten

Anmerkung: Die Themen des DB2 Information Center werden häufiger aktualisiert als die PDF- und Hardcopybücher. Um stets die neuesten Informationen zur Verfügung zu haben, sollten Sie die Dokumentationsaktualisierungen installieren, sobald diese verfügbar sind, oder das DB2 Information Center unter ibm.com aufrufen.

Darüber hinaus können Sie auf zusätzliche technische Informationen zu DB2, wie beispielsweise technische Hinweise (Technotes), White Papers und IBM Redbooks, online über ibm.com zugreifen. Rufen Sie dazu die Website 'DB2 Information Management - Software - Library' unter http://www.ibm.com/software/data/sw-library/ auf.

Feedback zur Dokumentation

Senden Sie uns Ihr Feedback zur DB2-Dokumentation! Wenn Sie Anregungen zur Verbesserung der DB2-Dokumentation haben, senden Sie eine E-Mail an db2docs@ca.ibm.com. Das DB2-Dokumentationsteam bearbeitet das gesamte Feedback, kann jedoch nicht im Einzelnen auf Ihre E-Mails antworten. Nennen Sie uns, wenn möglich, konkrete Beispiele, sodass wir die Problemstellung besser beurteilen können. Wenn Sie uns Feedback zu einem bestimmten Thema oder einer bestimmten Hilfedatei senden, geben Sie den entsprechenden Titel sowie die URL an.

Verwenden Sie diese E-Mail-Adresse nicht, wenn Sie sich an den DB2-Kundendienst wenden möchten. Wenn ein technisches Problem bei DB2 vorliegt, das Sie mithilfe der Dokumentation nicht beheben können, fordern Sie beim zuständigen IBM Service-Center Unterstützung an.

Bibliothek mit technischen Informationen zu DB2 im Hardcopy- oder PDF-Format

Die folgenden Tabellen enthalten eine Beschreibung der DB2-Bibliothek, die im IBM Publications Center unter www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss zur Verfügung steht. Über die folgende Adresse können Sie englische Handbücher im PDF-Format sowie übersetzte Versionen zu DB2 Version 10.1 herunterladen: www.ibm.com/support/docview.wss?rs=71&uid=swg2700947.

In den Tabellen sind die Bücher, die in gedruckter Form zur Verfügung stehen, gekennzeichnet; möglicherweise sind diese in Ihrem Land oder Ihrer Region jedoch nicht verfügbar.

Die Formnummer wird bei jeder Aktualisierung eines Handbuchs erhöht. Anhand der nachfolgenden Liste können Sie sicherstellen, dass Sie die jeweils neueste Version des Handbuchs lesen.

Anmerkung: Das *DB2 Information Center* wird häufiger aktualisiert als die PDF- und Hardcopybücher.

Tabelle 41. Technische Informationen zu DB2

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Administrative API Reference	SC27-3864-00	Ja	April 2012
Administrative Routines and Views	SC27-3865-00	Nein	April 2012
Call Level Interface Guide and Reference Volume 1	SC27-3866-00	Ja	April 2012
Call Level Interface Guide and Reference Volume 2	SC27-3867-00	Ja	April 2012
Command Reference	SC27-3868-00	Ja	April 2012
Datenbankverwaltung - Konzepte und Konfiguration - Referenzinformationen	SC12-4673-00	Ja	April 2012
Dienstprogramme für das Versetzen von Daten - Handbuch und Referenz	SC12-4691-00	Ja	April 2012
Datenbanküberwachung - Handbuch und Referenz	SC12-4674-00	Ja	April 2012
Datenrecovery und hohe Verfügbarkeit - Handbuch und Referenz	SC12-4692-00	Ja	April 2012
Datenbanksicherheit	SC12-4693-00	Ja	April 2012
DB2 Workload Management - Handbuch und Referenz	SC12-4683-00	Ja	April 2012

Tabelle 41. Technische Informationen zu DB2 (Forts.)

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Developing ADO.NET and OLE DB Applications	SC27-3873-00	Ja	April 2012
Developing Embedded SQL Applications	SC27-3874-00	Ja	April 2012
Developing Java Applications	SC27-3875-00	Ja	April 2012
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-3876-00	Nein	April 2012
Developing User-defined Routines (SQL and External)	SC27-3877-00	Ja	April 2012
Getting Started with Database Application Development	GI13-2046-00	Ja	April 2012
Installation und Verwaltung von DB2 unter Linux und Windows - Erste Schritte	GI11-3285-00	Ja	April 2012
Globalisierung	SC12-4694-00	Ja	April 2012
DB2-Server - Installation	SC12-4677-00	Ja	April 2012
IBM Data Server-Clients - Installation	SC12-4678-00	Nein	April 2012
Fehlernachrichten, Band 1	SC12-4686-00	Nein	April 2012
Fehlernachrichten, Band 2	SC12-4687-00	Nein	April 2012
Net Search Extender - Verwaltung und Benutzerhandbuch	SC12-4689-00	Nein	April 2012
Partitionierung und Clustering	SC12-4695-00	Ja	April 2012
pureXML - Handbuch	SC12-4684-00	Ja	April 2012
Spatial Extender - Benutzer- und Referenzhandbuch	SC12-4688-00	Nein	April 2012
SQL Procedural Languages: Application Enablement and Support	SC27-3896-00	Ja	April 2012
SQL Reference Volume 1	SC27-3885-00	Ja	April 2012
SQL Reference Volume 2	SC27-3886-00	Ja	April 2012
Text Search	SC12-4690-00	Ja	April 2012
Fehlerbehebung und Optimieren der Datenbankleistung	SC12-4675-00	Ja	April 2012

Tabelle 41. Technische Informationen zu DB2 (Forts.)

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
Upgrade auf DB2 Version 10.1	SC12-4676-00	Ja	April 2012
Neuerungen in DB2 Version 10.1	SC12-4682-00	Ja	April 2012
XQuery - Referenz	SC12-4685-00	Nein	April 2012

Tabelle 42. Technische Informationen zu DB2 Connect

Name	IBM Form	In gedruckter Form verfügbar	Letzte Aktualisierung
DB2 Connect - Installation und Konfiguration von DB2 Connect Personal Edition	SC12-4679-00	Ja	April 2012
DB2 Connect - Installation und Konfiguration von DB2 Connect-Servern	SC12-4680-00	Ja	April 2012
DB2 Connect - Benutzerhandbuch	SC12-4681-00	Ja	April 2012

Aufrufen der Hilfe für den SQL-Status über den Befehlszeilenprozessor

DB2-Produkte geben für Bedingungen, die aufgrund einer SQL-Anweisung generiert werden können, einen SQLSTATE-Wert zurück. Die SQLSTATE-Hilfe erläutert die Bedeutung der SQL-Statuswerte und der SQL-Statusklassencodes.

Vorgehensweise

Zum Starten der Hilfe für SQL-Statuswerte müssen Sie den Befehlszeilenprozessor öffnen und Folgendes eingeben:

```
? SQL-Status oder ? Klassencode
```

Hierbei steht *SQL-Status* für einen gültigen fünfstelligen SQL-Statuswert und *Klassencode* für die ersten beiden Ziffern dieses Statuswerts.
So kann beispielsweise durch die Eingabe von `? 08003` Hilfe für den SQL-Statuswert 08003 angezeigt werden, durch die Eingabe von `? 08` Hilfe für den Klassencode 08.

Zugriff auf verschiedene Versionen des DB2 Information Center

Die Dokumentation für andere Versionen der DB2-Produkte finden Sie in den jeweiligen Information Centers unter ibm.com.

Informationen zu diesem Vorgang

Für Themen aus DB2 Version 10.1 lautet die URL für das *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v10r1.

Für Themen aus DB2 Version 9.8 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/.

Für Themen aus DB2 Version 9.7 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/.

Für Themen aus DB2 Version 9.5 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

Für Themen aus DB2 Version 9.1 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v9/.

Für Themen aus DB2 Version 8 lautet die URL des *DB2 Information Center* http://publib.boulder.ibm.com/infocenter/db2luw/v8/.

Aktualisieren des auf Ihrem Computer oder Intranet-Server installierten DB2 Information Center

Ein lokal installiertes DB2 Information Center muss regelmäßig aktualisiert werden.

Vorbereitende Schritte

Ein DB2 Version 10.1 Information Center muss bereits installiert sein. Einzelheiten hierzu finden Sie unter „Installation des DB2 Information Center mit dem DB2-Installationsassistenten" in *DB2-Server - Installation*. Alle für die Installation des Information Center geltenden Voraussetzungen und Einschränkungen gelten auch für die Aktualisierung des Information Center.

Informationen zu diesem Vorgang

Ein vorhandenes DB2 Information Center kann automatisch oder manuell aktualisiert werden:

- Mit automatischen Aktualisierungen werden vorhandene Komponenten und Sprachen des Information Center aktualisiert. Ein Vorteil von automatischen Aktualisierungen ist, dass das Information Center im Vergleich zu einer manuellen Aktualisierung nur für einen kurzenZeitraum nicht verfügbar ist. Darüber hinaus können automatische Aktualisierungen so konfiguriert werden, dass sie als Teil anderer, regelmäßig ausgeführter Stapeljobs ausgeführt werden.
- Mit manuellen Aktualisierungen können Sie vorhandene Komponenten und Sprachen des Information Center aktualisieren. Automatische Aktualisierungen reduzieren die Ausfallzeiten während des Aktualisierungsprozesses, Sie müssen jedoch den manuellen Prozess verwenden, wenn Sie Komponenten oder Sprachen hinzufügen möchten. Beispiel: Ein lokales Information Center wurde ursprünglich sowohl mit englischer als auch mit französischer Sprachunterstützung installiert; nun soll auch die deutsche Sprachunterstützung installiert werden. Bei einer manuellen Aktualisierung werden sowohl eine Installation der deutschen Sprachunterstützung als auch eine Aktualisierung der vorhandenen Komponenten und Sprachen des Information Center durchgeführt. Sie müssen jedoch bei einer manuellen Aktualisierung das Information Center manuell stoppen, aktualisieren und erneut starten. Das Information Center ist während des gesamten Aktualisierungsprozesses nicht verfügbar. Während des automatischen Aktualisierungsprozesses kommt es zu einem Ausfall des Information Center, und es wird erst wieder nach der Aktualisierung erneut gestartet.

Dieser Abschnitt enthält Details zum Prozess der automatischen Aktualisierung. Anweisungen zur manuellen Aktualisierung finden Sie im Abschnitt „Manuelles Aktualisieren des auf Ihrem Computer oder Intranet-Server installierten DB2 Information Center".

Vorgehensweise

Gehen Sie wie folgt vor, um das auf Ihrem Computer bzw. Intranet-Server installierte DB2 Information Center automatisch zu aktualisieren:

1. Unter Linux:
   a. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das DB2 Information Center im Verzeichnis `/opt/ibm/db2ic/V10.1` installiert.
   b. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis `doc/bin`.
   c. Führen Sie das Script `update-ic` aus:
      ```
      update-ic
      ```
2. Unter Windows:
   a. Öffnen Sie ein Befehlsfenster.
   b. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das DB2 Information Center im Verzeichnis `<Programme>\IBM\DB2 Information Center\Version 10.1` installiert, wobei `<Programme>` das Verzeichnis der Programmdateien angibt.
   c. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis `doc\bin`.
   d. Führen Sie die Datei `update-ic.bat` aus:
      ```
      update-ic.bat
      ```

Ergebnisse

Das DB2 Information Center wird automatisch erneut gestartet. Standen Aktualisierungen zur Verfügung, zeigt das Information Center die neuen und aktualisierten Abschnitte an. Waren keine Aktualisierungen für das Information Center verfügbar, wird eine entsprechende Nachricht zum Protokoll hinzugefügt. Die Protokolldatei befindet sich im Verzeichnis `doc\eclipse\configuration`. Der Name der Protokolldatei ist eine Zufallszahl. Beispiel: `1239053440785.log`.

Manuelles Aktualisieren des auf Ihrem Computer oder Intranet-Server installierten DB2 Information Center

Wenn Sie das DB2 Information Center lokal installiert haben, können Sie Dokumentationsaktualisierungen von IBM abrufen und installieren.

Informationen zu diesem Vorgang

Zur manuellen Aktualisierung des lokal installierten *DB2 Information Center* sind die folgenden Schritte erforderlich:

1. Stoppen Sie das *DB2 Information Center* auf Ihrem Computer und starten Sie das Information Center im Standalone-Modus erneut. Die Ausführung des Information Center im Standalone-Modus verhindert, dass andere Benutzer in Ihrem Netz auf das Information Center zugreifen, und ermöglicht das Anwenden von Aktualisierungen. Die Workstationversion des DB2 Information Center wird stets im Standalone-Modus ausgeführt.

2. Verwenden Sie die Aktualisierungsfunktion, um zu prüfen, welche Aktualisierungen verfügbar sind. Falls Aktualisierungen verfügbar sind, die Sie installieren müssen, können Sie die Aktualisierungsfunktion verwenden, um diese abzurufen und zu installieren.

   Anmerkung: Wenn es in der verwendeten Umgebung erforderlich ist, die Aktualisierungen für das *DB2 Information Center* auf einer Maschine zu installieren, die nicht über ein Verbindung zum Internet verfügt, spiegeln Sie die Aktualisierungssite auf ein lokales Dateisystem und verwenden Sie dabei eine Maschine, die mit dem Internet verbunden ist und auf der das *DB2 Information Center* installiert ist. Wenn viele Benutzer Ihres Netzes die Dokumentationsaktualisierungen installieren sollen, können Sie die Zeit, die jeder einzelne Benutzer für die Aktualisierungen benötigt, reduzieren, indem Sie die Aktualisierungssite lokal spiegeln und ein Proxy dafür erstellen.
   Ist dies der Fall, verwenden Sie die Aktualisierungsfunktion, um die Pakete abzurufen. Die Aktualisierungsfunktion ist jedoch nur im Standalone-Modus verfügbar.
3. Stoppen Sie das im Standalone-Modus gestartete Information Center und starten Sie das *DB2 Information Center* auf Ihrem Computer erneut.

Anmerkung: Unter Windows 2008 und Windows Vista (und neueren Versionen) müssen die in diesem Abschnitt aufgeführten Befehle mit Administratorberechtigung ausgeführt werden. Zum Öffnen einer Eingabeaufforderung oder eines Grafiktools mit vollen Administratorberechtigungen klicken Sie mit der rechten Maustaste die Verknüpfung an und wählen Sie **Als Administrator ausführen** aus.

Vorgehensweise

Gehen Sie wie folgt vor, um das auf Ihrem Computer bzw. Intranet-Server installierte *DB2 Information Center* zu aktualisieren:

1. Stoppen Sie das *DB2 Information Center*.
   - Unter Windows: Klicken Sie **Start** > **Systemsteuerung** > **Verwaltung** > **Dienste** an. Klicken Sie mit der rechten Maustaste das **DB2 Information Center** an und wählen Sie **Beenden** aus.
   - Unter Linux: Geben Sie den folgenden Befehl ein:
     ```
     /etc/init.d/db2icdv10 stop
     ```
2. Starten Sie das Information Center im Standalone-Modus.
   - Unter Windows:
     a. Öffnen Sie ein Befehlsfenster.
     b. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das *DB2 Information Center* im Verzeichnis *Programme*\IBM\DB2 Information Center\Version 10.1 installiert, wobei *Programme* das Verzeichnis der Programmdateien angibt.
     c. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis doc\bin.
     d. Führen Sie die Datei help_start.bat aus:
        ```
        help_start.bat
        ```
   - Unter Linux:
     a. Navigieren Sie zu dem Pfad, in dem das Information Center installiert ist. Standardmäßig ist das *DB2 Information Center* im Verzeichnis /opt/ibm/db2ic/V10.1 installiert.
     b. Navigieren Sie vom Installationsverzeichnis in das Verzeichnis doc/bin.

c. Führen Sie das Script `help_start` aus:

```
help_start
```

Der standardmäßig auf dem System verwendete Web-Browser wird geöffnet und zeigt die Standalone-Version des Information Center an.

3. Klicken Sie die Aktualisierungsschaltfläche () an. (JavaScript muss im verwendeten Browser aktiviert sein.) Klicken Sie im rechten Fenster des Information Center die Schaltfläche für die Suche nach Aktualisierungen an. Eine Liste der Aktualisierungen für die vorhandene Dokumentation wird angezeigt.
4. Wählen Sie zum Initiieren des Installationsprozesses die gewünschten Aktualisierungen aus und klicken Sie anschließend die Schaltfläche für die Installation der Aktualisierungen an.
5. Klicken Sie nach Abschluss des Installationsprozesses **Fertigstellen** an.
6. Stoppen Sie das im Standalone-Modus gestartete Information Center:
   - Unter Windows: Navigieren Sie innerhalb des Installationsverzeichnisses zum Verzeichnis `doc\bin`, und führen Sie die Datei `help_end.bat` aus:

     ```
     help_end.bat
     ```

     Anmerkung: Die Stapeldatei `help_end` enthält die Befehle, die erforderlich sind, um die Prozesse, die mit der Stapeldatei `help_start` gestartet wurden, ordnungsgemäß zu stoppen. Verwenden Sie nicht die Tastenkombination `Strg+C` oder eine andere Methode, um `help_start.bat` zu stoppen.
   - Unter Linux: Navigieren Sie innerhalb des Installationsverzeichnisses zum Verzeichnis `doc/bin`, und führen Sie das Script `help_end` aus:

     ```
     help_end
     ```

     Anmerkung: Das Script `help_end` enthält die Befehle, die erforderlich sind, um die Prozesse, die mit dem Script `help_start` gestartet wurden, ordnungsgemäß zu stoppen. Verwenden Sie keine andere Methode, um das Script `help_start` zu stoppen.
7. Starten Sie das *DB2 Information Center* erneut.
   - Unter Windows: Klicken Sie **Start** > **Systemsteuerung** > **Verwaltung** > **Dienste** an. Klicken Sie mit der rechten Maustaste das **DB2 Information Center** an und wählen Sie **Start** aus.
   - Unter Linux: Geben Sie den folgenden Befehl ein:

     ```
     /etc/init.d/db2icdv10 start
     ```

Ergebnisse

Im aktualisierten *DB2 Information Center* werden die neuen und aktualisierten Themen angezeigt.

DB2-Lernprogramme

Die DB2-Lernprogramme unterstützen Sie dabei, sich mit den unterschiedlichen Aspekten der DB2-Produkte vertraut zu machen. Die Lerneinheiten bieten eine in einzelne Schritte unterteilte Anleitung.

Vorbereitungen

Die XHTML-Version des Lernprogramms kann über das Information Center unter http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/ angezeigt werden.

In einigen der Lerneinheiten werden Beispieldaten und Codebeispiele verwendet. Informationen zu bestimmten Voraussetzungen für die Ausführung der Tasks finden Sie in der Beschreibung des Lernprogramms.

DB2-Lernprogramme

Klicken Sie zum Anzeigen des Lernprogramms den Titel an.

„pureXML" in *pureXML - Handbuch*
Einrichten einer DB2-Datenbank, um XML-Daten zu speichern und Basisoperationen mit dem nativen XML-Datenspeicher auszuführen.

Informationen zur Fehlerbehebung in DB2

Es steht eine breite Palette verschiedener Informationen zur Fehlerbestimmung und Fehlerbehebung zur Verfügung, um Sie bei der Verwendung von DB2-Datenbankprodukten zu unterstützen.

DB2-Dokumentation
Informationen zur Fehlerbehebung stehen im Handbuch *Fehlerbehebung und Optimieren der Datenbankleistung* oder im Abschnitt mit grundlegenden Informationen zu Datenbanken im *DB2 Information Center* zur Verfügung, darunter:

- Informationen zum Eingrenzen und Aufdecken von Problemen mithilfe der Diagnosetools und -dienstprogramme von DB2.
- Lösungsvorschläge zu den am häufigsten auftretenden Problemen.
- Ratschläge zum Lösen anderer Probleme, die bei Verwendung der DB2-Datenbankprodukte auftreten können.

IBM Support Portal
Im IBM Support Portal finden Sie Informationen zu Problemen und den möglichen Ursachen und Fehlerbehebungsmaßnahmen. Die Website mit technischer Unterstützung enthält Links zu den neuesten DB2-Veröffentlichungen, technischen Hinweisen (TechNotes), APARs (Authorized Program Analysis Reports) und Fehlerkorrekturen, Fixpacks sowie weiteren Ressourcen. Sie können diese Wissensbasis nach möglichen Lösungen für aufgetretene Probleme durchsuchen.

Sie können auf das IBM Support Portal über die folgende Website zugreifen: http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows.

Bedingungen

Die Berechtigungen zur Nutzung dieser Veröffentlichungen werden Ihnen auf der Basis der folgenden Bedingungen gewährt.

Anwendbarkeit: Diese Bedingungen gelten zusätzlich zu den Nutzungsbedingungen für die IBM Website.

Persönliche Nutzung: Sie dürfen diese Veröffentlichungen für Ihre persönliche, nicht kommerzielle Nutzung unter der Voraussetzung vervielfältigen, dass alle Eigentumsvermerke erhalten bleiben. Sie dürfen diese Veröffentlichungen oder Teile dieser Veröffentlichungen ohne ausdrückliche Genehmigung von IBM nicht weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Kommerzielle Nutzung: Sie dürfen diese Veröffentlichungen nur innerhalb Ihres Unternehmens und unter der Voraussetzung, dass alle Eigentumsvermerke erhalten bleiben, vervielfältigen, weitergeben und anzeigen. Sie dürfen diese Veröffentlichungen oder Teile dieser Veröffentlichungen ohne ausdrückliche Genehmigung von IBM außerhalb Ihres Unternehmens nicht vervielfältigen, weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Rechte: Abgesehen von den hier gewährten Berechtigungen erhalten Sie keine weiteren Berechtigungen, Lizenzen oder Rechte (veröffentlicht oder stillschweigend) in Bezug auf die Veröffentlichungen oder darin enthaltene Informationen, Daten, Software oder geistiges Eigentum.

IBM behält sich das Recht vor, die in diesem Dokument gewährten Berechtigungen nach eigenem Ermessen zurückzuziehen, wenn sich die Nutzung der Veröffentlichungen für IBM als nachteilig erweist oder wenn die obigen Nutzungsbestimmungen nicht genau befolgt werden.

Sie dürfen diese Informationen nur in Übereinstimmung mit allen anwendbaren Gesetzen und Vorschriften, einschließlich aller US-amerikanischen Exportgesetze und Verordnungen, herunterladen und exportieren.

IBM übernimmt keine Gewährleistung für den Inhalt dieser Informationen. Diese Veröffentlichungen werden auf der Grundlage des gegenwärtigen Zustands (auf "as-is"-Basis) und ohne eine ausdrückliche oder stillschweigende Gewährleistung für die Handelsüblichkeit, die Verwendungsfähigkeit oder die Freiheit der Rechte Dritter zur Verfügung gestellt.

IBM Marken: IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der International Business Machines Corporation. Weitere Produkt- oder Servicenamen können Marken von oder anderen Herstellern sein. IBM oder anderen Herstellern sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite www.ibm.com/legal/copytrade.shtml.

Anhang B. Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Die Informationen über Produkte anderer Hersteller als IBM basieren auf den zum Zeitpunkt der ersten Veröffentlichung dieses Dokuments verfügbaren Informationen und können geändert werden.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Defense
France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Canada Limited
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
CANADA

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Dokument aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichen Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Diese Veröffentlichung kann Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes enthalten. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren; sie können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Musteranwendungsprogramme, die in Quellensprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Musterprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Musterprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten. Die Musterprogramme werden ohne Wartung (auf "as-is"-Basis) und ohne jegliche Gewährleistung zur Verfügung gestellt. IBM haftet nicht für Schäden, die durch Verwendung der Musterprogramme entstehen.

Kopien oder Teile der Musterprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (*Name Ihrer Firma*) (*Jahr*). Teile des vorliegenden Codes wurden aus Musterprogrammen der IBM Corp. abgeleitet. © Copyright IBM Corp. *_Jahr/Jahre angeben_*. Alle Rechte vorbehalten.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der IBM Corporation in den USA und/oder anderen Ländern. Weitere Produkt- oder Servicenamen können Marken von oder anderen Herstellern sein. IBM oder anderen Herstellern sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite „Copyright and trademark information" unter www.ibm.com/legal/copytrade.shtml.

Die folgenden Namen sind Marken oder eingetragene Marken anderer Unternehmen.

- Linux ist eine eingetragene Marke von Linus Torvalds in den USA und/oder anderen Ländern.
- Java und alle auf Java basierenden Marken und Logos sind Marken oder eingetragene Marken von Oracle und/oder ihren verbundenen Unternehmen.
- UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.
- Intel, das Intel-Logo, Intel Inside, Intel Inside logo, Celeron, Intel SpeedStep, Itanium und Pentium sind Marken oder eingetragene Marken der Intel Corporation oder deren Tochtergesellschaften in den USA und anderen Ländern.
- Microsoft, Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

Weitere Unternehmens-, Produkt- oder Servicenamen können Marken anderer Hersteller sein.

Index

A

Abfragen
  auszuführende räumliche Funktionen 97
  räumliche Indizes verwenden 98
  Schnittstellen zum Übergeben von räumlichen 97
Abstand
  Funktion ST_Distance 297
  ST_DistanceToPoint, Funktion 300
  ST_PointAtDistance, Funktion 396
Abstandsinformationen für Geometrien 261
Abstandstreue Projektionen 41
Aktivieren
  räumliche Operationen 31, 32
Aktualisierungen
  DB2 Information Center 469, 470
Anwendungen
  Beispielprogramm 103
Anwendungen schreiben
  Spatial Extender 101
ArcExplorer
  als Schnittstelle verwenden 97
Aufrufen von Prozeduren
  DB2 Spatial Extender 102
Automatische Geocodierung 66
Azimutale Projektionen 41

B

Bedingungen
  Veröffentlichungen 473
Befehle
  db2se 135
  db2trc 116
  Spatial Extender 135
Befehlszeilenprozessor (CLP)
  Nachrichten 114
  Spatial Extender-Befehle 135
Beispiele
  Spatial Extender 103
Bemaßungsinformationen abrufen 259
Bemerkungen 475

C

CREATE INDEX, Anweisung
  räumlicher Rasterindex 85

D

Datenbanken
  für räumliche Operationen aktivieren
    Übersicht 31
  Migration
    Spatial Extender 182
  räumliche Operationen aktivieren 32
Datenformate
  DB2 Spatial Extender 449
  Formdarstellung 456
  GML (Geography Markup Language) 456
Datenformate *(Forts.)*
  WKB-Darstellung (WKB = Well-Known Binary) 454
  WKT-Darstellung (WKT = Well-Known Text) 449
Datentypen
  alle Objekte
    Spatial Extender 59
  einteilige Objekte
    Spatial Extender 58
  mehrteilige Objekte
    Spatial Extender 58
Datentypinformationen abrufen 258
DB2 Information Center
  Aktualisierung 469, 470
  Versionen 468
DB2 Spatial Extender
  Befehle
    create_cs 140
    db2se alter_cs 136
    db2se alter_srs 138
    db2se create_srs 142
    db2se disable_autogc 146
    db2se disable_db 147
    db2se drop_srs 150
    db2se enable_autogc 151
    db2se enable_db 152
    db2se export_shape 153
    db2se import_shape 157
    db2se migrate 182
    db2se register_gc 164
    db2se register_spatial_column 167
    db2se remove_gc_setup 169
    db2se restore_indexes 170
    db2se run_gc 172
    db2se save_indexes 171
    db2se setup_gc 174
    db2se shape_info 177
    db2se unregister_gc 178
    db2se unregister_spatial_column 179
    db2se upgrade 180
    drop_cs 149
  Datenbanken für räumliche Operationen aktivieren 32
  Datenformate 449
  einrichten 13
  Funktionen 243
  Kopfdatendatei 101
  Linux und UNIX
    Installation 23
  Projekte erstellen 15
  Prozeduren aufrufen 102
  Trace für Probleme durchführen 116
  Tracefunktion
    Probleme bei DB2 Spatial Extender 116
  Überprüfung
    Installation 24
  Windows
    installieren 21
db2se-Befehle 135
  alter_cs 136
  alter_srs 138
  create_cs 140
  create_srs 142

db2se-Befehle *(Forts.)*
disable_autogc 146
disable_db 147
drop_cs 149
drop_srs 150
enable_autogc 151
enable_db 152
export_shape 153
import_shape 157
migrate 182
register_gc 164
register_spatial_column 167
remove_gc_setup 169
restore_indexes 170
run_gc 172
save_indexes 171
setup_gc 174
shape_info 177
unregister_gc 178
unregister_spatial_column 179
upgrade 180
db2trc, Befehl
Probleme bei DB2 Spatial Extender 116
DE_HDN _SRS_1004
räumliches Bezugssystem 46
DEFAULT _SRS
räumliches Bezugssystem 46
Dokumentation
gedruckt 466
Nutzungsbedingungen 473
PDF-Dateien 466
Übersicht 465

E

Eigenschaften von Geometrien
räumliche Funktionen
räumliches Bezugssystem 260
räumliche Funktionen für
Begrenzungsinformationen 260
Datentypinformationen 258
Dimensionsinformationen 260
Geometrien innerhalb einer Geometrie 259
Konfigurationsinformationen 260
Koordinaten- und Bemaßungsinformationen 259
Übersicht 9
Einheiten für Offsetwerte und Maßstabsfaktoren 49, 52
Einrichten
DB2 Spatial Extender 13
Geocoder
automatische Übersetzung 70
Geocodierungsoperationen 68
Einrichtung
Spatial Extender 19
Entscheidung zur Erstellung eines neuen räumlichen Bezugssystems 45
Ergebnistabellenfunktion
räumliche Spalten 265, 280, 281, 324, 325, 440
Erstellen
räumliche Spalten 59
Erstellen von Indizes
räumliche Rasterindizes 84
Erstellen von Projekten
DB2 Spatial Extender 15
Export von Daten
Formdatei 65

F

Faktoren, Umwandlung
Koordinaten 49, 52
Fehlerbehebung
Forminformationsnachrichten 114
Funktionen 113
Lernprogramme 473
Migrationsnachrichten 114
Onlineinformationen 473
Spatial Extender 109
gespeicherte Prozeduren 111
Nachrichten 109
Systemverwaltung, Protokoll mit Benachrichtigungen 118
Fehlerbestimmung
Lernprogramme 473
verfügbare Informationen 473
Flächentreue Projektionen 41
Formdarstellung, Datenformat 456
Formdatei
Daten exportieren 65
Formdateien
Export 153
Import 157
Informationen anzeigen 177
Formdaten
Import in Tabelle 64
Formeln für die Geocodierung 49, 52
Funktion ST_DistanceToPoint 300
Funktion ST_PointAtDistance 396
Funktionen
räumliche
Kategorien 247
Umwandlungen in Datenaustauschformate 247
räumliche Daten generieren 4
Funktionsnachrichten 113

G

GCS_NORTH _AMERICAN _1927
Koordinatensystem 46
GCS_NORTH _AMERICAN _1983
Koordinatensystem 46
GCS_WGS _1984
Koordinatensystem 46
GCSW _DEUTSCHE _HAUPTDRE IECKSNETZ
Koordinatensystem 46
Geocoder
einrichten
automatische Übersetzung 70
im Stapelmodus ausführen 71, 172
Katalogsicht ST_GEOCODER_ PARAMETERS 122
Katalogsicht ST_GEOCODERS 123
Katalogsicht ST_GEOCODING 124
Katalogsicht ST_GEOCODING_ PARAMETERS 125
Katalogsicht ST_SIZINGS 127
Registrierung 33, 164
Registrierung zurücknehmen 178
Übersicht 66
Geocoder verwenden
Spatial Extender 66
Geocodierte Spalten
automatische Geocodierung aktivieren 151
automatische Geocodierung inaktivieren 146
Geocodierung
Konfiguration 174
Konfiguration entfernen 169

Geocodierung *(Forts.)*
Übersicht 66
Geocodierungsoperationen
einrichten 68
Geografische Objekte
Beschreibung 1
Vergleichsfunktionen 253
Geografisches Koordinatensystem 35
Geometrieeigenschaften
Dimension 11
einfach 11
geschlossen 11
Innenbereich, Begrenzung und Außenbereich 10
Koordinaten 10
leer 11
M-Koordinaten 10
minimal einschließendes Rechteck 11
nicht einfach 11
nicht leer 11
Typen 9
X- und Y-Koordinaten 10
Z-Koordinaten 10
Geometrien
Client/Server-Datenübertragung 443
Eigenschaften
Übersicht 9
neu, aus vorhandenen Geometriebemaßungen 263
neue generieren
eine aus vielen 262
geänderte Formen 263
neue Bereichskonfigurationen 262
Übersicht 261
Umwandlung in andere 262
Räumliche Daten 5
Räumliche Funktionen 258
Übersicht 7
Geschäftsdaten
Quelle für räumliche Daten 4
Gespeicherte Prozeduren
DB2 Spatial Extender 185
Fehler 111
ST_ALTER_COORDSYS 186
ST_ALTER_SRS 188
ST_CREATE_COORDSYS 192
ST_CREATE_SRS 194
ST_DISABLE_AUTOGEOCODING 202
ST_DISABLE_DB 203
ST_DROP_COORDSYS 205
ST_DROP_SRS 206
ST_ENABLE_AUTOGEOCODING 208
ST_ENABLE_DB 210
ST_EXPORT_SHAPE 212
ST_IMPORT_SHAPE 215
ST_REGISTER_GEOCODER 224
ST_REGISTER_SPATIAL_COLUMN 228
ST_REMOVE_GEOCODING_ SETUP 230
ST_RUN_GEOCODING 232
ST_SETUP_GEOCODING 235
ST_UNREGISTER_GEOCODER 239
ST_UNREGISTER_SPATIAL_ COLUMN 240
GET GEOMETRY, Befehl
Syntax 93
GML (Geography Markup Language), Datenformat 456
Größe von Rasterzellen
Spatial Extender 80
gse_disable_autogc, gespeicherte Prozedur 202
gse_disable_db, gespeicherte Prozedur 203
gse_disable_sref, gespeicherte Prozedur 206
gse_enable_autogc, gespeicherte Prozedur 208
gse_enable_db, gespeicherte Prozedur 210
gse_enable_sref, gespeicherte Prozedur 194
gse_export_shape 212
gse_import_shape, gespeicherte Prozedur 215
gse_register_gc, gespeicherte Prozedur 224
gse_register_layer, gespeicherte Prozedur 228
gse_run_gc, gespeicherte Prozedur 232
gse_unregist_gc, gespeicherte Prozedur 239
gseidx, Befehl 93

H

Hardware
Anforderungen
Spatial Extender 20
Hilfe
SQL-Anweisungen 468

I

Import
Formdaten 64
räumliche Daten 5
Indexadvisor
Befehl GET GEOMETRY für Aufruf 93
Einsatzmöglichkeiten 79
Verwendungszweck 77, 87
Indexinformationen für Geometrien 261
Indizes
Indexadvisorbefehl 93
Räumliche Funktionen 258
räumliche Rasterindizes
Beschreibung 77
CREATE INDEX, Anweisung 85
Installation
DB2 Spatial Extender
Linux und UNIX 23
Systemanforderungen 20
Windows 21
Spatial Extender 19

K

Kartenprojektionen
Koordinatensysteme 457
Katalogsichten
Spatial Extender 119
SPATIAL_REF_SYS 132
ST_COORDINATE_SYSTEMS 119
ST_GEOCODER_ PARAMETERS 122
ST_GEOCODERS 123
ST_GEOCODING 124
ST_GEOCODING_ PARAMETERS 125
ST_GEOMETRY_ COLUMNS 120
ST_SIZINGS 127
ST_SPATIAL_ REFERENCE_SYSTEMS 128
ST_UNITS_OF_ MEASURE 131
Konstruktorfunktionen
Beispiele
Spatial Extender 252
ESRI-Formdarstellung 250
Geometriewerte aus Datenaustauschformaten 248
Geometriewerte aus Koordinaten 252

Konstruktorfunktionen *(Forts.)*
GML-Darstellung (GML = Geography Markup Language) 251
WKB-Darstellung (Well-Known Binary Representation; bekannte Binärdarstellung) 249
WKT-Darstellung (Well-Known Text Representation; bekannte Textdarstellung) 248
Koordinaten
abrufen 259
Räumliche Bezugssysteme 44
Umwandlung in räumlichen Bezugssystemen 44
Umwandlung zur Leistungsverbesserung 49, 52
Koordinatensysteme
ändern 136
erstellen 42, 140
Katalogsicht ST_COORDINATE_SYSTEMS 119
Katalogsicht ST_SPATIAL_ REFERENCE_SYSTEMS 128
löschen 149
Übersicht 35
unterstützte 457
vorhandene Koordinatensysteme verwenden 42
Koordinatensysteme verwenden
Spatial Extender 35
Kopfdatendatei
DB2 Spatial Extender 101

L

Leistung
Umwandlung von Koordinatendaten 49, 52
Lernprogramme
Fehlerbehebung 473
Fehlerbestimmung 473
Liste 472
pureXML 472
Lineare Einheiten
Koordinatensysteme 457
Linienfolgen 7

M

Maßstabsfaktoren
Übersicht 49, 52
Mehrlinienfolgen, homogene Gruppe von Spatial Extender 7
Mehrpunktangaben, homogene Gruppe von Spatial Extender 7
Minimal einschließendes Rechteck
Geometrieeigenschaften 11
Minimal einschließendes Rechteck (MBR)
Definition 9
in räumlichen Rasterindizes 77
MQTs
Räumliche Daten 73
Multiplikatoren zur Leistungsverbesserung
Koordinatenverarbeitung 49, 52
Multipolygon, homogene Gruppe von Spatial Extender 7

N

Nachrichten
Formdaten 114
Funktionen 113
Migrationsdaten 114
Spatial Extender
Befehlszeilenprozessor (CLP) 114
Bestandteile 109
Nachrichten *(Forts.)*
Spatial Extender *(Forts.)*
Gespeicherte Prozeduren 111
NAD27_ SRS_1002 (räumliches Bezugssystem) 46
NAD83_ SRS_1 (räumliches Bezugssystem) 46
Nullmeridiane
Koordinatensysteme 457

O

Offsetwerte
Übersicht 49, 52
Optimierung, räumliche Rasterindizes
mit Indexadvisor 87

P

Partitionierte Datenbanken 73, 74
Partitionierung 73
Polygone
Geometrietyp 7
Probleme ermitteln
Spatial Extender 109
Projizierte Koordinatensysteme 41
Projiziertes Koordinatensystem 35
Protokolle
diagnostisch 118
Punkte 7

R

Rasterebenen
Spatial Extender 79
Rastergrößen
räumlicher Rasterindex 87
Rasterindizes
optimieren 87
Übersicht 77
Räumliche Bezugssysteme
ändern 138
Bemaßungen 53
Beschreibung 44
Definition entfernen 150
erstellen 142, 194
in DB2 Spatial Extender bereitgestellt 46
Katalogsicht SPATIAL_REF_SYS 132
Koordinaten
Minimal- und Maximalkoordinaten 53
Maßstabsfaktoren
berechnen 52
neues räumliches Bezugssystem erstellen 45
Offsetwerte
berechnen 53
vorhandene räumliche Bezugssysteme auswählen 45
Räumliche Bezugssysteme konfigurieren
Spatial Extender 44
Räumliche Daten 73, 74
abrufen und analysieren
Funktionen 97
Schnittstellen 97
Verwenden von Indizes 98
analysieren und generieren 97
aus Geschäftsdaten abgeleitet 4
Beschreibung 1
Export 63
Funktionen 4

Räumliche Daten *(Forts.)*
Geocodierung 66
gespeicherte Prozeduren 185
Import 5, 63
Indizes verwenden 77
Kennung des räumlichen Bezugssystems 12
Projekte erstellen 15
replizierte MQTs 73
Sichten verwenden 77
ST_GEOMETRY_ COLUMNS 120
Übertragung vom Client an den Server 443
Ursprung 4
verwenden 5
Räumliche Daten analysieren 97
Räumliche Daten generieren 97
Räumliche Datenbanken
Spatial Extender aktivieren 152
Spatial Extender inaktivieren 147
Räumliche Funktionen
Abstandsinformationen 261
Beispiele 97
Datentypen 243
Datenumwandlung zwischen Koordinatensystemen 263
Eigenschaften von Geometrien 258
Begrenzungsinformationen 260
Datentypinformationen 258
Dimensionsinformationen 260
Geometrien innerhalb einer Geometrie 259
Konfigurationsinformationen 260
Koordinaten- und Bemaßungsinformationen 259
räumliches Bezugssystem 260
EnvelopesIntersect 263
Geometrien umwandeln 247
Indexinformationen 261
Indizes 258
Kategorien 247
Konstruktorfunktionen 247
MBR-Ergebnistabelle 265, 280, 324
neue Geometrien generieren
aus vorhandenen Geometriebemaßungen 263
eine aus vielen 262
geänderte Formen 263
Geometrien umwandeln 262
neue Bereichskonfigurationen 262
Übersicht 261
räumliche Indizes verwenden 98
ST_AppendPoint 267
ST_Area 268
ST_AsBinary 270
ST_AsGML 272
ST_AsShape 273
ST_AsText 274
ST_Boundary 275
ST_Buffer 277
ST_Centroid 283
ST_ChangePoint 284
ST_Contains 285
ST_ConvexHull 288
ST_CoordDim 290
ST_Crosses 291
ST_Difference 292
ST_Dimension 294
ST_Disjoint 295
ST_Distance 297
ST_DistanceToPoint 300
ST_Endpoint 301
ST_Envelope 301
Räumliche Funktionen *(Forts.)*
ST_EnvIntersects 303
ST_EqualCoordsys 304
ST_Equals 305
ST_EqualSRS 307
ST_ExteriorRing 308
ST_FindMeasure
ST_LocateAlong 309
ST_Generalize 310
ST_GeomCollection 312
ST_GeomCollFromTxt 314
ST_GeomCollFromWKB 316
ST_Geometry 317
ST_GeometryN 319
ST_GeometryType 320
ST_GeomFromText 321
ST_GeomFromWKB 322
ST_GetIndexParms 327
ST_InteriorRingN 329
ST_Intersection 330
ST_Intersects 332
ST_Is3d 335
ST_IsClosed 336
ST_IsEmpty 337
ST_IsMeasured 338
ST_IsRing 339
ST_IsSimple 340
ST_IsValid 342
ST_Length 343
ST_LineFromText 344
ST_LineFromWKB 346
ST_LineString 347
ST_LineStringN 348
ST_LocateAlong
ST_FindMeasure 309
ST_LocateBetween 359, 361
ST_M 350
ST_MaxM 351
ST_MaxX 352
ST_MaxY 354
ST_MaxZ 355
ST_MBR 357
ST_MBRIntersects 358
ST_MeasureBetween 359, 361
ST_MidPoint 362
ST_MinM 363
ST_MinX 365
ST_MinY 366
ST_MinZ 367
ST_MLineFromText 369
ST_MLineFromWKB 370
ST_MPointFromText 372
ST_MPointFromWKB 373
ST_MPolyFromText 375
ST_MPolyFromWKB 376
ST_MultiLineString 378
ST_MultiPoint 380
ST_MultiPolygon 381
ST_NumGeometries 383
ST_NumInteriorRing 384
ST_NumLineStrings 385
ST_NumPoints 386
ST_NumPolygons 387
ST_Overlaps 388
ST_Perimeter 390
ST_PerpPoints 391
ST_Point 394

Räumliche Funktionen *(Forts.)*
ST_PointAtDistance 396
ST_PointFromText 397
ST_PointFromWKB 399
ST_PointN 400
ST_PointOnSurface 401
ST_PolyFromText 402
ST_PolyFromWKB 403
ST_Polygon 405
ST_PolygonN 407
ST_Relate 408
ST_RemovePoint 409
ST_SRID 411, 412
ST_SrsID 411, 412
ST_SrsName 413
ST_StartPoint 414
ST_SymDifference 415
ST_ToGeomColl 418
ST_ToLineString 419
ST_ToMultiLine 420
ST_ToMultiPoint 421
ST_ToMultiPolygon 422
ST_ToPoint 423
ST_ToPolygon 424
ST_Touches 425
ST_Transform 427
ST_Union 429
ST_Within 431
ST_WKBToSQL 434
ST_WKTToSQL 435
ST_X 436
ST_Y 437
ST_Z 439
Überlegungen 243
Übersicht 243
Umwandlungen in Datenaustauschformate 247
ESRI-Formdarstellung 250
Geometriewerte aus Datenaustauschformaten 248
Geometriewerte aus Koordinaten 252
GML-Darstellung (GML = Geography Markup Language) 251
WKB-Darstellung (Well-Known Binary Representation; bekannte Binärdarstellung) 249
WKT-Darstellung (Well-Known Text Representation; bekannte Textdarstellung) 248
Union-Gesamtverknüpfungen 281, 325, 440
Vergleich von Geometrien
Containerbeziehungen 256
DE-9IM-Mustermatrixfolge 258
Geometriehüllen 256
identische Geometrien 257
Schnittpunkte 256, 257
Vergleichsfunktionen 253
Räumliche Funktionen verwenden
Abfragen 78
Räumliche Rasterindizes
erstellen 84
generieren 77
Rasterebenen und -größen 77, 79
verwenden 98
Räumliche Rasterindizes generieren 77
Räumliche Rasterindizes verwenden
Abfrage 78
Räumliche Ressourcen konfigurieren
räumliche Datenbanken 31
Spatial Extender-Projekte 35
Räumliche Spalten
Darstellungstools 57
Daten füllen 63
Einrichtung 57
erstellen 59
Geocodierung 66
Registrierung 60, 167
Registrierung zurücknehmen 179
Sichten 96
Räumliche Spalten füllen 63
Räumliche Umsetzungsgruppe
ST_GML 447
ST_Shape 446
ST_WellKnownBinary 444
Räumliche Umsetzungsgruppen
ST_WellKnownText 443
Räumlicher Bereich
Definition 44
Räumlicher Rasterindex
CREATE INDEX, Anweisung 85
Indexadvisorbefehl 93
Rastergrößen berechnen 87
Statistikdaten analysieren 88
Verwendung durch räumliche Funktionen 85
Verwendung durch SQL-Anweisungen 85
räumliches Bezugssystem
erstellen 50, 54
Registrierung
Geocoder 33
räumliche Spalten 60
Richtungstreue Projektionen 41
Ringe
Beschreibung 9

S

Schnittstellen
DB2 Spatial Extender 13
Sichten
räumliche Spalten 96
Softwarevoraussetzungen
Spatial Extender 20
Spatial Extender
Abfragen, die räumliche Funktionen verwenden 78
Abfragen, die räumliche Rasterindizes verwenden 78
aktualisieren
Server 27
Anwendungen schreiben 101
Befehle 135
gseidx 93
Befehlszeilenprozessor (CLP) 13
bereitgestellte räumliche Bezugssysteme 46
Datenformate 449
Datentypen
alle Objekte 59
einteilige Objekte 58
mehrteilige Objekte 58
Einheiten für Offsetwerte und Maßstabsfaktoren 50
einrichten 13
erste Schritte 19
Fehlerbehebung 109
Funktionen nach Eingabetyp 245
Geocoder verwenden 66
Katalogsichten 119
Konfiguration und Installation 19
Konstruktorfunktionen 247
Beispiele 252

Spatial Extender *(Forts.)*
Koordinatensysteme verwenden 35
Maßstabsfaktoren 49
Offsetwerte 49
Projekte erstellen 15
räumliche Bezugssysteme konfigurieren 44
räumliche Datentypen 57
räumliche Ressourcen konfigurieren
Datenbanken 31
Projekt 35
Schnittstellen 13
Syntax von Koordinatensystemen 457
unterstützte Kartenprojektionen 462
unterstützte lineare Einheiten 459
unterstützte Nullmeridiane 462
unterstützte Sphäroide 460
unterstützte Winkeleinheiten 459
Upgrade durchführen
32-Bit-Systeme auf 64-Bit-Systeme 28
Übersicht 27
SPATIAL_REF_SYS 132
Sphäroide
Koordinatensysteme 457
SQL-Anweisungen
Hilfe
anzeigen 468
ST_ALTER_COORDSYS, gespeicherte Prozedur 186
ST_ALTER_SRS 188
ST_COORDINATE_SYSTEMS 119
ST_CREATE_COORDSYS, gespeicherte Prozedur 192
ST_CREATE_SRS 194
ST_DISABLE_AUTOGEOCODING 202
ST_DISABLE_DB, gespeicherte Prozedur 203
ST_Distance 297
ST_DROP_COORDSYS, gespeicherte Prozedur 205
ST_DROP_SRS 206
ST_ENABLE_AUTOGEOCODING, gespeicherte Prozedur 208
ST_ENABLE_DB, gespeicherte Prozedur 210
ST_EXPORT_SHAPE, gespeicherte Prozedur 212
ST_GEOCODER_ PARAMETERS 122
ST_GEOCODERS 123
ST_GEOCODING 124
ST_GEOCODING_ PARAMETERS 125
ST_GEOMETRY_ COLUMNS 120
ST_Geometry-Werte
untergeordnete Typen 244
ST_IMPORT_SHAPE, gespeicherte Prozedur 215
ST_REGISTER_GEOCODER, gespeicherte Prozedur 224
ST_REGISTER_SPATIAL_COLUMN, gespeicherte Prozedur 228
ST_REMOVE_GEOCODING_SETUP, gespeicherte Prozedur 230
ST_RUN_GEOCODING, gespeicherte Prozedur 232
ST_SETUP_GEOCODING, gespeicherte Prozedur 235
ST_SIZINGS 127
ST_SPATIAL_ REFERENCE_SYSTEMS 128
ST_UNITS_OF_ MEASURE 131
ST_UNITS_OF_ MEASURE, Katalogsicht 131
ST_UNREGISTER_GEOCODER, gespeicherte Prozedur 239
ST_UNREGISTER_SPATIAL_COLUMN, gespeicherte Prozedur 240
Stapelbetrieb, Geocodierung 66
Stapelmodus
Geocoder ausführen 71
Statistikdaten für Indizes
räumlicher Rasterindex 88
Syntax von Koordinatensystemen
Spatial Extender 457
Systemverwaltung, Protokoll mit Benachrichtigungen
Details 118
Szenarios
Konfiguration von Spatial Extender 13

T

Tasks
Konfiguration von Spatial Extender 13

U

Überprüfung
Installation
DB2 Spatial Extender 24
Umsetzungsgruppen
Übersicht 443
Umwandlung
Koordinatenverarbeitung verbessern 49, 52
räumliche Daten zwischen Koordinatensystemen 263
Union-Gesamtverknüpfungsfunktionen 281, 325, 440
Unterstützte Kartenprojektionen
Spatial Extender 462
Unterstützte lineare Einheiten
Spatial Extender 459
Unterstützte Nullmeridiane
Spatial Extender 462
Unterstützte Sphäroide
Spatial Extender 460
Unterstützte Winkeleinheiten
Spatial Extender 459
Upgrades
Datenbanken, für Spatial Extender aktiviert 180
Spatial Extender
Übersicht 27
Vorgehensweise 27
Vorgehensweise (32-Bit- auf 64-Bit-Systeme) 28

V

Vergleichsfunktionen
Containerbeziehungen 256
DB2 Spatial Extender 253
DE-9IM-Mustermatrixfolge 258
Geometriehüllen 256
identische Geometrien 257
Schnittpunkte zwischen Geometrien 256, 257
Vorhandene räumliche Bezugssysteme auswählen 45

W

Well-Known Binary (WKB), Darstellung, Datenformat 454
Well-Known Text (WKT), Darstellung, Datenformat 449
WGS84_ SRS_1003
räumliches Bezugssystem 46
Winkeleinheiten
Koordinatensysteme 457
Winkeltreue Projektionen 41

Z

Zugriff auf räumliche Daten
Indizes 77

Zugriff auf räumliche Daten *(Forts.)*
Sichten 77

SC12-4688-00

Spine information:

IBM DB2 10.1 for Linux, UNIX and Windows

Spatial Extender - Benutzer- und Referenzhandbuch