DB2 Spatial Extender 및 Geodetic Data Management Feature

버전 **9**

Linux, UNIX 및 Windows

사용자 안내 및 참조서

SA30-3019-00

DB2 Spatial Extender 및 Geodetic Data Management Feature

버전 **9**

Linux, UNIX 및 Windows

사용자 안내 및 참조서

SA30-3019-00

주!

이 정보와 이 정보가 제공하는 제품을 사용하기 전에, 반드시 주의사항을 참조하십시오.

개정판 주의사항

이 문서는 IBM의 특허 정보에 대해 설명합니다. 해당 정보는 라이센스 계약에서 제공하며 저작권 법의 보호를 받습니다. 본 서적에 포함된 정보는 어떠한 제품 보증을 포함하지 않으며 본 매뉴얼에서 제공되는 어떠한 사항도 이와 같이 해석되지 않습니다.

IBM 책은 온라인 또는 해당 지역 IBM 담당자를 통해 주문할 수 있습니다.

- 온라인으로 책을 주문하려면 IBM Publications Center(www.ibm.com/shop/publications/order)를 방문하십시오.
- 해당 지역 IBM 담당자를 찾으려면 IBM Directory of Worldwide Contacts(www.ibm.com/planetwide)를 방문하십시오.

미국 또는 캐나다의 DB2 Marketing and Sales에서 DB2 책을 주문하려면 1-800-IBM-4YOU(426-4968)로 문의하십시오.

IBM에 정보를 보내면, IBM은 귀하의 권리를 침해하지 않는 범위 내에서 IBM이 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

© Copyright International Business Machines Corporation 1998, 2006. All rights reserved.

목차

제 1 장 DB2 Spatial Extender 제품 정보 1
DB2 Spatial Extender의 목적 1
데이터가 지리 대상물을 나타내는 방법. 2
공간 데이터 특성 3
측지 데이터 특성 4
공간 데이터 출처 4
지형, 공간 정보, 공간 데이터 및 Geometry를 함께 사용하는 방법 6

제 2 장 Geometry 정보 9
Geometry 9
Geometry 등록 정보 11
유형. 11
Geometry 좌표 11
X 및 Y 좌표 12
Z 좌표. 12
M 좌표 12
내부, 바운더리 및 외부 12
단순 또는 비단순 12
닫힘. 12
빈 또는 비어있지 않음 12
MBC(Minimum Bounding Circle) 13
차원. 13
공간 참조 시스템 ID 13

제 3 장 DB2 Spatial Extender 사용법 15
DB2 Spatial Extender 사용법 15
DB2 Spatial Extender 및 연관 기능에 대한 인터페이스 15
DB2 Spatial Extender 설정 및 프로젝트 작성을 위해 수행하는 태스크 16

제 4 장 DB2 Spatial Extender 시작하기. . . . 21
Spatial Extender 설정 및 설치. 21
Spatial Extender 설치에 대한 시스템 요구사항 . . 22
Windows용 DB2 Spatial Extender 설치 23
AIX용 DB2 Spatial Extender 설치 24
DB2 설정 마법사 25
db2_install 스크립트 26
SMIT 26
HP-UX용 DB2 Spatial Extender 설치 26
DB2 설치 마법사를 사용한 HP-UX용 Spatial Extender 설치 27
db2_install 스크립트를 사용한 HP-UX용 Spatial Extender 설치 27
swinstall 명령을 사용한 HP-UX용 Spatial Extender 설치 27
Solaris 운영 환경용 DB2 Spatial Extender 설치 28
DB2 설치 마법사를 사용하여 Solaris 운영 환경용 DB2 Spatial Extender를 설치하십시오. . . . 29
db2_install 스크립트를 사용한 DB2 Spatial Extender 설치 30
pkgadd 명령을 사용하여 Solaris용 DB2 Spatial Extender를 설치하십시오.. 30
Linux용 DB2 Spatial Extender 설치 30
DB2 설치 마법사를 사용하여 DB2 Spatial Extender 설치 31
db2_install 스크립트를 사용한 DB2 Spatial Extender 설치 32
DB2 Spatial Extender 인스턴스 환경 작성. . . . 32
Spatial Extender 설치 검증. 34
사후 설치 고려사항. 35
DB2용 ArcExplorer 다운로드 35

제 5 장 DB2 버전 9 데이터베이스 시스템으로 Spatial Extender 환경 이주 37
공간 사용 데이터베이스 이주 37
db2se migrate 명령 37
32비트에서 64비트 환경으로 이주. 39

제 6 장 데이터베이스 설정 41
공간 데이터를 수용할 수 있도록 데이터베이스 구성 41
트랜잭션 로그 특성 조정 41
응용프로그램 힙 크기 조정 42
응용프로그램 제어 힙 크기 조정 43

제 7 장 데이터베이스용 공간 자원 설정. 45
데이터베이스에서 자원 설정 방법 45
데이터베이스에 제공되는 자원 재고 45
공간 작업에 데이터베이스 사용. 46
참조 데이터에 대한 작업 방법 46
참조 데이터 46
참조 데이터에 대한 액세스 설정 47
지오코더(Geocoder) 등록. 48

제 8 장 프로젝트용 공간 자원 설정 49

© Copyright IBM Corp. 1998, 2006

좌표 시스템 사용법 49
좌표 시스템 49
지리 좌표 시스템 49
투영 좌표 시스템 55
좌표 시스템 선택 또는 작성 56
공간 참조 시스템 설정 방법 57
공간 참조 시스템 57
디폴트 공간 참조 시스템을 사용할 것인지 또는 새 시스템을 작성할 것인지 여부 판별 59
DB2 Spatial Extender에서 제공하는 공간 참조 시스템 60
좌표 데이터를 정수로 변환하는 변환 인수 . . . 62
공간 참조 시스템 작성 63
스케일 인수 계산 65
좌표 데이터를 정수로 변환하는 변환 인수 . . . 66
최대 및 최소 좌표와 도량 단위 판별 66
오프셋 값 계산 67
공간 참조 시스템 작성 67

제 9 장 공간 컬럼 설정 71
공간 컬럼 71
볼 수 있는 컨텐츠를 포함한 공간 컬럼 71
공간 데이터 유형 71
공간 컬럼 작성 73
공간 컬럼 등록 75

제 10 장 공간 컬럼 데이터 상주 77
공간 데이터 임포트 및 익스포트 정보 77
공간 데이터 임포트 78
새 테이블 또는 기존 테이블로 모양 데이터 임포트 78
새 테이블 또는 기존 테이블로 SDE 전송 데이터 임포트 79
공간 데이터 익스포트 80
shapefile로 데이터 익스포트 80
SDE 전송 파일로 데이터 익스포트 81
지오코더(Geocoder) 사용법 82
지오코더(Geocoder) 및 지오코딩(Geocoding) . . 82
지오코딩(Geocoding) 조작 설정 84
자동으로 실행되도록 지오코더(Geocoder) 설정 87
일괄처리 모드로 지오코더(Geocoder) 실행 . . . 88

제 11 장 인덱스 및 뷰를 사용하여 공간 데이터에 액세스 91
공간 인덱스 유형 91
공간 격자 인덱스 92
공간 격자 인덱스 생성 92
쿼리에서 공간 함수 사용 93
쿼리의 공간 격자 인덱스 사용 방법 93
인덱스 레벨 수 및 격자 크기에 대한 고려사항 . . . 94
격자 레벨 수 94
격자 셀 크기 95
공간 격자 인덱스 작성 98
SQL CREATE INDEX를 사용하여 공간 격자 인덱스 작성 100
공간 격자 인덱스에 대한 CREATE INDEX문 . . 100
Index Advisor로 공간 격자 인덱스 조정 101
Index Advisor로 공간 격자 인덱스 조정 개요 101
공간 격자 인덱스의 격자 크기 판별 102
공간 격자 인덱스 통계 분석 103
gseidx 명령 108
뷰를 사용하여 공간 컬럼 액세스 111

제 12 장 공간 정보 분석 및 생성 113
공간 분석 수행 환경 113
공간 함수 작동 방법 예 113
인덱스를 사용하여 쿼리를 최적화하는 함수 . . . 115

제 13 장 DB2 Spatial Extender 명령 117
DB2 Spatial Extender 설정 및 프로젝트 개발 명령 호출 117

제 14 장 샘플 프로그램 사용 및 응용프로그램 작성 125
DB2 Spatial Extender용 응용프로그램 작성 . . . 125
공간 응용프로그램에 DB2 Spatial Extender 헤더 파일 포함 125
응용프로그램에서 DB2 Spatial Extender 스토어드 프로시저 호출 126
DB2 Spatial Extender 샘플 프로그램 127

제 15 장 DB2 Spatial Extender 문제점 식별 135
DB2 Spatial Extender 메시지 해석 방법 135
DB2 Spatial Extender 스토어드 프로시저 출력 매개변수 137
DB2 Spatial Extender 함수 메시지 139
DB2 Spatial Extender CLP 메시지 141
DB2 제어 센터 메시지 143
db2trc 명령을 사용하여 DB2 Spatial Extender 문제점 추적 144
관리 통지 파일 145

제 16 장 DB2 Geodetic Data Management Feature 147
DB2 Geodetic Data Management Feature . . . 147

DB2 Geodetic Data Management Feature 및 DB2 Spatial Extender 사용 시기 147
측지 기준면 148
측지 위도 및 경도. 149
측지 거리 150
측지 영역 151

제 17 장 DB2 Geodetic Data Management Feature 설정 155
DB2 Geodetic Data Management Feature 설정 및 사용 155
Informix Geodetic DataBlade에서 DB2 Geodetic Data Management Feature로 이주 156
측지 데이터로 공간 컬럼 채우기 163

제 18 장 측지 인덱스 165
측지 Voronoi 인덱스. 165
Voronoi 셀 구조 166
대체 Voronoi 셀 구조 선택 시 고려사항 167
측지 Voronoi 인덱스 작성 168
측지 Voronoi 인덱스에 대한 CREATE INDEX문 169
DB2 Geodetic Data Management Feature에 제공된 Voronoi 셀 구조 171
세계, 인구 밀도 기반(Voronoi ID: 1) 173
미국(Voronoi ID: 2). 174
캐나다(Voronoi ID: 3) 175
인도(Voronoi ID: 4). 176
일본(Voronoi ID: 5). 177
아프리카(Voronoi ID: 6) 178
호주(Voronoi ID: 7). 179
유럽(Voronoi ID: 8). 180
북아메리카(Voronoi ID: 9) 181
남아프리카(Voronoi ID: 10) 182
지중해 지역(Voronoi ID: 11). 183
세계, 단일 데이터 분산, 중간 해상도 – dodeca04 (Voronoi ID: 12) 184
세계, 산업 국가 - G7 국가(Voronoi ID: 13) 185
세계, 단일 데이터 분산, 저해상도 - 아이소타입 (Voronoi ID: 14). 186

제 19 장 측지 데이터 사용과 공간 데이터 사용의 차이점. 187
최소 및 최대 x 및 y 속성 187
평면형 지구 표현과 구형 지구 표현 방법의 차이점 188
180도 경선을 지나는 선 세그먼트 188
180도 경선을 지나는 다각형 189
극을 감싸는 다각형 192
반구, 적도 지역 및 지구 전체를 표현하는 다각형 193
DB2 Geodetic Data Management Feature에서 지원하는 공간 함수 197
DB2 Geodetic Data Management Feature 스토어드 프로시저 및 카탈로그 뷰 201
DB2 Geodetic Data Management Feature에서 지원하는 기준면 202
측지 회전 타원체 209

제 20 장 스토어드 프로시저 211
GSE_export_sde 212
GSE_import_sde 214
ST_alter_coordsys 216
ST_alter_srs 218
ST_create_coordsys 222
ST_create_srs 225
ST_disable_autogeocoding. 232
ST_disable_db. 234
ST_drop_coordsys 235
ST_drop_srs 236
ST_enable_autogeocoding 238
ST_enable_db 240
ST_export_shape 242
ST_import_shape 246
ST_register_geocoder 254
ST_register_spatial_column 259
ST_remove_geocoding_setup 261
ST_run_geocoding 263
ST_setup_geocoding. 267
ST_unregister_geocoder 270
ST_unregister_spatial_ column 272

제 21 장 카탈로그 뷰 275
DB2GSE.ST_GEOMETRY_COLUMNS 카탈로그 뷰 275
DB2GSE.SPATIAL_REF_SYS 카탈로그 뷰. . . 276
DB2GSE.ST_COORDINATE_SYSTEMS 카탈로그 뷰 277
DB2GSE.ST_GEOMETRY_COLUMNS 카탈로그 뷰 278
DB2GSE.ST_GEOCODER_PARAMETERS 카탈로그 뷰 279
DB2GSE.ST_GEOCODERS 카탈로그 뷰. . . . 280
DB2GSE.ST_GEOCODING 카탈로그 뷰. . . . 281
DB2GSE.ST_GEOCODING_PARAMETERS 카탈로그 뷰 282

DB2GSE.ST_SIZINGS 카탈로그 뷰 283
DB2GSE.
ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰 284
DB2GSE.ST_UNITS_OF_MEASURE 카탈로그 뷰 286

제 22 장 공간 함수: 범주 및 사용 289
공간 함수: 범주 및 사용 289
Geometry 값을 데이터 교환 형식으로 변환하는 공간 함수 289
컨스트럭터 함수 289
데이터 교환 형식에 대해 수행되는 함수 . . . 290
좌표에서 Geometry를 작성하는 함수 291
예 292
WKT(Well-Known Text) 표현으로 변환 294
WKB(Well-Known Binary) 표현으로 변환 . . . 295
ESRI 모양 표현으로 변환 296
GML(Geography Markup Language) 표현으로 변환 297
지리 대상물을 비교하는 함수 298
비교 함수 299
공간 비교 함수. 300
한 Geometry가 다른 Geometry를 포함하는지 여부를 점검하는 함수 301
ST_Contains 301
ST_Within 302
Geometry 사이의 교차를 점검하는 함수 304
ST_Intersects 305
ST_Crosses 305
ST_Overlaps 307
ST_Touches 308
Geometry의 인벨로프를 비교하는 함수. 309
ST_EnvIntersects 310
ST_MBRIntersects 310
두 개가 동일한지 점검하는 함수 310
ST_EqualCoordsys 310
ST_Equals 310
ST_EqualSRS 311
두 Geometry에 교차가 없는지 점검하는 함수 . . 312
Geometry와 DE-9IM 패턴 행렬 문자열을 비교하는 함수 313
Geometry 등록 정보에 대한 정보를 리턴하는 함수 313
데이터 유형 정보를 리턴하는 함수 313
좌표 및 측정 정보를 리턴하는 함수 313
ST_CoordDim 314
ST_IsMeasured 314
ST_IsValid 314
ST_Is3D 314
ST_M 315
ST_MaxM 315
ST_MaxX 315
ST_MaxY 315
ST_MaxZ 315
ST_MinM 315
ST_MinX 315
ST_MinY 315
ST_MinZ 315
ST_X 315
ST_Y 315
ST_Z 315
Geometry 내의 Geometry에 대한 정보를 리턴하는 함수 316
ST_Centroid 316
ST_EndPoint 316
ST_GeometryN 316
ST_LineStringN 316
ST_MidPoint 316
ST_NumGeometries 317
ST_NumLineStrings 317
ST_NumPoints 317
ST_NumPolygons 317
ST_PointN 317
ST_PolygonN 317
ST_StartPoint 317
바운더리, 인벨로프 및 링에 대한 정보를 표시하는 함수 317
ST_Envelope 318
ST_EnvIntersects 318
ST_ExteriorRing 318
ST_InteriorRingN. 318
ST_MBR 318
ST_MBRIntersects 318
ST_NumInteriorRing 318
ST_Perimeter 318
Geometry 차원에 대한 정보를 리턴하는 함수 . . 319
ST_Area 319
ST_Dimension 319
ST_Length 319
Geometry가 닫혔는지, 비었는지 또는 단순한지 여부를 표시하는 함수 319
ST_IsClosed 319
ST_IsEmpty 319

ST_IsSimple 320
Geometry의 공간 참조 시스템을 식별하는 함수 320
ST_SrsId(ST_SRID라고도 함) 320
ST_SrsName 320
기존 Geometry에서 새 Geometry를 생성하는 함수 320
Geometry에서 다른 Geometry로 변환하는 함수 321
ST_Polygon 321
ST_ToGeomColl 321
ST_ToLineString 321
ST_ToMultiLine 321
ST_ToMultiPoint 321
ST_ToMultiPolygon 321
ST_ToPoint. 321
ST_ToPolygon. 321
다른 스페이스 구성으로 새 Geometry를 작성하는 함수 322
ST_Buffer 322
ST_ConvexHull 323
ST_Difference 323
ST_Intersection 324
ST_SymDifference 325
다수의 Geometry에서 한 Geometry를 파생시키는 함수 325
MBR 집계 326
ST_Union 326
총계 326
도량 단위에 기초한 새 Geometry를 파생시키는 함수 326
ST_FindMeasure(ST_LocateAlong이라고도 함) 327
ST_MeasureBetween(ST_LocateBetween이라고도 함). 327
기존 Geometry의 수정된 양식을 작성하는 함수 327
ST_AppendPoint 328
ST_ChangePoint 328
ST_Generalize. 328
ST_M 328
ST_PerpPoints. 328
ST_RemovePoint 328
ST_X. 329
ST_Y. 329
ST_Z. 329
거리 정보를 리턴하는 함수. 329
인덱스 정보를 리턴하는 함수 330
좌표 시스템 사이의 변환 330

제 23 장 공간 함수: 구문 및 매개변수 331
공간 함수: 고려사항 및 연관된 데이터 유형 . . . 331
고려사항 332
ST_Geometry 값을 부속 유형 값으로 처리 . . 332
입력 유형에 따라 나열된 공간 함수. 333
EnvelopesIntersect 336
MBR 집계 339
ST_AppendPoint 340
ST_Area. 342
ST_AsBinary 345
ST_AsGML 347
ST_AsShape 348
ST_AsText 349
ST_Boundary 350
ST_Buffer 352
ST_Centroid 355
ST_ChangePoint 356
ST_Contains 358
ST_ConvexHull 360
ST_CoordDim. 362
ST_Crosses. 363
ST_Difference. 364
ST_Dimension. 366
ST_Disjoint. 368
ST_Distance 370
ST_Edge_GC_USA 373
ST_Endpoint 377
ST_Envelope 378
ST_EnvIntersects. 379
ST_EqualCoordsys 380
ST_Equals 381
ST_EqualSRS 383
ST_ExteriorRing 384
ST_FindMeasure 또는 ST_LocateAlong 386
ST_Generalize. 387
ST_GeomCollection 389
ST_GeomCollFromTxt 391
ST_GeomCollFromWKB 393
ST_Geometry 395
ST_GeometryN 396
ST_GeometryType 398
ST_GeomFromText 399
ST_GeomFromWKB 400
ST_GetIndexParms 402
ST_InteriorRingN. 404
ST_Intersection 405
ST_Intersects 407
ST_Is3d 409

ST_IsClosed 410
ST_IsEmpty 412
ST_IsMeasured 413
ST_IsRing 414
ST_IsSimple 416
ST_IsValid 417
ST_Length 418
ST_LineFromText 420
ST_LineFromWKB 421
ST_LineString 422
ST_LineStringN 424
ST_M 425
ST_MaxM 427
ST_MaxX 428
ST_MaxY 430
ST_MaxZ 432
ST_MBR 433
ST_MBRIntersects 435
ST_MeasureBetween, ST_LocateBetween . . . 437
ST_MidPoint 438
ST_MinM 439
ST_MinX 441
ST_MinY 442
ST_MinZ 444
ST_MLineFromText 446
ST_MLineFromWKB 447
ST_MPointFromText 449
ST_MPointFromWKB 450
ST_MPolyFromText 451
ST_MPolyFromWKB 453
ST_MultiLineString 455
ST_MultiPoint 457
ST_MultiPolygon 458
ST_NumGeometries 460
ST_NumInteriorRing 461
ST_NumLineStrings 462
ST_NumPoints 463
ST_NumPolygons 464
ST_Overlaps 465
ST_Perimeter 467
ST_PerpPoints 469
ST_Point 472
ST_PointFromText 474
ST_PointFromWKB 475
ST_PointN 477
ST_PointOnSurface 478
ST_PolyFromText 479
ST_PolyFromWKB 480
ST_Polygon 481
ST_PolygonN 484
ST_Relate 485
ST_RemovePoint 486
ST_SrsId, ST_SRID 488
ST_SrsName 489
ST_StartPoint 490
ST_SymDifference 491
ST_ToGeomColl 494
ST_ToLineString 495
ST_ToMultiLine 496
ST_ToMultiPoint 497
ST_ToMultiPolygon 498
ST_ToPoint 500
ST_ToPolygon 501
ST_Touches 502
ST_Transform 503
ST_Union 506
ST_Within 508
ST_WKBToSQL 510
ST_WKTToSQL 511
ST_X 512
ST_Y 513
ST_Z 515
총계 516

제 24 장 그룹 변환 519
그룹 변환 519
ST_WellKnownText 변환 그룹 519
ST_WellKnownBinary 변환 그룹 521
ST_Shape 변환 그룹 522
ST_GML 변환 그룹 523

제 25 장 데이터 형식 지원 527
Well-Known Text 표현 527
Well-Known Binary 표현 532
모양 표현 534
GML(Geography Markup Language) 표현 . . . 534

제 26 장 지원되는 좌표 시스템 535
좌표 시스템 구문 535
  지원되는 선형 단위 537
지원되는 각도 단위 537
표준 회전 타원체 538
지원되는 측지학 기준면 540

지원되는 자오선 542
지원되는 맵 투영 543

제 27 장 DB2 제어 센터의 공간 태스크 545
좌표 시스템 변경 545
좌표 시스템 작성 545
공간 컬럼 작성. 545
공간 인덱스 작성 546
지오코딩(Geocoding) 실행 546
지오코딩(Geocoding) 설정 547
공간 참조 시스템 변경 547
공간 데이터 임포트 547

DB2 기술 라이브러리(PDF 형식) 549

인쇄된 DB2 책 주문. 551

DB2 문제점 해결 정보 553

주의사항 555

IBM에 문의 559

색인 561

제 1 장 DB2 Spatial Extender 제품 정보

이 섹션에서는 DB2 Spatial Extender를 소개하면서 목적 및 지원되는 데이터를 설명하고 해당 기본 개념을 함께 사용하는 방법에 대해 설명합니다.

DB2 Spatial Extender의 목적

DB2® Spatial Extender를 사용하여 지리 대상물에 대한 공간 정보를 생성 및 분석하고 이 정보의 기초가 되는 데이터를 저장 및 관리하십시오. 지리 대상물(여기서는 간단히 대상물이라고도 함)은 현실 세계에서 식별 가능한 위치를 가지는 모든 것 또는 식별 가능한 위치에 존재하는 것으로 상상할 수 있는 모든 것입니다. 대상물은 다음이 될 수 있습니다.

- 오브젝트(즉, 모든 종류의 구체적 엔티티). 예를 들면, 강, 숲 또는 산맥.
- 스페이스. 예를 들면, 위험 지역 주변의 안전 지대 또는 특정 비즈니스가 서비스를 제공하는 마케팅 영역.
- 정의 가능한 위치에서 발생하는 이벤트. 예를 들면, 특정 교차로에서 발생한 자동차 사고 또는 특정 상점에서의 상거래.

대상물은 여러 환경에 존재할 수 있습니다. 예를 들어, 앞의 목록에서 언급한 오브젝트(예: 강, 숲 및 산맥)는 자연 환경에 속합니다. 도시, 빌딩 및 사무실과 같은 다른 오브젝트는 문화 환경에 속합니다. 그리고 공원, 동물원 및 농장과 같은 오브젝트는 자연과 문화 환경의 조합을 나타냅니다.

이 설명에서 공간 정보는 DB2 Spatial Extender가 사용자에게 제공하는 정보 종류, 즉 지리 대상물의 위치에 대한 사실 및 수치를 의미합니다. 공간 정보의 예는 다음과 같습니다.

- 지도상의 지리 대상물 위치(예: 도시의 위치를 정의하는 경도 및 위도 값)
- 주변 환경과 관련한 지리 대상물의 위치(예: 한 도시에서 병원 및 진료소가 위치한 지점 또는 지역적 지진대에 대한 도시 거주지의 근접도)
- 지리 대상물이 서로 연관된 방식(예: 특정 강 체계가 특정 영역 내에 있거나 강 체계의 지류를 통과하는 영역의 특정 다리와 같은 정보)
- 하나 이상의 지리 대상물에 적용하는 수치(예: 사무실 빌딩과 해당 구획선 사이의 거리 또는 조류 보존 지역의 둘레)

공간 정보는 그 자체만으로 또는 전형적인 관계 데이터와 조합되어 서비스를 제공할 영역을 정의하고 가능한 시장 위치를 결정하는 데 도움을 줄 수 있습니다. 예를 들어, 도

© Copyright IBM Corp. 1998, 2006

복지 정책 관리자는 복지 정책의 신청자와 수혜자가 해당 영역에 실제로 살고 있는지 검증해야 합니다. DB2 Spatial Extender는 해당 영역의 위치와 신청자 및 수혜자의 주소에서 이러한 정보를 얻을 수 있습니다.

또는 음식점 체인의 소유자가 가까운 도시에서 영업을 원한다고 가정하십시오. 새로운 음식점을 열 장소를 결정하기 위해 소유자는 다음과 같은 질문에 대한 응답이 필요합니다. 즉, 외형적으로 자신의 음식점에 자주 들르게 될 단골들이 이 도시의 어느 지역에 집중되어 있는가? 주요 도로가 어디에 있는가? 범죄 발생률이 가장 낮은 곳이 어디인가? 경쟁 음식점이 어디에 위치하고 있는가? DB2 Spatial Extender 및 DB2는 이러한 질문에 응답하는 정보를 생성할 수 있습니다. 더우기 프론트엔드 도구가(필수 도구는 아님) 도움이 될 수 있습니다. 시각화 도구는 DB2 Spatial Extender에서 생성한 정보(예: 단골이 집중된 위치와 제시한 음식점에 대한 주요 도로의 근접도)를 지도에 그래픽 양식으로 표시할 수 있습니다. 비즈니스 인텔리전스 도구는 연관된 정보(예: 경쟁 음식점의 이름과 설명)를 보고서 양식으로 표시할 수 있습니다.

데이터가 지리 대상물을 나타내는 방법

DB2® Spatial Extender에서 지리 대상물은 하나 이상의 데이터 항목으로 표시될 수 있습니다(예: 테이블의 한 행에 있는 데이터 항목). (데이터 항목은 관계형 테이블의 셀에 있는 값입니다.) 예를 들어, 사무실 빌딩 및 거주 지역을 고려해 보십시오. 그림 1에서 BRANCHES 테이블의 각 행은 은행의 지점 사무실을 나타냅니다. 마찬가지로, 그림 1에 있는 CUSTOMERS 테이블의 각 행은 전체적으로 은행의 고객을 나타냅니다. 그러나 각 행의 서브세트(특히, 고객 주소를 구성하는 데이터 항목)는 고객의 소재를 표시합니다.

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

그림 1. 지리 대상물을 나타내는 데이터. BRANCHES 테이블의 데이터 행은 은행의 지점 사무실을 나타냅니다. CUSTOMERS 테이블의 주소 데이터는 고객의 거주지를 나타냅니다. 두 테이블의 이름 및 주소는 모두 가상입니다.

그림 1의 테이블에는 은행의 지점 및 고객을 나타내고 서술하는 데이터가 들어 있습니다. 여기서는 이러한 데이터를 비즈니스 데이터라고 합니다.

비즈니스 데이터의 서브세트(지점 및 고객의 주소를 나타내는 값)는 공간 정보를 생성하는 값으로 변환될 수 있습니다. 예를 들어, 그림 1에 표시된 것처럼 한 지방 사무소의

주소는 92467 Airzone Blvd., San Jose, CA 95141, USA입니다. 고객 주소는 9 Concourt Circle, San Jose, CA 95141, USA입니다. DB2 Spatial Extender는 지점 주소와 고객 주소를 상대방 주소에 대한 상대적 위치를 표시하는 값으로 변환할 수 있습니다. 그림 2에서는 해당 값을 포함하도록 지정된 새 컬럼이 있는 BRANCHES 및 CUSTOMERS 테이블을 표시합니다.

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

그림 2. 공간 컬럼이 추가된 테이블. 각 테이블의 LOCATION 컬럼에는 주소에 해당되는 좌표가 들어 있습니다.

공간 정보는 LOCATION 컬럼에 저장된 데이터 항목에서 도출되므로 이 설명에서는 데이터 항목을 공간 데이터라고 합니다.

공간 데이터 특성

공간 데이터는 위치를 식별하는 좌표로 구성됩니다. Spatial Extender는 x와 y 또는 위도와 경도 값으로 지정된 2차원 좌표로 작업합니다.

좌표는 다음을 표시하는 숫자입니다.

- 길이 단위를 고려한, 원점에 상대적인 축상의 위치.
- 각도 수치 단위를 고려한, 기준선 또는 기준 평면에 상대적인 방향

예를 들어, 위도는 적도 평면에 상대적인 각도(일반적으로 도 단위)를 표시하는 좌표입니다. 경도는 그리니치 자오선에 대해 상대적인 각도(이 또한 일반적으로 도 단위 사용)를 표시하는 좌표입니다. 따라서 지도에서 Yellowstone 국립공원의 위치는 적도 기준 북위 44.45도와 그리니치 경선 기준 서경 110.40도로 정의됩니다. 보다 정확히 표현하면 이 좌표는 미국 Yellowstone 국립공원의 중심을 나타냅니다.

경도 및 위도의 정의, 점, 선 및 참조 평면, 측량 단위 및 기타 연관된 매개변수를 통틀어 좌표 시스템이라고 합니다. 위도 및 경도 이외의 값을 기초로 하는 좌표 시스템도 있습니다. 이러한 좌표 시스템에는 고유한 참조점, 참조선, 참조 평면, 도량 단위 및 추가 연관 매개변수(예: 투영 변환)가 있습니다.

가장 간단한 공간 데이터 항목은 단일 지리적 위치를 정의하는 하나의 좌표 쌍으로 구성됩니다. 보다 광범위한 공간 데이터 항목은 도로 또는 강과 같은 선형 경로를 정의하는 여러 좌표로 구성됩니다. 세 번째 종류는 영역의 경계를 정의하는 좌표로 구성됩니다(예: 토지 구획 또는 범람 지역의 경계).

각 공간 데이터 항목은 공간 데이터 유형의 인스턴스입니다. 단일 위치를 표시하는 좌표의 데이터 유형은 ST_Point이고 선형 경로를 정의하는 좌표의 데이터 유형은 ST_LineString이며 영역의 바운더리를 정의하는 좌표의 데이터 유형은 ST_Polygon입니다. 이 유형은 다른 공간 데이터 유형과 함께 단일 계층 구조에 속하는 구조화된 유형입니다.

측지 데이터 특성

측지 데이터는 곡면, 연속면 및 닫힌 면을 설명하는 좌표 시스템에서 경도 및 위도 좌표로 표시되는 공간 데이터입니다.

DB2 Geodetic Data Management Feature에서는 Spatial Extender와 동일한 데이터 유형 및 함수를 사용하여 DB2 데이터베이스에 지리 데이터를 저장합니다. 지구를 평면 지도로 처리하는 Spatial Extender와는 달리, Geodetic Data Management Feature에서는 극 또는 날짜 변경선에 끝 또는 틈이 없는 구로 지구를 처리합니다. 평면 지도의 경우 구형 좌표를 평면형 좌표로 변환하려면 투영 좌표가 필요합니다. 반면 Geodetic Data Management Feature에서는 지구 표면의 타원체 모델에서 경도 및 위도를 사용합니다. 선 교차, 영역 겹침, 거리 및 영역과 같은 계산은 위치에 관계 없이 정확합니다.

공간 데이터 출처

사용자는 다음을 통해 공간 데이터를 얻을 수 있습니다.

- 비즈니스 데이터로부터 공간 데이터 도출
- 공간 함수로부터 공간 데이터 생성
- 외부 소스로부터 공간 데이터 임포트

비즈니스 데이터를 소스 데이터로 사용

DB2 Spatial Extender는 주소와 같은 비즈니스 데이터로부터(2 페이지의 『데이터가 지리 대상물을 나타내는 방법』에 언급된 바와 같이) 공간 데이터를 도출할 수 있습니다. 이를 지오코딩이라고 합니다. 해당 시퀀스를 알려면 3 페이지의 그림 2를 『이전』 그림으로 5 페이지의 그림 3을 『이후』 그림으로 간주하십시오. 3 페이지의 그림 2에서는 BRANCHES 테이블과 CUSTOMERS 테이블 모두에 공간 데이터에 지정된 컬럼이 있음을 보여줍니다. DB2 Spatial Extender는 이 테이블에 있는 주소를 지오코딩하여 주소에 해당하는 좌표를 확보한 후 좌표를 컬럼에 넣는다고 가정하십시오. 5 페이지의 그림 3에서는 이 결과에 대해 설명합니다.

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

그림 3. 소스 데이터에서 파생된 공간 데이터를 포함하는 테이블. CUSTOMERS 테이블의 LOCATION 컬럼에는 ADDRESS, CITY, POSTAL CODE, STATE_PROV 및 COUNTRY 컬럼의 주소에서 도출된 좌표가 들어 있습니다. 또한 BRANCHES 테이블의 LOCATION 컬럼에는 이 테이블의 ADDRESS, CITY, POSTAL CODE, STATE_PROV 및 COUNTRY 컬럼의 주소에서 도출된 좌표가 들어 있습니다.

DB2 Spatial Extender는 지오코더(Geocoder) 함수를 사용하여 비즈니스 데이터를 좌표로 변환함으로써 데이터에서 공간 함수를 사용할 수 있도록 합니다.

함수를 사용하여 공간 데이터 생성

함수를 사용하여 입력 데이터에서 공간 데이터를 생성할 수 있습니다.

공간 데이터는 지오코더(Geocoder)뿐만 아니라 다른 함수에 의해서도 생성될 수 있습니다. 예를 들어, 지점이 BRANCHES 테이블에 정의되어 있는 은행에서는 각 지점의 5마일 이내에 얼마나 많은 고객이 있는지 알고자 합니다. 은행이 데이터베이스에서 이러한 정보를 얻으려면 먼저 각 지점 주위의 5마일 반경 이내에 있는 구역을 정의해야 합니다. DB2 Spatial Extender 함수인 ST_Buffer가 해당 정의를 작성할 수 있습니다. 각 지점의 좌표를 입력으로 사용하여 ST_Buffer는 해당 구역의 주계를 구별하는 좌표를 생성할 수 있습니다. 그림 4에서는 ST_Buffer가 제공하는 정보가 있는 BRANCHES 테이블을 보여줍니다.

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

그림 4. 기존 공간 데이터로부터 파생된 새 공간 데이터가 들어 있는 테이블. SALES_AREA 컬럼 좌표는 LOCATION 컬럼 좌표에서 ST_Buffer 함수에 의해 도출되었습니다. LOCATION 컬럼의 좌표와 같이 SALES_AREA 컬럼의 좌표도 시뮬레이트된 것이며 실제 데이터가 아닙니다.

DB2 Spatial Extender는 ST_Buffer 이외에 기존 공간 데이터에서 새 공간 데이터를 도출하는 몇 개의 다른 함수를 제공합니다.

공간 데이터 임포트

Spatial Extender에서는 모양 파일 및 SDE 익스포트 형식으로 공간 데이터를 임포트하는 서비스를 제공합니다.

모양 파일에 있는 공간 데이터는 인터넷을 통해 여러 소스에서 사용할 수 있습니다. KR용 데이터 파일과 Spatial Extender 웹 사이트의 다운로드 섹션에 있는 국가, 주, 도시, 강 등의 전세계 대상물을 다운로드할 수 있습니다.

외부 데이터 소스에서 제공하는 파일로부터 공간 데이터를 임포트할 수 있습니다. 이러한 파일에는 일반적으로 지도에 적용되는 데이터 즉, 거리 네트워크, 범람원, 지진 단층 등이 포함됩니다. 이러한 데이터를 사용자가 생성한 공간 데이터와 조합하여 사용함으로써 사용 가능한 공간 정보를 증가시킬 수 있습니다. 예를 들어, 공공 토목 공사 부서에서 주택 지역이 얼마나 많은 위험에 노출되어 있는지 판별해야 할 경우에는 ST_Buffer를 사용하여 그 지역 주변의 구역을 정의할 수 있습니다. 그런 다음, 공공 토목 공사 부서는 범람원 및 지진 단층에 대한 데이터를 임포트하여 이러한 문제 영역의 어느 부분이 그 구역과 겹치는지 알 수 있습니다.

지형, 공간 정보, 공간 데이터 및 Geometry를 함께 사용하는 방법

이 섹션에서는 DB2® Spatial Extender(지리 대상물, 공간 정보, 공간 데이터 및 Geometry) 조작의 기본이 되는 여러 기본 개념을 요약합니다.

DB2 Spatial Extender를 사용하면 지리적(지구에서의 위치 또는 지역)으로 정의할 수 있는 것에 관련된 사실 또는 수치를 얻을 수 있습니다. DB2 문서에서는 이러한 사실 및 수치를 *공간 정보*라고 하며, 이에 해당하는 물체를 *지리 대상물*(여기서는 간단히 *지형*이라고 함)이라고 합니다.

예를 들어, DB2 Spatial Extender를 사용하여 거주 지역이 쓰레기 매립 예상 지역과 겹치는지를 판별할 수 있습니다. 거주 지역 및 예상 지역은 지형입니다. 겹치는 부분이 있는지를 찾는 것은 공간 정보의 예제가 될 수 있습니다. 겹치는 부분이 있는 경우 이 사항의 Extent도 공간 정보의 예제가 될 수 있습니다.

공간 정보를 작성하려면 DB2 Spatial Extender가 지형의 위치를 정의하는 데이터를 처리해야 합니다. 이러한 데이터를 *공간 데이터*라고 하며 지도 또는 유사한 프로젝션의 위치를 참조하는 좌표로 구성됩니다. 예를 들어, 한 지형이 다른 지형과 겹치는지를 판별하려면 DB2 Spatial Extender는 한 지형의 좌표가 다른 지형의 좌표와 일치하는지를 판별해야 합니다.

공간 정보 기술에서는 지형을 *Geometry*라고 하는 기호로 표시하는 것이 일반적입니다. Geometry는 일부는 시각적이고 일부는 수학적입니다. 시각적 측면을 고려해 보겠습니다. 공원 또는 도시와 같이 너비와 폭을 갖고 있는 지형에 대한 기호는 변이 여러 개인

그림입니다. 이러한 Geometry를 *다각형*이라고 합니다. 강 또는 길과 같은 선형 지형에 대한 기호는 선입니다. 이러한 Geometry를 *선스트링*이라고 합니다.

Geometry는 이 사항이 나타내는 지형에 관한 사실에 해당하는 등록 정보를 갖고 있습니다. 이러한 대부분의 등록 정보는 수학적으로 표현될 수 있습니다. 예를 들어, 지형에 대한 좌표는 공동으로 지형의 해당 Geometry 특성 중 하나를 구성합니다. *차원*이라고 하는 다른 특성은 지형의 길이 또는 폭을 나타내는 숫자 값입니다.

Geometry 관점에서 공간 데이터 및 특정 공간 정보를 볼 수 있습니다. 예를 들어, 앞에서 설명한 거주 지역과 쓰레기 매립 예상 지역을 고려해 보겠습니다. 거주 지역에 대한 공간 데이터는 DB2 데이터베이스의 테이블 컬럼에 저장된 좌표를 포함합니다. 이 규약은 단순히 데이터로 저장된 것이 아니라 실제 Geometry로 저장된 내용에 대한 것입니다. 거주 지역에는 너비와 폭이 있기 때문에 이러한 Geometry가 다각형임을 알 수 있습니다.

공간 데이터와 마찬가지로 특정 공간 정보도 Geometry 관점에서 볼 수 있습니다. 예를 들어, 거주 지역이 쓰레기 매립 예정 지역과 겹치는지를 판별하려면 DB2 Spatial Extender는 지역을 기호화하는 다각형의 좌표를 거주 지역을 나타내는 다각형의 좌표와 비교해야 합니다. 결과 정보(즉, 겹치는 영역)는 다각형(좌표, 차원 및 기타 특성이 있는 Geometry)으로 간주됩니다.

제 2 장 Geometry 정보

이 장에서는 좌표로 구성되고 지리 대상물을 나타내는 Geometry라고 하는 정보 엔티티에 대해 설명합니다. 다루게 되는 주제는 다음과 같습니다.

- Geometry
- Geometry 등록 정보

Geometry

Webster의 개정된 대사전에서 *Geometry*는 『관계, 특성 및 부피, 면적, 선, 각도의 측정법에 대해 연구하는 수학의 분야; 크기의 관계 및 특성을 다루는 과학; 공간에 관련된 과학』이라고 정의되어 있습니다. Geometry라는 단어는 과거 수 천년 동안 지도 제작자들이 세계 지도를 만드는 데 사용한 기하학적 지형을 나타내는 데 사용되었습니다. 이 Geometry의 새 의미에 대한 추상적 정의는 『지형을 나타내는 점 또는 점의 집계』입니다.

DB2 Spatial Extender에서 Geometry의 조작상 정의는 『지리 대상물의 모델』입니다. 모델은 지형 좌표로 표현될 수 있습니다. 모델은 정보를 전달합니다. 예를 들어, 좌표는 고정된 참조점에 대한 지형 위치를 나타냅니다. 또한 모델은 정보를 생성하는 데 사용될 수도 있습니다. 예를 들어, ST_Overlaps 함수는 두 개의 근접 영역 좌표를 입력으로 사용하고 영역이 서로 겹쳐지는지 여부에 관한 정보를 리턴할 수 있습니다.

Geometry가 표현하는 지형 좌표를 Geometry의 등록 정보로 간주합니다. 또한 여러 종류의 Geometry에는 기타 등록 정보(예: 영역, 길이 및 바운더리)가 있습니다.

DB2 Spatial Extender에서 지원하는 Geometry는 다음 그림에 나와 있는 계층 구조를 형성합니다. Geometry 계층 구조는 OGC(OpenGIS Consortium, Inc.) ″OpenGIS Simple Features Specification for SQL″ 문서에 정의되어 있습니다. 계층 구조의 7개 구성원은 인스턴스화할 수 있습니다. 즉, 특정 좌표 값으로 정의한 후 그림에서처럼 시각적으로 렌더링할 수 있습니다.

© Copyright IBM Corp. 1998, 2006

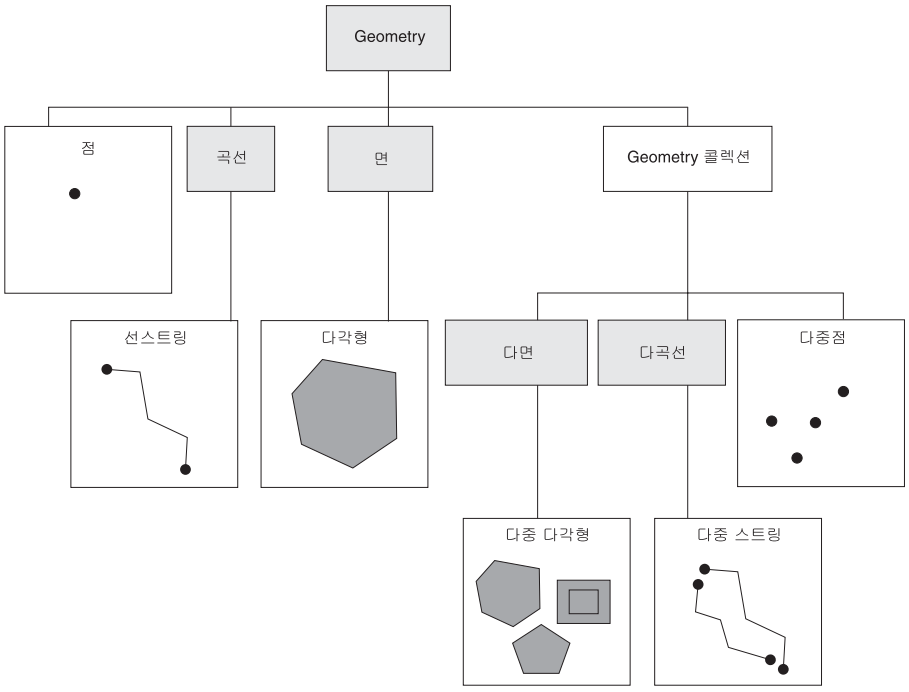


그림 5. DB2 Spatial Extender에서 지원하는 Geometry의 계층 구조. 인스턴스화할 수 있는 이 그림의 Geometry에는 시각적으로 표현할 수 있는 방법에 대한 예가 포함됩니다.

그림에 표시된 Geometry 구현이 DB2 Spatial Extender가 지원하는 공간 데이터 유형입니다.

그림에서와 같이 Geometry라는 슈퍼 클래스는 계층 구조의 루트입니다. 계층 구조의 루트 유형 및 기타 적합한 부속 유형은 인스턴스화할 수 없습니다. 또한 사용자는 인스턴스화할 수 있거나 인스턴스화할 수 없는 적합한 부속 유형을 정의할 수 있습니다.

부속 유형은 기본 Geometry 유형과 동종 콜렉션 유형의 두 유형으로 나눠집니다.

기본 Geometry에는 다음이 포함됩니다.

점 단일 점. 동서 좌표선(예: 위도)과 남북 좌표선(예: 자오선)이 교차하는 장소를 점유할 때 인식되는 이산 지형을 나타냅니다. 예를 들어, 세계 지도 표시에서는 지도 상의 각 도시가 위선과 경선의 교차점에 놓여 있음을 알 수 있습니다. 점은 각 도시를 나타낼 수 있습니다.

선스트링
둘 이상의 점을 연결하는 선. 직선일 필요는 없습니다. 선형 지리 대상물(예: 거리, 운하 및 송수관)을 나타냅니다.

다각형 다각형 또는 다각형 내 표면. 측정 단위 지리 대상물(예: 복지 지역, 숲 및 야생 동물 서식지)을 나타냅니다.

동종 콜렉션에는 다음이 포함됩니다.

다중점 다중점 Geometry 콜렉션. 구성요소 각각이 동서 좌표선 및 남북 좌표선의 교차점에 있는 여러 파트로 이루어진 대상물(예: 구성원 각각이 위선 및 자오선의 교차점에 들어 맞는 섬들의 체인)을 나타냅니다.

다중 선스트링

스트링이 여러 개인 다중 곡선 Geometry 콜렉션. 구성된 여러 파트로 이루어진 대상물(예: 강 체계 및 고속도로망)을 나타냅니다.

다중 다각형

다각형이 여러 개인 다중 표면 Geometry 콜렉션. 여러 면으로 이루어진 단위 또는 구성요소(예: 특정 영역의 집단 농장 또는 호수 체계)를 나타냅니다.

이름이 뜻하는 대로 동종 콜렉션은 기본 Geometry의 콜렉션입니다. 기본 Geometry 등록 정보를 공유하는 것 이외에도 동종 콜렉션에는 자신의 등록 정보도 일부 가지고 있습니다.

Geometry 등록 정보

이 주제는 Geometry의 특성에 대해 설명합니다. 이러한 특성은 다음과 같습니다.

- Geometry가 속하는 유형
- Geometry 좌표
- Geometry의 내부, 바운더리 및 외부
- 단순하거나 단순하지 않은 특성
- 빈 또는 비어 있지 않은 특성
- Geometry의 최소 바운딩 직사각형 또는 인벨로프(envelope)
- 차원
- Geometry가 연관된 공간 참조 시스템의 ID

유형

각 Geometry는 DB2 Spatial Extender에서 지원하는 Geometry 계층 구조 내의 한 유형에 속합니다. 계층 구조의 7개 유형(점, 선스트링, 다각형, Geometry 콜렉션, 다중점, 다중 선스트링 및 다중 다각형)은 특정 좌표 값으로 정의할 수 있습니다.

Geometry 좌표

좌표를 전혀 포함하지 않는 빈 Geometry가 아니라면 모든 Geometry는 최소한 하나의 X 및 하나의 Y 좌표를 포함합니다. 이 외에도 하나의 Geometry에는 하나 이상의 Z 좌표와 M 좌표가 포함될 수 있습니다. X, Y, Z 및 M 좌표는 배정밀도 수로 표시됩니다. 다음 주제에 대해서도 설명합니다.

- X 및 Y 좌표
- Z 좌표

- M 좌표

X 및 Y 좌표

X 좌표값은 동쪽 또는 서쪽 방향으로 참조점에 대해 상대적인 위치를 나타냅니다. Y 좌표값은 북쪽 또는 남쪽 방향으로 참조점에 대해 상대적인 위치를 나타냅니다.

Z 좌표

일부 Geometry에는 관련된 높이 또는 깊이가 있습니다. 지형의 Geometry를 형성하는 각 점에는 지표면에 수직인 높이나 깊이를 나타내는 선택적인 Z 좌표가 포함될 수 있습니다.

M 좌표

M 좌표(측정 단위)는 지리 대상물에 대한 정보를 전달하고 지형의 위치를 정의하는 좌표와 함께 저장되는 값입니다. 예를 들어, 응용프로그램에서 고속도로를 표현한다고 가정하십시오. 응용프로그램에서 직선 거리 또는 이정표를 나타내는 값을 처리하게 하려면 고속도로의 위치를 정의하는 좌표와 함께 이 값을 저장할 수 있습니다. M 좌표는 배정밀도 수로 표시됩니다.

내부, 바운더리 및 외부

모든 Geometry는 내부, 바운더리 및 외부에 의해 정의된 공간에 위치를 차지합니다. Geometry의 외부란 Geometry가 점유하지 않은 모든 공간을 말합니다. Geometry의 바운더리는 내부와 외부 사이의 인터페이스로서 작용합니다. 내부란 Geometry가 점유한 공간을 말합니다.

단순 또는 비단순

일부 Geometry 부속 유형의 값(선스트링, 다중점 및 다중 선스트링)은 단순 또는 비단순형입니다. 부속 유형에 부과된 모든 Geometry의 규칙에 따르면 Geometry는 단순한 것이고 따르지 않으면 단순하지 않은 것입니다. 선스트링이 그의 내부와 교차하지 않으면 이 선스트링은 단순합니다. 다중점의 어떤 요소도 동일한 좌표 공간을 점유하지 않으면 그 다중점은 단순합니다. 점, 표면, 다중 표면 및 빈 Geometry는 항상 단순합니다.

닫힘

곡선은 시작점과 끝점이 동일한 경우 닫힙니다. 다중 곡선은 해당 요소 모두가 닫힌 경우에만 닫힙니다. 링은 닫힌 단순한 곡선입니다.

빈 또는 비어있지 않음

Geometry에 아무 점도 없으면 그 Geometry는 비어 있습니다. 빈 Geometry의 인벨로프, 바운더리, 내부 및 외부는 정의되지 않고 널(NULL)로 표시됩니다. 빈 Geometry는 항상 단순합니다. 빈 다각형 및 다중 다각형의 면적은 0입니다.

MBC(Minimum Bounding Circle)

Geometry의 MBR은 최소 및 최대 (X,Y) 좌표에 의해 형성된 바운딩 Geometry입니다. 다음과 같은 특수한 경우는 예외로 하고 대부분 Geometry의 MBR은 바운더리 직사각형을 형성합니다.

- 점의 MBR은 점 그 자체이며 이는 최소 및 최대 XY 좌표가 동일하기 때문입니다.
- 수평 또는 수직 선스트링의 MBR은 소스 선스트링의 바운더리(끝점)에 의해 표현되는 선스트링입니다.

차원

Geometry에는 -1, 0, 1 또는 2 차원이 있을 수 있습니다. 차원은 다음과 같이 나열됩니다.

–1 비어 있습니다.

0 길이가 없으며 면적이 0입니다.

1 길이가 0보다 길며 면적이 0입니다.

2 면적이 0보다 큽니다.

점 및 다중점 부속 유형은 0 차원을 가집니다. 점은 단일 좌표로 모델화될 수 있는 차원 지형을 나타내는 반면 다중점 부속 유형은 점 세트로 모델화되어야 하는 데이터를 나타냅니다.

선스트링 및 다중 선스트링은 1차원을 가집니다. 이는 도로 세그먼트, 분기 강 체계 및 자연적으로 직선 형태인 기타 다른 지형을 저장합니다.

다각형 및 다중 다각형 부속 유형은 2차원입니다. 주계가 숲, 토지 필지 및 호수와 같이 정의 가능한 영역을 에워싸는 지형은 다각형 또는 다중 다각형 데이터 유형으로 표시될 수 있습니다.

공간 참조 시스템 ID

공간 참조 시스템에 대한 숫자 ID로 Geometry를 표시하는 데 사용되는 공간 참조 시스템을 판별합니다.

데이터베이스에 알려진 모든 공간 참조 시스템은 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰를 통해 액세스될 수 있습니다.

제 3 장 DB2 Spatial Extender 사용법

DB2 Spatial Extender 사용법

DB2® Spatial Extender 지원 및 사용에는 DB2 Spatial Extender 설정 및 공간 데이터를 사용하는 프로젝트에서의 작업이라는 두 가지 기본 활동이 포함되어 있습니다. 이 주제는 공간 태스크를 수행하는 데 사용할 수 있는 인터페이스를 소개합니다.

DB2 Spatial Extender 및 연관 기능에 대한 인터페이스

몇 가지 인터페이스를 통해 DB2 Spatial Extender를 설정하고 공간 데이터를 사용하는 프로젝트를 작성할 수 있습니다. 이러한 인터페이스는 다음과 같습니다.

- DB2 Spatial 및 Geodetic Data Management Feature 지원이 포함되는 소스 프로젝트를 여십시오. 이 지원이 포함된 일부 열린 소스 프로젝트에는 다음이 포함됩니다.
  - GeoTools(http://www.geotools.org/), 공간 응용프로그램을 빌드하기 위한 Java 라이브러리
  - GeoServer (http://docs.codehaus.org/displaly/GEOS/Home), 웹 맵 서버 및 웹 기능 서버
  - uDIG(http://udig.refractions.net/confluence/display/UDIG/Home), 데스크탑 공간 데이터 시각화 및 분석 응용프로그램
- DB2 제어 센터, 창을 포함하는 GUI, 노트북 및 DB2 Spatial Extender를 지원하는 메뉴 선택 항목입니다.
- DB2 Spatial Extender에서 제공하는 명령행 처리기(CLP). 이것은 db2se CLP라고 합니다.
- DB2 Spatial Extender의 스토어드 프로시저를 호출하는 응용프로그램.

공간 정보의 생성을 가능하게 하는 인터페이스도 있습니다. 이러한 인터페이스에는 다음이 포함됩니다.

- DB2 CLP, DB2 제어 센터의 쿼리 창 또는 응용프로그램에서 수행하는 SQL 쿼리.
- 공간 정보를 그래픽 양식으로 렌더링하는 시각화 도구. 한 가지 예는 DB2용 ArcExplorer로서 IBM의 ESRI(Environmental Systems Research Institute)에서 작성한 것입니다. DB2용 ArcExplorer는 다음 DB2 Spatial Extender 웹 사이트에서 다운로드할 수 있습니다.

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

© Copyright IBM Corp. 1998, 2006

DB2 Spatial Extender 설정 및 프로젝트 작성을 위해 수행하는 태스크

이 절에서는 DB2 Spatial Extender를 설정하기 위해 수행하고 공간 데이터를 사용하는 프로젝트에서 작동하는 태스크의 개요를 제공합니다. 여기에는 이러한 태스크를 설명하는 시나리오가 포함되어 있습니다. 태스크는 다음과 같은 두 가지 범주로 나뉩니다.

- DB2 Spatial Extender 설정
- 공간 데이터를 사용하는 프로젝트 작성

DB2 Spatial Extender 설정

이 절에서는 DB2 Spatial Extender를 설정하기 위해 수행하는 태스크를 나열하고 시나리오를 사용하여 가상의 회사가 각 태스크에 접근하는 방법을 설명합니다.

DB2 Spatial Extender를 설정하려면 다음을 수행하십시오.

1. 계획을 세우고 준비하십시오(작성할 프로젝트와 사용할 인터페이스를 결정하고, DB2 Spatial Extender를 관리하고 프로젝트를 작성할 담당자를 선택하는 등).

   시나리오: Safe Harbor Real Estate Insurance Company의 정보 시스템 환경에는 DB2 데이터베이스 시스템과 공간 데이터 전용의 별도 파일 시스템이 포함되어 있습니다. 어느 정도까지는 쿼리 결과에 두 시스템으로부터의 데이터 조합이 포함될 수 있습니다. 예를 들어, DB2 테이블은 수입에 관한 정보를 저장하고 파일 시스템의 파일은 회사의 지점 사무실 위치를 저장합니다. 따라서 지정된 금액의 수입을 올리는 사무실을 찾은 다음 해당 사무실의 위치를 판별할 수 있습니다. 그러나 두 시스템의 데이터를 통합할 수는 없습니다. (예를 들어, 사용자가 DB2 컬럼과 파일 시스템 레코드를 조인할 수 없으며 파일 시스템이 쿼리 최적화와 같은 DB2 서비스를 사용할 수 없습니다.) 이러한 단점을 없애기 위해 Safe Harbor에서는 DB2 Spatial Extender를 사용하여 새 공간 개발 부서(간단히 공간 부서라고도 함)를 설립합니다.

   공간 개발부의 첫 번째 임무는 Safe Harbor의 DB2 환경에 DB2 Spatial Extender를 포함시키는 것입니다.

   - 이 부서의 관리팀은 공간 관리 팀에 DB2 Spatial Extender를 설치 및 구현하도록 지시하고 공간 분석팀에 공간 정보를 생성 및 분석하도록 지시합니다.
   - 관리팀에는 강력한 UNIX® 백그라운드가 있으므로 db2se CLP를 사용하여 DB2 Spatial Extender를 관리하기로 결정합니다.
   - Safe Harbor의 사업상 결정은 주로 고객의 요구사항에 좌우되므로, 관리팀은 고객 관련 정보가 들어 있는 데이터베이스에 DB2 Spatial Extender를 설치하기로 결정합니다. 이 정보의 대부분은 CUSTOMERS 테이블에 저장됩니다.

2. DB2 Spatial Extender를 설치하십시오.

   시나리오: 공간 관리팀이 DB2 환경의 UNIX 머신에 DB2 Spatial Extender를 설치합니다.

3. DB2 Spatial Extender 버전 8이 있으면 공간 데이터를 DB2 버전 9로 이주시키십시오.

   시나리오: 버전 9 릴리즈는 Safe Harbor에서 획득한 첫 번째 버전입니다. 이주가 필요하지 않습니다.

4. 공간 데이터를 수용할 수 있도록 데이터베이스를 구성하십시오. 데이터베이스가 공간 함수, 로그 파일 및 DB2 Spatial Extender 응용프로그램을 수용할 수 있을 만큼 충분한 메모리 및 스페이스를 가지도록 구성 매개변수를 조정하십시오.

   시나리오: 공간 관리팀 구성원은 트랜잭션 로그 특성, 응용프로그램 힙 크기 및 응용프로그램 제어 힙 크기를 DB2 Spatial Extender의 요구사항에 적합한 값으로 조정합니다.

5. 데이터베이스에 대해 공간 자원을 설정하십시오. 이러한 자원에는 시스템 카탈로그, 공간 데이터 유형, 공간 함수, 지오코더(Geocoder) 및 기타 오브젝트가 포함됩니다. 이러한 자원을 설정하는 태스크를 공간 작업에 대한 데이터베이스 사용 가능화라고 합니다.

   DB2 Spatial Extender에서 제공하는 지오코더(Geocoder)는 미국 주소를 공간 데이터로 변환합니다. DB2SE_USA_GEOCODER라고 합니다. 사용자 소속 및 기타 소속에서는 다른 종류의 데이터뿐만 아니라 미국 내 또는 미국 외부의 주소를 공간 데이터로 변환하는 지오코더(Geocoder)를 제공할 수 있습니다.

   시나리오: 공간 관리팀이 계획 중인 프로젝트에 필요한 자원을 설정합니다.

   - 팀의 구성원이 공간 데이터 관련 작업에 대해 데이터베이스를 사용 가능화하는 자원을 획득하는 명령을 발행합니다. 이러한 자원에는 DB2 Spatial Extender 카탈로그, 공간 데이터 유형, 공간 함수 등이 포함됩니다.
   - Safe Harbor는 캐나다로 비즈니스를 확장할 계획이므로, 공간 관리팀은 캐나다 주소를 공간 데이터로 변환하는 지오코더(Geocoder)를 캐나다 벤더에게 권유하기 시작합니다. Safe Harbor에서는 몇 달이 지난 후에야 그러한 지오코더(Geocoder)를 획득할 수 있을 것으로 예상합니다. 따라서, 데이터를 모을 첫 번째 위치는 미국이 됩니다.

공간 데이터를 사용하는 프로젝트 작성

DB2 Spatial Extender를 설정한 후에는 공간 데이터를 사용하는 프로젝트를 작성할 수 있습니다. 이 절에서는 프로젝트 작성과 관련된 태스크를 나열하고 Safe Harbor Real Estate Insurance Company가 비즈니스 및 공간 데이터의 통합을 추구하는 시나리오를 계속합니다.

공간 데이터를 사용하는 프로젝트를 작성하려면 다음을 수행하십시오.

1. 계획을 세우고 준비하십시오(프로젝트에 대한 목표 설정, 필요한 테이블 및 데이터 결정, 사용할 좌표 시스템 결정 등).

시나리오: 공간 개발부가 프로젝트 개발을 준비합니다. 예를 들면, 다음과 같습니다.

- 관리팀이 프로젝트에 대해 다음과 같은 목표를 설정합니다.
  - 새 지점 사무실을 설립할 장소 결정
  - 위험 지역(교통 사고, 범죄, 홍수, 지진 등의 발생률이 높은 지역)에 대한 고객의 근접도를 기초로 프리미엄 조정
- 이러한 특정 프로젝트는 미국 내의 고객 및 사무실과 관련이 있습니다. 그러므로 공간 관리팀은 다음을 결정합니다.
  - DB2 Spatial Extender에서 제공하는 미국에 대한 좌표 시스템 사용. 이것을 GCS_NORTH_AMERICAN_1983이라고 합니다.
  - 미국 주소를 지오코딩(Geocoding)하도록 설계된 DB2SE_USA_GEOCODER 사용.
- 공간 관리팀은 프로젝트의 목표를 충족시키기 위해 어떤 데이터가 필요한지 그리고 이 데이터가 어느 테이블에 들어 있는지를 판별합니다.

2. 필요한 경우, 좌표 시스템을 작성하십시오.

   시나리오: Safe Harbor가 GCS_NORTH_AMERICAN_1983 사용을 결정했으므로 회사에서는 이 단계를 무시할 수 있습니다.

3. 기존 공간 참조 시스템이 사용자의 요구를 충족시키는지 판별하십시오. 사용자의 요구를 충족시키는 시스템이 없으면 새로 작성하십시오.

   공간 참조 시스템은 다음을 포함하는 매개변수 값 세트입니다.

   - 제공된 좌표 범위가 참조하는 스페이스의 최대 가능 범위를 정의하는 좌표. 사용 중인 좌표 시스템에서 판별 가능한 좌표의 최대 가능 범위를 판별하고 이 범위를 반영하는 공간 참조 시스템을 선택하거나 작성해야 합니다.
   - 좌표가 도출되는 좌표 시스템의 이름
   - 입력으로서 수신된 좌표를 최대 효율로 처리 가능한 값으로 변환하기 위해 수학적 연산에서 사용되는 숫자. 좌표는 변환된 양식으로 저장되고 사용자에게는 원래의 양식으로 리턴됩니다.

   시나리오: DB2 Spatial Extender는 GCS_NORTH_AMERICAN_1983과 함께 사용되도록 설계된 공간 참조 시스템 NAD83_SRS_1을 제공합니다. 공간 관리팀은 NAD83_SRS_1을 사용하기로 합니다.

4. 필요에 따라 공간 컬럼을 작성합니다. 공간 컬럼의 데이터가 시각화 도구에 의해 읽혀지면, 해당 컬럼은 자신이 속한 테이블이나 뷰의 유일한 공간 컬럼일 경우가 많습니다. 컬럼이 테이블에 있는 여러 공간 컬럼 중 하나일 경우, 이 컬럼은 다른 공간 컬럼이 없는 뷰에 포함될 수 있으며 시각화 도구는 이 뷰에서 이 공간 컬럼의 데이터를 읽을 수 있습니다.

   시나리오: 공간 관리팀이 공간 데이터를 포함할 컬럼을 정의합니다.

- 팀에서는 LOCATION 컬럼을 CUSTOMERS 테이블에 추가합니다. 테이블에는 고객 주소가 이미 들어 있습니다. DB2SE_USA_GEOCODER는 이들을 공간 데이터로 변환합니다. 그러면 DB2가 이 데이터를 LOCATION 컬럼에 저장합니다.
- 팀은 이제 별도의 파일 시스템에 저장되는 데이터를 포함할 OFFICE_LOCATIONS 테이블 및 OFFICE_SALES 테이블을 작성합니다. 이 데이터에는 Safe Harbor의 지점 사무소 주소, 지오코더(Geocoder)에 의해 이 주소로부터 파생된 공간 데이터 및 각 사무소 주변 5마일 반경 이내의 구역을 정의하는 공간 데이터가 포함됩니다. 지오코더(Geocoder)에서 도출된 데이터는 OFFICE_LOCATIONS 테이블의 LOCATION 컬럼에 입력되며, 구역을 정의하는 데이터는 OFFICE_SALES 테이블의 SALES_AREA 컬럼에 입력됩니다.

5. 필요한 경우, 시각화 도구가 액세스할 수 있도록 공간 컬럼을 설정하십시오. DB2 Spatial Extender 카탈로그에 해당 컬럼을 등록함으로써 이를 수행할 수 있습니다. 공간 컬럼을 등록할 때, DB2 Spatial Extender는 해당 컬럼의 모든 데이터가 동일한 공간 참조 시스템에 속해야 한다는 제한조건을 부과합니다. 이 제한조건은 데이터의 무결성(대부분의 시각화 도구의 일반적인 요구사항)을 강화합니다.

   시나리오: 공간 관리팀은 시각화 도구를 사용하여 LOCATION 컬럼 및 SALES_AREA 컬럼의 컨텐츠를 지도에 그래픽으로 표현하고자 합니다. 따라서 팀은 세 컬럼을 모두 등록합니다.

6. 공간 컬럼을 채우려면 다음을 수행하십시오.

   공간 데이터를 임포트해야 하는 프로젝트의 경우에는 데이터를 임포트하십시오.

   지오코더(Geocoder)가 필요한 프로젝트의 경우에는 다음을 수행하십시오.

   - 지오코더(Geocoder)를 호출할 때 필요한 제어 정보를 미리 설정하십시오.
   - 옵션으로서, 데이터베이스에 새 주소가 추가되거나 기존 주소가 갱신될 때마다 지오코더(Geocoder)가 자동으로 실행되도록 설정하십시오.

   필요한 경우, 지오코더(Geocoder)를 일괄처리 모드로 실행하십시오.

   공간 함수로 공간 데이터를 작성해야 하는 프로젝트의 경우에는 해당 함수를 실행하십시오.

   시나리오: 공간 관리팀은 CUSTOMER 테이블의 LOCATION 컬럼, OFFICE_LOCATIONS 테이블, OFFICE_SALES 테이블 및 새 HAZARD_ZONES 테이블을 채웁니다.

   - 팀이 DB2SE_USA_GEOCODER를 사용하여 CUSTOMER 테이블에 있는 주소를 지오코딩(Geocoding)합니다. 지오코딩에서 생성한 좌표는 테이블의 LOCATION 컬럼에 설정됩니다.
   - 팀이 유틸리티를 사용하여 사무실 데이터를 파일 시스템에서 파일로 로드합니다. 그런 다음, 이 데이터를 새 OFFICE_LOCATIONS 테이블로 임포트합니다.

- 팀이 HAZARD_ ZONES 테이블을 작성하고 해당 공간 컬럼을 등록하며 데이터를 이 컬럼으로 임포트합니다. 데이터는 맵 벤더가 제공한 파일에서 옵니다.

7. 필요한 경우, 공간 컬럼으로의 액세스를 용이하게 하십시오. 여기에는 DB2가 공간 데이터에 빨리 액세스할 수 있게 하는 인덱스를 정의하는 것과 사용자가 상호 관련된 데이터를 효율적으로 검색할 수 있게 하는 뷰를 정의하는 것이 포함됩니다. 시각화 도구가 뷰의 공간 컬럼에 액세스할 수 있게 하려면 이러한 컬럼을 DB2 Spatial Extender에 등록해야 합니다.

   시나리오: 공간 관리팀이 등록된 컬럼에 대한 인덱스를 작성합니다. 그런 다음, CUSTOMERS 및 HAZARD ZONES 테이블의 컬럼을 조인하는 뷰를 작성합니다. 다음에는 공간 컬럼을 이 뷰에 등록합니다.

8. 공간 정보와 관련 비즈니스 정보를 생성하십시오. 이 정보를 분석하십시오. 이 태스크에는 공간 컬럼 및 관련된 비공간 컬럼을 쿼리하는 것이 포함됩니다. 이러한 쿼리에는 광범위한 정보(예: 유해 폐기물 처리 지역 주위의 예상 안전 지역 또는 이 지역과 가장 근접한 공공 건물 사이의 최소 거리를 정의하는 좌표)를 리턴하는 DB2 Spatial Extender 함수를 포함할 수 있습니다.

   시나리오: 공간 분석팀이 쿼리를 실행하여 원래의 목표(새 지점 사무실 설립 위치 결정과 위험 지역에 대한 고객의 근접도를 토대로 프리미엄 조정)를 달성하는 데 도움이 되는 정보를 획득합니다.

제 4 장 DB2 Spatial Extender 시작하기

이 장에서는 AIX, HP-UX, Windows, Linux, IBM 시스템 z의 Linux 및 Solaris 운영 환경에 대한 Spatial Extender 설치 및 구성 관련 지시사항을 제공합니다. 또한 이 장에는 Spatial Extender를 호출할 때 발생할 수 있는 몇 가지 설치 및 구성 문제점에 대한 해결 방법에 대해서도 설명되어 있습니다.

Spatial Extender 설정 및 설치

전제조건

DB2 Spatial Extender를 설치하려면 클라이언트 및 서버에 DB2 소프트웨어가 설치 및 구성되어 있어야 합니다.

DB2 Spatial Extender 시스템은 DB2 데이터베이스 시스템, DB2 Spatial Extender 및 geobrowser(대부분 응용프로그램의 경우)로 구성됩니다. geobrowser는 필수는 아니지만 일반적으로 맵의 형태로 공간 쿼리의 결과를 시각적으로 렌더링하기 위해 유용합니다. 공간 데이터 관련 작업에 사용할 수 있는 데이터베이스는 서버에 있습니다. 클라이언트 응용프로그램을 사용하여 DB2 Spatial Extender 스토어드 프로시저 및 공간 쿼리를 통해 공간 데이터를 액세스할 수 있습니다. 클라이언트와 서버 모두가 동일한 머신에 상주하는 구성의 독립형 환경에서 DB2 Spatial Extender를 구성할 수 있습니다. 클라이언트-서버와 독립형 구성 모두에서 개방형 소스 geobrowser, ArcExplorer for DB2 또는 ArcSDE와 함께 실행되는 ESRI의 ArcGIS 도구 세트와 같은 geobrowser를 사용하여 공간 데이터를 볼 수 있습니다.

IBM의 DB2 Spatial Extender 웹 사이트에서 DB2용 ArcExplorer 무료 사본을 다운로드할 수 있습니다.

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

이 태스크를 수행하려면

1. 시스템이 모든 소프트웨어 요구사항을 충족하는지 확인하십시오.
2. Spatial Extender를 설치하십시오. 단계는 사용하는 운영 체제에 따라 다릅니다.
   - Windows
   - AIX
   - HP-UX
   - Solaris 운영 환경
   - Linux

© Copyright IBM Corp. 1998, 2006

3. UNIX 플랫폼의 경우: DB2 Spatial Extender 인스턴스 환경을 작성하십시오.
4. 설치를 확인하십시오.
5. 필요하면 문제점 해결 추가 정보를 참조하여 문제점을 정정하는 적절한 조치를 수행하십시오.
6. 컴퓨터에서 DB2 문서에 액세스하려 하지만 아직 DB2 정보 센터가 설치되어 있지 않은 경우, DB2 정보 센터 설치(UNIX) 또는 DB2 정보 센터 설치(Windows)를 참조하십시오. DB2 정보 센터에는 DB2 데이터베이스 시스템 및 DB2 데이터베이스 관련 제품에 대한 문서가 있습니다.

관련 태스크

32 페이지의『DB2 Spatial Extender 인스턴스 환경 작성』

Spatial Extender 설치에 대한 시스템 요구사항

DB2 Spatial Extender를 설치하기 전에 시스템이 아래 설명한 소프트웨어 및 디스크 스페이스 요구사항 모두를 충족하는지 확인하십시오.

운영 체제

Intel 기반 시스템의 32비트 Windows 또는 Linux에 DB2 Spatial Extender를 설치할 수 있습니다. AIX, HP-UX, Solaris, Linux 및 Linux for system z와 같은 64비트 UNIX 시스템에도 DB2 Spatial Extender를 설치할 수 있습니다.

소프트웨어 요구사항

DB2 Spatial Extender를 설치하려면 서버에 다음 DB2 소프트웨어를 설치 및 구성해야 합니다.

서버 소프트웨어

DB2 Spatial Extender를 설치하기 *전에* DB2를 설치해야 합니다. DB2 제어 센터를 사용할 계획이면 DB2 Administration Server(DAS)를 작성 및 구성하십시오. DAS 작성 및 구성에 대한 자세한 정보는 *IBM DB2 관리 안내서: 구현*을 참조하십시오.

공간 클라이언트 소프트웨어

Windows에서 DB2 Spatial Extender를 설치할 경우, Spatial Extender의 기본 설치는 Spatial Client를 포함합니다. AIX, HP-UX, Solaris, Linux for Intel 또는 System z의 Linux에 설치된 DB2 Spatial Extender의 경우, 관리 또는 Application Development Client와 함께 DB2 서버를 설치할 때 공간 클라이언트를 설치할 수 있습니다. 이 클라이언트를 설치하지 않을 경우에는 사용자 정의 설치 옵션을 선택하여 수동으로 Spatial Client를 설치해야 합니다.

디스크 스페이스 요구사항

Spatial Extender를 설치하려면, 시스템이 다음 테이블에 나열된 디스크 스페이스 요구사항에 만족해야 합니다. DB2 Spatial Extender용 라이브러리 코드는 DB2 Geodetic Data Management Feature용 코드를 통합하지만 측지 라이센스 키는 통합하지 않습니다.

표 1. DB2 Spatial Extender의 디스크 스페이스 요구사항

DB2 Spatial Extender 소프트웨어	**디스크 스페이스**
DB2 Spatial Extender의 서버 소프트웨어: • Spatial Extender 및 Geodetic Data Management Feature 서버 라이브러리 코드 및 문서	596MB 전체 디스크 스페이스: • 33MB

표 1에서는 Windows 일반 설치 또는 AIX, HP-UX, Solaris 운영 환경, Intel용 Linux 및 System z용 Linux에서 사전 선택된 구성요소로 DB2 데이터베이스 시스템 및 DB2 Spatial Extender를 설치할 때 필요한 디스크 스페이스를 지정합니다.

시스템이 모든 소프트웨어 및 디스크 스페이스 요구사항에 만족하면, Spatial Extender를 설치할 수 있습니다.

Windows용 DB2 Spatial Extender 설치

DB2 Spatial Extender 기능을 설치하려면 DB2 서버 제품이 설치되어 있어야 합니다.

이 태스크는 DB2 Spatial Extender 설치에 대한 기본 태스크 중 일부입니다.

DB2 설치 마법사 또는 응답 파일을 사용하여 Windows 운영 체제에 DB2 Spatial Extender를 설치할 수 있습니다.

권장사항:
DB2 설치 마법사를 사용하여 Spatial Extender를 설치하십시오. 설치 마법사는 설치 도움말, 자동화된 사용자 및 그룹 작성, 프로토콜 구성 및 인스턴스 작성이 포함된 사용하기 쉬운 그래픽 인터페이스를 제공합니다.

DB2 설치 마법사를 사용하여 Spatial Extender를 설치할 경우, 설치 중에 언제라도 **취소**를 눌러 프로세스를 종료할 수 있습니다.

이 태스크를 수행하려면

1. DB2 설치 마법사를 사용하여 Windows용 Spatial Extender를 설치하려면 다음을 수행하십시오.
   a. CD 드라이브에 Spatial Extender CD를 삽입하십시오. DB2 Spatial Extender를 설치할 수 있는 인터페이스인 DB2 설치 런치패드가 열립니다.
   b. **제품 설치**를 누르십시오.

c. 설치할 제품으로 DB2 Spatial Extender를 선택하고 **다음**을 누르십시오.DB2 설치 마법사가 실행됩니다. **다음**을 누르십시오. DB2 설치 마법사가 전체 설치 과정을 단계별로 안내합니다. 설치 중 언제라도 **도움말**을 눌러 온라인 설치 도움말을 시작할 수 있습니다.

2. 응답 파일을 사용하여 Windows용 Spatial Extender를 설치하려면 다음을 수행하십시오.

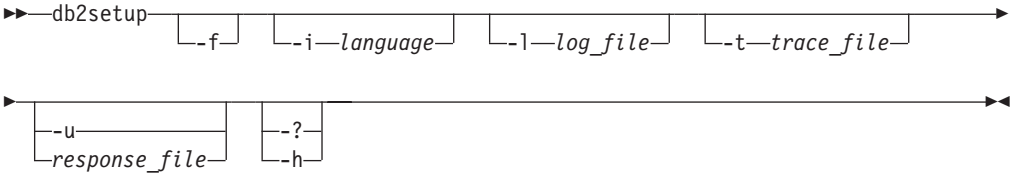
   a. 설치를 수행할 때 사용할 사용자 어카운트로 시스템에 로그온하십시오.

   b. Spatial Extender CD를 삽입하십시오. 자세한 정보는 *DB2 설치 및 구성 보충 설명서*를 참조하십시오.

   c. 명령 프롬프트에서 db2setup 명령을 실행하여 설치 프로그램을 실행하십시오.

   d. 설치가 완료되면 로그 파일에 있는 메시지를 점검하십시오.

db2setup 명령



여기서

-f 설치하기 전에 DB2 프로세스를 강제로 중지시킵니다.

-i(*language*)
설치를 수행할 언어를 나타내는 두 자로 된 언어 코드입니다.

-l(*log_file*)
사용할 로그 파일의 전체 경로 및 파일 이름입니다.

-t(*trace_file*)
설치 추적 정보를 사용하여 완전한 파일을 생성합니다.

-u(*response_file*)
완전한 응답 파일 이름을 지정합니다. 제공되는 샘플 응답 파일을 변경하고 이름을 바꾼 경우에는 이 매개변수가 새 이름과 일치하도록 해야 합니다. 이 매개변수는 필수입니다. 응답 파일은 DB2 Spatial Extender 설치 CD의 db2₩Windows₩samples₩db2gse.rsp에 있습니다.

-?, -h 사용법 정보를 생성합니다.

AIX용 DB2 Spatial Extender 설치

전제조건

AIX에 Spatial Extender를 설치하기 전에,

- 시스템이 모든 소프트웨어, 메모리 및 디스크 스페이스 요구사항을 충족시키는지 확인하십시오.
- 구성 매개변수를 갱신하고 AIX의 모든 DB2 클라이언트 및 서버에 대해 시스템을 재시작하십시오.
- 서버나 독립형 환경에 설치할 경우에는 DB2 버전 9 서버 제품이 설치되어 있어야 합니다.

  주: DB2 Spatial Client가 이미 설치되어 있는지 점검하십시오. Spatial Extender 클라이언트와 샘플 구성요소는 DB2 클라이언트 및 서버에서 사용할 수 있습니다. DB2 사용자 정의 설치 유형을 사용하고 **클라이언트 지원**에서 **Spatial Extender 클라이언트** 기능을 선택하며 **응용프로그램 개발 도구**에서 **Spatial Extender 샘플** 기능을 선택하면 이 공간 구성요소를 설치할 수 있습니다. 공간 클라이언트 기능만 필요하고 DB2와 함께 이 공간 구성요소가 이미 설치되어 있는 경우, 다음 DB2 Spatial Extender 설치 프로시저를 수행할 필요가 없습니다.
- 루트 권한이 있어야 합니다.

DB2 설치 마법사, db2_install 스크립트 또는 SMIT(System Management Interface Tool)를 사용하여 AIX용 DB2 Spatial Extender를 설치할 수 있습니다.

권장사항: DB2 설치 마법사를 사용하여 Spatial Extender를 설치하십시오. 설치 마법사는 설치 도움말, 자동화된 사용자 및 그룹 작성, 프로토콜 구성 및 인스턴스 작성이 포함된 사용하기 쉬운 그래픽 인터페이스를 제공합니다. 마법사를 사용하지 않을 경우에는 db2_install 스크립트 또는 AIX의 SMIT(System Management Interface Tool)를 사용하여 Spatial Extender를 설치할 수 있습니다. SMIT를 사용하여 Spatial Extender를 설치하는 것은 고급 사용자가 설정 프로세스에 대한 수동 제어를 필요로 하는 상황에서 권장됩니다.

Spatial Extender는 기타 다른 DB2 제품과 함께 /usr/opt/db2_09_01 디렉토리에 설치됩니다. Spatial Extender를 설치한 후에는 아직 작성하지 않은 경우, DB2 인스턴스 환경을 작성한 다음 설치를 검증하십시오.

DB2 설정 마법사

1. 루트 권한을 가진 사용자로서 로그인하십시오.
2. Spatial Extender CD를 삽입하여 마운트하십시오. DB2 Spatial Extender를 설치할 수 있는 인터페이스인 DB2 설치 런치패드가 열립니다. CD를 마운트하는 방법에 대한 정보는 *DB2 설치 및 구성 보충 설명서*를 참조하십시오.
3. 설치할 제품으로 DB2 Spatial Extender를 선택하고 **다음**을 누르십시오.
4. DB2 설치 마법사 창이 열립니다. DB2 설치 마법사가 전체 설치 과정을 단계별로 안내합니다. 설치 중 언제라도 **도움말**을 눌러 온라인 설치 도움말을 시작할 수 있습니다.

db2_install 스크립트

1. 루트 권한을 가진 사용자로서 로그인하십시오.
2. 해당 CD를 삽입하여 마운트하십시오.
3. db2_install 스크립트를 시작하여 ./db2_install 명령을 입력하십시오. db2_install 스크립트는 DB2 제품 CD의 루트 디렉토리에 있습니다. db2_install 스크립트에서 제품 키워드를 입력하라는 프롬프트가 표시됩니다.
4. **GSE**를 입력하여 DB2 Spatial Extender를 설치하십시오.

SMIT

1. 루트 권한을 가진 사용자로서 로그인하십시오.
2. Spatial Extender CD를 삽입하여 마운트하십시오.
3. smit install_latest 명령을 입력하십시오
4. 소프트웨어 필드의 입력 디바이스/디렉토리에 /cdrom/db2를 입력하십시오.
5. 설치 디렉토리가 있는지 확인하려면 DO를 누르거나 Enter를 누르십시오.
6. 설치할 소프트웨어 필드에서 클라이언트 구성요소를 설치할 것인지 또는 서버 구성요소를 설치할 것인지 표시하십시오. DB2 Spatial Extender에 대해 설치해야 하는 구성요소의 전체 목록은 DB2 Spatial Extender CD의 ComponentList.htm 파일을 참조하십시오.
7. DO 또는 Enter를 누르십시오. 설치 매개변수의 확인을 요구하는 프롬프트가 표시됩니다. Enter를 눌러 확인하십시오. 로그아웃하십시오.

HP-UX용 DB2 Spatial Extender 설치

HP-UX용 DB2 Spatial Extender 제품을 설치하기 전에,

- 시스템이 모든 하드웨어, 소프트웨어 및 메모리 요구사항을 충족시키는지 확인하십시오.
- DB2 서버 제품이 설치되어 있어야 합니다.

  주: DB2 Spatial Client가 이미 설치되어 있는지 점검하십시오. Spatial Extender 클라이언트와 샘플 구성요소는 DB2 클라이언트 및 서버에서 사용할 수 있습니다. DB2 사용자 정의 설치 유형을 사용하고 **클라이언트 지원**에서 **Spatial Extender 클라이언트** 기능을 선택하며 **응용프로그램 개발 도구**에서 **Spatial Extender 샘플** 기능을 선택하면 이 공간 구성요소를 설치할 수 있습니다. 공간 클라이언트 기능만 필요하고 DB2와 함께 이 공간 구성요소가 이미 설치되어 있는 경우, 다음 DB2 Spatial Extender 설치 프로시저를 수행할 필요가 없습니다.
- 구성 매개변수를 갱신하고 HP-UX의 모든 DB2 클라이언트 및 서버에 대해 시스템을 재시작하십시오.

- 루트 권한이 있어야 합니다.

DB2® 설치 마법사, db2_install 스크립트 또는 swinstall 명령을 사용하여 Spatial Extender를 설치할 수 있습니다.

권장사항:
DB2 설치 마법사를 사용하여 Spatial Extender를 설치하십시오. 설치 마법사는 설치 도움말, 자동화된 사용자 및 그룹 작성, 프로토콜 구성 및 인스턴스 작성이 포함된 사용하기 쉬운 그래픽 인터페이스를 제공합니다. 마법사를 사용하지 않을 경우에는 db2_install 스크립트 또는 swinstall 명령을 사용하여 HP-UX용 Spatial Extender를 설치할 수 있습니다. HP-UX swinstall 명령을 사용하여 Spatial Extender를 설치하는 것은 고급 사용자가 설정 프로세스에 대한 수동 제어를 필요로 하는 상황에서 권장됩니다.

이 태스크를 수행하려면

DB2 설치 마법사를 사용한 HP-UX용 Spatial Extender 설치

1. DB2 Spatial Extender CD를 삽입하여 마운트하십시오. DB2 Spatial Extender를 설치할 수 있는 인터페이스인 DB2 설치 런치패드가 열립니다.
2. 설치할 제품으로 DB2 Spatial Extender를 선택하고 **다음**을 누르십시오.DB2 설치 마법사가 실행됩니다. **다음**을 누르십시오. DB2 설치 마법사가 전체 설치 과정을 단계별로 안내합니다. 설치 중 언제라도 **도움말**을 눌러 온라인 설치 도움말을 시작할 수 있습니다.

db2_install 스크립트를 사용한 HP-UX용 Spatial Extender 설치

1. 루트 권한을 가진 사용자로서 로그인하십시오.
2. 해당 CD를 삽입하여 마운트하십시오.
3. db2_install 스크립트를 시작하여 ./db2_install 명령을 입력하십시오. db2_install 스크립트는 DB2 제품 CD의 루트 디렉토리에 있습니다. db2_install 스크립트에서 제품 키워드를 입력하라는 프롬프트가 표시됩니다.
4. **GSE**를 입력하여 DB2 Spatial Extender를 설치하십시오.

swinstall 명령을 사용한 HP-UX용 Spatial Extender 설치

1. 루트 권한을 가진 사용자로서 로그인하십시오.
2. Spatial Extender CD를 삽입하여 마운트하십시오.
3. swinstall 프로그램을 실행하십시오. 다음 명령을 사용하십시오.

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

   이 명령은 소프트웨어 선택 창 및 소스 지정 창을 엽니다. 필요한 경우, 소스 지정 창의 **소스 호스트 이름** 필드에 있는 값을 변경하십시오.

4. **소스 저장소 경로** 필드에 /cdrom/db2/hpux를 입력하십시오. 여기서 /cdrom은 CD 마운트 디렉토리를 나타냅니다.
5. **확인**을 눌러 소프트웨어 선택 창으로 되돌아가십시오. 소프트웨어 선택 창에 설치 가능한 소프트웨어 목록이 들어 있습니다.
6. 설치 허가된 제품을 선택하십시오.
7. **조치** 메뉴에서 **설치 표시**를 선택하여 설치할 제품을 선택하십시오. 다음과 유사한 메시지가 표시됩니다.

   ```
   표시한 소프트웨어 이외에, 종속성을 해결하기 위해 다른 소프트웨어가
   자동으로 표시되었습니다. 이 메시지는 다시 나타나지 않습니다.
   ```

8. **확인**을 선택하십시오.
9. **조치** 메뉴에서 **설치(분석)**를 선택하여 제품 설치를 시작하고 설치 분석 창을 여십시오.
10. **상태** 필드에 준비 메시지가 표시되면 설치 분석 창에서 **확인**을 선택하십시오.
11. 구성 창에서 **예**를 선택하여 소프트웨어를 설치할 것임을 확인하십시오.

    상태 필드에 준비가 표시되고 참고 창이 열릴 때까지, 소프트웨어가 설치되는 동안 설치 창에서 처리 데이터를 읽으십시오. swinstall 프로그램은 파일 세트를 로드하며 파일 세트에 대한 제어 스크립트를 실행합니다.
12. **파일** 메뉴에서 **종료**를 선택하여 swinstall을 종료하십시오.
13. 설치가 완료되면 Spatial Extender가 다른 DB2 제품과 함께 /opt/IBM/db2/V9.1 디렉토리에 설치됩니다.
14. Spatial Extender를 설치한 후에는 아직 작성하지 않은 경우, DB2 인스턴스 환경을 작성한 다음 설치를 검증하십시오.

Solaris 운영 환경용 DB2 Spatial Extender 설치

Solaris 운영 환경용 DB2 Spatial Extender 제품을 설치하기 전에

- 시스템이 모든 하드웨어, 소프트웨어 및 메모리 요구사항을 충족시키는지 확인하십시오.
- 서버 또는 독립형 환경에 설치하는 경우 DB2 버전 9 서버 제품이 설치되어 있어야 합니다.

  주: DB2 Spatial Client가 이미 설치되어 있는지 점검하십시오. Spatial Extender 클라이언트와 샘플 구성요소는 DB2 클라이언트 및 서버에서 사용할 수 있습니다. DB2 사용자 정의 설치 유형을 사용하고 **클라이언트 지원**에서 **Spatial Extender 클라이언트** 기능을 선택하며 **응용프로그램 개발 도구**에서 **Spatial Extender 샘플** 기능을 선택하면 이 공간 구성요소를 설치할 수 있습니다. 공간 클라이언트

기능만 필요하고 DB2와 함께 이러한 공간 구성요소가 이미 설치되어 있는 경우 다음 프로시저를 수행할 필요가 없습니다.

- 구성 매개변수를 갱신하고 Solaris 운영 환경의 모든 DB2 클라이언트 및 서버에서 시스템을 재시작하십시오.
- 루트 권한이 있어야 합니다.

DB2® 설치 마법사, db2_install 스크립트 또는 pkgadd 명령을 사용하여 Spatial Extender를 설치할 수 있습니다.

권장사항: DB2 설치 마법사를 사용하여 DB2 Spatial Extender를 설치하십시오.

설치 마법사는 설치 도움말, 자동화된 사용자 및 그룹 작성, 프로토콜 구성 및 인스턴스 작성이 포함된 사용하기 쉬운 그래픽 인터페이스를 제공합니다. 마법사를 사용하지 않을 경우에는 db2_install 스크립트 또는 Solaris 운영 환경 pkgadd 명령을 사용하여 Spatial Extender를 설치할 수 있습니다. Solaris 운영 환경 pkgadd 명령은 설치 프로세스에 대한 보다 강력한 수동 제어가 필요한 경우에 한해 고급 사용자가 사용할 수 있습니다.

DB2 Spatial Extender는 Solaris 운영 환경에서 패키지라고 하는 다양한 함수 및 구성요소로 구성됩니다. pkgadd 명령을 사용하여 Spatial Extender를 설치할 때는 필수 패키지 및 사용하고자 하는 선택적 함수와 연관된 패키지를 설치해야 합니다. DB2 Spatial Extender CD의 ComponentList.htm 파일은 DB2 Spatial Extender에 대해 설치해야 하는 전체 패키지 목록을 포함합니다. ComponentList.htm 파일은 */cdrom*/db2/solaris에 있습니다. 여기서, */cdrom*은 DB2 Spatial Extender CD의 마운트 지점입니다.

아직 DB2 인스턴스 환경을 작성하지 않은 경우 작성한 다음 설치를 검증하십시오.

이 태스크를 수행하려면

DB2 설치 마법사를 사용하여 Solaris 운영 환경용 DB2 Spatial Extender를 설치하십시오.

1. 루트 권한을 가진 사용자로서 로그인하십시오.
2. DB2 Spatial Extender CD를 삽입하여 마운트하십시오. DB2 Spatial Extender를 설치할 수 있는 인터페이스인 DB2 설치 런치패드가 열립니다. CD 마운트 방법에 대해서는 *UNIX용 DB2 빠른 시작*을 참조하십시오.
3. 설치할 제품으로 **Spatial Extender**를 선택하고 **다음**을 누르십시오.
4. DB2 설치 마법사가 실행됩니다. DB2 설치 마법사가 전체 설치 과정을 단계별로 안내합니다. 설치 중에 언제라도 **도움말**을 눌러 온라인 설치 도움말을 실행할 수 있습니다.

설치가 완료되면 Spatial Extender 소프트웨어가 /opt/ibm/db2/V9.1 디렉토리에 설치됩니다.

db2_install 스크립트를 사용한 DB2 Spatial Extender 설치

1. 루트 권한을 가진 사용자로서 로그인하십시오.
2. 해당 CD를 삽입하여 마운트하십시오.
3. db2_install 스크립트를 시작하여 ./db2_install 명령을 입력하십시오. db2_install 스크립트는 DB2 제품 CD의 루트 디렉토리에 있습니다. db2_install 스크립트에서 제품 키워드를 입력하라는 프롬프트가 표시됩니다.
4. **GSE**를 입력하여 DB2 Spatial Extender를 설치하십시오. **SPATIAL_EXTENDER CLIENT**를 입력하여 DB2 Spatial Extender 및 DB2 Administration Client를 모두 설치할 수 있습니다.

설치가 완료되면 Spatial Extender 소프트웨어가 /opt/ibm/db2/V9.1 디렉토리에 설치됩니다.

pkgadd 명령을 사용하여 Solaris용 DB2 Spatial Extender를 설치하십시오.

1. 루트 권한을 가진 사용자로서 로그인하십시오.
2. DB2 Spatial Extender CD를 삽입하여 마운트하십시오.
3. 설치할 필수 패키지 및 선택적 패키지를 식별하십시오. DB2 Spatial Extender에 대해 설치해야 하는 전체 구성요소 목록에 대해서는 CD에 있는 ComponentList.htm 파일을 참조하십시오.
4. 설치하려는 각 패키지에 대해 **pkgadd** 명령을 실행하십시오.

   ```
   pkgadd package_name
   ```

   이 명령에서 *package_name*은 설치하려는 패키지입니다. 예를 들어, Spatial Extender 기본 서버 지원을 설치하려면 다음 명령을 입력하여 db2gssg81 패키지를 설치해야 합니다.

   ```
   pkgadd db2gssg91
   ```

설치가 완료되면 Spatial Extender 소프트웨어가 /opt/ibm/db2/V9.1 디렉토리에 설치됩니다.

Linux용 DB2 Spatial Extender 설치

DB2 설치 마법사를 사용하거나 db2_install 스크립트를 사용하거나 rpm 명령을 사용하여 Linux용 DB2 Spatial Extender를 설치할 수 있습니다.

설치가 완료되면 Spatial Extender가 다른 DB2 제품과 함께 /opt/IBM/db2/V9.1 디렉토리에 설치됩니다. Spatial Extender를 설치한 후에는 아직 작성하지 않은 경우, DB2 인스턴스 환경을 작성한 다음 설치를 검증하십시오.

DB2 설치 마법사를 사용하여 DB2 Spatial Extender 설치

Linux용 DB2 Spatial Extender 제품을 설치하기 전에 다음을 수행하십시오.

- 시스템이 모든 하드웨어, 소프트웨어 및 메모리 요구사항을 충족시키는지 확인하십시오.
- 서버 환경 또는 독립형 환경에서 설치하는 경우 DB2 서버 제품이 설치되어 있는지 확인하십시오.

주:

- DB2 Spatial Client가 이미 설치되어 있는지 점검하십시오. DB2 Spatial Extender 클라이언트 및 샘플 구성요소는 DB2 클라이언트 및 서버에서 사용 가능합니다. DB2 사용자 정의 설치 유형을 사용하는 경우 이 공간 구성요소를 설치할 수 있으며, **클라이언트 지원** 아래의 **Spatial Extender 클라이언트** 기능을 선택하고 **응용프로그램 개발 도구** 아래의 **Spatial Extender 샘플** 기능을 선택합니다. 공간 클라이언트 기능만 필요하고 DB2와 함께 이 공간 구성요소가 이미 설치되어 있는 경우, 다음 DB2 Spatial Extender 설치 프로시저를 수행할 필요가 없습니다.
- 구성 매개변수를 갱신하고 Linux의 모든 DB2 클라이언트 및 서버용 시스템을 재시작하십시오.
- 루트 권한이 있는 확인하십시오.

팁: DB2 설치 마법사를 사용하여 Spatial Extender를 설치하십시오. 설치 마법사는 설치 도움말, 자동화된 사용자 및 그룹 작성, 프로토콜 구성 및 인스턴스 작성이 포함된 사용하기 쉬운 그래픽 인터페이스를 제공합니다. 마법사를 사용하지 않을 경우에는 db2_install 스크립트를 사용하거나 rpm 명령을 사용하여 Spatial Extender를 설치할 수 있습니다. 설치 프로세스에 보다 강력한 수동 제어가 필요한 경우는 고급 사용자가 Linux rpm 명령을 사용하여 Spatial Extender를 설치하는 것이 좋습니다.

1. 루트 권한을 가진 사용자로서 로그인하십시오.
2. DB2 Spatial Extender CD를 삽입하여 마운트하십시오. DB2 Spatial Extender를 설치할 수 있는 인터페이스인 DB2 설치 런치패드가 열립니다. CD를 마운트하는 방법에 대한 정보는 *DB2 설치 및 구성 보충 설명서*를 참조하십시오.
3. **제품 설치**를 누르십시오.
4. 설치할 제품으로 **Spatial Extender**를 선택하고 **다음**을 누르십시오.
5. DB2 설치 마법사 창에서 원하는 옵션을 선택하십시오. **DB2 Spatial Extender** 또는 **DB2 Application Development Client** 중 하나를 설치할 수 있습니다.

DB2 설치 마법사가 전체 설치 과정을 단계별로 안내합니다.

db2_install 스크립트를 사용한 DB2 Spatial Extender 설치

1. 루트 권한을 가진 사용자로서 로그인하십시오.
2. 해당 CD를 삽입하여 마운트하십시오.
3. db2_install 스크립트를 시작하여 ./db2_install 명령을 입력하십시오. db2_install 스크립트는 DB2의 루트 디렉토리에 있습니다. db2_install 스크립트에서 제품 키워드를 입력하라는 프롬프트가 표시됩니다.
4. DB2 Spatial Extender를 설치하려면 **SPATIAL_EXTENDER**를 입력하십시오. **SPATIAL_EXTENDER CLIENT**를 입력하여 DB2 Spatial Extender 및 DB2 Administration Client를 모두 설치할 수 있습니다.

DB2 Spatial Extender 인스턴스 환경 작성

이 태스크는 Spatial Extender 설정에 대한 기본 태스크 중 일부입니다.

제한사항: 이 섹션은 UNIX 시스템에 적용됩니다.

Spatial Extender를 설치하기 전에 작성한 DB2 인스턴스는 해당 인스턴스 환경에 DB2 Spatial Extender를 포함시키지 않습니다. 기존 DB2 인스턴스를 갱신하려면 **db2iupdt** 명령을 사용하십시오. DB2 제어 센터를 사용하며 DB2 Spatial Extender를 설치하기 전에 DB2 Administration Server에 대한 인스턴스를 작성한 경우에는 이 인스턴스를 갱신해야 합니다.

이 태스크를 수행하려면

1. db2iupdt 명령을 사용하여 인스턴스를 갱신하려면 다음을 수행하십시오.
   a. 루트 권한을 가진 사용자로서 로그인하십시오.
   b. 다음 명령을 실행하십시오.

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

      여기서

      DB2DIR

      DB2 설치 디렉토리

      - AIX에서 DB2 설치 디렉토리는 /usr/opt/db2_09_01입니다.
      - 다른 모든 UNIX 기반 운영 체제에서 설치 디렉토리는 /opt/ibm/db2/V9.1입니다.

      -a AuthType

      인스턴스에 대한 인증 유형을 나타냅니다. AuthType은 SERVER,

CLIENT, DCS, SERVER_ENCRYPT, DCS_ENCRYPT 중 하나가 될 수 있습니다. SERVER가 디폴트값입니다. 이 매개변수는 선택적입니다.

-u FencedID

분리(fenced) 사용자 정의 함수 및 스토어드 프로시저가 실행되는 사용자의 이름을 나타냅니다. DB2 클라이언트에서 인스턴스를 작성할 경우에는 이 플래그가 없어도 됩니다. 작성한 분리 사용자의 이름을 지정하십시오.

InstName

인스턴스의 이름을 나타냅니다. 인스턴스의 이름은 인스턴스 소유 사용자의 이름과 동일해야 합니다. 작성한 인스턴스 소유 사용자의 이름을 지정하십시오. 인스턴스는 인스턴스 소유 사용자의 홈 디렉토리에 작성됩니다.

2. db2icrt를 사용하여 인스턴스를 작성하기: 인스턴스를 작성한 후 Health Monitor용 통지를 구성하려 할 수 있습니다. 이 태스크는 Health Center 또는 CLP를 사용하여 수행할 수 있습니다. 자세한 정보는 *DB2 설치 및 구성 보충 설명서*를 참조하십시오.
   a. 루트 권한을 가진 사용자로서 로그인하십시오.
   b. 다음 명령을 실행하십시오.

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

      여기서

      DB2DIR

      DB2 설치 디렉토리

      - AIX에서 DB2 설치 디렉토리는 /usr/opt/db2_09_01입니다.
      - 다른 모든 UNIX 기반 운영 체제에서 설치 디렉토리는 /opt/ibm/db2/V9.1입니다.

      -a AuthType

      인스턴스에 대한 인증 유형을 나타냅니다. AuthType은 SERVER, CLIENT, DCS, SERVER_ENCRYPT, DCS_ENCRYPT 중 하나가 될 수 있습니다. SERVER가 디폴트값입니다. 이 매개변수는 선택적입니다.

      -u FencedID

      분리(fenced) 사용자 정의 함수 및 스토어드 프로시저가 실행되는 사용자의 이름을 나타냅니다. DB2 클라이언트에서 인스턴스를 작성할 경우에는 이 플래그가 없어도 됩니다. 작성한 분리 사용자의 이름을 지정하십시오.

InstName

인스턴스의 이름을 나타냅니다. 인스턴스의 이름은 인스턴스 소유 사용자의 이름과 동일해야 합니다. 작성한 인스턴스 소유 사용자의 이름을 지정하십시오. 인스턴스는 인스턴스 소유 사용자의 홈 디렉토리에 작성됩니다.

예를 들어, 서버 인증을 사용하고 분리 사용자가 db2fenc1이며 인스턴스 소유 사용자가 db2inst1인 경우, 다음 명령을 사용하여 AIX 시스템에서 인스턴스를 작성하십시오.

```
/usr/opt/db2_09_01/instance/db2icrt -a server -u db2fenc1 db2inst1
```

Spatial Extender 설치 검증

runGseDemo 프로그램을 실행하기 전에 다음을 수행하십시오.

- DB2 Spatial Extender 제품을 적절한 환경에 설치했는지 확인하십시오.
- 연관된 공간 데이터 관련 작업이 없는 새 데이터베이스를 사용하십시오.
- UNIX(AIX, HP-UX, Solaris 운영 환경, Intel용 Linux 및 System z용 Linux) 설치의 경우, DB2 Spatial Extender 인스턴스 환경을 설정했는지 점검하십시오. **db2ilist** 프로그램을 실행하여 인스턴스를 점검하는 방법에 대해서는 *DB2 설치 및 구성 보충 설명서*를 참조하십시오. db2start 명령을 실행하여 DB2 인스턴스를 시작할 필요가 없습니다.
- 응용프로그램 힙 크기를 증가시키십시오. 자세한 정보는 41 페이지의 『공간 데이터를 수용할 수 있도록 데이터베이스 구성』을 참조하십시오.

이 태스크는 Spatial Extender 설치 및 구성에 대한 기본 태스크 중 일부입니다.

DB2 Spatial Extender를 설치한 후에는, 데이터베이스를 작성하고 설치 확인 프로그램을 실행하여 DB2 Spatial Extender가 제대로 설치되고 구성되었는지 검증할 수 있습니다.

DB2 Spatial Extender 샘플 프로그램 runGseDemo를 사용하여 설치를 검증할 수 있습니다. runGseDemo 프로그램은 설치 문제를 진단하도록 설계되어 있습니다. 설치 검증 중에 특정 시스템 문제점을 진단하는 데 도움이 되는 오류 메시지를 수신할 수도 있습니다. 대부분의 오류 메시지는 소수의 전형적인 문제점으로 인해 발생합니다. 이러한 오류를 피하려면 ″전제조건″을 참조하십시오.

이 섹션의 검증 단계는 Windows, AIX, HP-UX, Solaris 운영 환경, Intel용 Linux 및 System z용 Linux 운영 체제에 적용됩니다.

이 태스크를 수행하려면

1. UNIX 전용: 인스턴스 소유자로서 로그온하십시오.

2. 데이터베이스를 작성하십시오. DB2 명령 창을 열고 다음을 입력하십시오.

   ```
   db2 create database mydb
   ```

   여기서 *mydb*는 데이터베이스 이름입니다.
3. 검증 프로세스 도중 오류 메시지를 수신하면 설치 문제점을 해결해야 합니다.
4. 설치 확인 프로그램을 찾으십시오.
   - UNIX 운영 체제의 경우, 다음을 입력하십시오.

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

     여기서 *$HOME*은 인스턴스 소유자의 홈 디렉토리입니다.
   - Windows의 경우, 다음을 입력하십시오.

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

     여기서 *c:¥Program Files¥IBM¥sqllib*는 DB2 Spatial Extender를 설치한 디렉토리입니다.
5. 설치 확인 프로그램을 실행하십시오. DB2 명령행에 runGseDemo 명령을 입력하십시오.

   ```
   runGseDemo mydb userID password
   ```

   여기서 *mydb*는 데이터베이스 이름입니다.

사후 설치 고려사항

Spatial Extender를 설치한 후에는 다음을 고려하십시오.

- ArcExplorer 다운로드
- 지오코더(Geocoder) 참조 데이터에 액세스

DB2용 ArcExplorer 다운로드

IBM은 IBM용으로 ESRI(Environmental Systems Research Institute)에서 작성한 브라우저를 제공합니다. 이 브라우저는 중간 데이터 서버 없이도 시각적인 DB2 Spatial Extender 데이터 쿼리 결과를 생성합니다. 이 브라우저는 DB2용 ArcExplorer입니다. 다음의 IBM Spatial Extender 웹 사이트에서 DB2용 ArcExplorer의 무료 사본을 다운로드할 수 있습니다.

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

ArcExplorer for DB2 설치 및 사용에 대한 정보는 *Using ArcExplorer*를 참조하십시오. 이 정보는 DB2 Spatial Extender 웹 사이트의 ArcExplorer for DB2 제품 다운로드 파트에 있습니다.

중요: DB2와 다른 디렉토리에 DB2용 ArcExplorer를 설치하십시오.

제 5 장 DB2 버전 9 데이터베이스 시스템으로 Spatial Extender 환경 이주

이 섹션에서는 DB2 Spatial Extender를 버전 8에서 9로 이주하는 방법에 대해 설명합니다. 또한 이주 유틸리티를 사용하여 32비트 환경에서 64비트 환경으로 이주하는 방법에 대해서도 설명합니다.

공간 사용 데이터베이스 이주

전제조건

이주 프로세스를 시작하기 전에 다음을 수행하십시오.

- 이주 유틸리티를 실행하기 전에 모든 데이터베이스 연결을 종료하십시오.
- 시스템이 DB2 Spatial Extender 버전 9.1의 설치 요구사항을 충족하는지 확인하십시오.
- 데이터베이스를 백업하려면 데이터베이스에 대해 SYSADM, SYSCTRL 또는 SYSMAINT 권한이 있어야 합니다.
- 데이터베이스를 이주시키려면 SYSADM 권한이 있어야 합니다.

DB2 Spatial Extender 버전 8을 사용 중인 경우 DB2 Spatial Extender 버전 9.1 또는 DB2 Geodetic Data Management Feature 버전 9.1에서 기존의 공간 사용 데이터베이스를 사용하기 전에 다음 단계를 완료해야 합니다. 이 주제는 이전 DB2 Spatial Extender 버전에서 공간 사용 데이터베이스를 이주하기 위해 필요한 단계에 대해 설명합니다.

이 태스크를 수행하려면

1. *DB2 이주 안내서*에 설명된 대로 DB2 서버를 버전 9.1로 이주하십시오.
2. Spatial Extender 명령 `db2se migrate`를 사용하여 버전 8에서 버전 9.1로 공간 사용 데이터베이스를 이주하십시오.

수신하는 오류에 대한 세부사항은 메시지 파일을 점검하십시오. 메시지 파일에는 인덱스, 뷰 및 이주된 지오코딩(Geocoding) 설정과 같은 유용한 정보도 들어 있습니다.

db2se migrate 명령

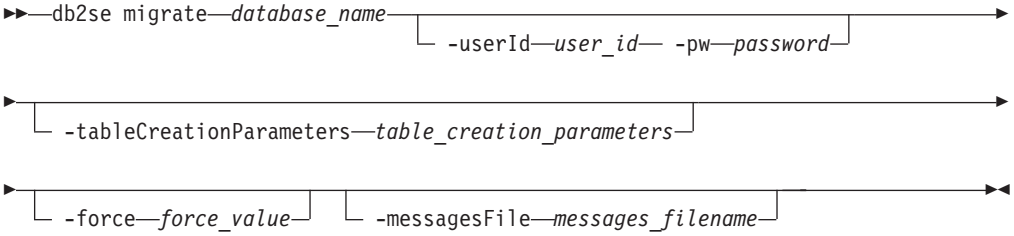
db2se migrate 명령을 사용하여 버전 8에서 버전 9로 공간 사용 데이터베이스를 이주하십시오.

© Copyright IBM Corp. 1998, 2006

이 명령은 테이블 크기에 따라 많은 시간이 걸릴 수 있는 이주를 완료하기 위해 공간 인덱스를 삭제(drop)하고 다시 작성할 수도 있습니다. 예를 들어, 데이터를 32비트 인스턴스에서 64비트 인스턴스로 이동하거나 데이터가 DB2 버전 8 Fixpack 6 또는 이전에 있는 경우 인덱스가 삭제되고 다시 작성됩니다.

팁: 추가 이주 처리를 수행하지 않고 이주해야 하는 인덱스를 찾으려면 db2se migrate 명령을 `-force 0` 옵션과 함께 실행하고 메시지 파일을 지정하십시오.

db2se migrate 명령



여기서

database_name

이주시킬 데이터베이스의 이름

user_id

이주시킬 데이터베이스에 대해 SYSADM 또는 DBADM 권한이 있는 데이터베이스 사용자 ID

password

사용자 암호

table_creation_parameters

Spatial Extender 카탈로그 테이블 작성에 사용되는 매개변수

force_value

- 0: 디폴트값. 이주를 시도하지만 응용프로그램 정의 오브젝트(뷰, 함수, 트리거 또는 공간 인덱스)가 Spatial Extender 오브젝트에 기반한 경우 중지합니다.
- 1: 응용프로그램 정의 오브젝트를 자동으로 저장하고 리스토어합니다. 필요하면 공간 인덱스를 저장하고 리스토어합니다.
- 공간 인덱스를 저장하고 리스토어합니다.
- 2: 응용프로그램 정의 오브젝트를 자동으로 저장하고 리스토어합니다. 공간 인덱스 정보를 저장하지만 자동으로 공간 인덱스를 리스토어하지 않습니다.

messages_filename

이주 조치 보고서가 있는 파일 이름. 사용자가 제공하는 파일 이름은 서버에 완전한 파일 이름이어야 합니다.

팁: 이주 문제점 해결을 위한 도움을 받으려면 이 매개변수를 지정하십시오.

제한사항: 기존 파일을 지정할 수 없습니다.

이주하는 동안 다음 오류 중 하나 이상을 수신할 수 있습니다.

- 데이터베이스가 현재 공간 데이터 관련 작업에 대해 사용 가능하지 않습니다.
- 데이터베이스 이름이 유효하지 않습니다.
- 데이터베이스와의 다른 연결이 있습니다. 실행할 수 없습니다.
- 공간 카탈로그가 일관되지 않습니다.
- 사용자에게 권한이 없습니다.
- 암호가 유효하지 않습니다.
- 몇몇 사용자 오브젝트를 이주시킬 수 없습니다.

32비트에서 64비트 환경으로 이주

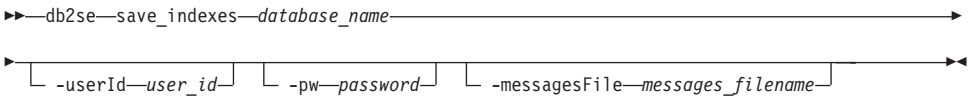
32비트 환경에서 작성된 공간 인덱스가 있고 64비트 환경으로 이주하려는 경우, 다음 단계를 완료하십시오.

32비트 버전 8 환경에서 64비트 버전 9 환경으로 이주할 경우, db2se 이주 명령이 인덱스를 이주하므로 이 태스크를 수행하지 않아도 됩니다.

이 태스크를 수행하려면 다음을 수행하십시오.

1. 데이터베이스를 백업하십시오.
2. 운영 체제 명령 프롬프트에서 **db2se save_indexes** 명령을 입력하여 정의되는 공간 인덱스를 저장하십시오.

db2se save_indexes 명령



여기서,

database_name
이주시킬 데이터베이스의 이름

user_id
이주시킬 데이터베이스에 대해 SYSADM 또는 DBADM 권한이 있는 데이터베이스 사용자 ID

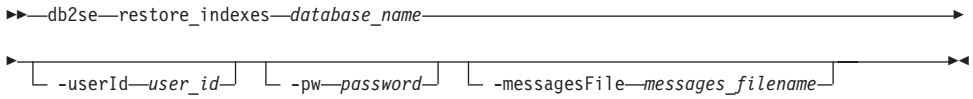
password
사용자 암호

messages_filename

이주 조치 보고서가 있는 파일 이름. 사용자가 제공하는 파일 이름은 서버에 완전한 파일 이름이어야 합니다.

3. V8 데이터베이스를 32비트에서 64비트 환경으로 이주하십시오. 32비트 환경에서 64비트 환경으로의 전환에 대한 자세한 정보는 *DB2 이주 안내서*를 참조하십시오.
4. 운영 체제 프롬프트에서 **db2se restore_indexes** 명령을 사용하여 공간 인덱스를 리스토어하십시오.

db2se restore_indexes 명령



여기서,

database_name

이주시킬 데이터베이스의 이름

user_id

이주시킬 데이터베이스에 대해 SYSADM 또는 DBADM 권한이 있는 데이터베이스 사용자 ID

password

사용자 암호

messages_filename

이주 조치 보고서가 있는 파일 이름. 사용자가 제공하는 파일 이름은 서버에 완전한 파일 이름이어야 합니다.

제 6 장 데이터베이스 설정

이 장에서는 공간 데이터를 수용하기 위해 데이터베이스를 구성하는 방법에 대해 논의됩니다.

공간 데이터를 수용할 수 있도록 데이터베이스 구성

이 주제는 DB2 Spatial Extender 작업에 영향을 주는 DB2 구성 매개변수를 식별합니다.

DB2 데이터베이스 환경에서 실행되는 DB2 Spatial Extender는 대부분의 디폴트 DB2 구성 값으로 작동합니다. 그러나 몇몇 구성 매개변수는 공간 데이터 관련 작업에 영향을 미칩니다. 공간 응용프로그램이 가능한 한 효율적으로 기능할 수 있도록 하려면 이러한 매개변수를 조정해야 합니다. 임의 데이터베이스에 대해 이러한 매개변수의 값을 수정하면, 변경사항은 해당 데이터베이스에만 영향을 미칩니다. 어떤 경우에는 공간 데이터 관련 작업을 수행하는 데 디폴트 값 이외의 값이 요구됩니다. 또한 사용하는 응용프로그램 및 전체 DB2 환경에 따라 디폴트 값 이외의 값을 선택하는 것이 바람직한 경우도 있습니다.

다음 절에서는 DB2 Spatial Extender 조작에 영향을 미치는 DB2 데이터베이스 관리 프로그램 및 데이터베이스 구성 매개변수를 조정하는 방법에 대해 설명합니다.

트랜잭션 로그 특성 조정

공간 데이터 관련 작업에 대해 데이터베이스를 사용하려면 먼저 트랜잭션 로그 용량이 충분한지 확인해야 합니다. 계획에 다음이 포함될 경우, 트랜잭션 로그 구성 매개변수의 디폴트값은 충분한 트랜잭션 로그 용량을 제공하지 못합니다.

- Windows 환경에서 공간 데이터 관련 작업에 대해 데이터베이스 사용 가능화
- ST_import_shape 스토어드 프로시저를 사용하여 모양 파일로부터 임포트
- 커미트 범위가 큰 지오코딩(Geocoding) 사용
- 동시 트랜잭션 실행

지금 또는 차후에 위의 항목 중 어느 것이라도 계획에 포함되면, 하나 이상의 트랜잭션 로그 구성 매개변수를 증가시켜 데이터베이스에 대한 트랜잭션 로그의 용량을 증가시켜야 합니다. 그렇지 않은 경우에는 디폴트 특성을 사용해도 됩니다.

권장사항: 세 개의 트랜잭션 로그 구성 매개변수에 대해 권장되는 최소값은 다음 테이블을 참조하십시오.

© Copyright IBM Corp. 1998, 2006

표 2. 트랜잭션 구성 매개변수에 대한 권장 최소값

매개변수	설명	디폴트값	권장 최소값
LOGFILSIZ	로그 파일 크기를 4KB 블록 수로 지정합니다.	1000	1000
LOGPRIMARY	복구 로그 파일에 사전 할당되는 1차 로그 파일 수를 지정합니다.	3	10
LOGSECOND	2차 로그 파일의 수를 지정합니다.	2	2

트랜잭션 로그 용량이 부적절하면, 공간 데이터 관련 작업에 대한 데이터베이스를 사용하려고 할 때 다음 오류 메시지가 발행됩니다.

GSE0010N DB2에 사용 가능한 로그 스페이스가 충분하지 않습니다.

하나 이상의 구성 매개변수 값을 증가시키려면 다음을 수행하십시오.

1. GET DATABASE CONFIGURATION 명령을 실행하여 LOGFILSIZ, LOGPRIMARY 및 LOGSECOND 매개변수의 현재 값을 찾거나 DB2 제어 센터의 **데이터베이스 구성** 창을 보십시오.
2. 위 테이블에 표시된 대로 하나, 둘 또는 세 개의 값을 변경할 것인지 결정하십시오.
3. 수정하고자 하는 각 값을 변경하십시오. 다음 명령 중 하나 이상을 발행하여 값을 변경할 수 있습니다. 여기서 *db_name*은 데이터베이스를 식별합니다.

```
UPDATE DATABASE CONFIGURATION FOR db_name USING LOGFILSZ 1000

UPDATE DATABASE CONFIGURATION FOR db_name USING LOGPRIMARY 10

UPDATE DATABASE CONFIGURATION FOR db_name USING LOGSECOND 2
```

변경하는 유일한 매개변수가 LOGSECOND이면 변경사항이 즉시 적용됩니다.

LOGFILSIZ, LOGPRIMARY 또는 둘 다를 변경할 경우에는 다음을 수행하십시오.

1. 데이터베이스로부터 모든 응용프로그램의 연결을 끊으십시오.
2. 데이터베이스를 명시적으로 활성화한 경우에는 데이터베이스를 비활성화하십시오.

LOGFILSIZ, LOGPRIMARY 또는 둘 다에 대한 변경사항은 다음 번에 데이터베이스를 활성화하거나 데이터베이스와의 연결을 설정할 때 적용됩니다.

응용프로그램 힙 크기 조정

데이터베이스 구성 매개변수 APPLHEAPSZ를 사용하여 응용프로그램 힙 크기(4KB 페이지 수)를 지정합니다. 이 매개변수는 데이터베이스 관리 프로그램이 특정 에이전트 또는 서브에이전트를 대신하여 사용할 수 있는 개인용 메모리 페이지 수를 정의합니다. 힙은 에이전트 또는 서브에이전트가 응용프로그램에 대해 초기화될 때 할당됩니다. 할당되는 양은 에이전트 또는 서브에이전트에 대한 요청을 처리하는 데 필요한 최소량입

니다. 에이전트 또는 서브에이전트는 보다 큰 SQL문을 처리하기 위해 보다 많은 힙 스페이스를 필요로 하므로, 데이터베이스 관리 프로그램은 요구되는 메모리를 이 매개변수에 지정된 최대값까지 할당합니다. 응용프로그램 힙은 에이전트의 개인용 메모리로부터 할당됩니다.

APPLHEAPSZ 매개변수의 디폴트값은 128(4KB 페이지)입니다. ST_enable_db 스토어드 프로시저를 실행할 때, 이 값은 2048이상이어야 합니다.

권장사항: 대부분의 DB2 Spatial Extender 응용프로그램에서 특히 모양 파일에서 임포트 또는 익스포트하는 경우 2048 이상의 APPLHEAPSZ 매개변수 값을 사용하십시오.

APPLHEAPSZ가 부적절한 값으로 설정되면, 공간 데이터 관련 작업에 대한 데이터베이스를 사용하려고 할 때 다음 오류 메시지가 발행됩니다.

```
GSE0009N DB2 응용프로그램 힙의 공간이 충분하지 않습니다.

GSE0213N  바인드 조작이 실패했습니다.
SQLERROR = "SQL0001N 바인딩 또는 프리컴파일이 성공적으로
           완료되지 않았습니다. SQLSTATE=00000".
```

응용프로그램 힙 크기를 변경하려면 다음을 수행하십시오.

1. GET DATABASE CONFIGURATION 명령을 실행하여 APPLHEAPSZ 매개변수의 현재 값을 찾거나 DB2 제어 센터의 **데이터베이스 구성** 창을 보십시오.
2. 값을 권장값인 2048 이상으로 변경하십시오. 다음 명령을 발행하여 값을 2048로 변경할 수 있습니다. 여기서 *db_name*은 데이터베이스를 식별합니다.

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

3. 데이터베이스로부터 모든 응용프로그램의 연결을 끊으십시오.
4. 데이터베이스를 명시적으로 활성화한 경우에는 데이터베이스를 비활성화하십시오.

변경사항은 다음에 데이터베이스를 활성화하거나 데이터베이스와의 연결을 설정할 때 적용됩니다.

응용프로그램 제어 힙 크기 조정

데이터베이스 구성 매개변수를 수정하려면 데이터베이스에 연결되어 있어야 합니다. 데이터베이스 매개변수 값을 수정하면 해당 데이터베이스에만 영향을 미칩니다.

모든 DB2 Spatial Extender 응용프로그램의 경우, 특히 모양 파일에서 임포트하거나 모양 파일로 익스포트하는 응용프로그램은 응용프로그램 제어 힙 크기에 권장값을 사용하는 것이 유익합니다. 이 특성은 APP_CTL_HEAP_SZ 매개변수로 지정합니다. 이 매개변수는 응용프로그램 제어 공유 메모리의 최대값(4KB 페이지 단위)을 지정합니다. 응용프로그램 제어 힙은 이 공유 메모리로부터 할당됩니다. 응용프로그램이 활동 중인 데이터베이스에서(파티션된 데이터베이스 시스템의 경우, 응용프로그램이 활동 중인 각

데이터베이스 파티션에서) 응용프로그램당 하나의 응용프로그램 제어 힙이 할당됩니다. 데이터베이스에 있는 응용프로그램에 대한 요청을 수신하는 첫 번째 에이전트에 의해 연결 처리 중에 힙이 할당됩니다. 힙은 동일한 응용프로그램을 대신하여 작업을 수행하는 에이전트 간에 정보를 공유하는 데 사용됩니다. (파티션된 데이터베이스 환경에서는 데이터베이스 파티션 레벨에서 공유가 발생하고 데이터베이스 파티션 간에는 공유가 발생하지 않습니다.) APP_CTL_HEAP_SZ 매개변수의 디폴트값은 128입니다.

권장사항: 대부분의 DB2 Spatial Extender 응용프로그램에서 1024(4KB 페이지) 이상의 APP_CTL_HEAP_SZ 매개변수 값을 사용하십시오.

APP_CTL_HEAP_SZ가 부적절한 값으로 설정되면, 모양 파일에서 데이터베이스로 데이터가 임포트될 때 다음 오류 메시지가 발행됩니다.

```
GSE0214N INSERT문이 실패했습니다.
SQLERROR = "SQL0973N 명령문을 처리하기 위해 "APP_CTL_HEAP" 힙에
           사용 가능한 스토리지가 충분하지 않습니다."
```

1. GET DATABASE CONFIGURATION 명령을 실행하여 APP_CTL_HEAP_SZ 매개변수의 현재 값을 찾거나 DB2 제어 센터의 **데이터베이스 구성** 창을 보십시오.
2. 값을 권장값 1024(4KB 페이지) 이상으로 변경하십시오. 다음 명령을 실행하십시오. 여기서 *db_name*은 응용프로그램 제어 힙을 1024KB로 늘릴 데이터베이스를 식별합니다.

   ```
   UPDATE DATABASE CONFIGURATION FOR db_name USING APP_CTL_HEAP_SZ 1024
   ```

3. 데이터베이스로부터 모든 응용프로그램의 연결을 끊으십시오.
4. 데이터베이스를 명시적으로 활성화한 경우에는 데이터베이스를 비활성화하십시오.

변경사항은 다음에 데이터베이스를 활성화하거나 데이터베이스와의 연결을 설정할 때 적용됩니다.

제 7 장 데이터베이스용 공간 자원 설정

공간 데이터를 수용하기 위해 데이터베이스를 설정했으면, 공간 컬럼을 작성 및 관리하고 공간 데이터를 분석할 때 필요한 자원을 데이터베이스에 제공할 준비가 되었습니다. 이러한 자원에는 다음이 포함됩니다.

- 공간 데이터 관련 작업을 지원하기 위해 Spatial Extender에서 제공하는 오브젝트(예 : 데이터베이스를 관리하기 위한 스토어드 프로시저, 공간 데이터 유형, 공간 데이터를 지오코드 및 임포트 또는 익스포트하기 위한 공간 유틸리티)
- 참조 데이터: DB2SE_USA_GEOCODER가 개별 주소를 좌표로 변환하는 데 사용하는 주소 범위입니다.
- 사용자가 벤더가 제공하는 모든 지오코더(Geocoder)

이 장에서는 이러한 자원에 대해 설명하고 이러한 자원을 사용 가능하게 만드는 태스크(공간 조작을 위한 데이터베이스 사용, 참조 데이터에 대한 액세스 설정 및 디폴트가 아닌 지오코더(Geocoder) 등록)를 소개합니다.

데이터베이스에서 자원 설정 방법

공간 데이터를 수용할 데이터베이스를 설정한 후 수행하는 첫 번째 태스크는 공간 작업(공간 데이터로 테이블을 채우고 공간 쿼리를 처리하는 작업)을 지원할 수 있도록 데이터베이스를 렌더링하는 것입니다. 이 태스크에는 DB2 Spatial Extender에서 제공하는 특정 자원을 사용한 데이터베이스 로드가 포함됩니다. 이 절에서는 이러한 자원을 설명하고 태스크에 대해 개괄적으로 설명합니다.

데이터베이스에 제공되는 자원 재고

데이터베이스에서 공간 작업을 지원하도록 하기 위해 DB2® Spatial Extender는 데이터베이스에 다음 자원을 제공합니다.

- 스토어드 프로시저. 공간 조작을 요청할 경우(예: 공간 데이터를 임포트하는 명령을 실행할 경우) DB2 Spatial Extender는 다음 스토어드 프로시저 중 하나를 호출하여 조작을 수행합니다.
- 공간 데이터 유형. 공간 데이터가 저장될 각 테이블 또는 뷰에 공간 데이터 유형을 지정해야 합니다.
- DB2 Spatial Extender의 카탈로그. 특정 조작은 이 카탈로그에 종속됩니다. 예를 들어, 시각화 도구에서 공간 컬럼에 액세스하려면 도구에서 공간 컬럼을 카탈로그에 등록해야 합니다.
- 공간 격자 인덱스. 공간 컬럼에 대한 격자 인덱스를 정의할 수 있도록 합니다.

© Copyright IBM Corp. 1998, 2006

- 공간 함수. 이를 사용하여 여러 가지 면에서 공간 데이터에 대해 작업할 수 있습니다. 예를 들어, Geometry 사이의 관계를 판별하고 공간 데이터를 추가로 생성하기 위해 작업할 수 있습니다.
- 좌표 시스템 정의.
- 디폴트 공간 참조 시스템.
- 두 개의 스키마 DB2GSE 및 ST_INFORMTN_SCHEMA. DB2GSE에는 위에 나열된 오브젝트(스토어드 프로시저, 공간 데이터 유형, DB2 Spatial Extender 카탈로그 등)가 포함됩니다. 카탈로그에 있는 뷰는 ST_INFORMTN_SCHEMA에서도 사용 가능하며 SQL/MM 표준을 따릅니다.

공간 작업에 데이터베이스 사용

공간 데이터 관련 작업에 대해 데이터베이스를 사용하려면 사용자 ID에 해당 데이터베이스에 대한 SYSADM 또는 DBADM 권한이 있어야 합니다.

다음 방법 중 하나를 사용하여 공간 데이터 관련 작업에 대해 데이터베이스를 사용 가능화할 수 있습니다.

- DB2 Spatial Extender 메뉴 옵션에 있는 데이터베이스 사용 창을 사용하십시오. 이 메뉴 옵션은 DB2 제어 센터의 데이터베이스 오브젝트에 있습니다.
- db2se enable_db 명령을 실행하십시오.
- db2gse.ST_enable_db 스토어드 프로시저를 호출하는 응용프로그램을 실행하십시오.

DB2 Spatial Extender 카탈로그가 상주할 테이블 스페이스를 명시적으로 선택할 수 있습니다. 명시적으로 선택하지 않으면 DB2가 디폴트 테이블 스페이스를 사용합니다.

DB2 Spatial Extender가 공간 컬럼을 작성하고 공간 데이터를 조작하는 데 필요한 자원을 데이터베이스에 제공하도록 하는 태스크를 『공간 데이터 관련 작업에 대해 데이터베이스 사용 가능화』라고 합니다.

참조 데이터에 대한 작업 방법

이 절에서는 참조 데이터를 설명하고 이 데이터에 액세스하기 위해 수행해야 하는 내용에 대해 나와 있습니다.

참조 데이터

참조 데이터는 DB2SE_USA_GEOCODER가 개별 주소를 좌표로 변환하는 데 사용하는 주소 범위입니다. 이 데이터는 United States Census Bureau에서 수집한 가장 최근의 주소 범위로 구성됩니다. DB2SE_USA_GEOCODER는 데이터베이스에서 하나의 주소를 읽을 때 참조 데이터를 검색하여 다음을 찾습니다.

- 해당 주소의 우편번호가 지정하는 영역 내에 있는 특정 거리의 이름. 지오코더(Geocoder)는 주소에 있는 거리의 이름과 지정된 등급 이상으로(예: 80퍼센트 이상) 일치하는 이름을 찾습니다.
- 주소 번호에 해당하는 주소 범위

일치하는 주소는 있으나 필수 스코어는 없을 경우 지오코더(Geocoder)는 읽은 주소의 좌표를 리턴합니다. 일치하는 주소도 없고 필수 스코어도 없을 경우 지오코더(Geocoder)는 널(NULL)을 리턴합니다.

*로케이터 파일*이라고 하는 고급 구성은 지오코더(DB2SE_USA_GEOCODER)가 수행하는 처리 옵션을 자세히 구성하는 데 사용됩니다. DB2® Spatial Extender에서 제공하는 디폴트 구성은 일반적으로 이 파일에서 변경하지 않아도 됩니다.

참조 데이터에 대한 액세스 설정

DB2SE_USA_GEOCODER의 참조 데이터는 Spatial Extender가 제공하는 CD 중 하나에 있습니다. 이 절에서는 참조 데이터에 대한 액세스를 준비하는 방법에 대해 설명합니다.

이 태스크를 수행하려면

1. 참조 데이터를 CD에 보존할 것인지 또는 하드 드라이브에 저장할 것인지 여부를 결정하십시오. CD에 보존하면 하드 드라이브에서 차지하는 스페이스(약 700MB)를 절약할 수 있습니다. 하드 드라이브에 저장하는 경우, CD에서 검색하는 것보다 빨리 검색할 수 있습니다.
2. 하드드라이브에 참조 데이터 저장
   a. 하드 드라이브에 데이터를 저장할 충분한 스페이스가 있는지 확인하십시오(약 700MB).
   b. 데이터를 하드 드라이브로 복사하십시오. 지시사항은 참조 데이터와 함께 제공되는 README 파일을 참조하십시오.
   c. 복사가 완료되었는지 판별하십시오. UNIX에서 데이터가 올바르게 로드되었는지 확인하려면 $DB2INSTANCE/sqllib/gse/refdata/ 디렉토리를 찾아 보십시오. Windows에서 데이터가 올바르게 로드되었는지 확인하려면 디렉토리 %DB2PATH%₩gse₩refdata₩에서 찾아 보십시오.
3. DB2SE_USA_GEOCODER에 기본 맵과 로케이터 파일의 이름 및 위치를 알리십시오. DB2SE_USA_GEOCODER의 base_map 및 locator_file 매개변수를 적절한 값으로 설정함으로써 이를 수행할 수 있습니다. 자세한 정보는 데이터베이스 관리자 또는 IBM 담당자에게 문의하십시오.

지오코더(Geocoder) 등록

지오코더(Geocoder)를 등록하려면 사용자 ID에 지오코더(Geocoder)가 상주하는 데이터베이스에 대한 SYSADM 또는 DBADM 권한이 있어야 합니다.

공간 데이터 관련 작업에 대해 데이터베이스가 사용 가능할 때 DB2SE_USA_GEOCODER가 자동으로 DB2 Spatial Extender에 등록됩니다. 다른 지오코더(Geocoder)를 사용하려면 이들 역시 등록해야 합니다.

이 태스크를 수행하려면

다음 방법 중 하나로 지오코더를 등록할 수 있습니다.

- DB2 제어 센터의 지오코더(Geocoder) 등록 창에서 지오코더(Geocoder)를 등록하십시오.
- db2se register_gc 명령을 실행하십시오.
- db2gse.ST_register_geocoder 스토어드 프로시저를 호출하는 응용프로그램을 실행하십시오.

제 8 장 프로젝트용 공간 자원 설정

공간 데이터 관련 작업에 대해 데이터베이스를 사용한 후에는 공간 데이터를 사용하는 프로젝트를 작성할 준비가 되어 있습니다. 각 프로젝트에서 필요로 하는 자원 중 하나는 공간 데이터가 준수하는 좌표 시스템과 데이터별로 참조되는 지리적 영역의 범위를 정의하는 공간 참조 시스템입니다. 이 장에서는 다음 사항에 대해 다룹니다.

- 좌표 시스템의 특성에 대해 논의하며 작성 방법을 알려줍니다.
- 공간 좌표 시스템에 대해 설명하며 작성 방법을 알려줍니다.

좌표 시스템 사용법

공간 데이터를 사용하는 프로젝트를 계획할 때, 데이터가 Spatial Extender 카탈로그에 등록된 좌표 시스템 중 하나를 따라야 하는지 여부를 판별해야 합니다. 사용자 요구사항에 만족하는 좌표 시스템이 없는 경우, 이러한 좌표 시스템을 작성할 수 있습니다. 여기에서는 좌표 시스템의 개념을 설명하며, 사용할 좌표 시스템을 선택하고 새 좌표 시스템을 작성하는 태스크를 소개합니다.

좌표 시스템

좌표 시스템은 제공된 영역(예: 지구 표면의 한 영역 또는 지구 표면 전체)에서 사물의 상대적 위치를 정의하는 프레임워크입니다. DB2® Spatial Extender는 지리 대상물의 위치를 판별하기 위해 다음 유형의 좌표 시스템을 지원합니다.

지리 좌표 시스템

지리 좌표 시스템은 3차원 구면을 사용하여 지구 상의 위치를 판별하는 참조 시스템입니다. 지구 상의 모든 위치는 각도 측정 단위를 기반으로 위도와 경도 좌표를 사용하여 점으로 나타낼 수 있습니다.

투영 좌표 시스템

*투영 좌표 시스템*은 지구를 평평한 2차원으로 표현한 것으로서 이것은 선형 측정 단위를 기반으로 하는 직선(데카르트) 좌표를 사용합니다. 이 시스템은 구형(또는 회전 타원체) 지구 모델을 기반으로 하며 해당 좌표는 투영 변환에 의한 지리 좌표와 관련이 있습니다.

지리 좌표 시스템

*지리 좌표 시스템*은 3차원 구면을 사용하여 지구 상의 위치를 판별하는 시스템입니다. 지구 상의 모든 위치는 경도 및 위도 좌표의 점으로 나타낼 수 있습니다. 점 값의 측정 단위는 다음과 같습니다.

© Copyright IBM Corp. 1998, 2006

- 지리 좌표 시스템에 DB2® Geodetic Data Management Feature에서 인식하는 공간 참조 시스템 ID(SRID)가 있는 경우 선형 단위가 사용됩니다.
- 지리 좌표 시스템에 DB2 Geodetic Data Management Feature에서 인식하지 못하는 SRID가 있는 경우는 다음 단위가 사용됩니다.
  - 10진수(도)
  - 10진수 분
  - 10진수 초
  - 기울기
  - 라디안

예를 들어, 그림 6에는 동경 80도와 북위 55도의 좌표로 위치가 표현되는 지리 좌표 시스템이 표시됩니다.

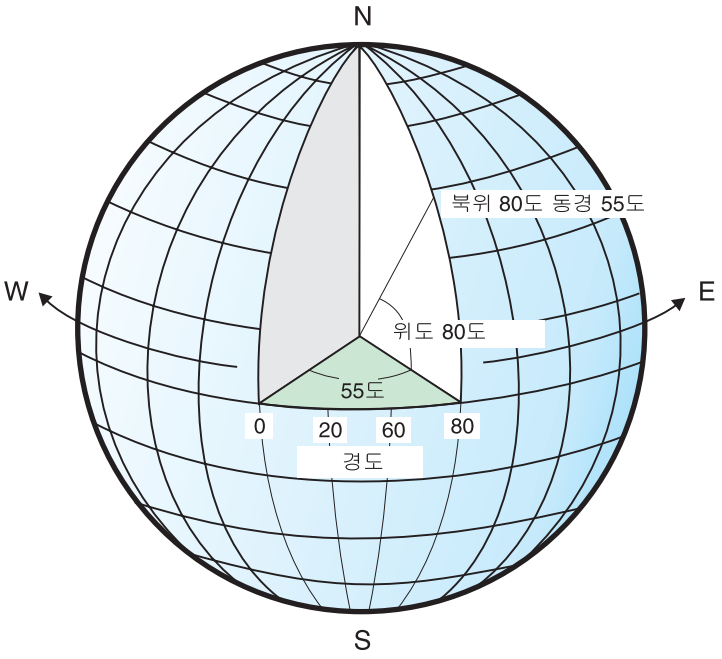


그림 6. 지리 좌표 시스템

동서 가로 방향의 각 선은 일정한 위도 값을 가지며 *위선*이라고 합니다. 이는 서로 동일한 간격으로 평행을 이루며 지구 둘레에 동심원을 형성합니다. *적도*는 가장 큰 원으로서 지구를 반으로 나눕니다. 적도는 양극까지의 거리가 동일하며 위선 값은 0입니다. 적도 북쪽은 0 - +90도 범위의 양의 위도이며 적도 남쪽은 0 - -90도 범위의 음의 위도입니다.

51 페이지의 그림 7은 위선을 보여줍니다.

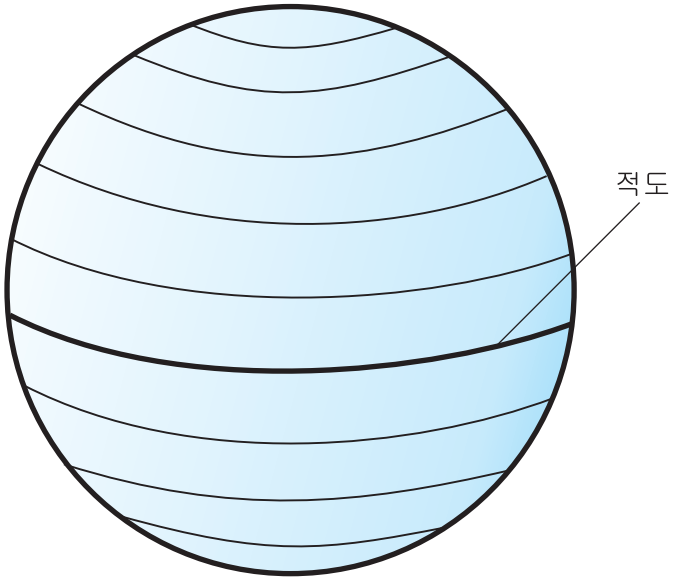


그림 7. 위선

남북 세로 방향의 각 선은 일정한 경도 값을 가지며 *경선*이라고 합니다. 이는 지구 둘레에 동일한 *크기*의 원을 형성하며 북극과 교차합니다. *본초 자오선*은 경도 좌표의 원점(0도)을 정의하는 경선입니다. 가장 일반적으로 사용되는 본초 자오선 위치 중 하나는 영국 그리니치를 통과하는 선입니다. 그러나 베른, 보고타 및 파리를 통과하는 다른 경선도 본초 자오선으로 사용됩니다. 본초 자오선 동쪽에서 *정반대* 경선까지의 위치(지구 반대편의 본초 자오선 연속)는 0 - +180도 범위의 양의 경도를 표시합니다. 본초 자오선 서쪽의 위치는 0 - -180도 범위의 음의 경도로 표시합니다.

52 페이지의 그림 8은 경선을 보여줍니다.

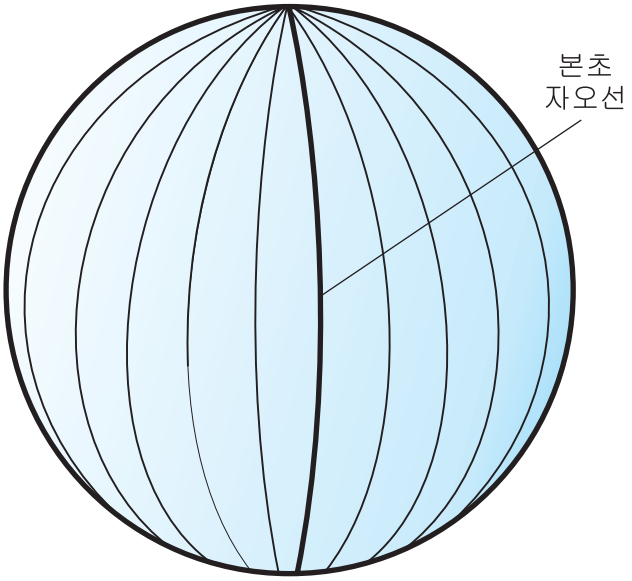


그림 8. 경선

경선과 위선은 지구를 둘러싸는 *경위선*이라는 격자를 형성합니다. 경위선의 원점은 (0,0)이며 이 지점이 적도와 본초 자오선이 교차하는 지점입니다. 적도는 위도 1도에 해당하는 선형 거리가 경도 1도에 해당하는 거리와 거의 동일한 유일한 위치입니다. 경선은 극 지점에서 모이므로 두 경선 간의 거리는 위선마다 다릅니다. 따라서 극 지점에 가까이 갈수록 위도 1도에 해당하는 거리는 경도 1도에 해당하는 거리보다 훨씬 커집니다.

또한 경위선을 사용하여 위선의 길이를 판별하기도 어렵습니다. 위선은 북극으로 갈수록 작아지는 동심원입니다. 위선은 자오선이 시작되는 북극에서 하나의 점을 이룹니다. 적도에서 경도 1도의 거리는 약 111.321km인 반면, 위도 60도에서 경도 1도는 55.802km에 불과합니다. (이 거리 근사값은 Clarke 1866 장구(spheroid)를 기반으로 합니다.) 그러므로 경도와 위도의 각 길이가 일정하지 않으므로 각도 측정 단위로는 점 간의 거리를 정확하게 측정할 수 없습니다.

53 페이지의 그림 9에서는 경위선 상에 있는 위치들 간의 다양한 치수를 보여줍니다.

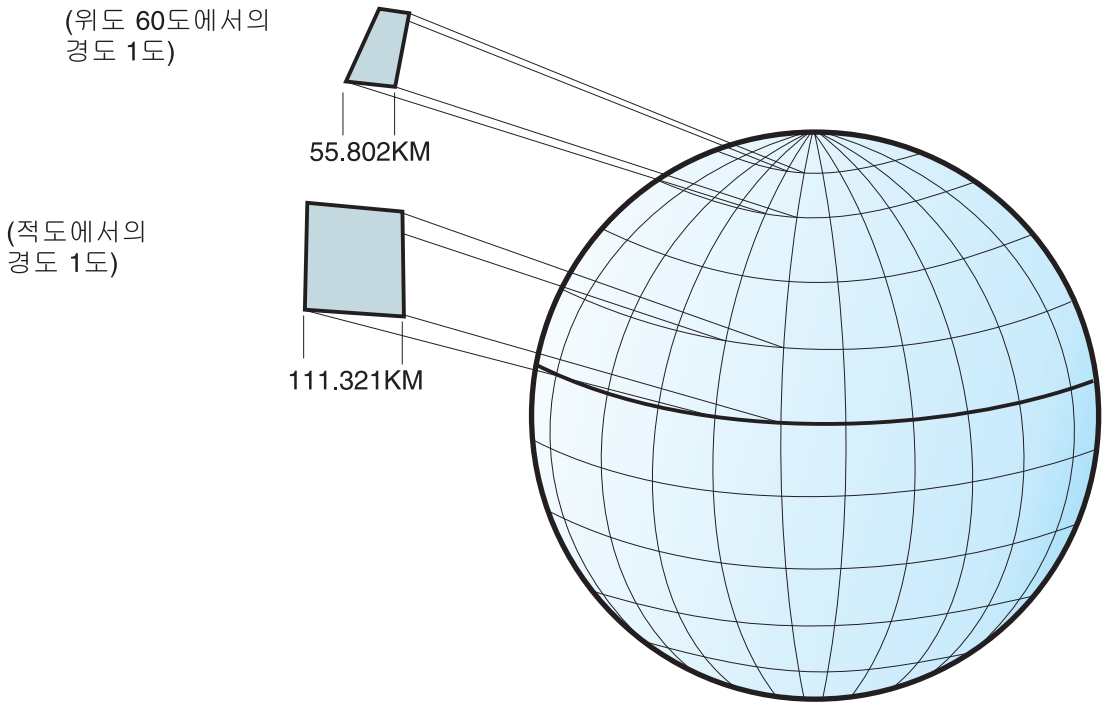


그림 9. 경위선 상에 있는 위치들 간의 다양한 차원

지구 모양과 근사한 구 또는 회전 타원체로 좌표 시스템을 정의할 수 있습니다. 지구는 완전한 구형이 아니므로 회전 타원체를 사용하면 지구 상의 위치에 따라 지도의 정확성을 유지하는 데 도움이 됩니다. *회전 타원체*는 타원에 기초한 타원체인 반면, 구는 원을 기초로 합니다.

타원의 모양은 두 개의 반지름으로 결정됩니다. 긴 반지름을 반장축이라 하고 짧은 반지름을 반단축이라 합니다. 타원체는 중심 축을 따라 타원을 회전시켜 형성되는 3차원 모양입니다.

54 페이지의 그림 10에서는 지구와 유사한 구 및 회전 타원체와 타원체의 장축 및 단축을 보여줍니다.

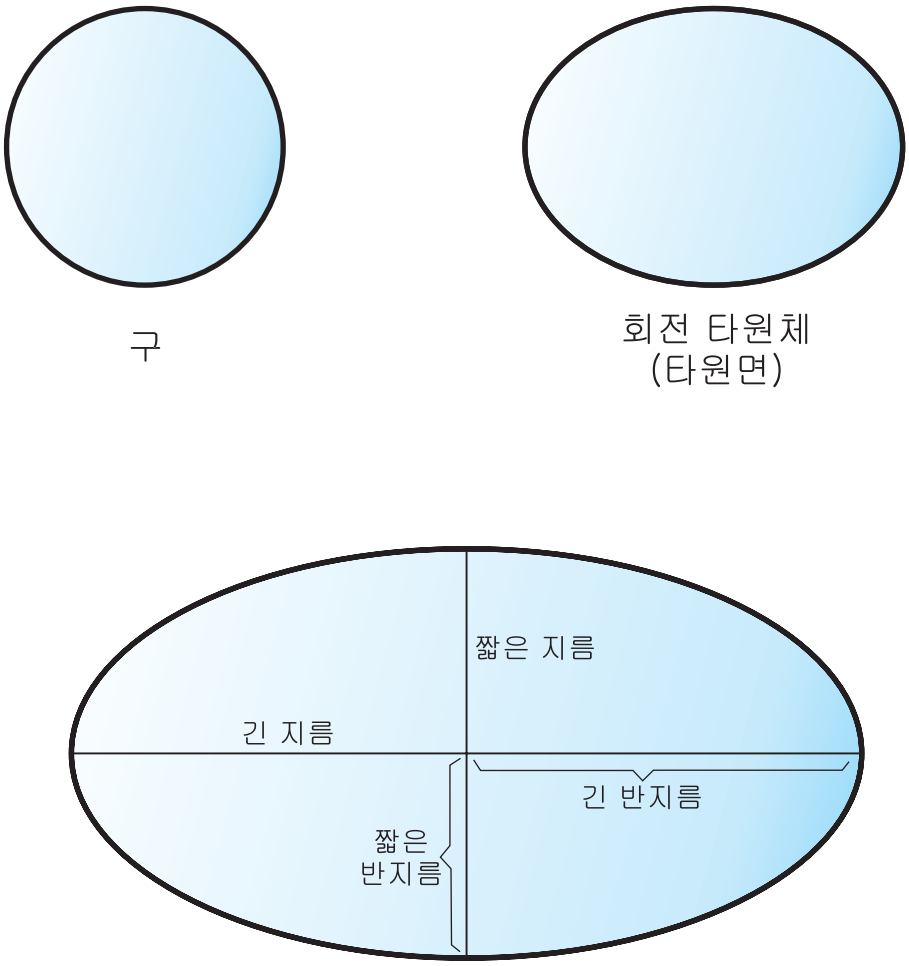


그림 10. 지구와 근사한 구 및 회전 타원체

*데이텀(Datum)*은 지구 중심에 상대적인 회전 타원체의 위치를 정의하는 값 세트입니다. 기준면은 위치 측정을 위한 참조 프레임을 제공하며 위선 및 경선의 원점과 방향을 정의합니다. 일부 기준면은 전역이며 전세계 어디에서나 평균 정확도를 제공합니다. 로컬 기준면은 해당 회전 타원체를 특정 영역의 지구 표면에 맞춥니다. 그러므로 좌표 시스템 측정치는 설계된 영역 이외의 영역에서 사용되면 정확하지 않습니다.

55 페이지의 그림 11에서는 다양한 데이텀이 지구 표면과 정렬되는 방법을 보여줍니다. 로컬 기준면인 NAD27은 이 특정 위치에서 지구 중심 기준면인 WGS84보다 지구 표면에 보다 가깝게 정렬됩니다.

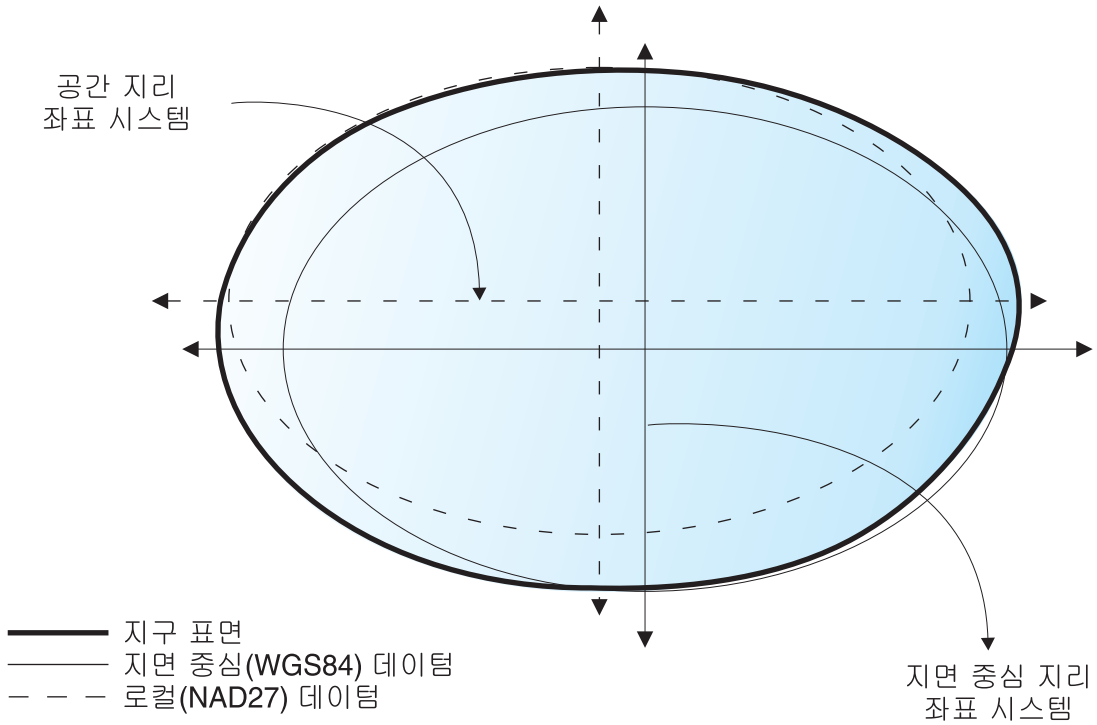


그림 11. 기준면 정렬

기준면을 변경하면 지리 좌표 시스템이 변경되고 좌표 값도 바뀝니다. 예를 들어, NAD(North American Datum) 1983을 사용할 경우 California Redlands에 있는 한 제어점의 DMS 좌표는 ″-117 12 57.75961 34 01 43.77884″입니다. NAD 1927에서 동일한 지점의 좌표는 ″-117 12 54.61539 34 01 43.72995″입니다.

투영 좌표 시스템

*투영 좌표 시스템*은 지구를 평평한 2차원으로 표현한 것입니다. 구 또는 회전 타원체 지리 좌표 시스템을 기초로 하지만 좌표에 선형 측정 단위를 사용하므로 거리 및 면적을 동일한 단위로 쉽게 계산할 수 있습니다.

위도 및 경도 좌표는 평면 투영에서는 x 및 y 좌표로 변환됩니다. x 좌표는 보통 한 점의 동쪽 방향이고 y 좌표는 보통 한 점의 북쪽 방향입니다. 동서로 이동하는 중심선을 x 축이라고 하고 남북으로 이동하는 중심선을 y 축이라고 합니다.

x 축과 y 축의 교차점이 원점이며 일반적으로 좌표는 (0,0)입니다. x 축 위에 있는 값은 양수값을 가지고 x 축 아래에 있는 값은 음수값을 가집니다. x 축에 평행한 선은 거리가 서로 같습니다. y 축 오른쪽에 있는 값은 양수값을 가지고 y 축 왼쪽에 있는 값은 음수값을 가집니다. y 축과 평행인 선은 거리가 서로 같습니다.

3차원의 지리 좌표 시스템을 2차원의 평면 투영 좌표 시스템으로 변환하는 데에는 수학적 공식이 사용됩니다. 이러한 변환을 *지도 투영*이라고 합니다. 지도 투영은 주로 사용되는 투영 표면(예: 원뿔, 원통 및 평면)별로 분류됩니다. 사용되는 투영에 따라 다양

한 공간 등록 정보들이 왜곡되어 나타납니다. 투영은 데이터 특성 중 하나 또는 둘의 왜곡을 최소화하도록 설계되었으나, 거리, 영역, 모양, 방향 또는 이러한 등록 정보의 조합이 여전히 모델링되는 데이터를 정확히 표현하지 못할 수도 있습니다. 사용할 수 있는 여러 프로젝션 유형이 있습니다. 대부분의 지도 투영이 공간 등록 정보의 정확도를 어느 정도 보존하려고 하는 반면, *Robinson* 투영과 같이 전체적 왜곡을 최소화하려는 투영도 있습니다. 가장 일반적인 유형의 지도 투영은 다음과 같습니다.

등적 프로젝션

이 투영은 특정 대상물의 영역을 보존합니다. 이 투영은 모양, 각도 및 스케일을 왜곡합니다. *Albers Equal Area Conic* 투영이 등적 투영의 한 예입니다.

등각 프로젝션

이 투영은 작은 영역의 지역 모양을 보존합니다. 이 투영은 지도에서 90도 각도로 교차하는 수직 격자선을 표시함으로써 공간 관계를 설명하는 개별 각도를 보존합니다. 모든 각도가 보존되나, 지도의 영역이 왜곡됩니다. *Mercator* 및 *Lambert Conformal Conic* 투영은 등각 투영의 예입니다.

등거리 프로젝션

이 투영은 제공된 데이터 세트의 스케일을 유지함으로써 특정 지점 간의 거리를 보존합니다. 일부 거리가 실제 거리(지구와 동일한 스케일에서 동일한 거리)가 됩니다. 데이터 세트를 벗어나면 스케일은 더욱 왜곡됩니다. *Sinusoidal* 투영 및 *Equidistant Conic* 투영이 정거 투영의 예입니다.

정방향 또는 방위 프로젝션

이 투영은 대원호의 일부를 유지함으로써 한 지점에서 다른 모든 지점으로의 방향을 보존합니다. 이 투영은 중심과 관련하여 지도에 있는 모든 지점의 방향 또는 방위를 올바르게 제공합니다. 방위 지도는 등적, 등각 및 등거리 투영과 조합할 수 있습니다. *Lambert Equal Area Azimuthal* 투영 및 *Azimuthal Equidistant* 투영이 방위 투영의 예입니다.

좌표 시스템 선택 또는 작성

프로젝트 계획의 첫 번째 단계는 사용할 좌표 시스템을 결정하는 것입니다.

좌표 시스템을 작성하려면 사용자 ID에 공간 데이터 관련 작업에 대해 사용 가능화된 데이터베이스에 대한 SYSADM 또는 DBADM 권한이 있어야 합니다. 기존 좌표 시스템을 사용할 경우에는 아무런 권한이 필요하지 않습니다.

공간 데이터 관련 작업에 대해 데이터베이스를 사용 가능화한 후에는 공간 데이터를 사용하는 프로젝트를 계획할 수 있습니다. DB2 Spatial Extender에서 제공하는 좌표 시스템을 사용하거나 임의 장소에서 작성된 좌표 시스템을 사용할 수 있습니다. 2000개 이상의 좌표 시스템이 DB2 Spatial Extender와 함께 제공됩니다. 여기에는 다음이 포함됩니다.

- DB2 Spatial Extender가 미지정으로서 참조하는 좌표 시스템. 다음의 경우에 이 좌표 시스템을 사용하십시오.
  - 지구 표면과 직접적 관련이 없는 위치를 정의해야 할 때(예: 사무실 빌딩 내의 사무실 위치 또는 저장실 내의 선반 위치)
  - 10진값을 거의 또는 전혀 포함하지 않는 양의 좌표로 이러한 위치를 정의할 수 있을 때
- GCS_NORTH_AMERICAN_1983. 미국 내의 위치를 정의해야 할 때, 예를 들면, 다음의 경우에 이 좌표 시스템을 사용하십시오.
  - DB2 Spatial Extender에서 제공하는 맵 및 데이터 CD에서 미국에 대한 공간 데이터를 임포트할 때
  - DB2 Spatial Extender에서 제공하는 지오코더(Geocoder)를 사용하여 미국 내의 주소를 지오코딩(Geocoding)할 때

이러한 좌표 시스템에 대한 자세한 정보를 얻고 DB2 Spatial Extender에서 제공하는 다른 좌표 시스템 및 다른 사용자에 의해 작성된 좌표 시스템(있는 경우)을 보려면 DB2SE.ST_COORDINATE_SYSTEMS 카탈로그 뷰를 참조하십시오.

이 태스크를 수행하려면

어떤 방법을 사용하여 좌표 시스템을 작성할지 선택하십시오.

- DB2 제어 센터의 좌표 시스템 작성 창에서 좌표 시스템을 작성하십시오.
- db2se 명령행 처리기에서 db2se create_cs 명령을 실행하십시오.
- db2se.ST_create_coordsys 스토어드 프로시저를 호출하는 응용프로그램을 실행하십시오.

공간 참조 시스템 설정 방법

공간 데이터를 사용하는 프로젝트를 계획할 때, 사용 가능한 공간 참조 시스템이 이 데이터에 사용될 수 있는지 여부를 판별해야 합니다. 사용 가능한 시스템 중 어떤 것도 데이터에 적절하지 않으면, 적절한 시스템을 작성할 수 있습니다. 이 절에서는 공간 참조 시스템의 개념을 설명하고 사용할 시스템을 선택하고 시스템을 작성하는 태스크에 대해 설명합니다.

공간 참조 시스템

*공간 참조 시스템*은 다음을 포함하는 매개변수 세트입니다.

- 좌표가 도출되는 좌표 시스템의 이름
- 공간 참조 시스템을 고유하게 식별하는 숫자 ID
- 제공된 좌표 범위에 의해 참조되는 스페이스의 최대 가능 범위를 정의하는 좌표

- 특정 수학적 연산에서, 입력으로서 수신된 좌표를 최대 효율로 처리 가능한 값으로 변환하는 데 사용되는 숫자

다음 절은 ID, 최대 스페이스 Extent 및 변환 인수를 정의하는 매개변수 값에 대해 설명합니다.

공간 참조 시스템 ID

공간 참조 시스템 ID(SRID)는 다양한 공간 함수에 대한 입력 매개변수로 사용됩니다.

측지 공간 참조 시스템의 경우, SRID 값은 2000000000 - 2000001000 범위 내에 있어야 합니다. DB2® Geodetic Data Management Feature에서는 318개의 사전 정의된 측지 공간 참조 시스템(SRS)을 제공합니다.

공간 컬럼에 저장된 좌표를 포함하는 스페이스 정의

공간 컬럼의 좌표는 일반적으로 지구 일부분에 걸쳐 있는 위치를 정의합니다. 이러한 위치가 걸쳐 있는 스페이스(동쪽에서 서쪽 그리고 북쪽에서 남쪽)를 *공간 Extent*라고 합니다. 예를 들어, 공간 컬럼에 해당 좌표가 있는 범람원을 고려해 보십시오. 좌표의 가장 서쪽과 가장 동쪽의 위도는 각각 –24.556 및 –19.338이며, 좌표의 가장 북쪽과 가장 남쪽의 경도는 각각 18.819 및 15.809라고 가정하십시오. 이 범람원의 공간 범위는 두 위도 사이의 동서면과 두 경도 사이의 북남면에 걸친 스페이스입니다. 이러한 값을 특정 매개변수에 지정함으로써 공간 참조 시스템에 이 값을 포함시킬 수 있습니다. 공간 컬럼에 Z 좌표와 도량 단위가 포함되면 최고 및 최저 Z 좌표와 도량 단위도 공간 참조 시스템에 포함시켜야 합니다.

공간 Extent는 위치의 실제 거리뿐만 아니라 이전 단락에서와 같이 잠재적 거리도 참조할 수 있습니다. 앞 예의 범람원이 다음 5년간 확장될 것으로 예상되었다고 가정하십시오. 5년 후의 범람원의 가장 서쪽, 가장 동쪽, 가장 북쪽 및 가장 남쪽 좌표를 산정할 수 있습니다. 그런 다음, 현재 좌표가 아닌 이러한 산정값을 공간 범위에 대한 매개변수에 지정할 수 있습니다. 그러면 범람원이 확장되고 보다 넓은 위도 및 경도가 공간 컬럼에 추가되더라도 원래의 공간 참조 시스템을 보유할 수 있습니다. 공간 참조 시스템이 원래의 위도 및 경도로 제한되면, 범람원이 커짐에 따라 공간 참조 시스템을 변경해야 합니다.

성능을 향상시키는 값으로 변환

일반적으로 좌표 시스템에 있는 대부분의 좌표는 10진값이고 일부는 정수값입니다. 또한 원점을 기준으로 동쪽에 있는 좌표는 양수값이고 서쪽에 있는 좌표는 음수값입니다. Spatial Extender에 의해 저장되기 전에, 음의 좌표는 양수값으로 변환되고 10진 좌표는 정수로 변환됩니다. 결과적으로 모든 좌표가 Spatial Extender에 의해 양수로서 저장됩니다. 이는 좌표 처리 시의 성능을 향상시키기 위해서입니다.

공간 참조 시스템에 있는 특정 매개변수를 사용하여 이전 단락에서 설명한 변환을 수행합니다. *오프셋* 매개변수를 각각의 음의 좌표에서 빼면 양수값이 남게 됩니다. 각 10진수 좌표를 *스케일 인수*라고 하는 다른 매개변수로 곱하면 정밀도가 10진수 좌표와 동일한 정수가 됩니다. (오프셋을 음의 좌표뿐만 아니라 양의 좌표에서도 뺄 수 있으며 10진수 좌표와 함께 10진수 이외의 좌표에도 스케일 인수를 곱할 수 있습니다. 이러한 방식으로, 모든 비 10진 및 양의 좌표가 10진 및 음의 좌표와 비례하는 상태로 유지될 수 있습니다.)

이러한 변환은 내부적으로 발생하며 좌표를 검색할 때까지만 유효합니다. 입력 및 쿼리 결과에는 항상 변환되지 않은 원래 양식의 좌표가 포함됩니다.

디폴트 공간 참조 시스템을 사용할 것인지 또는 새 시스템을 작성할 것인지 여부 판별

어떤 좌표 시스템을 사용할지 결정한 다음에야 작업 중인 좌표 데이터에 맞는 공간 참조 시스템을 제공할 준비가 됩니다. DB2 Spatial Extender에서는 공간 데이터에 대해 다섯 개의 공간 참조 시스템을 제공하고 DB2 Geodetic Data Management Feature에서는 측지 데이터에 대해 318개의 측지 공간 참조 시스템을제공합니다.

다음 질문에 응답하여 디폴트 공간 참조 시스템 중 하나를 사용할 것인지 또는 사전 정의된 측지 참조 시스템을 사용할 것인지 여부를 판별하십시오.

1. 디폴트 공간 참조 시스템을 기초로 하는 좌표 시스템에 작업 중인 지리 영역이 포함되어 있습니까? 이러한 좌표 시스템은 60 페이지의 『DB2 Spatial Extender에서 제공하는 공간 참조 시스템』에 표시됩니다.
2. 지리 좌표 시스템의 데이터가 10진수 도(degree) 또는 기울기를 측정 단위로 사용합니까? 데이터가 지구 표면의 넓은 면적을 표현합니까? 정확한 거리, 길이 및 영역을 계산해야 합니까? 북극, 남극 또는 국제 날짜 변경선 부근에 데이터가 있습니까? 이러한 질문에 예로 응답하는 경우, 사전 정의된 318개의 측지 공간 참조 시스템 중 하나를 사용할 수 있습니다. 이러한 사전 정의된 측지 공간 참조 시스템에 대한 정보는 202 페이지의 『DB2 Geodetic Data Management Feature에서 지원하는 기준면』을 참조하십시오.
3. 디폴트 공간 참조 시스템 중 하나와 연관된 변환 인수가 좌표 데이터와 함께 작동합니까?

   Spatial Extender에서는 오프셋 값 및 스케일 인수를 사용하여 제공되는 좌표 데이터를 양수로 변환합니다. 좌표 데이터가 디폴트 공간 참조 시스템 중 하나에 제공된 오프셋 값 및 스케일 인수에 사용되는지 판별하려면 다음을 수행하십시오.

   a. 62 페이지의 『좌표 데이터를 정수로 변환하는 변환 인수』에 있는 정보를 검토하십시오.

b. 그런 다음 디폴트 공간 참조 시스템에 이 인수가 어떻게 정의되어 있는지 확인하십시오. 최소 X 및 Y 좌표에 오프셋 값을 적용한 후 이 좌표가 모두 0보다 크지 않으면, 새 공간 참조 시스템을 작성하고 오프셋을 정의해야 합니다. 새 공간 참조 시스템 작성법에 대한 자세한 정보는 63 페이지의 『공간 참조 시스템 작성』을 참조하십시오.

4. 작업 중인 데이터에 높이 및 깊이 좌표(Z 좌표) 또는 도량 단위(M 좌표)가 포함됩니까? Z 또는 M 좌표에 대한 작업을 수행해야 하는 경우, 데이터에 적합한 Z 또는 M 오프셋 및 스케일 인수를 사용하여 새 공간 참조 시스템을 작성해야 합니다.
5. 기존 공간 참조 시스템 또는 측지 참조 시스템이 데이터에 대해 작업하지 않는 경우, 63 페이지의 『공간 참조 시스템 작성』을 수행해야 합니다.

필요한 공간 참조 시스템을 결정한 후에는 다음 태스크를 수행할 때 Spatial Extender에 이 선택항목을 지정하십시오.

- 『DB2 Spatial Extender에서 제공하는 공간 참조 시스템』
- 202 페이지의 『DB2 Geodetic Data Management Feature에서 지원하는 기준면』

DB2 Spatial Extender에서 제공하는 공간 참조 시스템

공간 참조 시스템은 좌표 데이터를 양수로 변환합니다.

DB2 Spatial Extender에서는 아래 테이블에 표시되는 공간 참조 시스템과 함께 각 공간 참조 시스템의 기초가 되는 좌표 시스템 및 DB2 Spatial Extender가 좌표 데이터를 양수로 변환하는 데 사용하는 오프셋 값과 스케일 인수를 제공합니다. 이 공간 참조 시스템에 대한 정보는 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 있습니다.

10진수(도) 단위로 작업 중인 경우(DB2 Spatial Extender 샘플 데이터 CD에 있는 모든 데이터는 10진수(도) 단위로 되어 있음), 디폴트 공간 참조 시스템의 오프셋 값과 스케일 인수는 위도-경도 좌표의 전체 범위를 지원하며 대략 10cm에 해당하는 6개의 10진수 자리를 유지합니다.

미국 주소에만 사용되는 지오코더(Geocoder)를 사용하려는 경우, 미국 좌표(예: GCS_NORTH_AMERICAN_1983 좌표 시스템)를 처리하는 공간 참조 시스템을 선택 또는 작성해야 합니다. 공간 데이터를 도출할 좌표 시스템을 지정하지 않으면 Spatial Extender에서는 DEFAULT_SRS 공간 참조 시스템을 사용합니다.

디폴트 공간 참조 시스템이 요구사항에 맞지 않는 경우, 새 공간 참조 시스템을 작성할 수 있습니다.

표 3. DB2 Spatial Extender를 사용하여 제공된 공간 참조 시스템

공간 참조 시스템	SRS ID	좌표 시스템	오프셋 값	스케일 인수	사용 시기
DEFAULT _SRS	0	없음	xOffset = 0 yOffset = 0 zOffset = 0 mOffset = 0	xScale = 1 yScale = 1 zScale = 1 mScale = 1	데이터가 좌표 시스템에 독립적일 경우 이 시스템을 선택할 수 있거나, 임의의 한 시스템을 지정할 수 없거나 지정할 필요가 없습니다.
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	DB2 Spatial Extender에서 생략된 미국 샘플 데이터를 사용하려는 경우, 이 공간 참조 시스템을 선택할 수 있습니다. 1983년 이후 수집된 좌표 데이터로 작업 중인 경우, NAD27_SRS_1002 대신 이 시스템을 사용하십시오.
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	DB2 Spatial Extender에서 생략된 미국 샘플 데이터를 사용하려는 경우, 이 공간 참조 시스템을 선택할 수 있습니다. 1983년 이전 수집된 좌표 데이터로 작업 중인 경우, NAD83_SRS_1 대신 이 시스템을 사용하십시오. 이 시스템은 디폴트 공간 참조 시스템보다 더 월등한 정밀도를 제공합니다.
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	미국 이외의 데이터로 작업 중인 경우, 이 공간 참조 시스템을 선택할 수 있습니다. (이 시스템은 전세계의 좌표를 처리합니다.) 지오코더(Geocoder)는 미국 주소 전용이므로 DB2 Spatial Extender에서 생략된 디폴트 지오코더(Geocoder)를 사용하려는 경우, 이 시스템을 사용하지 마십시오.
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	이 공간 참조 시스템은 독일어 주소의 좌표 시스템에 기초합니다.

좌표 데이터를 정수로 변환하는 변환 인수

DB2 Spatial Extender에서는 오프셋 값 및 스케일 인수를 사용하여 제공하는 좌표 데이터를 양수로 변환합니다. 디폴트 공간 참조 시스템에는 이미 연관된 오프셋 값 및 스케일 인수가 있습니다. 새 공간 참조 시스템을 작성하는 경우 스케일 인수와 선택적으로 데이터에 가장 적합한 오프셋 값을 판별하십시오.

오프셋 값

오프셋 값은 모든 좌표에서 양수 값만을 남겨두고 빼는 수입니다. Spatial Extender는 다음 공식을 사용하여 조정된 모든 좌표 값이 0보다 크도록 좌표 데이터를 변환합니다.

공식 표기: 이 공식에서 표기 『min』은 『모든 최소』를 나타냅니다. 예를 들어, 『min(x)』은 『모든 x 좌표의 최소』를 의미합니다. 각 지역 방향의 오프셋은 dimensionOffset으로 표시됩니다. 예를 들어, xOffset은 모든 X 좌표에 적용되는 오프셋 값입니다.

$$\min(x) - xOffset \geq 0$$
$$\min(y) - yOffset \geq 0$$
$$\min(z) - zOffset \geq 0$$
$$\min(m) - mOffset \geq 0$$

스케일 인수

스케일 인수는 10진수 좌표 및 도량 단위를 곱할 때 자리수가 최소한 원래 좌표 및 도량 단위와 동일한 정수가 산출되는 값입니다. Spatial Extender는 다음 공식을 사용하여 조정된 모든 좌표 값이 양수가 되도록 10진수 좌표 데이터를 변환합니다. 변환된 값은 2^{53}(대략 $9 * 10^{15}$)를 초과할 수 없습니다.

공식 표기: 이 공식에서 표기 『max』는 『모든 최대』를 나타냅니다. 각 지리적 차원에 대한 오프셋은 dimensionOffset으로 표시됩니다. (예를 들어, xOffset은 모든 X 좌표에 적용되는 오프셋 값입니다.) 각 지리적 차원의 스케일 인수는 dimensionScale로 표시됩니다. (예를 들어, xScale은 X 좌표에 적용되는 스케일 인수입니다.)

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

좌표 데이터에 적합하게 작동하는 스케일 인수를 선택할 경우 다음을 확인하십시오.

- X 및 Y 좌표에 대해 동일한 스케일 인수를 사용합니다.
- 스케일 인수에 10진수 X 좌표 또는 10진수 Y 좌표를 곱하면 2^{53} 미만의 값이 산출됩니다. 한 가지 일반 기술은 스케일 인수를 10의 거듭제곱으로 사용하는 것입니다. 즉, 10의 1승(10), 10의 2승(100), 10의 3승(1000) 또는 필요한 경우 더 큰 인수를 사용할 수 있습니다.
- 스케일 인수는 새 정수의 자리수가 원래 10진수 좌표의 자리수와 동일할 수 있을 만큼 커야 합니다.

예

ST_Point 함수가 X 좌표 10.01, Y 좌표 20.03 및 공간 참조 시스템의 ID를 입력하는 구성이라고 가정하십시오. ST_Point가 호출될 때, 이 함수는 10.01 및 20.03 값을 X 및 Y 좌표에 대한 공간 참조 시스템의 스케일 인수로 곱합니다. 이 스케일 인수가 10인 경우, Spatial Extender가 저장하는 결과 정수는 각각 100 및 200입니다. 이 정수의 자리수(3)가 좌표의 자리수(4)보다 작으므로 Spatial Extender는 이 정수를 다시 원래 좌표로 변환하거나 이 좌표가 속하는 좌표 시스템과 일치하는 값을 추출할 수 없습니다. 그러나 스케일 인수가 100인 경우 DB2 Spatial Extender에서 저장하는 결과 정수는 1001 및 2003(다시 원래 좌표로 변환되거나 호환 가능한 좌표를 도출할 수 있는 값)입니다.

오프셋 값 및 스케일 인수의 단위

기존 공간 참조 시스템을 사용하든지 새 공간 참조 시스템을 작성하든지 오프셋 값 및 스케일 인수의 단위는 사용하는 좌표 시스템의 유형에 따라 다양합니다. 예를 들어, 좌표 시스템을 사용하는 경우, 값은 10진수(도)와 같은 각도 단위입니다. 투영 좌표 시스템을 사용하는 경우, 값은 미터 또는 피트와 같은 선형 단위입니다.

공간 참조 시스템 작성

DB2 Spatial Extender에서 제공하는 공간 참조 시스템이 사용자 데이터에 대해 작업하지 않는 경우 새 공간 참조 시스템을 작성하십시오.

이 태스크를 수행하려면

1. 인터페이스를 선택하십시오.

   다음 방법 중 하나로 공간 참조 시스템을 작성할 수 있습니다.

   - DB2 제어 센터의 공간 참조 시스템 작성 창을 사용하십시오. 이 창을 사용하는 방법에 대한 자세한 정보는 온라인 도움말을 참조하십시오.
   - db2se 명령행 처리기에서 db2se create_srs 명령을 실행하십시오.
   - db2se.ST_create_srs 스토어드 프로시저를 호출하는 응용프로그램을 실행하십시오.

2. 적합한 공간 참조 시스템 ID(SRID)를 지정하십시오.
   - 구형 지구 표현의 측지 데이터의 경우, 200000318 - 2000001000 범위의 SRID 값을 지정하십시오.
   - 평면형 지구 표현의 공간 데이터의 경우, 아직 정의되지 않은 SRID를 지정하십시오.

3. 원하는 정밀도 등급을 결정하십시오.

   다음 중 하나를 수행할 수 있습니다.

- 작업할 지리적 영역의 Extent 및 좌표 데이터에 사용할 스케일 인수를 지정하십시오. Spatial Extender는 사용자가 지정하는 Extent를 가져와 오프셋을 계산합니다. 다음 중 한 가지 방법으로 Extent를 지정할 수 있습니다.
  - 제어 센터의 공간 참조 시스템 작성 창에서 **Extent**를 선택하십시오.
  - db2se create_srs 명령 또는 db2se.ST_create_srs 스토어드 프로시저에 올바른 매개변수를 제공하십시오.
- 오프셋 값(음수값을 양수값으로 변환하는 Spatial Extender에 필요)과 스케일 인수(10진 값을 정수로 변환하는 Spatial Extender에 필요)를 모두 지정하십시오. 엄격한 정확도 또는 정밀도 기준을 준수해야 하는 경우 이 방법을 사용하십시오. 다음 중 한 가지 방법으로 오프셋 값 및 스케일 인수를 지정할 수 있습니다.
  - 제어 센터의 공간 참조 시스템 작성 창에서 **오프셋**을 선택하십시오.
  - db2secreate_srs 명령 또는 db2se.ST_create_srs 스토어드 프로시저에 올바른 매개변수를 제공하십시오.

4. Spatial Extender가 좌표 데이터를 양수로 변환해야 하는 변환 정보를 계산하고 사용자가 선택한 인터페이스에 이 정보를 제공하십시오.

   이 정보는 3단계에서 선택한 방법에 따라 다릅니다.

   - 3단계에서 Extent 방법을 선택한 경우 다음 정보를 계산해야 합니다.
     - 스케일 인수. 작업 중인 좌표 중 하나가 10진수 값인 경우 스케일 인수를 계산하십시오. 스케일 인수는 10진수 좌표 및 도량 단위를 곱할 때 그 자리수가 최소한 원래 좌표 및 도량 단위만큼 큰 정수를 생성하는 숫자입니다. 좌표가 정수인 경우에는 스케일 인수를 1로 설정할 수 있습니다. 좌표가 10진 값인 경우, 10진 비율을 정수값으로 변환하는 숫자로 스케일 인수를 설정해야 합니다. 예를 들어, 좌표 단위가 미터이고 데이터의 정확도가 1cm이면 스케일 인수 100이 필요합니다.
     - 좌표 및 도량 단위의 최소값 및 최대값
   - 3단계에서 오프셋 방법을 선택한 경우, 다음 정보를 계산해야 합니다.
     - 오프셋 값

       좌표 데이터에 음수 및 도량 단위가 포함된 경우, 사용할 오프셋 값을 지정해야 합니다. 오프셋은 나머지로 음수값만 남겨 두고 모든 좌표를 뺀 수입니다. 양수 좌표에 대해 작업하는 경우, 모든 오프셋 값을 0으로 설정하십시오. 양수 좌표에 대해 작업하지 않는 경우, 좌표 데이터에 적용될 때 최대 양수값 (9,007,199,254,740,992) 미만의 정수를 생성하는 오프셋을 선택하십시오.
     - 스케일 인수

       나타내려는 위치의 좌표에 10진수가 있는 경우, 사용할 스케일 인수를 판별하고 공간 참조 시스템 작성 창에 이 스케일 인수를 입력하십시오.

5. db2se create_srs 명령 또는 db2se.ST_create_srs 스토어드 프로시저를 제출하여 스토어드 프로시저를 작성하십시오.

   예를 들어, 다음 명령은 이름이 mysrs인 공간 참조 시스템을 작성합니다.

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

스케일 인수 계산

전제조건

공간 참조 시스템을 작성하고 작업 중인 좌표가 10진수 값인 경우, 좌표 및 도량 단위에 알맞는 스케일 인수를 계산하십시오. 스케일 인수는 10진수 좌표 및 도량 단위를 곱할 때 그 자리수가 최소한 원래 좌표 및 도량 단위만큼 큰 정수를 생성하는 숫자입니다.

스케일 인수를 계산한 후 Extent 값을 판별해야 합니다. 그런 다음 db2se create_srs 명령 또는 db2se.ST_create_srs 스토어드 프로시저를 제출하십시오.

스케일 인수를 계산하려면 다음을 수행하십시오.

1. 어느 X 및 Y 좌표가 10진수인지 또는 10진수가 될 것 같은지 판별하십시오. 예를 들어, 사용자가 다룰 다양한 X 및 Y 좌표 중에서 세 개가 10진수(1.23, 5.1235 및 6.789)임을 판별했다고 가정하십시오.
2. 가장 긴 10진 정밀도를 가진 10진 좌표를 찾으십시오. 그런 다음 정밀도가 동일한 정수를 산출하기 위해 이 좌표에 곱해야 하는 10의 승수를 판별하십시오. 예를 들어, 세 개의 10진 좌표 중에서 5.1235가 가장 긴 10진 정밀도를 가집니다. 이 좌표에 10의 4승(10000)을 곱하면 정수 51235가 생성됩니다.
3. 방금 설명한 곱셈으로 생성된 정수가 2^{53} 미만인지 판별하십시오. 51235는 그리 크지 않습니다. 1.23, 5.11235 및 6.789에 추가하여 X 및 Y 좌표의 범위에 네 번째 10진수 값인 10000000006.789876이 포함된다고 가정하십시오. 이 좌표의 10진 정밀도는 다른 세 개 수의 정밀도보다 크므로 이 좌표(5.1235가 아님)에 10을 곱합니다. 이를 정수로 변환하기 위해 이 좌표에 10의 6승(1000000)을 곱할 수 있습니다. 그러나 결과 값, 10000000006789876은 2^{53}보다 큽니다. DB2 Spatial Extender가 이 값을 저장하려고 하면 예기치 못한 결과가 발생합니다.

   이러한 문제점을 피하려면 원래 좌표로 곱할 때 DB2 Spatial Extender가 정밀도 손실을 최소화하면서 저장 가능한 정수로 절단할 수 있는 10진수를 산출하는 10의 승수를 선택하십시오. 이 경우 10의 5승(100000)을 선택할 수 있습니다. 100000에 10000000006.789876을 곱하면 1000000000678987.6이 됩니다. DB2 Spatial Extender는 이 수를 1000000000678988로 반올림하므로 정확도가 약간 떨어집니다.

좌표 데이터를 정수로 변환하는 변환 인수

DB2 Spatial Extender에서는 오프셋 값 및 스케일 인수를 사용하여 제공하는 좌표 데이터를 양수로 변환합니다. 디폴트 공간 참조 시스템에는 이미 연관된 오프셋 값 및 스케일 인수가 있습니다. 새 공간 참조 시스템을 작성하는 경우 스케일 인수와 선택적으로 데이터에 가장 적합한 오프셋 값을 판별하십시오.

최대 및 최소 좌표와 도량 단위 판별

다음과 같은 경우 이 프로시저를 사용하여 최소 및 최대 좌표와 도량 단위를 판별하십시오.

- DB2 Spatial Extender에서 제공하는 공간 참조 시스템이 데이터에 대해 작업하지 않으므로 새 공간 참조 시스템을 작성하도록 판별하십시오.
- Extent 변환을 사용하여 좌표를 변환하십시오.

공간 참조 시스템 작성시 Extent 변환을 지정하려는 경우, 최소 및 최대 좌표와 도량 단위를 판별하십시오.

Extent 값을 판별한 후 좌표가 10진수이면 스케일 인수를 계산해야 합니다. 그렇지 않은 경우 db2se create_srs 명령 또는 db2se.ST_create_srs 스토어드 프로시저를 제출하십시오.

나타낼 위치의 최소 및 최대 좌표 및 도량 단위를 판별하려면 다음을 수행하십시오.

1. 최소 및 최대 X 좌표를 판별하십시오. 최소 X 좌표를 찾으려면 가장 서쪽에 있는 도메인의 X 좌표를 식별하십시오. (해당 위치가 원점을 기준으로 서쪽에 있으면 이 좌표는 음수값이 됩니다.) 최대 X 좌표를 찾으려면 가장 동쪽에 있는 도메인의 X 좌표를 식별하십시오. 예를 들어, 유정을 표시하고자 하며 각 유정이 X 및 Y 좌표 쌍으로 정의될 경우, 가장 서쪽에 있는 유정의 위치를 가리키는 X 좌표는 최소 X 좌표이고 가장 동쪽에 있는 유정의 위치를 가리키는 X 좌표는 최대 X 좌표입니다.

   팁: 다중 다각형과 같은 다중 기능 유형의 경우 계산 중인 방향으로 가장 먼 다각형에서 가장 먼 점을 선택했는지 확인하십시오. 예를 들어, 최소 X 좌표의 식별을 시도할 경우, 다중 다각형의 가장 서쪽에 있는 다각형 중 가장 서쪽에 있는 X 좌표를 식별하십시오.
2. 최소 및 최대 Y 좌표를 판별하십시오. 최소 Y 좌표를 찾으려면 가장 남쪽에 있는 도메인의 Y 좌표를 식별하십시오. (해당 위치가 원점을 기준으로 남쪽에 있으면 이 좌표는 음수값이 됩니다.) 최대 Y 좌표를 찾으려면 가장 북쪽에 있는 도메인의 Y 좌표를 식별하십시오.
3. 최소 및 최대 Z 좌표를 판별하십시오. 최소 Z 좌표는 최대 좌표 깊이이고 최대 Z 좌표는 최대 높이 좌표입니다.

4. 최소 및 최대 도량 단위를 판별하십시오. 공간 데이터에 도량 단위를 포함시킬 경우, 최고와 최저 수치를 가지는 도량 단위를 판별하십시오.

오프셋 값 계산

좌표 데이터에 음수 또는 도량 단위가 포함되는 경우 오프셋 값을 지정해야 합니다.

공간 참조 시스템을 작성하고 좌표 데이터에 음수 또는 도량 단위가 포함되는 경우, 사용하려는 오프셋 값을 지정해야 합니다. 오프셋은 나머지로 음수값만 남겨 두고 모든 좌표를 뺀 수입니다. 좌표가 음수 또는 도량 단위가 아닌 양수인 경우 공간 작업의 성능을 향상시킬 수 있습니다.

작업 중인 좌표에 대한 오프셋 값을 계산하려면 다음을 수행하십시오.

1. 나타내고자 하는 위치에 대한 X, Y 및 Z 좌표 범위 내에서 최저 음의 좌표를 판별하십시오. 데이터에 음의 도량 단위를 포함시킬 경우에는 이러한 도량 단위의 최저값을 판별하십시오.
2. 선택적이지만 권장되는 사항: 나타내고자 하는 위치를 둘러싼 영역을 실제 *크기*보다 더 *크게* DB2 Spatial Extender에 표시하십시오. 공간 컬럼에 이 위치에 대한 데이터를 쓴 다음, 공간 참조 시스템을 다른 시스템으로 바꾸지 않고 도메인의 가장 바깥쪽 범위에 추가되는 대로 새 지형(feature)에 대한 데이터를 추가할 수 있습니다.

   1단계에서 식별한 각 좌표 및 수치에 해당 좌표 또는 수치의 5 - 10퍼센트에 해당하는 값을 추가하십시오. 결과를 *인수화된 값*이라고 합니다. 예를 들어, 가장 낮은 음의 X 좌표가 -100인 경우 -5를 추가할 수 있으므로 증가된 -105를 산출할 수 있습니다. 나중에 공간 참조 시스템을 작성할 때 가장 낮은 X 좌표를 실제값인 -100 대신 -105로 표시합니다. DB2 Spatial Extender가 -105를 도메인의 서쪽 한계로 해석합니다.
3. 인수화된 X 값으로부터 뺄 경우 0이 되는 값을 찾으십시오. 이 값이 X 좌표에 대한 오프셋 값입니다. DB2 Spatial Extender는 양수값만을 생성하기 위해 모든 X 좌표에서 이 숫자를 뺍니다.

   예를 들어, 증가된 X 값이 -105인 경우 여기서 -105를 빼야만 0이 됩니다. DB2 Spatial Extender는 표현하는 지형과 연관된 모든 X 좌표에서 -105를 뺍니다. 이 좌표 중 어느 것도 -100보다 *크기* 않으므로 모든 빼기 결과 값은 양수가 됩니다.
4. 증가된 Y 값, 증가된 Z 값 및 증가된 수치에 대해 3단계를 반복하십시오.

공간 참조 시스템 작성

DB2 Spatial Extender에서 제공하는 공간 참조 시스템이 사용자 데이터에 대해 작업하지 않는 경우 새 공간 참조 시스템을 작성하십시오.

이 태스크를 수행하려면

1. 인터페이스를 선택하십시오.

   다음 방법 중 하나로 공간 참조 시스템을 작성할 수 있습니다.

   - DB2 제어 센터의 공간 참조 시스템 작성 창을 사용하십시오. 이 창을 사용하는 방법에 대한 자세한 정보는 온라인 도움말을 참조하십시오.
   - db2se 명령행 처리기에서 db2se create_srs 명령을 실행하십시오.
   - db2se.ST_create_srs 스토어드 프로시저를 호출하는 응용프로그램을 실행하십시오.

2. 적합한 공간 참조 시스템 ID(SRID)를 지정하십시오.
   - 구형 지구 표현의 측지 데이터의 경우, 200000318 - 2000001000 범위의 SRID 값을 지정하십시오.
   - 평면형 지구 표현의 공간 데이터의 경우, 아직 정의되지 않은 SRID를 지정하십시오.

3. 원하는 정밀도 등급을 결정하십시오.

   다음 중 하나를 수행할 수 있습니다.

   - 작업할 지리적 영역의 Extent 및 좌표 데이터에 사용할 스케일 인수를 지정하십시오. Spatial Extender는 사용자가 지정하는 Extent를 가져와 오프셋을 계산합니다. 다음 중 한 가지 방법으로 Extent를 지정할 수 있습니다.
     – 제어 센터의 공간 참조 시스템 작성 창에서 **Extent**를 선택하십시오.
     – db2se create_srs 명령 또는 db2se.ST_create_srs 스토어드 프로시저에 올바른 매개변수를 제공하십시오.
   - 오프셋 값(음수값을 양수값으로 변환하는 Spatial Extender에 필요)과 스케일 인수(10진 값을 정수로 변환하는 Spatial Extender에 필요)를 모두 지정하십시오. 엄격한 정확도 또는 정밀도 기준을 준수해야 하는 경우 이 방법을 사용하십시오. 다음 중 한 가지 방법으로 오프셋 값 및 스케일 인수를 지정할 수 있습니다.
     – 제어 센터의 공간 참조 시스템 작성 창에서 **오프셋**을 선택하십시오.
     – db2secreate_srs 명령 또는 db2se.ST_create_srs 스토어드 프로시저에 올바른 매개변수를 제공하십시오.

4. Spatial Extender가 좌표 데이터를 양수로 변환해야 하는 변환 정보를 계산하고 사용자가 선택한 인터페이스에 이 정보를 제공하십시오.

   이 정보는 3단계에서 선택한 방법에 따라 다릅니다.

   - 3단계에서 Extent 방법을 선택한 경우 다음 정보를 계산해야 합니다.
     – 스케일 인수. 작업 중인 좌표 중 하나가 10진수 값인 경우 스케일 인수를 계산하십시오. 스케일 인수는 10진수 좌표 및 도량 단위를 곱할 때 그 자리수가 최소한 원래 좌표 및 도량 단위만큼 큰 정수를 생성하는 숫자입니다. 좌표

가 정수인 경우에는 스케일 인수를 1로 설정할 수 있습니다. 좌표가 10진 값인 경우, 10진 비율을 정수값으로 변환하는 숫자로 스케일 인수를 설정해야 합니다. 예를 들어, 좌표 단위가 미터이고 데이터의 정확도가 1cm이면 스케일 인수 100이 필요합니다.

  - 좌표 및 도량 단위의 최소값 및 최대값

- 3단계에서 오프셋 방법을 선택한 경우, 다음 정보를 계산해야 합니다.
  - 오프셋 값

    좌표 데이터에 음수 및 도량 단위가 포함된 경우, 사용할 오프셋 값을 지정해야 합니다. 오프셋은 나머지로 음수값만 남겨 두고 모든 좌표를 뺀 수입니다. 양수 좌표에 대해 작업하는 경우, 모든 오프셋 값을 0으로 설정하십시오. 양수 좌표에 대해 작업하지 않는 경우, 좌표 데이터에 적용될 때 최대 양수값 (9,007,199,254,740,992) 미만의 정수를 생성하는 오프셋을 선택하십시오.

  - 스케일 인수

    나타내려는 위치의 좌표에 10진수가 있는 경우, 사용할 스케일 인수를 판별하고 공간 참조 시스템 작성 창에 이 스케일 인수를 입력하십시오.

5. db2se create_srs 명령 또는 db2se.ST_create_srs 스토어드 프로시저를 제출하여 스토어드 프로시저를 작성하십시오.

   예를 들어, 다음 명령은 이름이 mysrs인 공간 참조 시스템을 작성합니다.

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

제 9 장 공간 컬럼 설정

프로젝트에 대한 공간 데이터를 확보하려고 할 때, 사용자는 좌표 시스템 및 공간 참조 시스템을 선택 또는 작성할 뿐만 아니라 데이터를 포함할 하나 이상의 테이블 컬럼을 제공하기도 합니다. 이 장에서는 다음 사항에 대해 다룹니다.

- 컬럼의 쿼리 결과가 그래픽으로 렌더링될 수 있으며, 컬럼에 대한 데이터 유형 선택 지침을 제공한다는 점에 대해 언급합니다.
- 컬럼 제공 태스크를 설명합니다.
- 그래픽 양식으로 해당 내용을 표시할 수 있는 도구에 컬럼이 액세스할 수 있게 해 주는 태스크에 대해 설명합니다.

공간 컬럼

볼 수 있는 컨텐츠를 포함한 공간 컬럼

DB2®용 ArcExplorer와 같은 시각화 도구를 사용하여 공간 컬럼을 쿼리하면, 도구는 그래픽 표시 양식으로 결과를 리턴합니다(예: 구획 바운더리 맵 또는 로드 시스템 레이아웃). 일부 시각화 도구는 컬럼의 모든 행이 동일한 공간 참조 시스템을 사용해야 한다는 제한조건을 가집니다. 이러한 제한조건을 적용하는 방법은 공간 참조 시스템에 컬럼을 등록하는 것입니다.

공간 데이터 유형

공간 데이터 관련 작업에 대해 데이터베이스를 사용 가능화하면 DB2 Spatial Extender가 해당 데이터베이스에 구조화된 데이터 유형 계층 구조를 제공합니다.

72 페이지의 그림 12에서 이 계층 구조를 보여줍니다. 이 그림에서 인스턴스화 가능 유형은 흰색 바탕이며 인스턴스화 불가능 유형은 회색 바탕입니다.

인스턴스화할 수 있는 데이터 유형은 ST_Point, ST_LineString, ST_Polygon, ST_GeomCollection, ST_MultiPoint, ST_MultiPolygon 및 ST_MultiLineString입니다.

인스턴스화할 수 없는 데이터 유형은 ST_Geometry, ST_Curve, ST_Surface, ST_MultiSurface 및 ST_MultiCurve입니다.

© Copyright IBM Corp. 1998, 2006

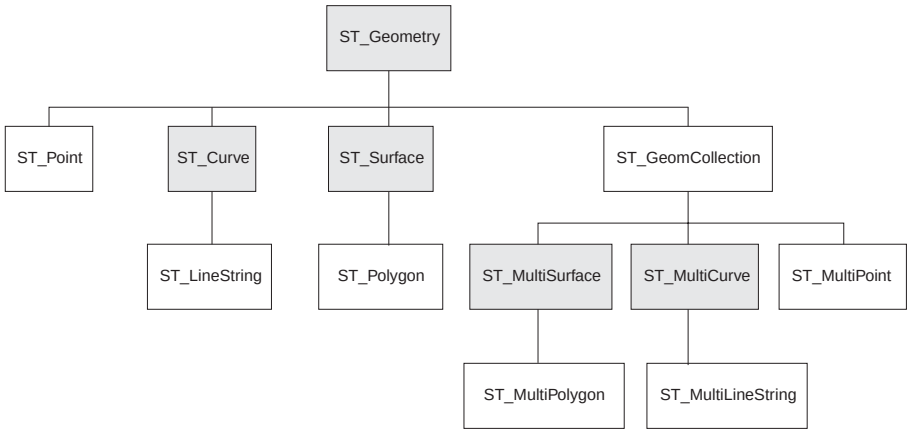


그림 12. 공간 데이터 유형의 계층 구조. 흰색 상자에 이름 지정된 데이터 유형은 인스턴스화 가능 유형입니다. 회색 상자에 이름 지정된 데이터 유형은 인스턴스화 불가능 유형입니다.

그림 12의 계층 구조는 다음을 포함합니다.

- 단일 단위를 형성한 것으로 인식될 수 있는 지리 대상물에 대한 데이터 유형(예: 개별 주택지 및 고립된 호수)
- 여러 단위 또는 구성요소로 구성된 지리 대상물에 대한 데이터 유형(예: 수로 체계 및 하나의 호수에 있는 섬 그룹)
- 모든 종류의 지리 대상물에 대한 데이터 유형

단일 단위 지형에 대한 데이터 유형

ST_Point, ST_LineString 및 ST_Polygon을 사용하여 단일 단위를 구성하는 것으로 인식할 수 있는 지형이 차지하는 스페이스를 정의하는 좌표를 저장하십시오.

- 이질적인 지리적 특성이 차지하고 있는 공간의 지점을 표시하고자 할 때 ST_Point를 사용하십시오. 지형은 우물과 같이 매우 작을 수도 있고 도시와 같이 매우 클 수도 있고 빌딩 또는 공원과 같이 중간 크기의 지형일 수도 있습니다. 각각의 경우에 공간에 있는 점은 동서 좌표선(예: 위선) 및 남북 좌표선(예: 자오선)의 교차점에 놓여 있을 수 있습니다. ST_Point 데이터 항목은 이러한 교차를 정의하는 X 좌표 및 Y 좌표를 포함합니다. X 좌표는 교차점이 동서쪽 선에 놓인 위치를 말하며, Y 좌표는 교차점이 남북선에 놓인 위치를 말합니다.
- 거리, 운하 및 송수관과 같은 선형 특징이 차지하는 공간을 정의하는 좌표에 대해 ST_Linestring을 사용하십시오.
- 선거구, 숲 또는 야생동물 서식지와 같은 다중 평면 대상물로 덮인 스페이스의 Extent를 표시하려면 ST_Polygon을 사용하십시오. ST_Polygon 데이터 항목은 그러한 대상물의 경계를 정의하는 좌표로 구성됩니다.

어떤 경우에는 ST_Polygon 및 ST_Point를 동일한 지형에 사용할 수 있습니다. 예를 들어, 아파트 단지에 대한 공간 정보가 필요하다고 가정하십시오. 아파트 단지에서 각 빌딩이 있는 지점을 나타내고자 할 경우, ST_Point를 사용하여 그러한 각 지점을 정의

하는 X 및 Y 좌표를 저장합니다. 또는 아파트 단지가 차지하는 영역 전체를 표시하려는 경우 ST_Polygon을 사용하여 이 영역의 바운더리를 정의하는 좌표를 저장합니다.

다중 단위 지형에 대한 데이터 유형

ST_MultiPoint, ST_MultiLineString 및 ST_MultiPolygon을 사용하여 다중 단위로 구성된 지형이 차지하는 스페이스를 정의하는 좌표를 저장하십시오.

- 해당 위치가 각각 X 및 Y 좌표로 참조되는 단위들로 구성된 대상물을 표시하려면 ST_MultiPoint를 사용하십시오. 예를 들어, 행이 섬 군도를 나타내는 테이블을 고려해 보십시오. 각 섬의 X 및 Y 좌표가 식별되어 있습니다. 테이블에 이러한 좌표와 각 군도 전체에 대한 좌표가 포함되도록 하려면 이러한 좌표를 포함하도록 ST_MultiPoint 컬럼을 정의하십시오.
- 선형 단위로 구성된 대상물을 표시하고 이러한 단위의 위치 및 각 대상물 전체의 위치에 대한 좌표를 저장하려면 ST_MultiLineString을 사용하십시오. 예를 들어, 행이 강 체계를 나타내는 테이블을 고려해 보십시오. 테이블에 강 체계와 해당 구성요소의 위치에 대한 좌표가 포함되도록 하려면 이러한 좌표를 포함하도록 ST_MultiLineString 컬럼을 정의하십시오.
- 다중 평면 단위로 구성된 대상물을 표시하고 이러한 단위의 위치 및 각 대상물 전체의 위치에 대한 좌표를 저장하려면 ST_MultiPolygon을 사용하십시오. 예를 들어, 해당 행이 농촌 지역과 각 지역에 있는 농장을 나타내는 테이블을 고려해 보십시오. 이 테이블에 해당 지역과 농장의 위치에 대한 좌표가 포함되도록 하려면 이러한 좌표를 포함하도록 ST_MultiPolygon 컬럼을 정의하십시오.

다중 단위는 개별적 엔티티의 콜렉션을 의미하지는 않습니다. 그보다 다중 단위는 전체를 구성하는 파트의 집계를 참조합니다.

모든 지형에 대한 데이터 유형

기타 다른 데이터 유형 중 어느 것을 사용할 것인지 확신하지 못할 때 ST_Geometry를 사용할 수 있습니다.

ST_Geometry는 기타 데이터 유형이 속하는 계층 구조의 루트이므로 ST_Geometry 컬럼은 기타 데이터 유형의 컬럼을 포함할 수 있는 같은 종류의 데이터 항목을 포함할 수 있습니다.

경고: 제공되는 지오코더(Geocoder)인 DB2SE_USA_GEOCODER를 사용하여 공간 컬럼의 데이터를 생성하려면 해당 컬럼의 유형이 ST_Point 또는 ST_Geometry이어야 합니다. 그러나 특정 시각화 도구는 ST_Geometry 컬럼을 지원하지 않고 ST_Geometry의 알맞은 부속 유형이 지정된 컬럼만을 지원합니다.

공간 컬럼 작성

공간 데이터를 저장하거나 검색하려면 공간 컬럼을 작성해야 합니다.

공간 컬럼을 작성하려면 사용자 ID가 DB2 SQL CREATE TABLE문 또는 ALTER TABLE문에 필요한 권한을 보유해야 합니다. 사용자 ID는 다음 권한 또는 특권 중 최소 하나를 보유해야 합니다.

- 컬럼을 포함하는 테이블이 상주하는 데이터베이스에 대한 SYSADM 또는 DBADM 권한
- 데이터베이스에 대한 CREATETAB 권한 및 테이블 스페이스에 대한 USE 특권과 다음 중 한 가지 특권
  - 내재되거나 명시된 인덱스 스키마가 없을 경우 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
  - 인덱스 스키마 이름이 기존의 스키마를 참조할 경우 스키마에 대한 CREATEIN 특권
- 변경될 테이블에 대한 ALTER 특권
- 변경될 테이블에 대한 CONTROL 특권
- 테이블의 스키마에 대한 ALTERIN 특권

이 태스크는 상위 태스크 "프로젝트의 공간 자원 설정"의 일부입니다. 좌표 시스템을 선택하고 데이터에 사용할 공간 참조 시스템을 결정한 후 기존 테이블에 공간 컬럼을 작성하거나 새 테이블로 공간 데이터를 임포트하십시오.

이 태스크를 수행하려면

다음과 같은 여러 가지 방법 중 하나로 데이터베이스에 공간 컬럼을 제공할 수 있습니다.

- DB2의 CREATE TABLE문을 사용하여 테이블을 작성하고 이 테이블에 공간 컬럼을 포함시키십시오.
- DB2의 ALTER TABLE문을 사용하여 기존 테이블에 공간 컬럼을 추가하십시오.
- DB2 제어 센터의 공간 컬럼 작성 창을 사용하십시오. 테이블에서 공간 컬럼 창을 여십시오.
- 모양 파일에서 공간 데이터를 임포트할 경우, DB2 Spatial Extender를 사용하여 테이블을 작성한 후 이 테이블에 데이터를 포함할 컬럼을 제공하십시오. "새 테이블 또는 기존 테이블로 모양 데이터 임포트"를 참조하십시오.
- SDE 전송 파일에서 공간 데이터를 임포트할 경우, DB2 Spatial Extender를 사용하여 테이블을 작성하고 이 테이블에 데이터를 포함할 컬럼을 제공한 후 시각화 도구가 이 컬럼에 액세스할 수 있도록 하십시오. "새 테이블 또는 기존 테이블로 SDE 전송 데이터 임포트"를 참조하십시오.

공간 자원 설정에서 다음 태스크는 『공간 컬럼 등록』입니다.

공간 컬럼 등록

공간 컬럼을 등록하면 테이블에 제한조건이 작성되어 가능한 경우 모든 Geometry가 지정된 공간 참조 시스템을 사용하는지 확인합니다.

전제조건

다음과 같은 경우 공간 컬럼을 등록할 수 있습니다.

- 시각화 도구로 액세스

  공간 컬럼에서 데이터의 그래픽 표시를 생성하기 위해 특정 시각화 도구(예: DB2용 ArcExplorer)를 원하는 경우, 해당 컬럼 데이터의 무결성을 보장해야 합니다. 컬럼의 모든 행이 동일한 공간 참조 시스템을 사용해야 하는 제한조건을 부과함으로써 이를 수행합니다. 이러한 제한조건을 부과하려면 해당 컬럼의 이름과 그 컬럼에 적용되는 공간 참조 시스템을 지정하여 컬럼을 등록해야 합니다.
- 공간 인덱스로 액세스

  공간 인덱스가 올바른 결과를 리턴하도록 하려면 인덱스를 작성할 공간 컬럼의 모든 데이터에 동일한 좌표 시스템을 사용하십시오. 모든 데이터를 제한할 공간 컬럼을 등록하여 동일한 공간 참조 시스템 및 그에 따라 동일한 좌표 시스템을 사용하십시오.

공간 컬럼을 등록하려면 사용자 ID에 다음 권한 중 하나가 있어야 합니다.

db2se 명령행 처리기 또는 응용프로그램을 사용하여 SDE 전송 파일에서 데이터를 임포트하는 경우, DB2 Spatial Extender를 사용하여 데이터를 보유하는 컬럼을 자동으로 작성하고 등록할 수 있습니다. 이 경우에는 사용자 ID에 데이터베이스에 대한 SYSADM 또는 DBADM 권한이 있어야 합니다.

이 태스크를 수행하려면

1. 다음 방법 중 하나로 공간 컬럼을 등록할 수 있습니다. DB2 제어 센터의 공간 컬럼 및 공간 참조 시스템 선택 창을 사용하여 컬럼을 등록하십시오. db2se register_spatial_column 명령을 실행하십시오. db2gse.ST_register_spatial_column 스토어드 프로시저를 호출하는 응용프로그램을 실행하십시오. SDE 전송 파일에서 공간 데이터를 임포트하려는 경우, 제어 센터의 공간 데이터 임포트 창, import_sde 명령 또는 db2gse.ST_import_sde 스토어드 프로시저를 사용하여 공간 컬럼이 있는 테이블을 작성하고 이 컬럼을 등록한 후 데이터를 이 컬럼으로 임포트할 수 있습니다.
2. 컬럼을 등록한 후 특정 컬럼에 대해 선택한 공간 참조 시스템을 검사하려면 DB2GSE.GSE_GEOMETRY_COLUMNS 뷰에서 SRS_NAME 컬럼을 참조하십시오.

제 10 장 공간 컬럼 데이터 상주

공간 컬럼을 작성하고 이러한 시각화 도구로 액세스되는 공간 컬럼을 등록했으면 공간 데이터로 컬럼을 채울 준비가 된 것입니다. 데이터를 제공하는 데에는 세 가지 방법이 있습니다. 이러한 방법으로는 데이터 임포트, 지오코더(Geocoder)를 사용하여 비즈니스 데이터에서 데이터 가져오기 또는 공간 함수를 사용하여 데이터 작성 또는 비즈니스 데이터나 기타 공간 데이터에서 데이터 가져오기가 있습니다.

공간 데이터 임포트 및 익스포트 정보

DB2® Spatial Extender를 사용하여 데이터와 외부 데이터 소스 사이에서 공간 데이터를 교환할 수 있습니다. 보다 정확히 말하면, 공간 데이터를 데이터 교환 파일이라는 파일 형식으로 데이터베이스에 전송함으로써 외부 소스에서 공간 데이터를 임포트할 수 있습니다. 또한 데이터베이스에서 데이터 교환 파일로 공간 데이터를 익스포트할 수 있으며, 외부 소스는 이 데이터 교환 파일로부터 공간 데이터를 획득합니다. 이 절에서는 공간 데이터를 임포트 및 익스포트하는 몇 가지 이유와 DB2 Spatial Extender가 지원하는 데이터 교환 파일의 특성에 대해 설명합니다.

공간 데이터 임포트 및 익스포트 이유

공간 데이터를 임포트함으로써, 산업계에서 이미 사용 가능한 상당량의 공간 정보를 획득할 수 있습니다. 공간 데이터를 익스포트함으로써, 기존 응용프로그램이 공간 데이터를 표준 파일 형식으로 사용 가능하게 할 수 있습니다. 다음과 같은 시나리오를 생각해 봅시다.

- 데이터베이스에 판매 사무실, 고객 및 기타 비즈니스 관련 정보를 나타내는 공간 데이터가 있습니다. 사용자 소속의 문화 환경(도시, 거리, 명승지 등)을 나타내는 공간 데이터로 이 데이터를 보충하려 합니다. 사용자가 원하는 데이터는 지도 벤더에서 제공합니다. DB2 Spatial Extender를 사용하여 벤더가 제공하는 데이터 교환 파일에서 이러한 데이터를 임포트할 수 있습니다.
- Oracle 시스템에서 DB2 환경으로 공간 데이터를 이주시키고자 합니다. Oracle 유틸리티를 사용하여 데이터를 데이터 교환 파일로 로드합니다. 그런 다음, DB2 Spatial Extender를 사용하여 이 파일에서 공간 데이터 관련 작업에 대해 사용 가능화한 데이터베이스로 데이터를 임포트합니다.
- DB2에 연결되어 있지 않으며, 공간형 검색기를 사용하여 고객에게 공간 정보의 비주얼 프리젠테이션을 표시하고자 합니다. 브라우저에는 작업할 파일만 필요하며, 데이터베이스에는 연결될 필요가 없습니다. DB2 Spatial Extender를 사용하여 데이터를 데이터 교환 파일로 익스포트한 다음, 브라우저를 사용하여 데이터를 비주얼 양식으로 렌더링할 수 있습니다.

© Copyright IBM Corp. 1998, 2006

모양 파일 및 SDE 전송 파일

DB2 Spatial Extender는 모양 파일 및 SDE 전송 파일이라는 두 가지 유형의 데이터 교환 파일을 지원합니다. 모양 파일은 실제로 이름은 동일하나 확장자는 다른 파일의 콜렉션을 지칭합니다. 콜렉션은 최대 4개의 파일을 포함할 수 있습니다. 이 함수는 다음과 같습니다.

- ESRI에서 개발한 사실상의 산업 표준 형식인 모양 형식의 공간 데이터가 들어 있는 파일. 이러한 데이터를 종종 모양 데이터라고 합니다. 모양 데이터가 있는 파일의 확장자는 .shp입니다.
- 모양 데이터로 정의된 위치와 관련한 비즈니스 데이터가 들어 있는 파일. 이 파일의 확장자는 .dbf입니다.
- 모양 데이터에 대한 인덱스가 들어 있는 파일. 이 파일의 확장자는 .shx입니다.
- .shp 파일에 있는 데이터의 기초가 되는 좌표 시스템의 스펙이 들어 있는 파일. 이 파일의 확장자는 .prj입니다.

모양 파일은 종종 파일 시스템에서 생성된 데이터를 임포트하고 파일 시스템에 있는 파일로 데이터를 익스포트하는 데 사용됩니다.

DB2 Spatial Extender를 사용하여 모양 데이터를 임포트하면 하나 이상의 .shp 파일이 수신됩니다. 대부분의 경우에 세 종류의 기타 모양 파일이 하나 이상 수신됩니다.

SDE 전송 파일은 종종 ESRI 데이터베이스에서 생성된 데이터를 임포트하는 데 사용됩니다. 각 파일에는 공간 데이터, 이 데이터에 대한 공간 참조 시스템 및 비즈니스 데이터가 들어 있습니다. ESRI 고유 형식의 공간 데이터는 DB2 Spatial Extender 카탈로그에 등록된 테이블 컬럼을 위한 것입니다. 비즈니스 데이터는 등록된 컬럼이 속하는 테이블의 다른 컬럼을 위한 것입니다.

공간 데이터 임포트

이 절에서는 사용자 데이터베이스로 모양 데이터와 SDE 전송 데이터를 임포트하는 태스크에 대한 개요를 제공합니다. 이 절에는 이러한 태스크를 수행하기 위해 사용자가 알아야 하는(예: 프로세스 및 매개변수) 특성에 대한 상호 참조가 포함되어 있습니다.

새 테이블 또는 기존 테이블로 모양 데이터 임포트

기존 테이블 또는 뷰로 모양 데이터를 임포트하거나, 단일 조작으로 테이블을 작성하여 모양 데이터를 새 테이블로 임포트할 수 있습니다.

기존 테이블이나 뷰로 공간 데이터를 임포트하려면 사용자 ID에 다음 권한 중 하나가 있어야 합니다.

- 테이블 또는 뷰가 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한
- 테이블 또는 뷰에 대한 CONTROL 특권

- 테이블 또는 뷰에 대한 INSERT 특권
- 테이블 또는 뷰에 대한 SELECT 특권(테이블에 IDENTITY 컬럼이 아닌 ID 컬럼이 있는 경우에만 필요함)
- 입력 파일 및 오류 파일이 속하는 디렉토리에 액세스할 수 있는 특권
- 입력 파일에 대한 읽기 특권 및 오류 파일에 대한 쓰기 특권

자동으로 테이블을 작성하고 모양 데이터를 새 테이블로 임포트하려면 사용자 ID에 다음과 같은 권한이 있어야 합니다.

- 테이블이 있는 데이터베이스에 대한 SYSADM, DBADM 또는 CREATETAB 권한
- 다음 사용 권한 중 하나
  - 테이블이 속하는 스키마에 대한 CREATEIN 특권(스키마가 이미 있을 때 필요함)
  - 테이블이 있는 데이터베이스에 대한 IMPLICIT_SCHEMA 권한(테이블에 대해 지정된 스키마가 실제로 존재하지 않을 때 필요함)
- 입력 파일 및 오류 파일이 속하는 디렉토리에 액세스할 수 있는 특권
- 입력 파일에 대한 읽기 특권 및 오류 파일에 대한 쓰기 특권

모양 데이터를 임포트하려면 다음 방법 중 하나를 선택하십시오.

- 기존 테이블, 갱신 가능한 기존 뷰 또는 INSERT에 대한 INSTEAD OF 트리거가 정의되어 있는 기존 뷰의 공간 컬럼으로 공간 데이터 임포트하십시오.
- 공간 컬럼이 있는 테이블을 자동으로 작성하고 이 컬럼으로 공간 데이터 임포트하십시오.

이 태스크를 수행하려면

모양 데이터를 임포트하는 데 사용할 방법을 선택하십시오.

- DB2 제어 센터의 모양 데이터 임포트 창을 사용하십시오.
- db2se import_shape 명령을 실행하십시오.
- db2gse.ST_import_shape 스토어드 프로시저를 호출하는 응용프로그램을 실행하십시오.

새 테이블 또는 기존 테이블로 SDE 전송 데이터 임포트

기존 테이블로 SDE 전송 데이터를 임포트하거나, 단일 조작으로 테이블을 작성하여 SDE 전송 데이터를 새 테이블로 임포트할 수 있습니다.

기존 테이블이나 뷰에 있는 컬럼으로 데이터를 임포트하려면 사용자 ID에 다음 권한 중 하나가 있어야 합니다.

- 테이블 또는 뷰가 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한

- 테이블 또는 뷰에 대한 CONTROL 특권
- 테이블 또는 뷰에 대한 INSERT 및 SELECT 특권

테이블을 자동으로 작성하여 모양 데이터를 새 테이블로 임포트하는 조작을 시작하려면 사용자 ID에 다음과 같은 권한이 있어야 합니다.

- 테이블이 있는 데이터베이스에 대한 SYSADM, DBADM 또는 CREATETAB 권한
- 다음 사용 권한 중 하나
  - 테이블이 속하는 스키마에 대한 CREATEIN 특권(스키마가 이미 있을 때 필요함)
  - 테이블이 있는 데이터베이스에 대한 IMPLICIT_SCHEMA 권한(테이블에 대해 지정된 스키마가 실제로 존재하지 않을 때 필요함)

SDE 전송 데이터를 임포트하려면 다음 방법 중 하나를 선택하십시오.

- DB2 Spatial Extender 카탈로그에 이미 등록된 공간 컬럼이 들어 있는 기존 테이블로 SDE 전송 데이터 임포트. 전송 데이터에는 이 컬럼용의 공간 데이터와 테이블에 있는 다른 컬럼용의 비즈니스 데이터가 포함될 수 있습니다.
- 공간 컬럼이 있는 테이블을 자동으로 작성하고 이 컬럼을 카탈로그에 등록한 다음 SDE 전송 데이터를 이 컬럼과 테이블의 다른 컬럼으로 임포트

이 태스크를 수행하려면

어떤 방법을 사용하여 SDE 전송 데이터를 임포트할지 선택하십시오.

- DB2 제어 센터의 임포트 창을 사용하십시오.
- db2se import_sde 명령을 실행하십시오.
- db2gse.GSE_import_sde 스토어드 프로시저를 호출하는 응용프로그램을 실행하십시오.

공간 데이터 익스포트

이 절에서는 모양 및 SDE 전송 파일로 공간 데이터를 익스포트하는 태스크에 대한 개요가 제공됩니다. 이 절에는 이러한 태스크를 수행하기 위해 사용자가 알아야 하는(예: 프로세스 및 매개변수) 특성에 대한 상호 참조가 포함되어 있습니다.

shapefile로 데이터 익스포트

쿼리 결과에서 리턴된 공간 데이터를 모양 파일로 익스포트할 수 있습니다.

모양 파일로 데이터를 익스포트하려면 사용자 ID에 다음과 같은 특권이 있어야 합니다.

- 익스포트할 결과를 리턴하는 subselect를 실행할 수 있는 특권

- 데이터를 익스포트할 파일이 있는 디렉토리에 대한 쓰기 특권
- 익스포트된 데이터를 포함할 파일을 작성할 수 있는 특권(해당 파일이 없는 경우에만 필요함)

해당 특권이 무엇인지와 해당 특권을 획득하는 방법에 대해서는 데이터베이스 관리자에게 문의하십시오.

shapefile로 익스포트하는 공간 데이터는 기본 테이블, 복수 테이블의 조인 또는 통합, 뷰를 쿼리할때 리턴된 결과 세트 또는 공간 함수의 출력과 같은 소스에서 비롯된 것일 수 있습니다.

데이터를 익스포트하려는 파일이 있으면, DB2 Spatial Extender가 이 파일에 해당 데이터를 추가할 수 있습니다. 그러한 파일이 없으면, DB2 Spatial Extender를 사용하여 파일을 작성할 수 있습니다.

이 태스크를 수행하려면

shapefile로 데이터를 익스포트하는 방법을 선택하십시오.

- DB2 제어 센터의 모양 파일 익스포트 창에서 익스포트를 시작하십시오.
- db2se 명령행 처리기에서 db2se export_shape 명령을 실행하십시오.
- db2gse.ST_export_shape 스토어드 프로시저를 호출하는 응용프로그램을 실행하십시오.

SDE 전송 파일로 데이터 익스포트

공간 데이터를 포함하는 테이블을 SDE 전송 파일로 익스포트할 수 있습니다.

SDE 전송 파일로 데이터를 익스포트하려면 사용자 ID에 다음과 같은 승인이 있어야 합니다.

- SYSADM 또는 DBADM 권한
- 익스포트할 테이블에 대한 SELECT 특권
- 데이터를 익스포트할 파일이 있는 디렉토리에 대한 쓰기 특권

제한사항:

- 각 익스포트 조작에서 하나의 공간 컬럼만을 익스포트할 수 있습니다.
- 익스포트하는 컬럼의 데이터 유형은 SDE 형식이 지원하는 유형이어야 합니다.
- 테이블은 단 하나의 공간 컬럼을 포함해야 합니다.
- 이 컬럼은 DB2 Spatial Extender 카탈로그에 등록되어 있어야 합니다.
- 기존 SDE 파일에 추가할 수 없습니다.

익스포트하는 테이블은 둘 이상의 공간 컬럼을 포함할 수 없습니다. 또한 이 컬럼은 DB2 Spatial Extender 카탈로그에 등록되어 있어야 합니다. 테이블에 비즈니스 데이터가 들어 있으면 이 데이터는 공간 데이터와 함께 익스포트됩니다. 테이블의 모든 행 또는 행의 서브세트만을 익스포트할 수 있습니다. 서브세트를 익스포트하려면 서브세트를 식별하는 WHERE절을 지정하십시오.

이 태스크를 수행하려면

공간 및 비즈니스 데이터를 SDE 전송 파일로 익스포트할 방법을 선택하십시오.

- DB2 제어 센터의 SDE 파일 익스포트 창을 사용하십시오.
- db2se export_sde 명령을 실행하십시오.
- db2gse.GSE_export_sde 스토어드 프로시저를 호출하는 응용프로그램을 실행하십시오.

지오코더(Geocoder) 사용법

이 절에서는 지오코딩(Geocoding) 개념을 설명하여 다음 태스크에 대해 소개합니다.

- 지오코더(Geocoder)가 수행할 작업 정의(예: 커미트를 발행하기 전에 지오코더(Geocoder)가 처리할 레코드 수 지정)
- 데이터가 테이블에 추가되거나 갱신되는 즉시 데이터를 지오코딩(Geocoding)하도록 지오코더(Geocoder) 설정
- 일괄처리 모드로 지오코더(Geocoder) 실행

지오코더(Geocoder) 및 지오코딩(Geocoding)

지오코더 및 지오코딩이라는 용어는 여러 문맥에서 사용됩니다. 여기서는 이러한 문맥을 분류하여 이러한 용어를 다룰 때 그 의미가 보다 명확해질 수 있도록 합니다. 여기서는 지오코더 및 지오코딩을 정의하고, 지오코더가 작동하는 모드를 설명하며, 지오코딩이 속하는 보다 큰 활동을 설명하고, 지오코딩과 관련한 사용자 태스크를 요약합니다.

DB2® Spatial Extender에서 지오코더(Geocoder)는 기존 데이터(함수의 입력)를 공간 개념에서 이해할 수 있는 데이터(함수의 출력)로 변환하는 스칼라 함수입니다. 일반적으로 기존 데이터는 하나의 위치를 설명하거나 이름을 지정하는 관계 데이터입니다. 예를 들어, DB2 Spatial Extender와 함께 제공되는 지오코더(Geocoder)인 DB2SE_USA_GEOCODER는 미국 주소를 ST_Point 데이터로 변환합니다. DB2 Spatial Extender는 벤더 제공 및 사용자 제공 지오코더(Geocoder)를 지원할 수 있으며, 해당 입력 및 출력은 DB2SE_USA_GEOCODER의 입력 및 출력과 유사하지 않아도 됩니다. 즉, 하나의 벤더 제공 지오코더(Geocoder)는 DB2가 파일에 저장하지 않고 파일에서 쓰는 좌표로 주소를 변환할 수도 있습니다. 또 하나의 벤더 제공 지오코더

(Geocoder)는 상가 빌딩의 사무실 번호를 해당 빌딩의 사무실 위치를 정의하는 좌표로 변환하거나 창고 선반의 ID를 창고에서의 선반 위치를 정의하는 좌표로 변환할 수도 있습니다.

다른 경우에는 지오코더(Geocoder)가 변환하는 기존 데이터가 공간 데이터일 수도 있습니다. 예를 들어, 사용자 제공 지오코더(Geocoder)가 X 및 Y 좌표를 DB2 Spatial Extender 데이터 유형 중 하나와 일치하는 데이터로 변환할 수도 있습니다.

DB2 Spatial Extender에서 지오코딩(Geocoding)은 지오코더(Geocoder)가 입력을 출력으로 변환하는(예: 주소를 좌표로 변환) 간단한 조작입니다.

모드

지오코더(Geocoder)는 두 가지 모드로 작동합니다.

- 일괄처리 모드에서, 지오코더는 단일 조작으로 단일 테이블로부터의 모든 입력을 변환합니다. 예를 들어, 일괄처리 모드에서는 DB2SE_USA_GEOCODER가 단일 테이블에 있는 모든 주소(또는 테이블의 지정된 행 서브세트에 있는 모든 주소)를 변환합니다.
- 자동 모드에서, 지오코더는 테이블에서 데이터가 삽입 또는 갱신되는 즉시 데이터를 변환합니다. 지오코더(Geocoder)는 테이블에 정의된 INSERT 및 UPDATE 트리거에 의해 활성화됩니다.

지오코딩(Geocoding) 프로세스

지오코딩(Geocoding)은 다른 데이터에서 DB2 테이블에 있는 공간 컬럼의 컨텐츠를 도출하는 몇 가지 조작 중 하나입니다. 여기서는 이러한 조작을 통틀어 지오코딩 프로세스라고 합니다. 지오코딩(Geocoding) 프로세스는 지오코더(Geocoder)에 따라 달라질 수 있습니다. 예를 들어, DB2SE_USA_GEOCODER는 알려진 주소 파일을 검색하여 입력으로서 수신되는 각 주소가 알려진 주소 및 제공된 등급과 일치하는지 판별합니다. 알려진 주소는 사람들이 연구할 때 찾는 참조 자료와 유사하므로, 이러한 주소를 참조 데이터라고 합니다. 다른 지오코더(Geocoder)는 참조 데이터를 필요로 하지 않을 수도 있습니다. 즉, 다른 방법으로 입력을 검증할 수도 있습니다. DB2SE_USA_GEOCODER가 참여하는 지오코딩(Geocoding) 프로세스는 다음과 같습니다.

1. DB2SE_USA_GEOCODER가 수행하도록 설계된 조작을 수행합니다.
   a. DB2SE_USA_GEOCODER가 입력으로서 수신되는 각 주소를 구문 분석합니다.
   b. DB2SE_USA_GEOCODER가 참조 데이터를 검색하여 구문 분석된 주소에 있는 거리 이름과 특정 등급으로 유사한 거리 이름을 찾습니다. 주소의 우편번호가 지정하는 영역으로 거리 검색 범위를 제한합니다.
   c. 검색이 성공적이면 DB2SE_USA_GEOCODER는 발견한 거리에 있는 주소가 구문 분석된 주소 및 특정 등급과 일치하는지 판별합니다.

d. 일치하는 주소가 있으면 DB2SE_USA_GEOCODER는 구문 분석된 주소를 지오코딩(Geocoding)합니다. 그렇지 않은 경우에는 널(NULL)을 리턴합니다.

2. DB2SE_USA_GEOCODER가 구문 분석된 주소를 지오코딩(Geocoding)하면 DB2가 결과 좌표를 지정된 공간 컬럼에 놓습니다.
3. DB2SE_USA_GEOCODER가 일괄처리 모드로 지오코딩(Geocoding)되면, DB2 Spatial Extender는 (a) DB2SE_USA_GEOCODER가 특정 수의 입력 레코드 처리를 완료할 때마다 또는 (b) DB2SE_USA_GEOCODER가 모든 입력 처리를 완료한 후에 커미트를 발행합니다.

사용자 태스크

DB2 Spatial Extender에서 지오코딩(Geocoding)과 관련한 태스크는 다음과 같습니다.

- 제공된 공간 컬럼에 대해 지오코딩(Geocoding) 프로세스의 특정 파트가 실행되는 방법을 지정하고(예를 들어, 입력 레코드의 거리 이름과 참조 데이터의 거리 이름이 일치하는 최소 등급 설정), 각 커미트 전에 처리되는 레코드 수를 결정합니다. 이 태스크를 지오코딩 설정 또는 지오코딩 조작 설정이라고 합니다.
- 테이블에서 데이터가 추가되거나 갱신될 때마다 데이터가 자동으로 지오코딩(Geocoding)되도록 지정합니다. 자동 지오코딩(Geocoding)이 발생하면, 사용자가 지오코딩(Geocoding) 조작을 설정할 때 지정한 명령어가 적용됩니다. (커미트 관련 명령어는 제외됩니다. 이는 일괄처리 지오코딩(Geocoding)에만 적용됩니다.) 이 태스크를 자동으로 실행되도록 지오코더 설정이라고 합니다.
- 일괄처리 모드로 지오코더(Geocoder)를 실행합니다. 사용자가 이미 지오코딩(Geocoding) 조작을 설정하였으면, 사용자가 해당 명령어를 겹쳐쓰지 않는 한 각 일괄처리 세션 동안 명령어가 여전히 효력을 가집니다. 사용자가 제공된 세션 이전에 지오코딩(Geocoding) 조작을 설정하지 않은 경우, 사용자는 해당 특정 세션에 대해 적용하려는 조작을 설정할 수 있습니다. 이 태스크를 일괄처리 모드로 지오코더 실행 및 일괄처리 모드로 지오코딩 실행이라고 합니다.

지오코딩(Geocoding) 조작 설정

DB2 Spatial Extender는 지오코더(Geocoder)를 호출할 때 수행되어야 할 작업을 미리 설정하도록 합니다.

특정 지오코더(Geocoder)의 지오코딩(Geocoding) 조작을 설정하려면 사용자 ID에 다음 권한 중 하나가 있어야 합니다.

- 지오코더(Geocoder)가 조작을 수행하는 테이블이 들어 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한
- 지오코더(Geocoder)가 조작을 수행하는 테이블에 대한 SELECT 특권과 CONTROL 또는 UPDATE 특권

지오코더(Geocoder)를 호출할 때 다음 매개변수를 지정할 수 있습니다.

- 지오코더(Geocoder)가 데이터를 제공하는 컬럼
- 지오코더(Geocoder)가 테이블이나 뷰에서 읽는 입력이 해당 테이블 또는 뷰의 특정 행 서브세트로 제한되는지의 여부
- 지오코더(Geocoder)가 하나의 작업 단위(UOW)에서 일괄처리 세션으로 지오코딩해야 하는 레코드 범위 또는 수.
- 지오코더(Geocoder) 고유 조작에 대한 요구사항. 예를 들어, DB2SE_USA_GEOCODER는 참조 데이터에 있는 상대 레코드와 지정된 등급 이상으로 일치하는 레코드만을 지오코딩(Geocoding)할 수 있습니다. 이 등급을 최소 일치 스코어라고 합니다.

자동 모드로 실행되도록 지오코더(Geocoder)를 설정하기 전에 위의 목록에 매개변수를 지정해야 합니다. 그런 다음에는, 지오코더(Geocoder)가 호출될 때마다(자동으로 또는 일괄처리 실행을 위해) 지오코딩(Geocoding) 조작이 스펙에 따라 수행됩니다. 예를 들어, 각 작업 단위(UOW) 내에서 45개의 레코드가 일괄처리 모드로 지오코딩(Geocoding)되도록 지정하면, 45번째 레코드가 지오코딩(Geocoding)될 때마다 커미트가 발행됩니다. (예외: 일괄처리 지오코딩(Geocoding)의 개별 세션에 스펙을 겹쳐쓸 수 있습니다.)

일괄처리 모드로 지오코더(Geocoder)를 실행하기 전에 지오코딩(Geocoding) 조작의 디폴트을 설정하지 않아도 됩니다. 일괄처리 세션을 시작할 때, 실행 기간 동안 조작이 수행되는 방법을 지정할 수 있습니다. 일괄처리 세션에 대해 디폴트값을 설정할 경우, 개별 세션에 대해서는 필요에 따라 디폴트값을 겹쳐쓸 수 있습니다.

이 태스크를 수행하려면

지역코딩(Geocoding) 조작을 설정할 방법을 선택하십시오.

- DB2 제어 센터의 지오코딩(Geocoding) 설정 창에서 지오코더(Geocoder)를 호출하십시오.
- db2se setup_gc 명령을 실행하십시오.
- db2gse.ST_setup_geocoding 스토어드 프로시저를 호출하는 응용프로그램을 실행하십시오.

권장사항: DB2SE_USA_GEOCODER는 주소 데이터의 레코드를 읽을 때 이 레코드를 참조 데이터에 있는 상대 레코드와 일치시킵니다. 대략 다음과 같이 수행합니다. 먼저 지오코더(Geocoder)가 참조 데이터를 검색하여 우편번호가 레코드의 우편번호와 동일한 거리를 찾습니다. 레코드에 있는 거리와 특정 최소 등급 이상으로 유사한 거리 이름을 찾으면 계속해서 전체 주소를 찾습니다. 레코드에 있는 전체 주소와 특정 최소 등급 이상으로 유사한 전체 주소를 찾으면 레코드를 지오코딩(Geocoding)합니다. 그러한 주소를 찾지 못하면 널(NULL)을 리턴합니다.

거리 이름이 일치해야 하는 최소 등급을 철자 민감도라고 합니다. 전체 주소가 일치해야 하는 최소 등급을 최소 일치 스코어라고 합니다. 예를 들어, 철자 민감도가 80이면, 거리 이름 간의 일치율이 80퍼센트 이상이어야 지오코더(Geocoder)가 전체 주소를 찾습니다. 최소 일치 스코어가 60이면, 주소 간의 일치율이 60퍼센트 이상이어야 지오코더(Geocoder)가 레코드를 지오코딩(Geocoding)합니다.

철자 민감도 및 최소 일치 스코어를 지정할 수 있습니다. 이를 조정해야 하는 경우도 있습니다. 예를 들어, 철자 민감도 및 최소 일치 스코어가 둘 다 95라고 가정합시다. 지오코딩(Geocoding)할 주소의 유효성이 주의 깊게 검증되지 않은 경우에는 95퍼센트로 정확히 일치하는 주소를 찾을 가능성이 매우 희박합니다. 결과적으로 지오코더(Geocoder)는 이러한 레코드를 처리할 때 널(NULL)을 리턴할 가능성이 큽니다. 그러한 경우에는 철자 민감도 및 최소 일치 스코어를 낮추어 지오코더(Geocoder)를 다시 실행하는 것이 바람직합니다. 철자 민감도 및 최소 일치 스코어의 권장 스코어는 각각 70 및 60입니다.

이 주제의 시작 부분에서 언급하였듯이, 지오코더(Geocoder)가 테이블이나 뷰에서 읽는 입력이 해당 테이블 또는 뷰의 특정 행 서브세트로 제한되는지를 결정할 수 있습니다. 예를 들어, 다음 시나리오를 고려하십시오.

- 지오코더(Geocoder)를 호출하여 테이블에 있는 주소를 일괄처리 모드로 지오코딩(Geocoding)합니다. 불행히도 최소 일치 스코어가 너무 높아 지오코더(Geocoder)가 대부분의 주소를 처리할 때 널(NULL)을 리턴합니다. 지오코더(Geocoder)를 다시 실행할 때 최소 일치 스코어를 줄입니다. 지오코딩(Geocoding)되지 않은 주소로 입력을 제한하기 위해, 지오코더(Geocoder)가 자신이 이전에 리턴한 널(NULL)을 포함하는 행만을 선택하도록 지정합니다.
- 지오코더(Geocoder)가 특정 날짜 이후에 추가된 행만을 선택합니다.
- 지오코더(Geocoder)가 특정 영역에 있는 주소가 포함된 행만을 선택합니다.

이 주제의 시작 부분에서 언급하였듯이, 지오코더(Geocoder)가 하나의 작업 단위(UOW)에서 일괄처리 모드로 지오코딩(Geocoding)하는 레코드 수를 결정할 수 있습니다. 지오코더(Geocoder)가 각 작업 단위에서 동일한 수의 레코드를 처리하게 하거나 또는 단일 작업 단위에서 테이블의 모든 레코드를 처리하게 할 수 있습니다. 후자를 택할 경우에는 다음에 주의하십시오.

- 전자에 비해 작업 단위 크기에 대한 제어 범위가 좁습니다. 결과적으로, 지오코더(Geocoder)가 작동할 때 작성되는 로그 항목 수나 걸리는 잠금 수를 제어할 수 없습니다.
- 지오코더(Geocoder)가 롤백이 필요한 오류를 발견하면, 모든 레코드에 대해 지오코더(Geocoder)를 다시 실행해야 합니다. 테이블이 지나치게 큰 경우 결과적으로 자원의 손실이 클 수 있고, 대부분의 레코드가 처리된 후 오류 및 롤백이 발생합니다.

자동으로 실행되도록 지오코더(Geocoder) 설정

테이블에서 데이터가 추가되거나 갱신되는 즉시 해당 데이터가 자동으로 변환되도록 지오코더(Geocoder)를 설정할 수 있습니다.

자동으로 실행되도록 지오코더(Geocoder)를 설정하려면, 다음을 수행하십시오.

- 지오코더(Geocoder)로부터의 출력으로 채워지는 각 공간 컬럼에 대해 지오코딩(Geocoding) 조작을 설정해야 합니다.
- 사용자 ID에 다음과 같은 권한이 있어야 합니다.
  - 지오코더(Geocoder)를 호출하는 트리거가 정의되는 테이블이 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한
  - 이 테이블에 대한 하나 이상의 사용 권한
    - CONTROL 특권
    - CONTROL 특권이 없으면 ALTER, SELECT 및 UPDATE 특권이 필요합니다.
  - 이 테이블에서 트리거를 작성하는 데 필요한 특권

일괄처리 모드에서 지오코더(Geocoder)를 호출하기 전에 지오코더(Geocoder)가 자동으로 실행되도록 설정할 수 있습니다. 이로 인해 자동 지오코딩(Geocoding)이 일괄처리 지오코딩(Geocoding)보다 먼저 실행될 수 있습니다. 이 경우에는 일괄처리 지오코딩(Geocoding)이 자동으로 이미 처리된 데이터를 처리할 가능성이 있습니다. 그러나 공간 데이터가 두 번 생성되면 두 번째 생성된 데이터가 첫 번째 데이터를 겹쳐쓰므로, 처리가 중복되더라도 데이터 중복을 초래하지는 않습니다. 그러나 이로 인해 성능이 저하될 수 있습니다.

일괄처리 모드와 자동 모드 중 어느 방식으로 테이블에 있는 주소를 지오코딩(Geocoding)할 것인지 결정하기 전에 다음을 고려하십시오.

- 자동 지오코딩(Geocoding)보다 일괄처리 지오코딩(Geocoding)의 성능이 우수합니다. 일괄처리 세션은 한 번의 초기화로 열리고 한 번의 정리로 닫힙니다. 자동 지오코딩(Geocoding)에서는 초기화로 시작되고 정리로 끝나는 단일 조작에서 각 데이터 항목이 지오코딩(Geocoding)됩니다.
- 대체로, 자동 지오코딩(Geocoding)에 의해 채워지는 공간 컬럼은 일괄처리 지오코딩(Geocoding)에 의해 채워지는 공간 컬럼에 비해 보다 최신화될 수 있습니다. 하나의 일괄처리 세션 이후, 주소 데이터가 누적되어 그 다음 세션까지 지오코딩(Geocoding)되지 않은 상태로 있을 수 있습니다. 그러나 자동 지오코딩(Geocoding)이 이미 사용 가능화되었다면 데이터베이스에 주소 데이터가 저장되는 즉시 이 주소 데이터가 지오코딩(Geocoding)됩니다.

이 태스크를 수행하려면

어떤 방법을 사용하여 자동 지역코딩(Geocoding)을 설정할지 선택하십시오.

- DB2 제어 센터의 지오코딩(Geocoding) 설정 창 또는 지오코딩(Geocoding) 창에서 자동 지오코딩(Geocoding)을 설정하십시오.
- db2se enable_autogc 명령을 실행하십시오.
- db2gse.ST_enable_autogeocoding 스토어드 프로시저를 호출하는 응용프로그램을 실행하십시오.

일괄처리 모드로 지오코더(Geocoder) 실행

지오코더(Geocoder)를 일괄처리 모드로 실행하는 경우 여러 레코드를 특정 컬럼으로 가는 공간 데이터로 변환합니다.

지오코더(Geocoder)를 일괄처리 모드로 실행하려면 사용자 ID에 다음 권한 중 하나가 있어야 합니다.

- 데이터를 지오코딩(Geocoding)할 테이블이 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한
- 이 테이블에 대한 CONTROL 또는 UPDATE 특권

각 커미트 전에 처리되는 레코드 수를 지정하려면 이 테이블에 대한 SELECT 특권도 있어야 합니다. 지오코더(Geocoder)가 조작을 수행하는 행을 제한하기 위해 Where절을 지정할 경우에도 Where 절에 참조하는 테이블 및 뷰에 대한 SELECT 특권이 필요할 수 있습니다. 데이터베이스 관리자에게 문의하십시오.

지오코더(Geocoder)를 실행하여 특정 공간 컬럼을 채우기 전에 언제라도 해당 컬럼에 대한 지오코딩(Geocoding) 조작을 설정할 수 있습니다. 조작 설정에는 지오코더(Geocoder)를 실행할 때 반드시 충족시켜야 하는 특정 요구사항의 지정이 포함됩니다. 예를 들어, 지오코더(Geocoder)가 100개의 입력 레코드를 처리할 때마다 DB2 Spatial Extender가 커미트를 발행해야 한다고 가정하십시오. 이러한 조작을 설정할 때는 필요한 숫자로 100을 지정할 것입니다.

지오코더(Geocoder)를 실행할 준비가 되었을 때, 조작을 설정할 때 지정한 값을 겹쳐 쓸 수 있습니다. 겹쳐쓴 값은 실행 기간 동안에만 적용됩니다.

조작을 설정하지 않을 경우에는, 지오코더(Geocoder)를 실행할 준비가 될 때마다 실행 시 충족시켜야 하는 요구사항을 지정해야 합니다.

이 태스크를 수행하려면

지오코더(Geocoder)를 호출하여 일괄처리 모드로 실행하는 방법을 선택하십시오.

- DB2 제어 센터의 지오코딩(Geocoding) 실행 창에서 지오코더(Geocoder)를 호출하십시오.
- db2se run_gc 명령을 실행하십시오.

- db2gse.ST_run_geocoding 스토어드 프로시저를 호출하는 응용프로그램을 실행하십시오.

제 11 장 인덱스 및 뷰를 사용하여 공간 데이터에 액세스

공간 컬럼을 쿼리하기 전에, 이들에 대한 액세스를 이용할 인덱스 및 뷰를 작성할 수 있습니다. 이 장에서는 다음 사항에 대해 다룹니다.

- Spatial Extender가 공간 데이터에 대한 액세스를 신속히 처리하기 위해 사용하는 인덱스의 특성을 설명합니다.
- 그러한 인덱스 작성 방법을 설명합니다.
- 뷰를 사용하여 공간 데이터에 액세스하는 방법을 설명합니다.

공간 인덱스 유형

쿼리 성능이 우수한지는 데이터베이스의 기본 테이블 컬럼에 인덱스가 효율적으로 정의되어 있는지에 연관되어 있습니다. 쿼리의 성능은 쿼리 중에 컬럼 값을 얼마나 빨리 찾는지에 직접 연관되어 있습니다. 인덱스를 사용하는 쿼리는 보다 빠르게 실행할 수 있으며 성능이 현저히 향상됩니다.

공간 쿼리는 일반적으로 둘 이상의 차원을 포함하는 쿼리입니다. 예를 들어, 공간 쿼리에서 점이 영역(다각형) 내에 포함되어 있는지 여부를 알고자 할 수 있습니다. 공간 쿼리의 다차원적 특성으로 인해 DB2® 원시(native) B 트리 인덱싱은 이 쿼리에 비효율적입니다.

공간 쿼리는 다음과 같은 유형의 인덱스를 사용할 수 있습니다.

- 공간 격자 인덱스

  DB2 Spatial Extender의 인덱스화 기술은 다차원 공간 데이터를 인덱스화하기 위해 설계된 *격자 인덱스화*를 이용하여 공간 컬럼을 인덱스화합니다. DB2 Spatial Extender는 지구를 평면 투영한 2차원 데이터용으로 최적화된 격자 인덱스를 제공합니다.
- 측지 Voronoi 인덱스

  DB2 Geodetic Data Management Feature에서는 다차원 측지 데이터가 있는 컬럼에서 인덱스를 작성할 수 있는 새로운 공간 액세스 메소드를 지원합니다. 측지 Voronoi 인덱스는 지구를 180도 경선의 가장자리 또는 극 지점 주위가 왜곡되지 않은 연속적인 구로 간주하므로 측지 데이터에 대해서는 격자 인덱스보다 적합합니다.

© Copyright IBM Corp. 1998, 2006

공간 격자 인덱스

인덱스는 특히 쿼리 테이블에 많은 행이 포함되어 있는 경우 응용프로그램 쿼리 성능을 향상시킵니다. 쿼리 옵티마이저가 쿼리를 실행하기 위해 선택하는 올바른 인덱스를 작성하면 처리할 행 수를 줄일 수 있습니다.

DB2 Spatial Extender는 2차원 데이터용으로 최적화된 격자 인덱스를 제공합니다. 인덱스는 Geometry의 X 및 Y 차원에서 작성됩니다.

격자 인덱스의 다음 측면을 이해하면 유용하게 활용할 수 있습니다.

- 인덱스의 생성
- 쿼리에서 공간 함수 사용
- 쿼리의 공간 격자 인덱스 사용 방법

공간 격자 인덱스 생성

Spatial Extender는 Geometry의 MBR(Minimum Bounding Rectangle)을 사용하여 공간 격자 인덱스를 생성합니다.

대부분의 Geometry에서 MBR은 Geometry를 둘러싸는 직사각형입니다.

공간 격자 인덱스는 인덱스 작성 시 지정하는 고정 *크기*의 논리 제곱 격자로 영역을 나눕니다. 공간 인덱스는 각 Geometry의 MBR과 격자 셀의 교차점에 대한 하나 이상의 항목을 만듦으로써 공간 컬럼에 대해 구축됩니다. 인덱스 항목은 격자 셀 ID, Geometry MBR 및 Geometry가 포함되는 행의 내부 ID로 구성됩니다.

최대 세 개까지의 공간 인덱스 레벨(격자 레벨)로 정의할 수 있습니다. 여러 격자 레벨을 사용하면 다양한 *크기*의 공간 데이터에 대해 인덱스를 최적화할 수 있으므로 유용합니다.

한 Geometry가 네 개 이상의 격자 셀과 교차하면 이 Geometry는 그 다음으로 큰 레벨로 승격됩니다. 일반적으로 Geometry가 클수록 보다 큰 레벨에서 인덱스화됩니다. Geometry가 최대 격자 *크기*에서 열 개 이상의 격자 셀을 교차하는 경우 시스템 정의 오버플로우 인덱스 레벨이 사용됩니다. 이 오버플로우 레벨을 사용하면 인덱스 항목이 너무 많이 생성되지 않습니다. 최적의 성능을 위해서는 이 오버플로우 레벨을 사용하지 않도록 격자 *크기*를 정의하십시오.

예를 들어, 여러 격자 레벨이 있는 경우, 인덱스화 알고리즘은 인덱스화되는 데이터에 대한 최상의 분석을 제공하기 위해 가능한 최저 격자 레벨을 사용하려고 합니다. Geometry가 제공된 레벨에서 네 개 이상의 격자 셀과 교차하면 Geometry는 그 다음으로 큰 레벨로 승격됩니다(다른 레벨이 있는 경우). 그러므로 10.0, 100.0 및 1000.0의 세 격자 레벨이 있는 공간 인덱스는 먼저 각 Geometry와 레벨 10.0 격자를 교차시킵니다. Geometry가 *크기*가 10.0인 네 개 이상의 격자 셀과 교차하는 경우, 이

Geometry는 승격되어 레벨 100.0 격자와 교차합니다. 100.0 레벨에서 교차점이 다섯 이상인 경우, 해당 Geometry는 1000.0 레벨로 승격됩니다. 1000.0 레벨에서 교차점이 11개 이상인 경우, 해당 Geometry는 오버플로우 레벨에서 인덱스화됩니다.

쿼리에서 공간 함수 사용

DB2 옵티마이저는 쿼리의 WHERE 절에 다음 함수 중 하나가 포함되는 경우 사용할 공간 격자 인덱스를 고려합니다.

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

쿼리의 공간 격자 인덱스 사용 방법

쿼리 옵티마이저가 공간 격자 인덱스를 선택하면 쿼리 실행에서는 다단계 필터 프로세스를 사용합니다.

필터 프로세스에는 다음 단계가 포함됩니다.

1. 쿼리 창을 교차하는 격자 셀을 판별하십시오. *쿼리 창*은 필요에 의해 공간 함수의 두 번째 매개변수로 지정하는 Geometry입니다(아래 예 참조).
2. 일치하는 격자 셀 ID가 있는 항목에 대한 인덱스를 스캔하십시오.
3. 인덱스 항목의 Geometry MBR 값과 쿼리 창을 비교하여 쿼리 창 외부의 값은 모두 버리십시오.
4. 필요한 경우 추가 분석 작업을 수행하십시오. 이전 단계의 Geometry 후보 세트가 추가 분석 작업을 수행하여 공간 함수(ST_Contains, ST_Distance 등) 충족 여부를 판별할 수 있습니다. 공간 함수 EnvelopesIntersect는 이 단계를 생략하여 일반적으로 최적의 성능을 제공합니다.

다음 공간 쿼리 예에는 C.GEOMETRY 컬럼에 공간 격자 인덱스가 있습니다.

```
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
```

첫 번째 예에서, 네 개의 좌표 값이 쿼리 창을 정의합니다. 이 좌표 값은 직사각형의 왼쪽 하단 구석과 오른쪽 상단 구석(42.0 –73.0 및 43.0 –72.0)을 지정합니다.

두 번째 예에서, Spatial Extender는 호스트 변수 geometry2에 의해 지정된 Geometry의 MBR을 계산하여 쿼리 창으로 사용합니다.

공간 격자 인덱스를 작성하는 경우, 공간 응용프로그램에서 사용할 가장 일반적인 쿼리 창 *크기*에 적합한 격자 *크기*를 지정해야 합니다. 격자 *크기*가 더 큰 경우, 쿼리 창 외부에 있는 Geometry의 인덱스 항목이 쿼리 창을 교차하는 격자 셀에 상주하므로 이 항목을 스캔해야 하며 이러한 추가 스캔 작업으로 성능이 저하됩니다. 그러나 격자 *크기*가 더 작아지면 각 Geometry에 보다 많은 인덱스 항목이 생성되어 보다 많은 인덱스 항목을 스캔해야 하므로 이 또한 쿼리 성능을 저하시킵니다.

DB2 Spatial Extender는 공간 컬럼 데이터를 분석하고 일반 쿼리 창 *크기*의 올바른 격자 *크기*를 제안하는 Index Advisor 유틸리티를 제공합니다.

인덱스 레벨 수 및 격자 크기에 대한 고려사항

공간 격자 인덱스의 올바른 격자 *크기*를 판별하려면 Index Advisor를 사용하십시오. 이 방법이 인덱스를 조정하고 공간 쿼리를 가장 효율적으로 만들 수 있는 가장 좋은 방법이기 때문입니다.

격자 레벨 수

세 개의 격자 레벨을 가질 수 있습니다.

공간 격자 인덱스의 각 격자 레벨에 대해 공간 쿼리를 수행하는 동안 별도의 인덱스 검색이 수행됩니다. 따라서 격자 레벨이 많은 경우 쿼리의 효율성이 떨어집니다.

공간 컬럼의 값이 상대적 *크기*와 동일한 경우 단일 격자 레벨을 사용하십시오. 그러나 일반 공간 컬럼에는 상대적 *크기*가 동일한 Geometry가 포함되지 않지만 공간 컬럼의 Geometry는 *크기*에 따라 그룹화됩니다. 이러한 Geometry 그룹화에 격자 레벨을 대응시켜야 합니다.

예를 들어, 대형 시골 필지로 둘러싸인 소도시 필지 그룹을 포함하는 공간 컬럼을 가진 시골 토지 필지 테이블이 있다고 가정합시다. 필지의 *크기*는 두 개의 그룹(소도시 그룹 및 대형 시골 필지 그룹)으로 나뉘어져 있으므로 공간 격자 인덱스에 대해 두 개의 격자 레벨을 지정합니다.

격자 셀 크기

일반 규칙은 인덱스 항목의 수를 최소화함과 동시에 최상의 해상도를 얻기 위해 가능한 많이 격자 *크기*를 줄이는 것입니다.

- 전체 인덱스를 컬럼에 있는 작은 Geometry에 대해 최적화하려면 고밀도 격자 *크기*에 대해 작은 값을 사용해야 합니다. 그러면 검색 영역에 없는 Geometry의 평가로 발생하는 오버헤드를 피할 수 있습니다. 그러나 고밀도 격자 *크기* 또한 가장 높은 수의 인덱스 항목을 생성합니다. 결과적으로, 쿼리시 처리되는 인덱스 항목의 수가 증가하고 따라서 인덱스에 필요한 스토리지의 양도 증가합니다. 이러한 요소는 전체 성능을 저하시킵니다.
- 보다 큰 격자 *크기*를 사용하면 보다 큰 Geometry에 대해 인덱스를 최적화할 수 있습니다. 보다 큰 격자 *크기*는 큰 Geometry에 대해 고밀도 격자 *크기*보다 적은 수의 인덱스 항목을 생성합니다. 결과적으로, 인덱스에 필요한 스토리지의 양이 줄고 전체 성능이 향상됩니다.

다음 그림은 격자 *크기*별 다른 효과를 보여줍니다.

그림 13은 토지 구역 지도를 보여줍니다. 각 구역은 다각형 Geometry로 표시됩니다. 검은색 직사각형은 쿼리 창을 표시합니다. MBR이 쿼리 창을 교차하는 모든 Geometry를 찾으려는 것으로 가정하십시오. 그림 13에서는 28개 Geometry(분홍색으로 강조표시됨)에 쿼리 창을 교차하는 MBR이 있음을 표시합니다.

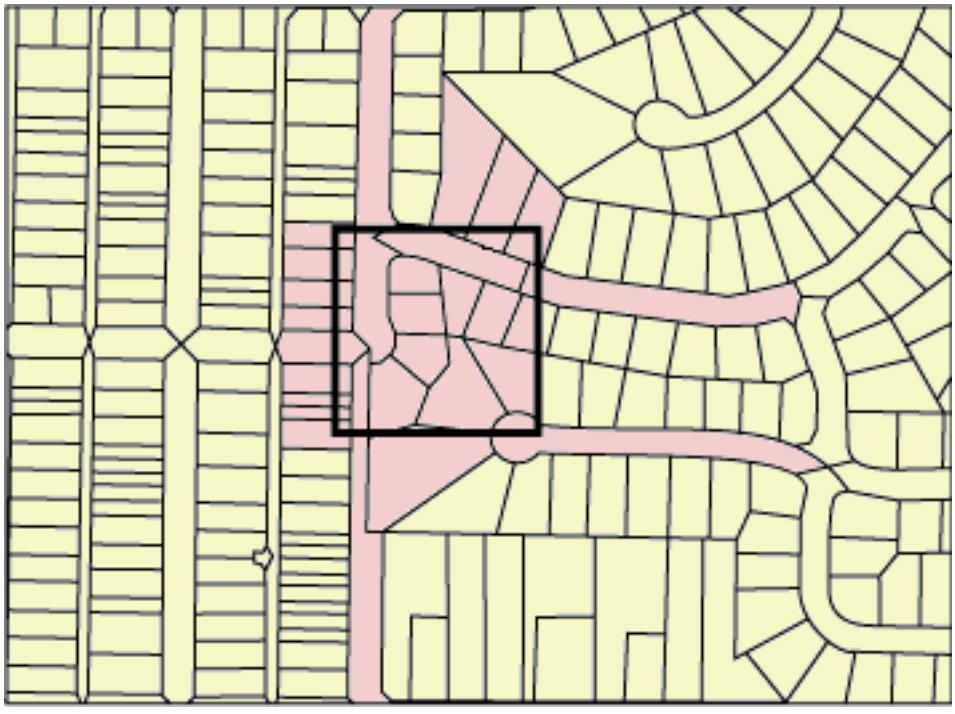


그림 13. 인근 토지 구역

96 페이지의 그림 14은 쿼리 창에 딱 맞는 소형 격자 *크기*(25)를 보여줍니다.

- 이 쿼리는 강조표시된 28개의 Geometry만 리턴하지만 쿼리시 MBR이 쿼리 창을 교차하는 세 개의 추가 Geometry를 검사하여 삭제해야 합니다.
- 이 소형 격자 *크기*를 사용하면 Geometry당 많은 인덱스 항목을 사용할 수 있습니다. 실행하는 동안 쿼리는 이 31개 Geometry의 모든 인덱스 항목에 액세스합니다. 그림 14에서는 쿼리 창을 겹쳐쓰는 256개 격자 셀을 표시합니다. 그러나 동일한 격자 셀을 사용하여 많은 Geometry를 인덱스화하므로 쿼리 실행시 578개의 인덱스 항목에 액세스합니다.

이 쿼리 창에 이 소형 격자 *크기*를 사용하면 많은 인덱스 항목을 스캔해야 합니다.

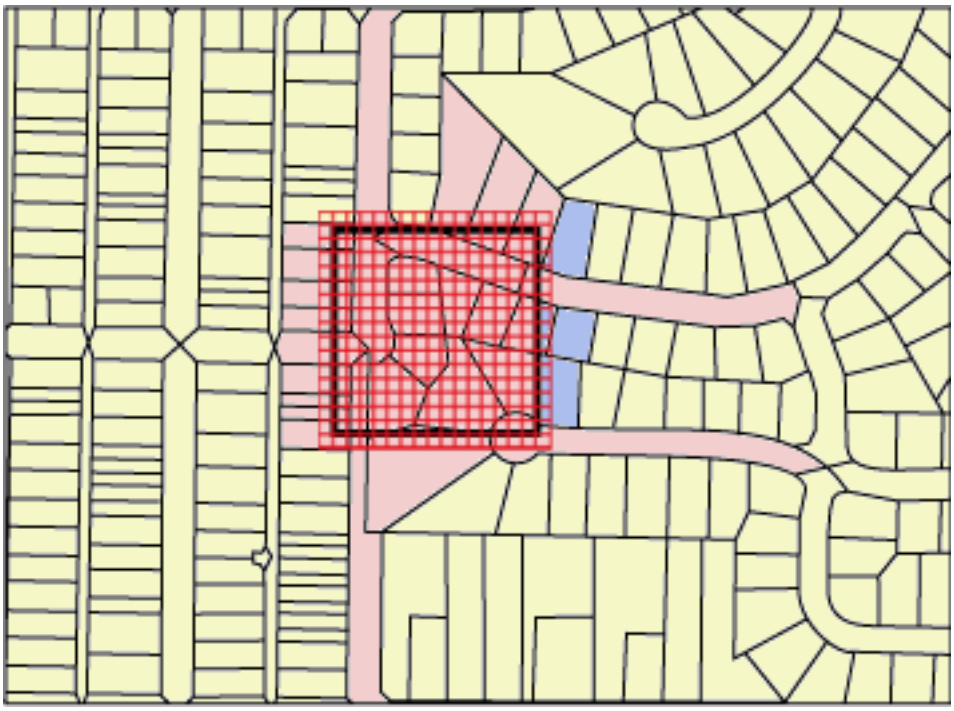


그림 14. 토지 구역의 소형 격자 크기(25)

97 페이지의 그림 15에서는 쿼리 창보다 많은 Geometry로 상당히 넓은 영역을 둘러싸는 대형 격자 *크기*(400)를 보여줍니다.

- 이 대형 격자 *크기*를 사용하면 Geometry당 하나의 인덱스 항목만 사용하게 되지만 쿼리시 MBR이 격자 셀을 교차하는 59개의 추가 Geometry를 검사하여 삭제해야 합니다.
- 쿼리 실행시 총 112개의 인덱스 항목 중에 쿼리 창을 교차하는 28개의 Geometry에 대한 모든 인덱스 항목과 59개의 추가 Geometry에 대한 인덱스 항목에 액세스해야 합니다.

이 쿼리 창의 경우 이 대형 격자 *크기*를 사용하면 과도하게 많은 Geometry를 검사해야 합니다.

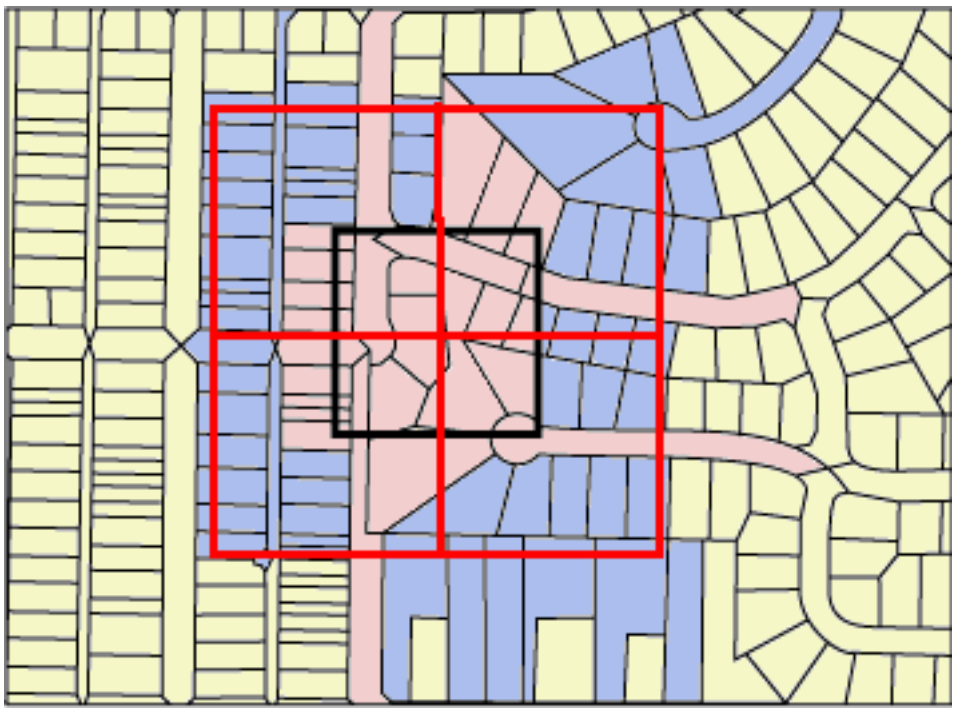


그림 15. 토지 구역의 대형 격자 크기(400)

98 페이지의 그림 16는 쿼리 창에 딱 맞는 대형 격자 크기(100)를 보여줍니다.

- 이 쿼리는 강조표시된 28개의 Geometry만 리턴하지만 쿼리시 MBR이 쿼리 창을 교차하는 다섯 개의 추가 Geometry를 검사하여 삭제해야 합니다.
- 쿼리 실행시 총 91개의 인덱스 항목 중에 쿼리 창을 교차하는 28개의 Geometry에 대한 모든 인덱스 항목과 다섯 개의 추가 Geometry에 대한 인덱스 항목에 액세스해야 합니다.

이 쿼리 창에는 이 중간 격자 크기가 가장 적합합니다. 이 크기를 사용하면 소형 격자 크기보다 훨씬 적은 인덱스 항목을 사용하고 쿼리 실행시 대형 격자 크기보다 적은 추가 Geometry를 검사할 수 있기 때문입니다.

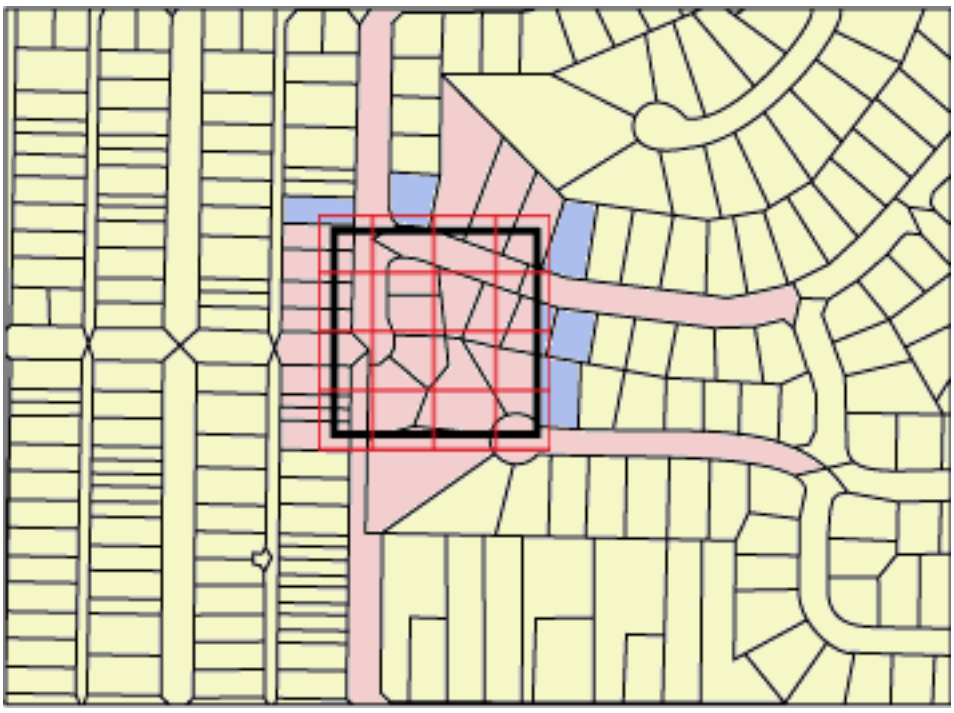


그림 16. 토지 구역의 중간 격자 크기(100)

공간 격자 인덱스 작성

공간 쿼리를 최적화하는 데 도움을 주는 2차원 격자 인덱스를 공간 컬럼에 정의하려면 공간 격자 인덱스를 작성하십시오.

공간 격자 인덱스를 작성하기 위한 전제조건은 다음과 같습니다.

- DB2 SQL CREATE INDEX문에 필요한 권한을 사용자 ID가 보유하고 있어야 합니다. 사용자 ID는 다음 권한 또는 특권 중 최소 하나를 보유해야 합니다.
  - 컬럼을 포함하는 테이블이 상주하는 데이터베이스에 대한 SYSADM 또는 DBADM 권한
  - 다음 권한 또는 특권 모두
    - 다음 테이블 특권 중 하나
      - 테이블에 대한 CONTROL 특권
      - 테이블에 대한 INDEX 특권
    - 스키마에 대한 다음 권한 또는 특권 중 하나
      - 내재되거나 명시된 인덱스 스키마가 없을 경우 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
      - 인덱스 스키마 이름이 기존의 스키마를 참조할 경우 스키마에 대한 CREATEIN 특권

- 완전한 공간 격자 인덱스 이름 및 인덱스에서 사용할 세 개의 격자 *크기*에 지정할 값을 알고 있어야 합니다.

권장사항:

- 컬럼에 공간 격자 인덱스를 작성하기 전에 Index Advisor를 사용하여 인덱스 매개변수를 판별하십시오. Index Advisor는 공간 컬럼 데이터를 분석하고 공간 격자 인덱스에 적절한 격자 *크기*를 제안합니다.
- 데이터를 컬럼으로 *초기 로드*하려는 경우, 로드 프로세스를 완료한 후 공간 격자 인덱스를 작성해야 합니다. 이런 방식에서는 Index Advisor를 사용한 데이터의 특성을 기본으로 하는 최적의 격자 셀 *크기*를 선택할 수 있습니다. 또한 인덱스를 작성하기 전에 데이터를 로드하면 로드 프로세스 중에 공간 격자 인덱스를 유지하지 않아도 되므로 로드 프로세스의 성능이 향상됩니다.

제한사항:

공간 격자 인덱스를 작성하는 경우 CREATE INDEX문을 사용하여 인덱스를 작성하면 동일한 제한사항이 적용됩니다. 즉, 인덱스를 작성하는 컬럼은 뷰 컬럼 또는 별칭 컬럼이 아닌 기본 테이블 컬럼이어야 합니다. DB2 데이터베이스 시스템은 이 프로세스에서 별명을 분석합니다.

공간 격자 인덱스를 작성하여 공간 컬럼에 대한 쿼리 성능을 향상시킬 수 있습니다.

공간 격자 인덱스를 작성하는 경우, 다음 정보를 제공해야 합니다.

- 이름
- 정의될 공간 컬럼의 이름
- 세 개의 격자 *크기*를 조합하면 쿼리를 충족시키기 위해 스캔해야 하는 인덱스 항목 수 및 총 인덱스 항목 수를 최소화하여 성능을 최적화할 수 있습니다.

다음 중 한 가지 방법으로 공간 격자 인덱스를 작성할 수 있습니다.

- DB2 제어 센터의 Spatial Extender 창을 사용하십시오.
- EXTEND USING절에서 SQL CREATE INDEX문과 db2gse.spatial_index 확장자를 함께 사용하십시오.
- DB2 Spatial Extender에 대한 작업을 수행하는 GIS 도구를 사용하십시오. 이 도구를 사용하여 인덱스를 작성하는 경우, 이 도구는 올바른 SQL CREATE INDEX문을 발행합니다.

이 주제는 처음 두 메소드에 대한 단계를 설명합니다. GIS 도구를 사용하여 공간 격자 색인을 작성하는 작업에 대한 정보는 해당 도구와 함께 제공되는 문서를 참조하십시오.

이 태스크를 수행하려면

SQL CREATE INDEX를 사용하여 공간 격자 인덱스 작성

1. EXTEND USING절 및 db2gse.spatial_index 격자 인덱스 확장자를 사용하여 CREATE INDEX문을 판별하십시오. 예를 들어, 다음 명령문은 공간 컬럼 TERRITORY가 있는 BRANCHES 테이블의 공간 격자 인덱스 TERRIDX를 작성합니다.

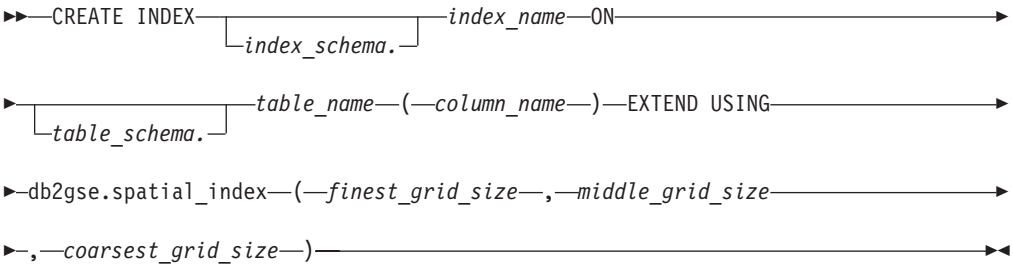
   ```
   CREATE INDEX terridx
      ON branches (territory)
      EXTEND USING db2gse.spatial_index (1.0, 10.0, 100.0)
   ```

2. DB2 명령 편집기, DB2 명령 창 또는 DB2 명령행 처리기에서 CREATE INDEX 명령을 발행하십시오.

공간 격자 인덱스에 대한 CREATE INDEX문

CREATE INDEX문에서 EXTEND USING절을 사용하여 공간 격자 인덱스를 작성하십시오.

구문



매개변수

index_schema.

작성 중인 인덱스가 속할 스키마의 이름. 이름을 지정하지 않으면 DB2 데이터베이스 시스템에서 CURRENT SCHEMA 특수 레지스터에 저장된 스키마 이름을 사용합니다.

index_name

작성 중인 격자 인덱스의 규정되지 않은 이름

table_schema.

*column_name*을 포함하는 테이블이 속하는 스키마의 이름. 이름을 지정하지 않을 경우 DB2는 CURRENT SCHEMA 특수 레지스터에 저장된 스키마 이름을 사용합니다.

table_name

*column_name*을 포함하는 테이블의 규정되지 않은 이름

column_name

공간 격자 인덱스를 작성한 공간 컬럼의 이름

finest_grid_size, middle_grid_size, coarsest_grid_size

공간 격자 인덱스의 격자 *크기*. 이 매개변수는 다음 조건을 준수해야 합니다.

- *finest_grid_size*는 0보다 커야 합니다.
- *middle_grid_size*는 *finest_grid_size*보다 *크거나* 0이어야 합니다.
- *coarsest_grid_size*는 *middle_grid_size*보다 *크거나* 0이어야 합니다.

제어 센터 또는 CREATE INDEX문을 사용하여 공간 격자 인덱스를 작성하는지에 따라 첫 번째 Geometry를 인덱스화할 때 격자 *크기*의 유효성을 점검합니다. 따라서 지정한 격자 *크기*가 해당 값 조건을 충족시키지 않는 경우, 다음과 같은 상황에 오류 조건이 발생합니다.

- 공간 컬럼의 모든 Geometry가 널(NULL)인 경우, Spatial Extender는 격자 *크기*의 유효성을 검증하지 않고 인덱스를 작성할 수 있습니다. 해당 공간 컬럼에 널(NULL)이 아닌 Geometry를 갱신하거나 삽입하면 Spatial Extender가 격자 *크기*의 유효성을 확인합니다. 지정된 격자 *크기*가 유효하지 않은 경우, 널(NULL)이 아닌 Geometry를 삽입 또는 갱신하면 오류가 발생합니다.
- 인덱스를 작성할 때 널(NULL)이 아닌 Geometry가 공간 컬럼에 있는 경우, Spatial Extender는 이 시점에서 격자 *크기*의 유효성을 확인합니다. 지정된 격자 *크기*가 유효하지 않은 경우, 즉시 오류가 발생하여 공간 격자 인덱스가 작성되지 않습니다.

예

다음 예 CREATE INDEX문은 BRANCHES 테이블의 공간 컬럼 TERRITORY에 TERRIDX 공간 격자 인덱스를 작성합니다.

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

Index Advisor로 공간 격자 인덱스 조정

Index Advisor로 공간 격자 인덱스 조정 개요

DB2® Spatial Extender에서는 Index Advisor라는 유틸리티를 제공하며 이 유틸리티는 다음 기능을 수행할 수 있습니다.

- 공간 격자 인덱스의 알맞은 격자 *크기*를 판별하십시오.

  Index Advisor는 공간 컬럼의 Geometry를 분석하여 공간 격자 인덱스에 최적인 격자 *크기*를 권장합니다.

- 기존 격자 인덱스를 분석합니다.

  Index Advisor는 공간 데이터의 검색을 용이하게 하는 현재 셀 *크기*를 판별할 수 있는 통계를 수집하여 표시할 수 있습니다.

공간 격자 인덱스의 격자 크기 판별

전제조건

인덱스화하려는 데이터를 분석하려면 다음을 수행해야 합니다.

- 사용자 ID는 이 테이블에 대해 SELECT 특권이 있어야 합니다.
- 테이블에 백만 개 이상의 행이 있는 경우 ANALYZE절을 사용하여 적절한 처리 시간을 확보할 수 있도록 행 서브세트를 분석할 수 있습니다. ANALYZE절을 사용하려면 USER TEMPORARY 테이블 스페이스를 사용할 수 있어야 합니다. 이 테이블 스페이스의 페이지 크기를 8KB 이상으로 설정하고 이에 대한 USE 특권이 있는지 확인하십시오. 예를 들어, 다음 DDL문은 사용자 임시 테이블 스페이스와 페이지 크기가 동일한 버퍼 풀을 작성하고 모든 사용자에게 USE 특권을 부여합니다.

  ```
  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;
  ```

  또한 DB2 제어 센터를 사용하여 사용자 테이블 스페이스 및 해당 버퍼 풀을 작성할 수 있습니다.

컬럼에 공간 격자 인덱스를 작성하기 전에 Index Advisor를 사용하여 적절한 격자 크기를 판별할 수 있습니다.

이 태스크를 수행하려면

공간 격자 인덱스의 올바른 격자 크기를 판별하려면 다음을 수행하십시오.

1. 인덱스 어드바이저에 작성할 인덱스에 권장되는 격자 셀 크기를 요청하십시오.
   a. ADVISE 키워드와 함께 Index Advisor를 호출하는 명령을 입력하여 격자 셀 크기를 요청하십시오. 예를 들어, COUNTIES 테이블의 SHAPE 컬럼에 대한 Index Advisor를 호출하려면 다음을 입력하십시오.

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

      제한사항: 운영 체제 프롬프트에서 위 gseidx 명령을 입력한 경우 한 행에 전체 명령을 입력해야 합니다. 또는 여러 행에 명령을 분할할 수 있는 CLP 파일에서 gseidx 명령을 실행할 수 있습니다.

      Index Advisor는 권장되는 격자 셀 크기를 리턴합니다. 예를 들어, ADVISE 키워드를 사용하는 위 gseidx 명령은 SHAPE 컬럼에 대해 다음 권장 셀 크기를 리턴합니다.

```
Query Window Size  Suggested Grid Sizes              Cost
-----------------  --------------------------       ----
      0.1           0.7,       2.8,      14.0        2.7
      0.2           0.7,       2.8,      14.0        2.9
      0.5           1.4,       3.5,      14.0        3.5
        1           1.4,       3.5,      14.0        4.8
        2           1.4,       3.5,      14.0        8.2
        5           1.4,       3.5,      14.0         24
       10           2.8,       8.4,      21.0         66
       20           4.2,      14.7,      37.0        190
       50           7.0,      14.0,      70.0        900
      100          42.0,         0,         0       2800
```

b. 화면에 표시하는 좌표의 너비를 기반으로 gseidx 출력에서 적절한 쿼리 창 크기를 선택하십시오.

   이 예에서는 10진수(도) 표시의 위도 및 경도 값으로 좌표를 나타냅니다. 일반 맵 표시장치의 너비가 약 0.5도(약 55KM)인 경우, 쿼리 창 *크기* 컬럼에서 값이 0.5인 행으로 찾아가십시오. 이 행은 격자 *크기*를 1.4, 3.5 및 14.0으로 제안합니다.

2. 제안되는 격자 *크기*로 인덱스를 작성하십시오. 이전 단계의 예의 경우, 다음 SQL 문을 실행할 수 있습니다.

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

공간 격자 인덱스 통계 분석

전제조건

인덱스화하려는 데이터를 분석하려면 다음을 수행해야 합니다.

- 사용자 ID는 이 테이블에 대해 SELECT 특권이 있어야 합니다.
- 테이블에 백만 개 이상의 행이 있는 경우 ANALYZE절을 사용하여 적절한 처리 시간을 확보할 수 있도록 행 서브세트를 분석할 수 있습니다. ANALYZE절을 사용하려면 USER TEMPORARY 테이블 스페이스를 사용할 수 있어야 합니다. 이 테이블 스페이스의 페이지 *크기*를 8KB 이상으로 설정하고 이에 대한 USE 특권이 있는지 확인하십시오. 예를 들어, 다음 DDL문은 사용자 임시 테이블 스페이스와 동일한 페이지 *크기*의 버퍼 풀을 작성하고 모든 사용자에게 USE 특권을 부여합니다.

```
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;
```

   또한, 사용자 테이블 스페이스 및 해당 버퍼 풀을 작성하기 위해 DB2 제어 센터를 사용할 수도 있습니다.

기존 공간 격자 인덱스에 대한 통계로 인덱스가 효율적인지 또는 좀 더 효율적인 인덱스로 바꾸어야 하는지 여부를 알 수 있습니다. Index Advisor를 사용하여 이러한 통계를 확보하고 필요한 경우 인덱스를 바꾸십시오.

팁: 인덱스를 조정하는 것만큼이나 쿼리에 사용되는지 확인하는 작업도 중요합니다. 공간 인덱스를 사용하는지 여부를 판별하려면 DB2 제어 센터에서 Visual Explain을 실행하거나 쿼리시 db2exfmt와 같은 명령행 도구를 실행하십시오. Explain 출력의 『Access Plan』 섹션에 EISCAN 연산자 및 공간 인덱스의 이름이 표시되는 경우, 쿼리에서 인덱스를 사용하고 있는 것입니다.

이 태스크를 수행하려면

공간 격자 인덱스에 대한 통계를 확보하고 필요하면 인덱스를 바꾸십시오.

1. 인덱스 어드바이저가 기존 인덱스의 격자 셀 *크기*를 기본으로 하는 통계를 수집하도록 하십시오. 인덱스화된 데이터의 서브세트 또는 모든 데이터에 대한 통계를 요청할 수 있습니다.
   - 행 서브세트의 인덱스화된 데이터에 대한 통계를 확보하려면, gseidx 명령을 입력한 후 기존 인덱스 절과 DETAIL 키워드와 함께 ANALYZE 키워드와 매개변수를 지정하십시오. Index Advisor가 통계를 얻기 위해 분석해야 하는 행의 수 또는 백분율을 지정할 수 있습니다. 예를 들어, COUNTIES_SHAPE_IDX 인덱스로 인덱스화된 데이터 서브세트에 대한 통계를 얻으려면 다음을 입력하십시오.

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

   - 인덱스화된 모든 데이터에 대한 통계를 얻으려면 gseidx 명령을 입력하고 기존 인덱스 절을 지정하십시오. DETAIL 키워드를 포함합니다. 예를 들어, COUNTIES_SHAPE_IDX 인덱스의 Index Advisor를 호출하려면 다음을 입력하십시오.

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

   Index Advisor는 통계, 데이터의 막대 그래프 및 기존 인덱스에 대한 권장 셀 크기를 리턴합니다. 예를 들어, 위 gseidx 명령은 COUNTIES_SHAPE_IDX로 인덱스화된 모든 데이터에 대해 다음 통계를 리턴합니다.

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

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

   Number of occupied Grid Cells : 2922
   Index Entry/Geometry ratio    : 4.154292
   Geometry/Grid Cell ratio      : 1.004791
   ```

```
Maximum number of Geometries per Grid Cell: 14
Minimum number of Geometries per Grid Cell: 1

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



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

Grid Size                      : 0.0
No geometries indexed on this level.


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

Grid Size                      : 0.0
No geometries indexed on this level.


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

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

2. 기존 인덱스의 격자 셀 *크기*가 검색에 얼마나 도움이 되는지 판별하십시오. 이전 단계에서 리턴된 통계를 평가하십시오.

   팁:

   - 『인덱스 항목/Geometry 비율』 통계는 1 - 4 범위의 값으로서 1에 가까운 것이 바람직합니다.
   - Geometry당 인덱스 항목 수는 최대 격자 *크기*에서 10보다 적어야 오버플로우 레벨을 피할 수 있습니다.

     인덱스 어드바이저 출력에 『Grid Level X』 섹션이 있으면 오버플로우 레벨이 존재함을 나타냅니다.

   이전 단계에서 COUNTIES_SHAPE_IDX에 대해 확보한 인덱스 통계는 다음과 같은 이유로 격자 *크기*(0.5, 0, 0)가 이 컬럼의 데이터에 적합하지 않음을 나타냅니다.

   - Grid Level 1의 경우, 『Index Entry/Geometry ratio』 값 `4.154292`는 지침 4보다 큽니다.

     『Index Entries』 행의 값은 1, 2, 3, 4 및 10으로서 Geometry당 인덱스 항목 수를 나타냅니다. 각 『Index Entries』 컬럼 아래의 『Absolute』 값은 해당 특정 인덱스 항목 수를 갖는 Geometry 수를 나타냅니다. 예를 들어, 이전 단계의 출

력은 1519개의 Geometry에 네 개의 인덱스 항목이 있는 것으로 표시합니다. 열 개의 인덱스 항목에 대한『Absolute』값은 695개의 Geometry에 5 - 10개의 인덱스 항목이 있음을 나타내는 695입니다.

- 『Grid Level X』섹션이 있는 경우 오버플로우 인덱스 레벨이 존재함을 나타냅니다. 통계는 205개의 Geometry에 각각 열 개가 넘는 인덱스 항목이 있는 것으로 표시합니다.

3. 통계 결과에 만족하지 않는 경우, 다음 인덱스 어드바이저 출력의 Query Window Size 및 Suggested Grid Sizes 컬럼에 있는 해당 행과 Histogram 섹션을 확인하십시오.

   a. Geometry 수가 가장 많은 MBR *크기*를 찾으십시오.『Histogram』섹션에는 해당 MBR *크기*를 갖는 Geometry 수와 MBR *크기*가 나열됩니다. 다음 샘플 막대 그래프에서 최대 Geometry 수(437)는 MBR *크기* 0.5에 있습니다.

```
Histogram:
----------
    MBR Size             Geometry Count
    -------------------- --------------------
                0.040000                    1
                0.045000                    3
                0.050000                    1
                0.055000                    3
                0.060000                    3
                0.070000                    4
                0.075000                    3
                0.080000                    4
                0.085000                    1
                0.090000                    2
                0.095000                    1
                0.150000                   10
                0.200000                    9
                0.250000                   15
                0.300000                   23
                0.350000                   83
                0.400000                  156
                0.450000                  282
                0.500000                  437
                0.550000                  397
                0.600000                  341
                0.650000                  246
                0.700000                  201
                0.750000                  154
                0.800000                  120
                0.850000                   66
                0.900000                   79
                0.950000                   59
                1.000000                   47
                1.500000                  230
                2.000000                   89
                2.500000                   34
                3.000000                   10
                3.500000                    5
                4.000000                    3
                5.000000                    3
```

```
 5.500000                  2
 6.000000                  2
 6.500000                  3
 7.000000                  2
 8.000000                  1
15.000000                  3
25.000000                  2
30.000000                  1
```

b. 제안되는 격자 *크기*(1.4, 3.5, 14.0)를 확보하려면 값이 0.5인 Query Window Size 행을 찾아십시오.

```
Query Window Size  Suggested Grid Sizes             Cost
-----------------  --------------------------      ----
      0.1           0.7,       2.8,      14.0       2.7
      0.2           0.7,       2.8,      14.0       2.9
      0.5           1.4,       3.5,      14.0       3.5
        1           1.4,       3.5,      14.0       4.8
        2           1.4,       3.5,      14.0       8.2
        5           1.4,       3.5,      14.0        24
       10           2.8,       8.4,      21.0        66
       20           4.2,      14.7,      37.0       190
       50           7.0,      14.0,      70.0       900
      100          42.0,         0,         0      2800
```

4. 권장 *크기*가 2단계의 지침을 충족시키는지 검증하십시오. 다음과 같이 제안되는 격자 *크기*로 gseidx 명령을 실행하십시오.

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

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

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

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

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


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

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

Number of occupied Grid Cells : 56
Index Entry/Geometry ratio    : 2.344262
```

```
Geometry/Grid Cell ratio      : 1.089286
Maximum number of Geometries per Grid Cell: 10
Minimum number of Geometries per Grid Cell: 1

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


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

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

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

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

이제 통계는 다음과 같이 지침 범위의 값을 표시합니다.

- 『Index Entry/Geometry ratio』 값은 Grid Level 1의 경우 1.941599, Grid Level 2의 경우 2.344262 및 Grid Level 3의 경우 1.866667입니다. 이 값은 모두 지침 값 범위(1 - 4) 내에 있습니다.
- 『Grid Level X』 섹션이 없는 것은 오버플로우 레벨에 인덱스 항목이 없음을 의미합니다.

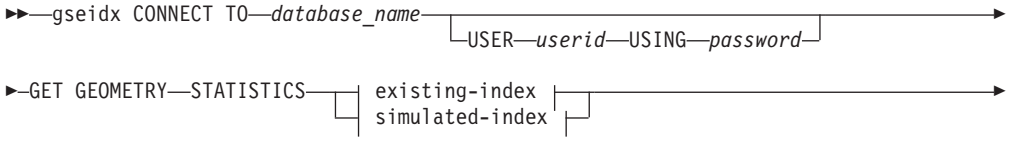
5. 기존 인덱스를 삭제하고 권장되는 격자 크기를 지정하는 인덱스로 대체하십시오. 이전 단계의 샘플의 경우, 다음 DDL 명령문을 실행하십시오.

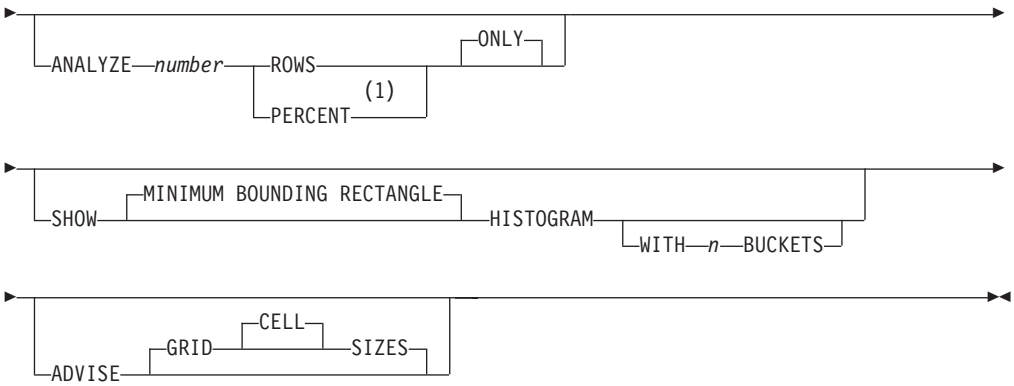
```
DROP INDEX userID.counties_shape_idx;
CREATE INDEX counties_shape_idx ON userID.counties(shape) EXTEND USING
DB2GSE.SPATIAL_INDEX(1.4,3.5,14.0);
```

gseidx 명령

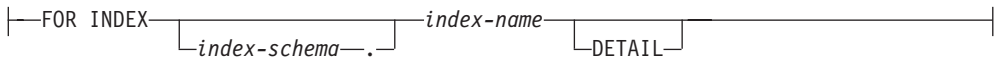
gseidx 명령을 사용하여 공간 격자 인덱스에 대한 Index Advisor를 호출하십시오.

구문

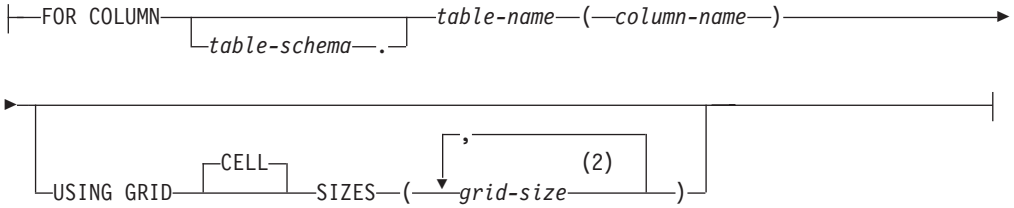




existing-index:



simulated-index:



주:

1 PERCENT 키워드 대신 백분율 부호(%)를 지정할 수 있습니다.

2 1, 2 또는 3의 격자 레벨에 셀 크기를 지정할 수 있습니다.

매개변수

database_name

공간 테이블이 상주하는 데이터베이스의 이름

userid 인덱스 또는 테이블이 상주하는 데이터베이스에 대해 SYSADM 또는 DBADM 권한이 있거나 테이블에 대해 SELECT 권한이 있는 사용자 ID. 데이터베이스 소유자의 사용자 ID로 DB2 명령 환경에 로그온하는 경우, gseidx 명령에 *userid* 및 *password*를 지정하지 않아도 됩니다.

password

사용자 ID의 암호

existing-index

기존 인덱스를 참조하여 통계를 수집합니다.

index-schema

기존 인덱스를 포함하는 스키마의 이름

index-name

기존 인덱스의 규정되지 않은 이름

DETAIL

각 격자 레벨에 대해 다음 정보가 표시됩니다.

- 격자 셀의 *크기*
- 인덱스화된 Geometry의 수
- 인덱스 항목의 수
- Geometry를 포함하는 격자 셀의 수
- Geometry당 평균 인덱스 항목 수
- 격자 셀당 평균 Geometry 수
- 가장 많은 Geometry를 포함하는 셀의 Geometry 수
- 가장 적은 Geometry를 포함하는 셀의 Geometry 수

simulated-index

테이블 컬럼 및 이 컬럼에 대해 시뮬레이트된 인덱스를 참조하십시오.

table-schema

시뮬레이트된 인덱스를 작성할 컬럼이 있는 테이블을 포함하는 스키마 이름

table-name

시뮬레이트된 인덱스를 작성할 컬럼이 있는 테이블의 규정되지 않은 이름

column-name

시뮬레이트된 인덱스를 작성할 테이블 컬럼의 규정되지 않은 이름

grid-size

시뮬레이트된 인덱스에 대한 각 격자 레벨(최고, 중간, 최저 레벨)의 셀 *크기*. 하나 이상의 레벨에 대해 셀 *크기*를 지정해야 합니다. 레벨을 포함시키지 않으려면 레벨에 대한 격자 셀 *크기*를 지정하지 말거나 격자 셀 *크기*를 제로(0.0)로 지정하십시오.

grid-size 매개변수를 지정할 경우, 인덱스 어드바이저는 기존의 인덱스 클로즈(clause)에 DETAIL 키워드를 포함시킬 때 리턴하는 것과 동일한 유형의 통계를 리턴합니다.

ANALYZE number ROWS | PERCENT ONLY

테이블 행의 서브세트에서 데이터에 대한 통계를 수집합니다. 테이블에 1백만 개가 넘는 행이 있는 경우, ANALYZE절을 사용하여 올바른 처리 시간을 확보할 수 있습니다. 이 서브세트에 포함시킬 행의 수 또는 백분율에 대한 근사치를 지정하십시오.

SHOW MINIMUM BOUNDING RECTANGLE HISTOGRAM

Geometry의 MBR(Minimum Bounding Rectangle) *크기* 및 MBR *크기*가 동일한 Geometry 수를 보여주는 도표를 표시합니다.

WITH n BUCKETS

분석된 모든 Geometry의 MBR에 대한 그룹화 수로 지정합니다. 작은 MBR은 다른 작은 Geometry에 그룹화됩니다. 대형 MBR은 다른 대형 Geometry에 그룹화됩니다.

이 매개변수를 지정하지 않거나 0 버켓을 지정하는 경우, Index Advisor가 대수 버켓 *크기*를 표시합니다. 예를 들어, MBR *크기*는 1.0, 2.0, 3.0,... 10.0, 20.0, 30.0,... 100.0, 200.0, 300.0,..과 같은 값일 수 있습니다.

0보다 큰 버켓 수를 지정하면, Index Advisor는 동일한 *크기*의 값을 표시합니다. 예를 들어, MBR *크기*가 8.0, 16.0, 24.0,... 320.0, 328.0, 334.0과 같은 동일한 *크기* 값이 될 수 있습니다.

디폴트값은 대수 *크기*의 버켓을 사용하는 것입니다.

ADVISE GRID CELL SIZES

최적에 근접한 격자 셀 *크기*를 계산합니다.

사용법 참고

운영 체제 프롬프트에서 gseidx 명령을 입력하는 경우, 단일 행에 전체 명령을 입력해야 합니다.

예

다음 예는 이름이 COUNTIES_SHAPE_IDX인 기존 격자 인덱스에 대한 세부 정보를 리턴하고 올바른 격자 인덱스 *크기*를 제안하도록 요청하는 것입니다.

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

뷰를 사용하여 공간 컬럼 액세스

다른 데이터 유형에 대해 DB2에 뷰를 정의한 방법과 동일하게 공간 컬럼을 사용하는 뷰를 정의할 수 있습니다.

공간 컬럼이 있는 테이블이 있고 뷰에서 이를 사용하려면 다음과 같은 소스 정보를 사용하십시오.

제 12 장 공간 정보 분석 및 생성

공간 컬럼을 채웠으면 공간 컬럼을 쿼리할 준비가 된 것입니다. 이 장에서는 다음 사항에 대해 다룹니다.

- 쿼리를 제출할 수 있는 환경에 대해 설명합니다.
- 쿼리에서 호출할 수 있는 여러 가지 공간 함수 유형에 대한 예를 제공합니다.
- 공간 인덱스와 결합하여 공간 함수를 사용하는 지침을 제공합니다.

공간 분석 수행 환경

다음과 같은 프로그래밍 환경에서 SQL 및 공간 함수를 사용하여 공간 분석을 수행할 수 있습니다.

- 대화식 SQL문

  DB2® Command Editor, DB2 명령창 또는 DB2 명령행 프로세서에서 대화식으로 SQL문을 입력할 수 있습니다.
- DB2에서 지원되는 모든 언어로 작성된 응용프로그램

공간 함수 작동 방법 예

DB2 Spatial Extender에는 공간 데이터에 대해 다양한 조작을 수행하는 함수가 제공됩니다. 일반적으로 이 함수는 수행하는 조작 유형에 따라 범주화될 수 있습니다. 표 4에서는 예와 함께 이 범주를 나열합니다. 다음 표 4의 텍스트는 예에 대한 코드입니다.

표 4. 공간 함수 및 조작

함수 범주	운영 예
특정 Geometry에 대한 정보 리턴	상점 10의 판매 영역에 대한 범위를 제곱 마일 단위로 리턴합니다.
비교 작성	고객의 집의 위치가 상점 10의 판매 영역 내에 있는지 여부를 판별합니다.
기존 Geometry에서 새 Geometry 파생	해당 위치로부터 상점의 판매 영역을 파생합니다.
Geometry를 데이터 교환 형식으로 (또는 그 반대로) 변환	DB2 데이터베이스에 정보를 추가할 수 있도록 GML 형식의 고객 정보를 Geometry로 변환합니다.

예 1: 특정 Geometry에 대한 정보 리턴

이 예에서 ST_Area 함수는 상점 10의 판매 영역을 나타내는 숫자값을 리턴합니다. 함수는 영역의 위치를 정의하는 데 사용된 좌표 시스템의 단위와 동일한 단위로 영역을 리턴합니다.

© Copyright IBM Corp. 1998, 2006

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

다음 예는 이전 예와 동일한 조작을 보여주나 메소드로 ST_Area를 호출하여 제곱 마일 단위로 영역을 리턴합니다.

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

예 2: 비교 작성

이 예에서 ST_Within 함수는 고객의 거주지를 나타내는 Geometry의 좌표를 상점 10의 판매 영역을 나타내는 Geometry의 좌표와 비교합니다. 함수의 출력은 거주지가 판매 영역 내에 있는지 여부를 나타냅니다.

```
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
```

예 3: 기존 Geometry에서 새 Geometry 도출

이 예에서 ST_Buffer 함수는 상점의 위치를 나타내는 Geometry에서 상점의 판매 영역을 나타내는 Geometry를 파생합니다.

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

다음 예에서는 이전 예와 동일한 조작을 보여주나 메소드로 ST_Buffer가 호출됩니다.

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

예 4: 데이터 교환 형식으로(또는 그 반대로) Geometry 변환

이 예에서 DB2 데이터베이스에 저장할 수 있도록 GML 형식의 고객 정보가 Geometry로 변환됩니다.

```
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) )
```

인덱스를 사용하여 쿼리를 최적화하는 함수

*비교 함수*라는 특수 공간 함수 그룹은 공간 격자 인덱스 또는 측지 Voronoi 인덱스(모두 *공간 인덱스*라고 함)를 사용하여 쿼리 성능을 향상시킬 수 있습니다. 이러한 각 함수는 두 개의 Geometry를 다른 Geometry와 비교합니다. 비교 결과 특정 기준을 충족시키는 경우, 함수는 1 값을 리턴합니다. 결과가 기준을 충족시키지 못하는 경우, 함수는 0 값을 리턴합니다. 비교를 수행할 수 없는 경우, 함수는 널(NULL) 값을 리턴할 수 있습니다.

예를 들어, ST_Overlaps 함수가 동일한 차원(예: 두 개의 선스트링 또는 두 개의 다각형)을 가진 두 개의 Geometry를 비교합니다. Geometry의 일부가 중복될 경우와 겹쳐지는 공간이 Geometry와 동일한 차원을 가질 경우, ST_Overlaps는 1 값을 리턴합니다.

표 5는 공간 격자 인덱스를 사용할 수 있는 비교 함수 및 측지 Voronoi 인덱스를 사용할 수 있는 비교 함수를 보여줍니다.

표 5. 공간 격자 인덱스 또는 측지 Voronoi 인덱스를 사용할 수 있는 비교 함수

비교 함수	공간 격자 인덱스 사용 가능	측지 **Voronoi** 인덱스 사용 가능
EnvelopesIntersect	예	예
ST_Contains	예	예
ST_Crosses	예	아니오
ST_Distance	예	예
ST_EnvIntersects	예	예
ST_Equals	예	아니오
ST_Intersects	예	예
ST_MBRIntersects	예	예
ST_Overlaps	예	아니오
ST_Touches	예	아니오
ST_Within	예	예

함수를 실행하려면 시간과 메모리가 필요하기 때문에 이를 실행하려면 상당한 처리가 필요할 수 있습니다. 또한 비교하는 Geometry가 복잡할 수록 비교 작업도 복잡하고 시간도 더 많이 소요됩니다. 위에 나열된 특수 함수는 공간 인덱스를 사용하여 Geometry를 찾을 수 있는 경우 해당 조작을 빨리 완료할 수 있습니다. 이러한 함수에서 공간 인덱스를 사용할 수 있도록 하려면 다음 규칙 모두를 준수하십시오.

- WHERE절에 함수를 지정해야 합니다. SELECT, HAVING 또는 GROUP BY절에 지정하는 경우, 공간 인덱스를 사용할 수 없습니다.
- 함수는 술어의 왼쪽에 있는 표현식이어야 합니다.

- 함수 결과와 다른 표현식을 비교하는 술어에서 사용하는 연산자는 ST_Distance 함수가 Less than 연산자를 사용해야 하는 경우를 제외하면 반드시 등호여야 합니다.
- 술어 오른쪽에 있는 표현식은 ST_Distance가 왼쪽의 함수인 경우를 제외하면 상수 1이어야 합니다.
- 해당 조작에는 공간 인덱스가 정의되어 있는 공간 컬럼 검색이 포함되어야 합니다.

예를 들면, 다음과 같습니다.

```
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
```

표 6에서는 공간 쿼리를 작성하여 공간 인덱스를 활용하는 올바른 방법과 잘못된 방법을 보여줍니다.

표 6. 공간 함수가 공간 인덱스를 활용하기 위한 규칙을 준수 및 위반할 수 있는 방법을 보여주는 예

공간 함수를 참조하는 쿼리	규칙 위반

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Contains(s.sales_zone,`
`ST_Point(-121.8,37.3, 1)) = 1`
	이 예에서는 어떤 조건도 위반되지 않았습니다.

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Length(s.location) > 10`
	공간 함수 ST_Length는 Geometry를 비교하지 않으므로 공간 인덱스를 활용할 수 없습니다.

`SELECT *`
`FROM stores AS s`
`WHERE 1=db2gse.ST_Within(s.location,:BayArea)`
	함수는 술어의 왼쪽에 있는 표현식이어야 합니다.

`SELECT *`
`FROM stores AS s`
`WHERE db2gse.ST_Contains(s.sales_zone,`
`ST_Point(-121.8,37.3, 1)) <> 0`
	등식 비교에서는 정수 상수 1을 사용해야 합니다.

`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`
	함수의 인수 어느 쪽에도 공간 인덱스가 없어 인덱스를 활용할 수 없습니다.

제 13 장 DB2 Spatial Extender 명령

이 장에서는 DB2 Spatial Extender를 설정하는 데 사용되는 명령에 대해 설명합니다. 프로젝트를 개발하기 위해 이러한 명령을 사용하는 방법에 대해서도 설명합니다.

DB2 Spatial Extender 설정 및 프로젝트 개발 명령 호출

db2se라는 명령행 처리기(CLP)를 사용하여 Spatial Extender를 설정하고 공간 데이터를 사용하는 프로젝트를 작성하십시오. 이 주제에서는 db2se를 사용하여 DB2 Spatial Extender 명령을 실행하는 방법에 대해 설명합니다.

전제조건

db2se 명령을 발행하려면 이러한 명령을 발행할 수 있는 권한이 있어야 합니다. 지정된 명령에 필요한 권한을 찾으려면, 명령과 연관된 스토어드 프로시저에 대한 자세한 정보는 118 페이지의 표 7을 참조하십시오. 예를 들어, db2se create_srs 명령에는 db2.ST_create_srs 스토어드 프로시저와 동일한 권한이 필요합니다.

예외: db2se shape_info 명령은 스토어드 프로시저를 호출하지 않습니다. 이 명령은 모양 파일의 컨텐츠에 대한 정보를 표시합니다.

운영 체제 프롬프트에서 db2se 명령을 입력하십시오.

지정 가능한 부속 명령 및 매개변수를 알아 보려면 다음을 수행하십시오.

- db2se 또는 db2se -h를 입력한 다음 Enter를 누르십시오. db2se 부속 명령 목록이 표시됩니다.
- db2se와 부속 명령 또는 db2se와 부속 명령 다음에 -h를 입력하십시오. 그런 다음, Enter를 누르십시오. 부속 명령에 필요한 구문이 표시됩니다. 이 구문에서,
  - 각 매개변수 앞에는 대시(dash)가 오고 뒤에는 매개변수 값을 위한 플레이스 홀더가 옵니다.
  - 대괄호를 두른 매개변수는 선택적입니다. 나머지 매개변수는 필수입니다.

중요사항: 편의를 위해 모니터에서 명령 구문을 대화식으로 검색할 수 있습니다. 다른 곳에서 구문을 찾지 않아도 됩니다.

db2se 명령을 발행하려면 `db2se`를 입력하십시오. 그런 다음 부속 명령을 입력하고 그 뒤에 부속 명령에 필요한 매개변수 및 매개변수 값을 입력하십시오. 마지막으로, Enter를 누르십시오.

© Copyright IBM Corp. 1998, 2006

지정한 데이터베이스로의 액세스를 부여하는 사용자 ID 및 암호를 입력해야 하는 경우도 있습니다. 예를 들어, 본인이 아닌 다른 사용자로서 데이터베이스에 연결하려면 해당 사용자 ID 및 암호를 입력하십시오. 항상 ID 앞에 매개변수 사용자 ID를 입력하고 암호 앞에는 매개변수 pw를 입력하십시오.

사용자 ID 및 암호를 지정하지 않으면 디폴트로 현재 사용자 ID 및 암호가 사용됩니다. 입력한 값에서는 디폴트로 대소문자를 구분하지 않습니다. 대소문자를 구분하게 하려면 값을 큰따옴표로 묶으십시오. 예를 들어, 소문자 테이블 이름 mytable을 지정하고 mytable을 입력하십시오.

시스템 프롬프트(쉘)에서 해석되지 않았는지 확인하려면 따옴표를 제거해야 합니다. 예를 들어, ₩″mytable₩″을 지정하십시오. 대소문자가 구분되는 값을 다른 대소문자가 구분되는 값으로 규정한 경우 두 값을 구분하십시오(예: ″myschema″.″mytable″). 문자열을 큰따옴표로 묶으십시오(예: ″select * from newtable″).

db2se 명령이 실행될 때, 그 명령에 해당하는 스토어드 프로시저가 호출되고 요청한 조작이 수행됩니다.

db2se 명령 개요

다음 테이블은 Spatial Extender 설정 및 공간 데이터를 사용하는 프로젝트 작성과 관련된 태스크를 수행하기 위해 발행하는 db2se 명령을 보여줍니다. 이 테이블에서는 db2se 명령의 예와 권한 및 명령 고유 매개변수에 대한 정보를 제공합니다. 태스크 오른쪽에 있는 두 번째 컬럼에 스토어드 프로시저 정보에 대한 링크 또는 참조가 있습니다. 이 스토어드 프로시저는 명령이 발행될 때 호출됩니다. 스토어드 프로시저를 사용할 수 있는 권한은 해당 명령을 사용할 수 있는 권한과 동일하며, 명령과 스토어드 프로시저는 동일한 매개변수를 공유합니다. 매개변수의 권한 부여 및 의미에 대한 자세한 정보는 참조에서 식별된 절을 참조하십시오.

표 7. 태스크별로 인덱스화된 db2se 명령

태스크	명령 및 예
좌표 시스템 작성	db2se create_cs 명령 고유 매개변수 및 필수 권한은 db2gse.ST_create_coordsys 스토어드 프로시저에 대한 매개변수 및 필수 권한과 동일합니다. 다음 예는 『mycoordsys』 좌표 시스템을 작성합니다. `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]]`

표 7. 태스크별로 인덱스화된 db2se 명령 (계속)

태스크	명령 및 예
공간 참조 시스템 작성	db2se create_srs 명령 고유 매개변수는 스토어드 프로시저의 매개변수와 동일합니다. 필요한 권한이 없습니다. 다음 예는 『mysrs』 공간 참조 시스템을 작성합니다. `db2se create_srs mydb -srsName ₩"mysrs₩" -srsID 100 -xScale 10 -coordsysName ₩"GCS_North_American_1983₩"`
공간 참조 시스템 삭제	db2se drop_srs 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 예는 이름이 『mysrs』인 공간 참조 시스템을 삭제합니다. `db2se drop_srs mydb -srsName ₩"mysrs₩"`
좌표 시스템 정의 삭제	db2se drop_cs 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 예는 『mycoordsys』 좌표 시스템을 삭제합니다. `db2se drop_cs mydb -coordsysName ₩"mycoordsys₩"`
데이터를 자동으로 지오코딩(Geocoding)하는 설정 사용 안함	db2se disable_autogc 명령 고유 매개변수 및 필수 권한은 db2gse.ST_disable_autogeocoding 스토어드 프로시저에 대한 매개변수 및 필수 권한과 동일합니다. 다음 예는 MYTABLE 테이블에서 MYCOLUMN이라는 지역코드화된 컬럼의 자동 지오코딩(Geocoding)을 사용 불가능하게 합니다. `db2se disable_autogc mydb -tableName ₩"mytable₩" -columnName ₩"mycolumn₩"`
공간 데이터 관련 작업에 대해 데이터베이스 사용 가능화	db2se enable_db 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 예는 공간 데이터 관련 작업에 대한 MYDB 데이터베이스를 사용할 수 있습니다. `db2se enable_db mydb`

표 7. 태스크별로 인덱스화된 db2se 명령 (계속)

태스크	명령 및 예

SDE 전송 파일로 데이터 익스포트	db2se export_sde
	
	명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다.
	
	다음 예는 MYSPATIALCOLUMN 공간 컬럼이 있는 MYSDETABLE 테이블에서 mysdefile이라는 SDE 전송 파일로 데이터를 익스포트합니다.
	`db2se export_sde mydb -tableName ₩"mySDEtable₩" -columnName ₩"mySpatialcolumn₩" -fileName /home/myaccount/mysdefile`
	
	다음 예는 데이터를 SPATIALTABLE 테이블에서 DB2 클라이언트에 작성할 SDE 파일 sdex를 익스포트합니다. 오류 및 정보 메시지(예: 익스포트가 시작 및 완료되는 시간 및 익스포트된 행 수)가 sdex.export.log라는 파일에 작성됩니다.
	`db2se export_sde mydb -client -fileName sdex -selectStatement "SELECT * FROM spatialTable" -messagesFile sdex.export.log`

모양 파일로 데이터 익스포트	db2se export_shape 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 예는 MYCOLUMN이라는 공간 컬럼 및 연관 테이블 MYTABLE을 myshapefile이라는 모양 파일로 익스포트합니다. `db2se export_shape mydb -fileName /home/myaccount/myshapefile -selectStatement "select * from mytable"`

SDE 전송 파일 임포트	db2se import_sde
	
	명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다.
	
	다음 예는 SDE 전송 파일 mysdefile을 MYSPATIALCOLUMN 공간 컬럼이 포함된 MYSDETABLE 테이블로 임포트합니다. 10개의 레코드마다 커미트가 발행됩니다.
	`db2se import_sde mydb -tableName ₩"mysdetable₩" -columnName ₩"mySpatialcolumn₩" -fileName /home/myaccount/"mysdefile" -commitScope 10`
	
	다음 예에서는 DB2 클라이언트가 상주하는 SDE 파일 sdex를 임포트하는 방법을 보여줍니다. 이 예에서 SDETABLE(컬럼 이름 지정된 ID로)이라는 테이블로 데이터가 임포트되고 100개의 레코드마다 커미트가 발행됩니다. 모든 오류는 sdex.exceptions 파일에 기록됩니다.
	`db2se import_sde mydb -client -filename sdex -srsId 1234 -tableName sdeTable -idColumn id -commitScope 100 -messagesFile sdex.exceptions`

표 7. 태스크별로 인덱스화된 db2se 명령 (계속)

태스크	명령 및 예
모양 파일 임포트	db2se import_shape 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 명령은 myfile 모양 파일을 MYTABLE 테이블로 임포트합니다. 임포트하는 동안 myfile의 공간 데이터는 MYTABLE 컬럼 MYCOLUMN에 삽입됩니다. `db2se import_shape mydb -fileName ₩"myfile₩" -srsName NAD83_SRS_1 -tableName ₩"mytable₩" -spatialColumnName ₩"mycolumn₩"`
지오코더(Geocoder) 등록	db2se register_gc 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 예는『myschema.myfunction』함수에서 구현된『mygeocoder』지오코더(Geocoder)를 등록합니다. `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"`
공간 컬럼 등록	db2se register_spatial_column 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 예는『USA_SRS_1』공간 참조 시스템과 함께 MYTABLE 테이블의 MYCOLUMN 공간 컬럼을 등록합니다. `db2se register_spatial_column mydb -tableName ₩"mytable₩" -columnName ₩"mycolumn₩" -srsName USA_SRS_1`
공간 데이터 관련 작업에 대해 데이터베이스를 사용 가능화하는 자원 제거	db2se disable_db 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 예는 공간 데이터 관련 작업에 대해 MYDB 데이터베이스를 사용할 수 있도록 하는 자원을 제거합니다. `db2se disable_db mydb`
지오코딩(Geocoding) 작업에 대한 설정 제거	db2se remove_gc_setup 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 예는 MYTABLE 테이블의 MYCOLUMN 공간 컬럼에 적용되는 지오코딩(Geocoding) 작업에 대한 설정을 제거합니다. `db2se remove_geocoding_setup mydb -tableName ₩"mytable₩" -columnName ₩"mycolumn₩"`

표 7. 태스크별로 인덱스화된 db2se 명령 (계속)

태스크	명령 및 예
일괄처리 모드로 지오코더(Geocoder) 실행	db2se run_gc 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 예는 일괄처리 모드에서 지오코더(Geocoder)를 실행하여 MYTABLE 테이블에서 MYCOLUMN 컬럼의 데이터를 채웁니다. `db2se run_gc mydb -tableName ₩"mytable₩" -columnName ₩"mycolumn₩"`
자동으로 실행되도록 지오코더(Geocoder) 설정	db2se enable_autogeocoding 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 예는 MYTABLE 테이블의 MYCOLUMN 컬럼에 대한 자동 지오코딩(Geocoding)을 설정합니다. `db2se enable_autogeocoding mydb -tableName ₩"mytable₩" -columnName ₩"mycolumn₩"`
지오코딩(Geocoding) 조작 설정	db2se setup_gc 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 예는 지오코딩(Geocoding) 작업을 설정하여 MYTABLE 테이블의 MYCOLUMN 공간 컬럼의 데이터를 채웁니다. `db2se setup_gc mydb -tableName ₩"mytable₩" -columnName ₩"mycolumn₩" -geocoderName ₩"db2se_USA_GEOCODER₩" -parameterValues "address,city,state,zip,2,90,70,20,1.1,'meter',4.." -autogeocodingColumns address,city,state,zip commitScope 10`

모양 파일 및 내용에 대한 정보 표시	db2se shape_info
	
	이 명령을 사용하려면 다음 조건이 충족되어야 합니다.
	• 명령을 참조하는 파일을 읽을 사용 권한을 얻으십시오.
	• 이 파일이 포함되는 데이터베이스에 연결할 수 있습니다(*–database* 매개변수를 사용하는 경우, 이 매개변수는 시스템이 호환 가능한 좌표 시스템 및 공간 참조 시스템에 대해 이름 지정된 데이터베이스를 검색하도록 지정함).
	
	다음 예에서는 현재 디렉토리에 있는 myfile 모양 파일에 대한 정보를 보여줍니다.
	
	`db2se shape_info -fileName myfile`
	
	다음 예에서는 샘플 UNIX 모양 파일 offices에 대한 정보를 보여줍니다. *–database* 매개변수는 이름 지정된 데이터베이스(이 경우 MYDB)에서 호환 가능한 모든 좌표 시스템 및 공간 참조 시스템을 찾습니다.
	
	`db2se shape_info -fileName ~/sqllib/samples/extenders/spatial/data/offices -database myDB`

표 7. 태스크별로 인덱스화된 db2se 명령 (계속)

태스크	명령 및 예

SDE 파일 및 내용에 대한 정보 표시	db2se sde_info
	
	이 명령을 사용하려면 다음 조건이 충족되어야 합니다.
	• 명령을 참조하는 파일을 읽을 사용 권한을 얻으십시오.
	• 이 파일이 포함되는 데이터베이스에 연결할 수 있습니다(*–database* 매개변수를 사용하는 경우, 이 매개변수는 시스템이 호환 가능한 좌표 시스템 및 공간 참조 시스템에 대해 이름 지정된 데이터베이스를 검색하도록 지정함).
	
	다음 예에서는 현재 디렉토리에 있는 SDE 파일 sdefile에 대한 정보를 보여줍니다.
	
	`db2se sde_info -fileName myfile`
	
	다음 예는 sdex라는 SDE 파일에 대한 정보를 보여주며, 호환 가능한 모든 좌표 시스템 및 공간 참조 시스템에 해당하는 이름 지정된 MYDB 데이터베이스를 검색합니다.
	
	`db2se sde_info -fileName data/sdex -database myDB`

지오코더(Geocoder) 등록 취소	db2se unregister_gc 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 예는 『mygeocoder』 지오코더(Geocoder)를 등록해제합니다. `db2se unregister_gc mydb -geocoderName W"mygeoco derW"`
공간 컬럼 등록 취소	db2se unregister_spatial_column 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 예는 MYTABLE 테이블의 MYCOLUMN 특수 컬럼을 등록해제합니다. `db2se unregister_spatial_column mydb -tableName W"mytableW" -columnName W"mycolumnW"`
좌표 시스템 정의 갱신	db2se alter_cs 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 예는 새 소속 이름을 사용하여 『mycoordsys』 좌표 시스템의 정의를 갱신합니다. `db2se alter_cs mydb -coordsysName W"mycoordsysW" -organization myNeworganizationb -tableName W"mytableW"`

표 7. *태스크별로 인덱스화된 db2se 명령 (계속)*

태스크	명령 및 예
공간 참조 시스템 정의 갱신	db2se alter_srs 명령 고유 매개변수와 필수 권한 부여는 스토어드 프로시저의 매개변수 및 필수 권한 부여와 동일합니다. 다음 예는 다른 xOffset 및 설명을 사용하여 『mysrs』 공간 참조 시스템을 변경합니다. `db2se alter_srs mydb -srsName ₩"mysrs₩" -xOffset 35 -description "This is my own spatial reference system."`

제 14 장 샘플 프로그램 사용 및 응용프로그램 작성

이 장에서는 Spatial Extender 응용프로그램 작성 방법에 대해 설명합니다.

DB2 Spatial Extender용 응용프로그램 작성

DB2 Spatial Extender 스토어드 프로시저 또는 함수를 호출하는 응용프로그램을 작성하려면 다음 태스크 및 참조 정보를 읽으십시오.

이 태스크를 수행하려면

공간 응용프로그램에 DB2 Spatial Extender 헤더 파일 포함

DB2 Spatial Extender는 DB2 Spatial Extender의 스토어드 프로시저 및 함수에서 사용 가능한 상수를 정의하는 헤더 파일을 제공합니다.

권장사항:
C 또는 C++ 프로그램에서 DB2 Spatial Extender 스토어드 프로시저 또는 함수를 호출하려면 공간 응용프로그램에 이 헤더 파일을 포함시키십시오.

이 태스크를 수행하려면

1. DB2 Spatial Extender 응용프로그램이 이 헤더 파일에 있는 필요한 정의를 사용할 수 있는지 확인하십시오.
   a. 사용하는 응용프로그램에 DB2 Spatial Extender 헤더 파일을 포함시키십시오. 헤더 파일의 이름은 다음과 같습니다.

      ```
      db2gse.h
      ```

      헤더 파일은 *db2path*/include 디렉토리에 있으며 *db2path*는 DB2 데이터베이스 시스템이 설치된 설치 디렉토리입니다.
   b. `makefile`에 포함 디렉토리의 경로를 해당 컴파일 옵션과 함께 지정하십시오.
2. Windows 32비트 시스템에서 Windows 64비트 응용프로그램을 빌드하는 경우 `samples/extenders/spatial/makefile.nt` 파일에서 DB2_LIBS 매개변수를 변경하여 64비트 응용프로그램과 호환되도록 하십시오. 필요한 변경사항은 아래에 강조표시되어 있습니다.

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

© Copyright IBM Corp. 1998, 2006

응용프로그램에서 DB2 Spatial Extender 스토어드 프로시저 호출

DB2 Spatial Extender 스토어드 프로시저를 호출하는 응용프로그램을 작성할 계획인 경우, SQL CALL문을 사용하고 해당 스토어드 프로시저의 이름을 지정합니다.

공간 데이터 관련 작업에 대해 데이터베이스를 사용 가능화할 때 DB2 Spatial Extender 스토어드 프로시저가 작성됩니다.

이 태스크를 수행하려면

1. ST_enable_db 스토어드 프로시저를 호출하여 공간 작업에 데이터베이스를 사용하십시오.

   다음과 같이 스토어드 프로시저를 지정하십시오.

   ```
   CALL db2gse!ST_enable_db
   ```

   이 호출에서 db2gse!는 DB2 Spatial Extender 라이브러리 이름을 나타냅니다. ST_enable_db 스토어드 프로시저는 호출에 느낌표(즉, db2gse!)를 포함시켜야 하는 유일한 프로시저입니다.
2. 기타 DB2 Spatial Extender 스토어드 프로시저를 호출하십시오. 다음 양식으로 스토어드 프로시저 이름을 지정하십시오. 여기서 db2gse는 모든 DB2 Spatial Extender 스토어드 프로시저의 스키마 이름이고 *spatial_procedure_name*는 스토어드 프로시저의 이름입니다. 호출에 느낌표는 포함시키지 마십시오.

   ```
   CALL db2gse.spatial_procedure_name
   ```

   다음 테이블에 DB2 Spatial Extender 스토어드 프로시저가 나와 있습니다.

표 8.

스토어드 프로시저	설명
GSE_export_sde	공간 컬럼 및 그와 연관된 테이블을 SDE 전송 파일로 익스포트합니다.
GSE_import_sde	SDE 전송 파일을 데이터베이스로 임포트합니다.
ST_alter_coordsys	데이터베이스에서 좌표 시스템의 속성을 갱신합니다.
ST_alter_srs	데이터베이스에서 공간 참조 시스템의 속성을 갱신합니다.
ST_create_coordsys	데이터베이스에 좌표 시스템을 작성합니다.
ST_create_srs	데이터베이스에 공간 참조 시스템을 작성합니다.
ST_disable_autogeocoding	DB2 Spatial Extender가 지역코드화된 컬럼과 그와 연관된 지오코딩(Geocoding) 컬럼의 동기화를 중지하도록 지정합니다.
ST_disable_db	DB2 Spatial Extender가 공간 데이터를 저장하고 이 데이터에 대해 수행되는 조작을 지원하는 데 사용되는 자원을 제거합니다.
ST_drop_coordsys	데이터베이스에서 좌표 시스템을 삭제합니다.
ST_drop_srs	데이터베이스에서 공간 참조 시스템을 삭제합니다.
ST_enable_autogeocoding	DB2 Spatial Extender가 지역코드화된 컬럼과 그와 연관된 지오코딩(Geocoding) 컬럼을 동기화하도록 지정합니다.

표 8. (계속)

스토어드 프로시저	설명
ST_enable_db	데이터베이스가 공간 데이터를 저장하고 조작을 지원하는 데 필요한 자원을 데이터베이스에 제공합니다.
ST_export_shape	데이터베이스의 선택된 데이터를 모양 파일로 익스포트합니다.
ST_import_shape	모양 파일을 데이터베이스로 임포트합니다.
ST_register_geocoder	DB2 Spatial Extender 제품의 일부인 DB2SE_USA_GEOCODER 이외의 지오코더(Geocoder)를 등록합니다.
ST_register_spatial_column	공간 컬럼을 등록하고 이 컬럼에 공간 참조 시스템을 연관시킵니다.
ST_remove_geocoding_setup	지역코드화된 컬럼에 대한 모든 지오코딩(Geocoding) 설정 정보를 제거합니다.
ST_run_geocoding	일괄처리 모드로 지오코더(Geocoder)를 실행합니다.
ST_setup_geocoding	지오코딩(Geocoding)될 컬럼을 지오코더(Geocoder)와 연관시키고 해당 지오코딩(Geocoding) 매개변수 값을 설정합니다.
ST_unregister_geocoder	DB2SE_USA_GEOCODER 이외의 지오코더(Geocoder)를 등록 취소합니다.
ST_unregister_spatial_ column	공간 컬럼의 등록을 제거합니다.

DB2 Spatial Extender 샘플 프로그램

DB2® Spatial Extender 샘플 프로그램인runGseDemo에는 두 가지 목적이 있습니다. 샘플 프로그램을 사용하여 DB2 Spatial Extender용 응용프로그램 프로그래밍에 익숙해지고 프로그램을 사용하여 DB2 Spatial Extender 설치를 검증할 수 있습니다.

- UNIX®의 경우 다음 경로에서 runGseDemo 프로그램을 찾을 수 있습니다.

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

  여기서 $HOME은 인스턴스 소유자의 홈 디렉토리입니다.

- Windows®의 경우 다음 경로에서 runGseDemo 프로그램을 찾을 수 있습니다.

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

  여기서 c:¥Program Files¥IBM¥sqllib는 DB2 Spatial Extender를 설치한 디렉토리입니다.

DB2 Spatial Extender runGseDemo 샘플 프로그램을 사용하면 응용프로그램을 보다 쉽게 프로그래밍할 수 있습니다. 이 샘플 프로그램을 사용하여 공간 데이터 관련 작업에 필요한 데이터베이스를 사용 가능하게 하고 해당 데이터베이스에서 데이터에 대한 공간 분석을 수행할 수 있습니다. 이 데이터베이스에는 고객 및 범람 지역에 대한 가상 정보가 있는 테이블이 있습니다. 이 정보로부터 Spatial Extender를 실험하여 어떤 고객이 홍수 피해를 입을 위험에 처해 있는지 판별할 수 있습니다.

샘플 프로그램으로 다음을 수행할 수 있습니다.

- 일반적으로 공간적으로 사용 가능한 데이터베이스를 작성 및 유지보수하는 데 필요한 단계를 참조하십시오.
- 응용프로그램에서 공간 스토어드 프로시저를 호출하는 방법에 대해 이해하십시오.
- 샘플 코드를 사용자의 응용프로그램으로 잘라내서 붙여넣으십시오.

DB2 Spatial Extender에 대한 태스크를 코딩하려면 다음 샘플 프로그램을 사용하십시오. 예를 들어, 데이터베이스 인터페이스를 사용하는 응용프로그램을 작성하여 DB2 Spatial Extender 스토어드 프로시저를 호출한다고 가정해 보십시오. 샘플 프로그램에서 응용프로그램을 사용자 정의하는 코드를 복사할 수 있습니다. DB2 Spatial Extender의 프로그래밍 단계에 친숙하지 않은 경우, 각 단계를 자세하게 보여주는 샘플 프로그램을 실행할 수 있습니다. 샘플 프로그램 실행 지시사항은 이 주제의 끝부분에 있는 『관련 태스크』를 참조하십시오.

다음 테이블에서는 샘플 프로그램의 각 단계에 대해 설명합니다. 각 단계에서 조치를 수행하고, 많은 경우에 해당 조치를 실행 취소합니다. 예를 들어, 첫 번째 단계에서 공간 데이터베이스를 사용한 다음 공간 데이터베이스를 사용하지 않게 할 수 있습니다. 이러한 방법으로 여러 Spatial Extender 스토어드 프로시저에 익숙해질 수 있습니다.

표 9. DB2 Spatial Extender 샘플 프로그램 단계

단계	조치 및 설명
공간 데이터베이스 사용 또는 사용 안함	• 공간 데이터베이스 사용 이것이 DB2 Spatial Extender를 사용하는 데 필요한 첫 단계입니다. 공간 데이터 관련 작업에 사용된 데이터베이스에는 공간 유형 세트, 공간 함수 세트, 공간 관련 술어 세트, 새 인덱스 유형, 공간 카탈로그 테이블 및 뷰 세트가 있습니다. • 공간 데이터베이스 사용 안함 이 단계는 일반적으로 잘못된 데이터베이스에 대해 공간 성능을 사용하거나 이 데이터베이스에서 공간 데이터 관련 작업을 수행할 필요가 없을 때 수행됩니다. 공간 데이터베이스를 사용하지 않을 때에는 공간 유형 세트, 공간 함수 세트, 공간 관련 술어 세트, 새 인덱스 유형, 해당 데이터베이스에 연관된 공간 카탈로그 테이블 및 뷰 세트를 제거합니다. • 공간 데이터베이스 사용 위와 동일합니다.

표 9. DB2 Spatial Extender 샘플 프로그램 단계 (계속)

단계	조치 및 설명
좌표 시스템을 작성 또는 삭제	• NORTH_AMERICAN이라는 좌표 시스템을 작성하십시오. 이 단계는 데이터베이스에 새 좌표 시스템을 작성합니다. • NORTH_AMERICAN이라는 좌표 시스템을 삭제하십시오. 이 단계에서는 데이터베이스에서 NORTH_AMERICAN 좌표 시스템을 삭제합니다. • KY_STATE_PLANE이라는 좌표 시스템을 작성하십시오. 이 단계에서는 다음 단계에서 작성되는 공간 참조 시스템에서 사용할 KY_STATE_PLANE이라는 새 좌표 시스템을 작성합니다.
공간 참조 시스템 작성 또는 삭제	• SRSDEMO1이라는 공간 참조를 작성하십시오. 이 단계는 좌표를 해석하는 데 사용되는 새 공간 참조 시스템(SRS)을 정의합니다. SRS는 공간적으로 사용 가능한 데이터베이스 컬럼에 저장할 수 있는 형식의 Geometry 데이터를 포함합니다. SRS를 특정 공간 컬럼에 등록한 후 해당 공간 컬럼에 적용 가능한 좌표를 연관된 CUSTOMERS 테이블 컬럼에 저장할 수 있습니다. • SRSDEMO1 SRS 삭제 이 단계는 데이터베이스에 더 이상 SRS가 필요하지 않은 경우에 수행됩니다. SRS를 삭제하면 데이터베이스에서 SRS 정의가 제거됩니다.
공간 테이블 작성 및 데이터 채우기	• KY_STATE_SRS SRS 작성 • CUSTOMERS 테이블 작성 • CUSTOMERS 테이블을 채우십시오. CUSTOMERS 테이블은 몇 년 동안 데이터베이스에 저장된 비즈니스 데이터를 나타냅니다. • LOCATION 컬럼을 추가하여 CUSTOMERS 테이블 변경 ALTER TABLE문은 ST_Point 유형의 새 컬럼(LOCATION)을 추가합니다. 이 컬럼은 다음 단계에서 주소 컬럼을 지오코딩(Geocoding)하여 데이터 상주됩니다. • OFFICES 테이블 작성 OFFICES 테이블은 보험 회사의 각 사무소에 대한 영업 구역을 나타냅니다. 전체 테이블은 다음 단계에서 DB2가 아닌 데이터베이스의 속성 데이터와 함께 데이터 상주됩니다. 이 다음 단계에는 SHAPE 파일로부터 OFFICES 테이블로 속성 데이터를 임포트하는 과정이 포함됩니다.

표 9. DB2 Spatial Extender 샘플 프로그램 단계 (계속)

단계	조치 및 설명
컬럼 채우기	• KY_STATE_GC라는 지오코더(Geocoder)를 사용하여 CUSTOMERS 테이블의 LOCATION 열에 대한 주소 데이터를 지오코드하십시오. 이 단계에서는 지오코더(Geocoder) 유틸리티를 호출하여 일괄처리 공간 지오코딩(Geocoding)을 수행합니다. 일괄처리 지오코딩(Geocoding)은 테이블의 중요 부분이 지오코딩(Geocoding) 또는 다시 지오코딩(Geocoding)되어야 할 때 보통 수행됩니다. • KY_STATE_SRS 공간 참조 시스템을 사용하여 모양 파일로부터 이전에 작성된 OFFICES 테이블을 로드하십시오. 이 단계에서는 모양 파일 형식의 기존 공간 데이터가 있는 OFFICES 테이블을 로드합니다. OFFICES 테이블이 존재하기 때문에 LOAD 유틸리티는 새로운 레코드를 기존의 테이블에 추가합니다. • KY_STATE_SRS 공간 참조 시스템을 사용하여 모양 파일에서 FLOODZONES 테이블 작성 및 로드 이 단계는 모양 파일의 양식에서 기존 데이터가 있는 FLOODZONES 테이블을 로드합니다. 테이블이 존재하지 않기 때문에 로드 유틸리티는 데이터를 로드하기 전에 테이블을 작성합니다. • KY_STATE_SRS 공간 참조 시스템을 사용하여 모양 파일에서 REGIONS 테이블 작성 및 로드
지오코더(Geocoder) 등록 및 등록 해제	• SAMPLEGC 지오코더(Geocoder) 등록 • SAMPLEGC 지오코더(Geocoder) 등록해제 • KY_STATE_GC 지오코더(Geocoder) 등록 이러한 단계는 SAMPLEGC 지오코더(Geocoder)를 등록 및 등록해제한 다음 새로운 KY_STATE_GC 지오코더(Geocoder)를 작성하여 샘플 프로그램에서 사용합니다.
공간 인덱스 작성	• CUSTOMERS 테이블의 LOCATION 컬럼에 대한 공간 격자 인덱스 작성 • CUSTOMERS 테이블의 LOCATION 컬럼에 대한 공간 격자 인덱스 삭제 • CUSTOMERS 테이블의 LOCATION 컬럼에 대한 공간 격자 인덱스 작성 • OFFICES 테이블의 LOCATION 컬럼에 대한 공간 격자 인덱스 작성 • FLOODZONES 테이블의 LOCATION 컬럼에 대한 공간 격자 인덱스 작성 • REGIONS 테이블의 LOCATION 컬럼에 대한 공간 격자 인덱스 작성 이 단계는 CUSTOMERS, OFFICES, FLOODZONES 및 REGIONS 테이블에 대한 공간 격자 인덱스를 작성합니다.

표 9. DB2 Spatial Extender 샘플 프로그램 단계 (계속)

단계	조치 및 설명
자동 지오코딩(Geocoding) 사용	• KY_STATE_GC지오코더(Geocoder)를 사용하여 CUSTOMERS 테이블의 LOCATION 컬럼에 대한 지오코딩(Geocoding)을 설정합니다. 이 단계에서는 CUSTOMERS 테이블의 LOCATION 컬러의 KY_STATE_GC 지오코더(Geocoder)에 연관시키며 해당 지오코딩(Geocoding) 매개변수 값을 설정합니다. • CUSTOMERS 테이블의 LOCATION 컬럼에 대한 자동 지오코딩(Geocoding) 사용 이 단계에서는 지오코더(Geocoder)의 자동 호출을 설정합니다. 자동 지오코딩(Geocoding)을 사용하면 CUSTOMERS 테이블의 LOCATION, STREET, CITY, STATE 및 ZIP 컬럼이 후속 삽입 및 갱신 조작에 대해 서로 동기화됩니다.
USTOMERS 테이블에 대한 삽입, 갱신 및 삭제 조작 수행	• 시가가 다른 일부 레코드 삽입 • 새 주소가 있는 일부 레코드 갱신 • 테이블에서 모든 레코드 삭제 이러한 단계는 CUSTOMERS 테이블의 STREET, CITY, STATE 및 ZIP 컬럼에 대한 삽입, 갱신 및 삭제 조작에 대해 설명합니다. 자동 지오코딩(Geocoding)을 사용 가능하게 하면, 이 컬럼에 삽입 또는 갱신된 데이터는 LOCATION 컬럼으로 자동 지오코딩(Geocoding)됩니다. 이 프로세스는 이전 단계에서 사용되었습니다.
자동 지오코딩(Geocoding) 사용 안함	• CUSTOMERS 테이블의 LOCATION 컬럼에 대한 자동 지오코딩(Geocoding) 사용 안함 • CUSTOMERS 테이블의 LOCATION 컬럼에 대한 지오코딩(Geocoding) 설정 제거 • CUSTOMERS 테이블의 LOCATION 컬럼에 대한 공간 인덱스 삭제 이 단계는 다음 단계에 대한 준비 과정으로 지오코더(Geocoder) 및 공간 인덱스의 자동 호출을 사용하지 않습니다. 다음 단계에서는 전체 CUSTOMERS 테이블을 다시 지오코드합니다. **권장사항:** 대량의 지오코드 데이터를 로드하는 경우, 데이터를 로드하기 전에 공간 인덱스를 삭제한 다음 데이터를 로드한 후 이를 다시 작성하십시오.

표 9. DB2 Spatial Extender 샘플 프로그램 단계 (계속)

단계	조치 및 설명
CUSTOMERS 테이블을 다시 지오코딩(Geocoding)	• 하위 정밀도 레벨을 사용하여 CUSTOMERS 테이블의 LOCATION 컬럼 지오코딩(Geocoding): 100% 대신 90% • CUSTOMERS 테이블의 LOCATION 컬럼에 대한 공간 인덱스 재작성 • 하위 정밀도 레벨을 사용하여 자동 지오코딩(Geocoding) 재사용: 100% 대신 90% 이러한 단계는 일괄처리 모드에서 지오코더(Geocoder)를 실행하고, 공간 인덱스를 재작성하고, 새 정밀도 레벨을 사용하여 자동 지오코딩(Geocoding)을 다시 사용합니다. 이러한 조치는 공간 관리자가 지오코딩(Geocoding) 프로세스에서 높은 실패율을 경고할 때 효과적입니다. 정밀도 레벨을 100%로 설정하면, 참조 자료에서 일치하는 주소를 찾을 수 없어 주소를 지오코딩(Geocoding)하는 데 실패할 수도 있습니다. 정밀도 레벨을 낮추면 지오코더(Geocoder)는 일치하는 데이터를 찾을 수 있는 확률이 높아집니다. 테이블이 다시 지오코딩(Geocoding)되면 자동 지오코딩(Geocoding)을 다시 사용할 수 있고 공간 인덱스가 재작성됩니다. 이렇게 하면 공간 후속 인덱스와 삽입 및 갱신 조작에 대한 공간 컬럼을 증분식으로 유지보수할 수 있습니다.
뷰를 작성하여 뷰에 공간 컬럼 등록	• CUSTOMERS 테이블 및 FLOODZONES 테이블의 조인에 기초하여 HIGHRISKCUSTOMERS 뷰 작성 • 뷰의 공간 컬럼 등록 이 단계는 뷰를 작성한 후 해당 공간 컬럼을 등록합니다.
공간 분석 수행	• 각 영역에서 제공하는 고객 수 찾기(ST_Within) • 동일한 영역을 가진 사무실 및 고객의 경우, 일정 거리 내에 있는 각 사무실의 고객 수 찾기(ST_Within, ST_Distance) • 각 영역의 경우, 각 고객의 평균 수입 및 상여금 찾기(ST_Within) • 각 사무소 지역과 겹치는 범람 지역(flood) 수 찾기(ST_Overlaps) • 사무소 지역(ST_Distance)의 중심에 있는 사무실이라는 가정 하에, 특정 고객 위치에서 가장 가까운 사무실 찾기 • 특정 범람 지역 바운더리와 가장 가까운 위치에 있는 고객 찾기 (ST_Buffer, ST_Intersects) • 특정 사무실에서 지정한 거리 내에 있는 위험 확률이 높은 고객 찾기 (ST_Within) 이 모든 단계는 sqlRunSpatialQueries 스토어드 프로시저를 사용합니다. 이 단계에서는 DB2 SQL로 된 공간 관련 술어와 함수를 사용하여 공간 분석을 수행합니다. DB2 쿼리 옵티마이저는 가능할 때마다 공간 컬럼의 공간 인덱스를 이용하여 쿼리 성능을 개선합니다.

표 9. DB2 Spatial Extender 샘플 프로그램 단계 (계속)

단계	조치 및 설명
공간 데이터를 모양 파일로 익스포트	• HIGHRISKCUSTOMERS 뷰를 모양 파일로 익스포트 이 단계에서는 HIGHRISKCUSTOMERS 뷰를 모양 파일로 익스포트하는 예를 보여줍니다. 데이터베이스 형식에서 다른 파일 형식으로 데이터를 익스포트하면 다른 도구(예: DB2용 ArcExplorer)에서 정보를 사용할 수 있습니다. 이 단계는 runGseDemo.c 프로그램에 포함되어 있으나 참조용으로 주석 처리되어 있습니다. 샘플 프로그램을 수정하여 모양 파일을 익스포트할 위치를 지정하고 샘플 프로그램을 재실행할 수 있습니다.
SDE 파일 익스포트 및 임포트	• CUSTOMERS 테이블을 SDE 전송 파일로 익스포트 • 새로 익스포트된 SDE 전송 파일에서 데이터 임포트 이 단계에서는 SDE 전송 파일의 익스포트 및 임포트 예제를 보여줍니다. 이 단계는 runGseDemo.c 프로그램에 포함되어 있으나 참조용으로 주석 처리되어 있습니다. 샘플 프로그램을 수정하여 SDE 파일을 익스포트할 위치를 지정하고 샘플 프로그램을 재실행할 수 있습니다.

제 15 장 DB2 Spatial Extender 문제점 식별

DB2 Spatial Extender에 대해 작업할 때 문제점이 발생하면 문제점의 원인을 판별해야 합니다.

다음 방법으로 DB2 Spatial Extender에 대한 문제점을 해결할 수 있습니다.

- 메시지 정보를 사용하여 문제점을 진단할 수 있습니다.
- Spatial Extender 스토어드 프로시저 및 함수에 대해 작업할 때 DB2는 스토어드 프로시저 또는 함수의 성공 또는 실패에 대한 정보를 리턴합니다. 리턴되는 정보는 DB2 Spatial Extender에 대해 작업할 때 사용하는 인터페이스에 따라 메시지 코드(정수 유형), 메시지 텍스트 또는 둘 다일 수 있습니다.
- 오류에 대한 진단 정보를 기록하는 DB2 관리 통지 파일을 볼 수 있습니다.
- DB2 Spatial Extender 문제점이 반복하여 다시 발생하는 경우, IBM 고객 지원 담당자는 사용자에게 DB2 추적 기능을 사용하여 문제점 진단을 도와줄 수 있습니다.

DB2 Spatial Extender 메시지 해석 방법

네 가지 다른 인터페이스를 통해 DB2® Spatial Extender에 대해 작업할 수 있습니다.

- DB2 Spatial Extender 스토어드 프로시저
- DB2 Spatial Extender 함수
- DB2 Spatial Extender 명령행 프로시저(CLP)
- DB2 제어 센터

모든 인터페이스는 요청한 공간 데이터 관련 작업이 완료되었는지 또는 오류가 발생했는지를 판별할 수 있도록 DB2 Spatial Extender 메시지를 리턴합니다.

다음 테이블은 이 샘플 DB2 Spatial Extender 메시지 텍스트의 각 파트에 대해 설명합니다.

```
GSE0000I: 조작이 완료되었습니다.
```

표 10. DB2 Spatial Extender 메시지 텍스트 파트

메시지 텍스트 파트	설명
GSE	메시지 ID. 모든 DB2 Spatial Extender 메시지는 세 자의 접두부인 GSE로 시작됩니다.
0000	메시지 번호. 0000 - 9999의 네 자리 숫자

© Copyright IBM Corp. 1998, 2006

표 10. DB2 Spatial Extender 메시지 텍스트 파트 (계속)

메시지 텍스트 파트	설명
I	메시지 유형. 메시지의 심각도를 표시하는 단일 문자
	C 중요한 오류 메시지
	N 중요하지 않은 오류 메시지
	W 경고 메시지
	I 정보용 메시지
조작이 완료되었습니다.	메시지 설명

메시지 텍스트에 나타나는 설명은 간략한 설명입니다. 문제점을 방지하거나 정정하기 위해 자세한 설명 및 제안을 포함하는 메시지에 대한 추가 정보를 검색할 수 있습니다. 이 추가 정보를 표시하려면 다음을 수행하십시오.

1. 운영 체제 시스템 명령 프롬프트를 여십시오.
2. 메시지 ID 및 메시지 번호와 함께 DB2 help 명령을 입력하여 메시지에 대한 추가 정보를 표시하십시오. 예를 들면, 다음과 같습니다.

   ```
   DB2 "? GSEnnnn"
   ```

   여기서 *nnnn*는 메시지 번호입니다.

대문자 또는 소문자로 메시지 유형을 표시하는 문자 및 GSE 메시지 ID를 입력할 수 있습니다. DB2″? GSE0000I″라고 입력하면 db2 ″? gse0000i″를 입력할 때와 동일한 결과가 생성됩니다.

명령을 입력할 때 메시지 번호 다음에 문자를 생략할 수 있습니다. 예를 들어, DB2 ″? GSE0000″라고 입력하면 DB2 ″? GSE0000I″를 입력할 때와 동일한 결과가 생성됩니다.

메시지 코드가 GSE4107N이라고 가정해 보십시오. 명령 프롬프트에 DB2 ″? GSE4107N″이라고 입력하면 다음과 같은 정보가 표시됩니다.

```
GSE4107N 격자 크기 값 "<grid-size>"이 사용된 위치에서 유효하지 않습니다.

설명: 지정된 격자 크기 "<grid-size>"가 유효하지 않습니다.

 CREATE INDEX문을 사용하여 격자 인덱스를 작성할 때 다음과 같은
 유효하지 않은 스펙 중 하나가 작성되었습니다.

   - 첫 번째, 두 번째, 세 번째 격자 레벨에 대한 격자 크기로서 0보다 작은 수를
     지정했습니다.

   - 첫 번째 격자 레벨에 대한 격자 크기로서 0(제로)이 지정되었습니다.

   - 두 번째 격자 레벨에 대해 지정한 격자 크기가 첫 번째 격자 레벨의
     격자 크기보다 작으나 0이 아닙니다.

   - 세 번째 격자 레벨에 대해 지정한 격자 크기가 두 번째
     격자 레벨의 격자 크기보다 작으나 0이 아닙니다.

   - 세 번째 격자 레벨에 대해 지정된 격자 크기는 0보다 크지만
```

두 번째 격자 레벨에 대해 지정된 격자 크기는 0입니다.

사용자 조치: 격자 크기에 대한 올바른 값을 지정하십시오.

msgcode: -4107

sqlstate: 38SC7

정보가 너무 많아 한 화면에 표시할 수 없을 때 운영 체제에서 **추가** 실행 가능 프로그램 및 파이프를 지원하면 다음 명령을 입력하십시오.

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

추가 프로그램을 사용하면 정보를 읽을 수 있도록 각 데이터 화면 다음에 화면이 강제로 일시정지됩니다.

DB2 Spatial Extender 스토어드 프로시저 출력 매개변수

DB2® Spatial Extender 스토어드 프로시저는 DB2 제어 센터에서 Spatial Extender를 사용 가능하게 설정한 후 사용할 때 또는 DB2 Spatial Extender CLP(db2se)를 사용할 때 내재적으로 호출됩니다. 응용프로그램 또는 DB2 명령행에서 명시적으로 스토어드 프로시저를 호출할 수 있습니다.

이 주제에서는 응용프로그램 또는 DB2 명령행에서 스토어드 프로시저를 명시적으로 호출할 때 문제점을 진단하는 방법에 대해 설명합니다. 내재적으로 호출된 스토어드 프로시저를 진단하기 위해 DB2 Spatial Extender CLP에서 리턴된 메시지 또는 DB2 제어 센터에서 리턴된 메시지를 사용합니다. 이러한 메시지는 별도의 주제에서 설명됩니다.

DB2 Spatial Extender 스토어드 프로시저에는 메시지 코드(msg_code) 및 메시지 텍스트(msg_text)의 두 가지 출력 매개변수가 있습니다. 매개변수 값은 스토어드 프로시저의 성공 또는 실패를 표시합니다.

msg_code

msg_code 매개변수는 정수이며 양수, 음수 또는 0일 수 있습니다. 양수는 경고, 음수는 오류(중요한 오류 및 중요하지 않은 오류 둘 다), 0은 정보 메시지에 사용됩니다.

msg_code의 절대값은 메시지 번호로서 msg_text에 포함됩니다. 예를 들면 다음과 같습니다.

- msg_code가 0이면 메시지 번호는 0000입니다.
- msg_code가 -219인 경우 메시지 번호는 0219입니다. 음수 msg_code는 메시지가 중요하거나 중요하지 않은 오류임을 나타냅니다.
- msg_code가 +1036이면, 메시지 번호는 1036입니다. 양수 msg_code는 메시지가 경고임을 나타냅니다.

Spatial Extender 스토어드 프로시저에 대한 msg_code 번호는 다음 테이블에 표시된 세 개의 범주로 나뉘어집니다.

표 11. 스토어드 프로시저 메시지 코드

코드	범주
0000 – 0999	일반 메시지
1000 – 1999	관리 메시지
2000 – 2999	메시지 임포트 및 익스포트

msg_text

msg_text 매개변수는 메시지 ID, 메시지 번호, 메시지 유형 및 설명으로 구성됩니다. 스토어드 프로시저 msg_text 값에 대한 예는 다음과 같습니다.

```
GSE0219N   An EXECUTE IMMEDIATE문이 실패했습니다.
            SQLERROR = "<sql-error>".
```

msg_text 매개변수에 나타나는 설명은 간단한 설명입니다. 문제점을 방지하거나 정정하기 위해 자세한 설명 및 제안을 포함하는 메시지에 대한 추가 정보를 검색할 수 있습니다.

msg_text 매개변수의 파트에 대한 자세한 설명 및 메시지에 대한 추가 정보를 검색하는 방법에 대한 정보는 DB2 Spatial Extender 메시지 해석 방법 주제를 참조하십시오.

응용프로그램에서 스토어드 프로시저에 대한 작업

응용프로그램에서 DB2 Spatial Extender 스토어드 프로시저를 호출할 경우 출력 매개변수로 msg_code 및 msg_text를 수신합니다. 다음을 수행할 수 있습니다.

- 응용프로그램 사용자에게 출력 매개변수 값을 리턴하도록 응용프로그램을 프로그램할 수 있습니다.
- 리턴된 msg_code 값 유형을 기본으로 일부 조치를 수행할 수 있습니다.

DB2 명령행에서 스토어드 프로시저에 대한 작업

DB2 명령행에서 DB2 Spatial Extender 스토어드 프로시저를 호출할 경우 msg_code 및 msg_text 출력 매개변수를 수신합니다. 이 출력 매개변수는 스토어드 프로시저의 성공 및 실패를 표시합니다.

데이터베이스에 연결하여 ST_disable_db 스토어드 프로시저를 호출하려 한다고 가정하십시오. 아래 예에서는 DB2 CALL 명령을 사용하여 공간 데이터 관련 작업에 대해 데이터베이스를 사용하지 않고 출력 값 결과를 표시합니다. 강제 매개변수 값 0과 함께 CALL 명령의 끝에 두 개의 물음표를 사용하여 msg_code 및 msg_text 출력 매개변수를 표시합니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

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

    출력 매개변수 값
  -------------------------
    매개변수 이름   : MSGCODE
    매개변수 값     : 0

    매개변수 이름   : MSGTEXT
    매개변수 값     : GSE0000I  조작이 완료되었습니다.

    리턴 상태 = 0
```

리턴된 msg_text가 GSE2110N이라고 가정하십시오. 메시지에 대한 자세한 정보를 표시하려면 DB2 help 명령을 사용하십시오. 예를 들면, 다음과 같습니다.

```
"? GSE2110"
```

다음 정보가 표시됩니다.

```
GSE2110N   "<row-number>"행의 Geometry에 대한
           공간 참조 시스템이 유효하지 않습니다.
           공간 참조 시스템의 숫자 ID가 "<srs-id>"입니다.

설명: row-number 행에서 익스포트할 Geometry가 유효하지 않은
      공간 참조 시스템을 사용합니다. Geometry를 익스포트할 수 없습니다.

사용자 응답: 표시된 Geometry를 정정하거나 SELECT문을 알맞게 수정하여
             익스포트 조작에서 행을 제외시키십시오.

msg_code: -2110

sqlstate: 38S9A
```

DB2 Spatial Extender 함수 메시지

DB2® Spatial Extender 함수에서 리턴한 메시지는 일반적으로 SQL 메시지에 임베디드(embedded)됩니다. 메시지에서 리턴되는 SQLCODE는 함수에서 오류가 발생했는지 또는 함수에 경고가 연관되어 있는지 여부를 표시합니다. 예를 들면, 다음과 같습니다.

- SQLCODE –443(메시지 번호 SQL0443)은 함수에 오류가 발생했음을 표시합니다.
- SQLCODE +462(메시지 번호 SQL0462)는 경고가 함수에 연관되어 있는지 여부를 표시합니다.

다음 테이블에서는 이 샘플 메시지의 중요한 파트에 대해 설명합니다.

```
DB21034E  명령이 올바른 명령행 처리기 명령이 아니어서 SQL문으로서
          처리되었습니다. SQL 처리 중에 다음이 리턴되었습니다.
SQL0443N  루틴 "DB2GSE.GSEGEOMFROMWKT"(특정 이름 "GSEGEOMWKT1")에서
          SQLSTATE 오류와 함께 "GSE3421N 다각형이 닫혀 있지 않습니다."라는
          진단 텍스트가 리턴됩니다. SQLSTATE=38SSL
```

표 12. *DB2 Spatial Extender 함수 메시지의 중요한 파트*

메시지 파트	설명
SQL0443N	SQLCODE는 문제점 유형을 나타냅니다.
GSE3421N	DB2 Spatial Extender 메시지 번호 및 메시지 유형. 함수의 메시지 번호의 범위는 GSE3000 - GSE3999입니다. 추가로 DB2 Spatial Extender 함수에 대해 작업할 때 일반 메시지가 리턴될 수 있습니다. 일반 메시지의 메시지 번호 범위는 GSE0001 - GSE0999입니다.
다각형이 닫히지 않았습니다.	DB2 Spatial Extender 메시지 설명
SQLSTATE=38SSL	오류를 자세히 식별하는 SQLSTATE 코드. 각 명령문 또는 행에 대해 SQLSTATE 코드가 리턴됩니다. • Spatial Extender 함수 오류에 대한 SQLSTATE 코드는 38S*xx*입니다. 여기서 각 *x*는 문자 또는 숫자입니다. • Spatial Extender 함수 경고에 대한 SQLSTATE 코드는 01HS*x*입니다. 여기서 *x*는 문자 또는 숫자입니다.

SQL0443 오류 메시지의 예

다음에 표시된 대로 다각형의 값을 POLYGON_TABLE에 삽입하려 한다고 가정하십시오.

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

다각형을 닫는 끝 값을 제공하지 않았기 때문에 오류 메시지가 발생합니다. 리턴된 오류 메시지는 다음과 같습니다.

```
DB21034E  명령이 올바른 명령행 처리기 명령이 아니어서 SQL문으로서
처리되었습니다. SQL 처리 중에 다음이 리턴되었습니다.
SQL0443N  루틴 "DB2GSE.GSEGEOMFROMWKT"(특정 이름 "GSEGEOMWKT1")에서
SQLSTATE 오류와 함께 "GSE3421N 다각형이 닫혀 있지 않습니다."라는
진단 텍스트가 리턴됩니다.
SQLSTATE=38SSL
```

SQL 메시지 번호 SQL0443N은 오류가 발생했음을 나타내며 메시지는 GSE3421N 다각형이 닫혀 있지 않습니다라는 Spatial Extender 메시지를 포함합니다.

이 유형의 메시지를 수신하면 다음과 같이 하십시오.

1. DB2 또는 SQL 오류 메시지 내에서 GSE 메시지 번호를 찾으십시오.
2. DB2 help 명령(DB2 ?)을 사용하여 Spatial Extender 메시지 설명 및 사용자 조치를 찾으십시오. 위의 명령을 사용하여 운영 체제 명령 프롬프트에 다음 명령을 입력하십시오.

   ```
   DB2 "? GSE3421"
   ```

자세한 설명 및 권장되는 사용자 조치와 함께 메시지가 반복됩니다.

DB2 Spatial Extender CLP 메시지

DB2® Spatial Extender CLP(db2se)는 다음에 대한 메시지를 리턴합니다.

- 내재적으로 호출될 경우 스토어드 프로시저
- DB2 Spatial Extender CLP에서 **shape_info** 부속 명령 프로그램을 호출할 경우 모양 정보. 이것은 정보 메시지입니다.
- 이주 조작
- 클라이언트 간 모양 임포트 및 익스포트 조작

DB2 Spatial Extender CLP에서 리턴한 스토어드 프로시저 메시지의 예

DB2 Spatial Extender CLP를 통해 리턴되는 대부분의 메시지는 DB2 Spatial Extender 스토어드 프로시저에 대한 것입니다. DB2 Spatial Extender CLP에서 스토어드 프로시저를 호출할 때마다 스토어드 프로시저의 성공 또는 실패를 표시하는 메시지를 수신합니다.

메시지 텍스트는 메시지 ID, 메시지 번호, 메시지 유형 및 설명으로 구성됩니다. 예를 들어, `db2se enable_db testdb` 명령을 사용하여 데이터베이스를 사용할 경우 Spatial Extender CLP에서 리턴되는 메시지 텍스트는 다음과 같습니다.

```
데이터베이스 사용 중. 잠시만 기다리십시오...
GSE1036W 조작이 완료되었습니다. 그러나 특정 데이터베이스 관리 프로그램 및
         데이터베이스 구성 매개변수의 값을 늘려야 합니다.
```

마찬가지로 `db2se disable_db testdb` 명령을 사용하여 데이터베이스를 사용하지 않을 경우 Spatial Extender CLP에서 리턴되는 메시지 텍스트는 다음과 같습니다.

```
GSE0000I 조작이 완료되었습니다.
```

메시지 텍스트에 나타나는 설명은 간략한 설명입니다. 문제점을 방지하거나 정정하기 위해 자세한 설명 및 제안을 포함하는 메시지에 대한 추가 정보를 검색할 수 있습니다. 이 정보를 검색하는 단계 및 메시지 텍스트의 파트를 해석하는 방법에 대한 자세한 설명은 별도의 주제에 설명되어 있습니다.

응용프로그램 또는 DB2 명령행을 통해 스토어드 프로시저를 호출할 경우, 출력 매개변수 진단에 대해 설명하는 별도의 주제가 있습니다.

Spatial Extender CLP에서 리턴한 모양 정보 메시지의 예

`office`라는 모양 파일에 대한 정보를 표시하려 한다고 가정하십시오. Spatial Extender CLP(db2se)를 통해 다음 명령을 발행합니다.

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

다음은 표시되는 정보의 예입니다.

```
모양 파일 정보
----------------------------------------
파일 코드                  = 9994
파일 길이(16비트 단어)     = 484
모양 파일 버전             = 1000
모양 유형                  = 1 (ST_POINT)
레코드 수                  = 31

Minimum X coordinate = -87.053834
Maximum X coordinate = -83.408752
Minimum Y coordinate = 36.939628
Maximum Y coordinate = 39.016477
Shapes do not have Z coordinates.
Shapes do not have M coordinates.

모양 인덱스 파일(extension .shx)이 표시됩니다.

속성 파일 정보
----------------------------------------
dBase 파일 코드               = 3
최종 갱신 날짜                = 1901-08-15
레코드 수                     = 31
헤더의 바이트 수              = 129
각 레코드의 바이트 수         = 39
컬럼 수                       = 3

컬럼 수  컬럼 이름  데이터 유형   길이  10진수

      1       NAME      C(문자)     16      0
      2  EMPLOYEES   N(Numeric™)    11      0
      3         ID      N(숫자)     11      0

좌표 시스템 정의: "GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433295]]"
```

Spatial Extender CLP에서 리턴한 이주 메시지의 예

이주 조작을 수행하는 명령을 호출할 경우 해당 조작의 성공 및 실패를 표시하는 메시지가 리턴됩니다.

db2se migrate mydb -messagesFile/tmp/migrate.msg 명령을 사용하여 mydb 데이터베이스의 이주를 호출한다고 가정하십시오. Spatial Extender CLP에서 리턴되는 메시지 텍스트는 다음과 같습니다.

```
데이터베이스 이주 중. 잠시만 기다리십시오...
GSE0000I 조작이 완료되었습니다.
```

DB2 제어 센터 메시지

DB2 Spatial Extender 메시지

DB2 제어 센터를 통해 DB2® Spatial Extender에 대해 작업하는 경우 메시지는 DB2 메시지 창에 표시됩니다. 발생하는 대부분의 메시지는 DB2 Spatial Extender 메시지입니다. 간혹 SQL 메시지를 수신합니다. 라이센스 부여, 잠금에 오류가 관련될 때 또는 DAS 서비스가 사용 불가능할 때 SQL 메시지가 리턴됩니다. 다음 절에서는 DB2 Spatial Extender 메시지 및 SQL 메시지가 어떻게 DB2 제어 센터에 나타나는지를 보여주는 예를 제공합니다.

제어 센터를 통해 DB2 Spatial Extender 메시지를 수신할 경우, DB2 메시지 창의 텍스트 영역에 전체 메시지 텍스트가 나타납니다. 예를 들면, 다음과 같습니다.

```
GSE0219N   EXECUTE IMMEDIATE문이 실패했습니다.
           SQLERROR = "<sql-error>".
```

SQL 메시지

제어 센터를 통해 DB2 Spatial Extender에 관련된 SQL 메시지를 수신할 경우 다음과 같이 표시됩니다.

- DB2 메시지 창의 왼쪽에 메시지 ID, 메시지 번호 및 메시지 유형이 나타납니다. 예를 들면, SQL0612N과 같습니다.
- DB2 메시지 창의 텍스트 영역에 메시지 텍스트가 나타납니다.

DB2 메시지 창에 나타나는 메시지 텍스트는 SQL 메시지 텍스트 및 SQLSTATE를 포함하거나 메시지 텍스트, 자세한 설명 및 사용자 조치를 포함할 수 있습니다.

SQL 메시지 텍스트 및 SQLSTATE를 포함하는 SQL 메시지의 예는 다음과 같습니다.

```
[IBM][CLI Driver][DB2/NT] SQL0612N "<name>"이 중복된 이름입니다.
SQLSTATE=42711
```

메시지 텍스트, 자세한 설명 및 사용자 조치를 포함하는 SQL 메시지의 예는 다음과 같습니다.

```
SQL8008N
"DB2 Spatial Extender" 제품에 유효한 라이센스 키가 없거나 기간이 만료되었습니다.

설명:

유효한 라이센스 키를 찾을 수 없거나 기간이 만료되었습니다.

사용자 응답:

제품의 완전한 권리부여 버전에 대한 사용권 키를 설치하십시오.
IBM® 담당자 또는 대리점에 문의하여 제품에 대한 라이센스 키를 구입할 수 있습니다.
```

db2trc 명령을 사용하여 DB2 Spatial Extender 문제점 추적

DB2 Spatial Extender 문제점이 반복하여 다시 발생하는 경우 DB2 추적 기능을 사용하여 문제점에 대한 정보를 캡처할 수 있습니다.

DB2 추적 기능은 **db2trc** 시스템 명령에 의해 활성화됩니다. DB2 추적 기능은 다음을 수행할 수 있습니다.

- 이벤트 추적
- 추적 데이터를 파일에 덤프
- 추적 데이터를 읽을 수 있는 형식으로 형식화

제한사항:

- DB2 기술 지원 담당자의 지시가 있을 경우에만 이 기능을 활성화하십시오.
- UNIX 운영 체제의 경우 SYSADM, SYSCTRL 또는 SYSMAINT 권한이 있어야만 DB2 인스턴스를 추적할 수 있습니다.
- Windows 운영 체제의 경우 특수한 권한은 필요하지 않습니다.

이 태스크를 수행하려면

1. 모든 다른 응용프로그램을 종료하십시오.
2. 추적을 온(on)으로 설정하십시오.

   DB2 기술 지원 담당자는 이 단계에 대한 특정 매개변수를 제공합니다. 기본 명령은 다음과 같습니다.

   ```
   db2trc on
   ```

   제한사항: db2trc 명령은 운영 체제 명령 프롬프트 또는 쉘 스크립트에서 입력해야 합니다. DB2 Spatial Extender 명령행 인터페이스(db2se) 또는 DB2 CLP에서 사용할 수 없습니다.

   메모리 또는 파일에 대해 추적할 수 있습니다. 선호하는 추적 메소드는 메모리에 대해 추적하는 것입니다. 재생된 문제점이 워크스테이션을 일시중단하고 추적 덤프를 방해하면 파일을 추적하십시오.
3. 문제점을 재생하십시오.
4. 문제점이 발생하면 즉시 파일로 추적을 덤프하십시오.

   예를 들면, 다음과 같습니다.

   ```
   db2trc dump january23trace.dmp
   ```

이 명령은 현재 디렉토리에 지정한 이름을 가진 파일(*january23trace.dmp*)을 작성하고 추적 정보를 해당 파일로 덤프합니다. 파일 경로를 포함하여 다른 디렉토리를 지정할 수 있습니다. 예를 들어, 덤프 파일을 /tmp/spatial/errors 디렉토리에 배치하려면 구문은 다음과 같습니다.

```
db2trc dump /tmp/spatial/errors/january23trace.dmp
```

5. 추적을 해제하십시오.

   예를 들면, 다음과 같습니다.

   ```
   db2trc off
   ```

6. 데이터를 ASCII 파일로 형식화하십시오. 데이터를 다음 두 가지 방법으로 정렬할 수 있습니다.
   - flw 옵션을 사용하여 프로세스 또는 스레드에서 데이터를 정렬하십시오. 예를 들면, 다음과 같습니다.

     ```
     db2trc flw january23trace.dmp january23trace.flw
     ```

   - fmt 옵션을 사용하여 모든 이벤트를 연대순으로 나열하십시오. 예를 들면, 다음과 같습니다.

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

관리 통지 파일

오류에 대한 진단 정보는 관리 통지 파일에 기록됩니다. 이 정보는 문제점 판별에 사용되며 DB2® 기술 지원에 사용됩니다.

관리 통지 파일에는 DB2 Spatial Extender뿐 아니라 DB2에 의해 로그된 텍스트 정보도 포함됩니다. 이 파일은 DIAGPATH 데이터베이스 관리 프로그램 구성 매개변수로 지정된 디렉토리에 위치합니다. Windows® NT, Windows 2000 및 Windows XP 시스템에서 DB2 관리 통지 파일은 이벤트 로그에 있으며 Windows 이벤트 표시기를 통해 검토할 수 있습니다.

DB2가 관리 로그에 기록하는 정보는 DIAGLEVEL 및 NOTIFYLEVEL 설정에 따라 결정됩니다.

문제점이 발생했을 것이라고 의심되는 시스템에 대한 파일을 보려면 텍스트 편집기를 사용하십시오. 가장 최근에 기록된 이벤트는 파일의 가장 아래에 있습니다. 일반적으로 각 항목은 다음과 같은 파트를 포함합니다.

- 시간소인.
- 오류 보고 위치. 응용프로그램 ID를 사용하여 서버 및 클라이언트 로그에 대한 응용프로그램 관련 항목을 일치시킬 수 있습니다.
- 오류에 대해 설명하는 진단 메시지(일반적으로 ″DIA″ 또는 ″ADM″으로 시작됨).

- SQLCA 데이터 구조 및 특별 덤프 또는 트랩 파일 위치에 대한 포인터와 같은 모든 사용 가능한 지원 데이터.

데이터베이스가 정상적으로 작동할 경우, 이러한 유형의 정보는 중요하지 않으며 무시됩니다.

관리 통지 파일은 연속적으로 커집니다. 너무 커지면 백업한 후 파일을 지우십시오. 새 파일은 시스템에서 필요로 할 때 자동으로 생성됩니다.

제 16 장 DB2 Geodetic Data Management Feature

이 장에서는 DB2 Geodetic Data Management Feature의 목적, 사용 시기 및 측지 개념에 대해 설명하여 DB2 Geodetic Data Management Feature를 소개합니다.

DB2 Geodetic Data Management Feature

DB2® Geodetic Data Management Feature를 사용하면 지구를 구로 처리할 수 있습니다. 다른 Spatial Extender 조작과 동일한 공간 데이터 유형 및 기능을 사용하면, Geodetic Data Management Feature를 사용하여 극 지점 주위의 데이터 및 180도 경선을 가로지르는 데이터의 쿼리를 완벽하게 실행할 수 있습니다. 지구 표면의 정확한 위치에 참조되는 데이터를 유지보수할 수 있습니다.

Geodetic Data Management Feature는 *측지학*이라는 학문 분야에서 이름이 지정된 것입니다. 측지학은 지구(또는 태양이나 천체와 같은 타원체 모양의 물체)의 *크기* 및 모양을 연구하는 학문입니다. Geodetic Data Management Feature는 지구 표면에 정의되는 오브젝트를 매우 정밀하게 처리할 수 있도록 설계되었습니다.

이러한 정밀도를 확보하기 위해 Geodetic Data Management Feature에서는 타원형의 지구 모델에 경도와 위도 좌표 시스템 또는 *측지 기준면*(평면이 아님) *x* 및 *y* 좌표 시스템을 사용합니다. 타원체 모델을 사용하면 평면 프로젝션 사용 시 발생할 수 있는 왜곡, 오류 및 부정확 등을 방지할 수 있습니다.

공간 데이터 관련 작업이 아닌 측지 조작을 수행하려면 데이터에 대한 측지 공간 참조 시스템을 정의해야 합니다. 이 시스템에는 2000000000 - 2000001000 범위 내의 공간 참조 시스템 ID(SRID)가 있습니다. DB2 Geodetic Data Management Feature에서는 318개의 사전 정의된 측지 공간 참조 시스템을 제공합니다.

DB2 Geodetic Data Management Feature를 사용하려면 DB2 Spatial Extender가 설치되어 있어야 합니다. Geodetic Data Management Feature를 사용하려면 별도의 라이센스를 구입해야 합니다.

DB2 Geodetic Data Management Feature 및 DB2 Spatial Extender 사용 시기

DB2® Spatial Extender 및 DB2 Geodetic Data Management Feature 모두는 DB2 데이터베이스에서 지리 정보 시스템(GIS) 데이터를 관리합니다. 각 Extender는 다른 핵심 기술을 사용하여 다른 문제점을 해결함으로써 상호 보완적인 기능을 수행합니다.

© Copyright IBM Corp. 1998, 2006

- Geodetic Data Management Feature에서는 지구를 구로 간주합니다. 타원체 지구 모델에 대한 위도 및 경도 좌표 시스템을 사용합니다. Geometry 조작은 위치에 관계 없이 정확하며 Geodyssey Limited로부터 사용 허가된 Hipparchus 라이브러리에서 빌드됩니다. 자세한 측지 정보는 http://www.geodyssey.com을 참조하십시오.

  Geodetic Data Management Feature는 지구 상의 넓은 영역을 대상으로 하는 글로벌 데이터 세트 및 응용프로그램에 가장 유용합니다(단일 맵 프로젝션에서는 응용프로그램에 필요한 정확도를 제공할 수 없음).
- Spatial Extender는 지구를 평면 지도로 간주하며 지구의 곡면을 평면에 투영하여 근사치로 표현하는 평면 Geometry를 사용합니다. 이러한 투영으로 인한 왜곡은 데이터의 Extent로 상쇄될 수 있지만 일반적으로 투영된 지역의 가장자리로 갈수록 증가합니다. 모든 평면 지도 투영에는 어느 정도의 왜곡이 발생합니다. Spatial Extender는 ESRI로부터 사용 허가된 ESRI 모양 라이브러리에서 빌드됩니다. 자세한 공간 정보는 http://www.esri.com을 참조하십시오.

  Spatial Extender는 투영 좌표에서 잘 표현되는 로컬 및 지역 데이터 세트와 위치 정확도가 중요하지 않은 응용프로그램에 가장 적합합니다. 예를 들어, 의료보험 회사가 특정 지역의 병원 및 의원의 위치 정보를 파악하려 할 수 있습니다.

측지 기준면

측지 기준면은 지구 표면에 대해 설명하는 참조 시스템입니다. 과학의 발전으로 새로운 지구 측정 도구가 개발되면서 오래 동안 많은 이러한 참조 시스템이 개발되어 왔습니다. 지표 측정 및 Satellite 측정 방법을 사용하여 기준면을 작성해 왔으며 기준면은 다시 평면 지도 투영을 작성하는 데 사용됩니다.

측지 데이텀은 *회전 타원체*를 사용한 일반적인 지구 모양의 근사값을 기반으로 합니다. 회전 타원체는 중심 축을 따라 회전하는 타원체로 설명되는 3차원 모양입니다.

정의되는 모든 공간 오브젝트는 특정 기준면을 참조해야 합니다. 공간 참조 시스템 ID(SRID)로 기준면을 지정합니다. DB2® Geodetic Data Management Feature에서 지원하는 기준면을 선택할 수 있습니다. 이러한 시스템의 SRID 범위는 2000000000 - 2000001000입니다.

- ″DB2 Geodetic Data Management Feature에서 지원하는 기준면″에는 Geodetic Data Management Feature에서 제공하는 318개의 사전 정의된 측지 공간 참조 시스템이 나열되어 있습니다.
- 2000000318 - 2000001000 범위 내의 ID를 사용하는 공간 참조 시스템을 작성하여 새 기준면을 정의할 수도 있습니다.

제한사항: 둘 이상의 지오 공간 오브젝트를 인수로 사용하는 함수는 기준면 조합을 처리할 수 없습니다. Geodetic Data Management Feature는 기준면 변환을 수행하지 않습니다.

측지 위도 및 경도

DB2 Geodetic Data Management Feature의 좌표 참조 시스템에서는 *측지 위도* 및 *경도*를 사용하여 지구의 상대 위치를 설명합니다. 측지 위도 및 경도는 항상 특정 기준면을 기반으로 합니다.

측지 위도

한 점의 측지 위도는 지구 표면 상의 한 점을 지나는 법선을 교차하는 수직선과 적도면 사이의 각도입니다.

측지 경도

*측지 경도*는 지구 중심과 본초 자오선을 연결하는 선 *a*와 점이 있는 경선과 중심을 연결하는 선 *b* 사이 적도면에서의 각도입니다. *경선*은 데이텀 표면의 직접 경로이자 양극 사이의 최단 거리입니다.

그림 17의 타원체는 측지 위도 및 경도를 나타내는 각도를 보여줍니다. 측지 경도의 각도는 지구가 타원형 모양이므로 정중앙에서 시작되지 않습니다.

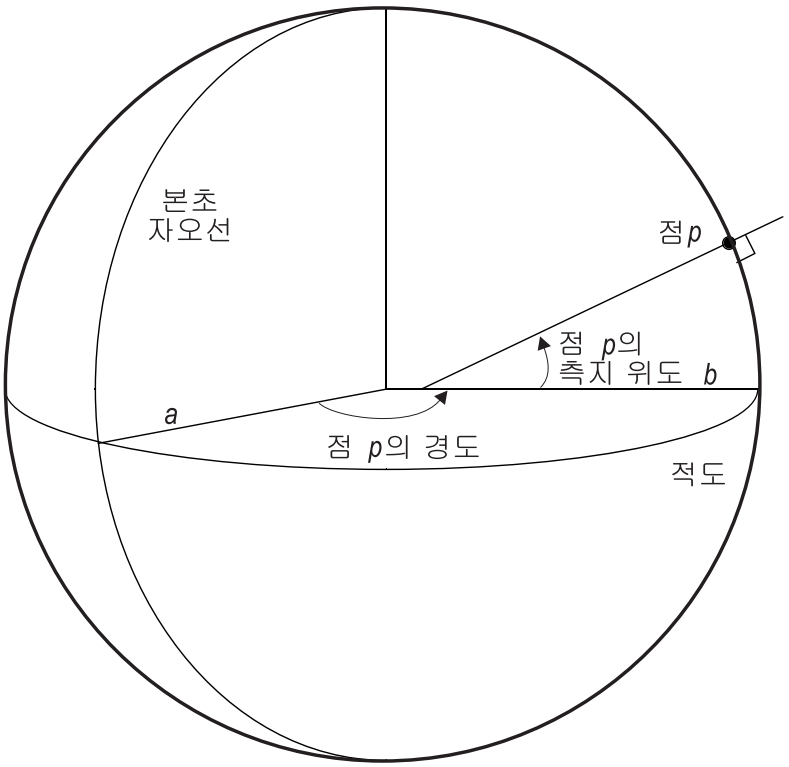


그림 17. 측지 위도 및 경도 각도

위도 및 경도 좌표는 도(소숫점 표기)로 표시됩니다. 경도는 본초 자오선(경도 0°)을 시작으로 동쪽으로 180°(+), 서쪽으로 180°(-), 총 360°로 표시됩니다. 위도는 적도(위도 0°)에서 시작하여 북극(위도 90°)과 남극(위도 -90°)까지 계속됩니다.

측지 거리

DB2® Geodetic Data Management Feature에서는 *측지선* 상의 두 점 사이 거리를 측정합니다. 측지선은 타원형 모형의 지구에 있는 두 점 사이의 최단 경로이며 이 최단 경로는 두개의 끝점이 동일한 위도에 있는 경우라도 연속 위도선과 일치하지 않을 수 있습니다.

선 구획은 측지선을 기준으로 계산하므로 네 개의 점이 넓게 분산되어 있는 다각형은 151 페이지의 그림 18에 나와 있는 것처럼 원하는 영역을 포함하지 않을 수 있습니다. 이 다각형은 약 120도 떨어진 경도선 사이의 영역을 표시하며 위쪽 두 점의 위도 값은 동일하고 맨 아래 두 점의 위도 값도 동일합니다. 두 경선 사이의 측지선은 타원체 모양의 지구상의 곡선과 일치합니다. 측지선의 위도는 측지선 양 끝보다 중앙이 20도 가량 더 큽니다.

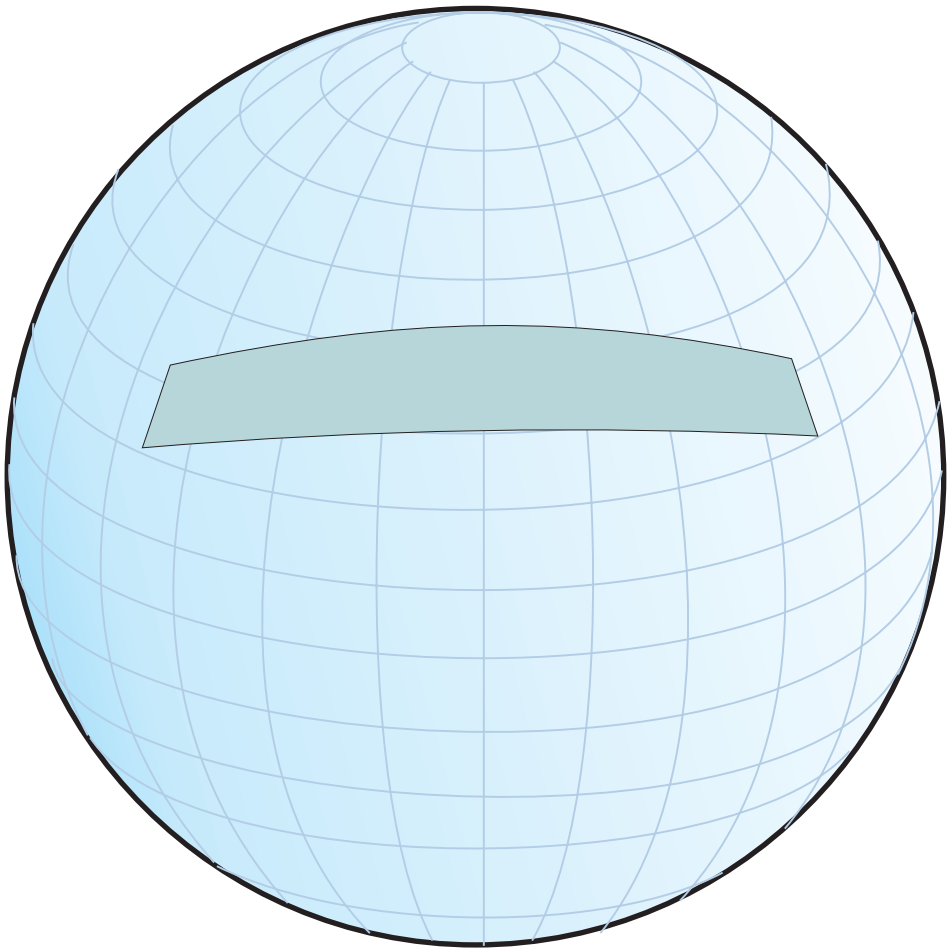


그림 18. 점이 넓게 분산되어 있는 다각형의 내부 영역

측지선이 아닌 경로를 표현하려면(예를 들어, 연속 위도를 따라 선을 표시하려면) 추가 중간 점을 삽입해야 합니다.

측지 영역

측지 영역(다각형)은 응용프로그램에 대한 고유한 특성이 있는 지구 표면의 한 영역입니다. 영역의 예로는 시장 영향력이 있는 지역 또는 지정된 시간 동안 위성에서 볼 수 있는 지역을 들 수 있습니다.

Geodetic Data Management Feature에서는 닫힌 링을 형성하는 정렬된 일련의 점으로 영역을 정의합니다. 다각형에서 점을 지정하는 순서가 매우 중요합니다. 정점에서 정점으로 정해진 순서대로 다각형을 따라가면 왼쪽 영역이 다각형의 내부가 됩니다.

ST_Polygon 데이터 유형을 사용하면 하나 이상의 링으로 둘러싸인 영역을 정의할 수 있습니다(152 페이지의 그림 19 참조). 해당 링을 구성하는 점(정점)의 위도 및 경도 좌표로 다각형을 정의하십시오.

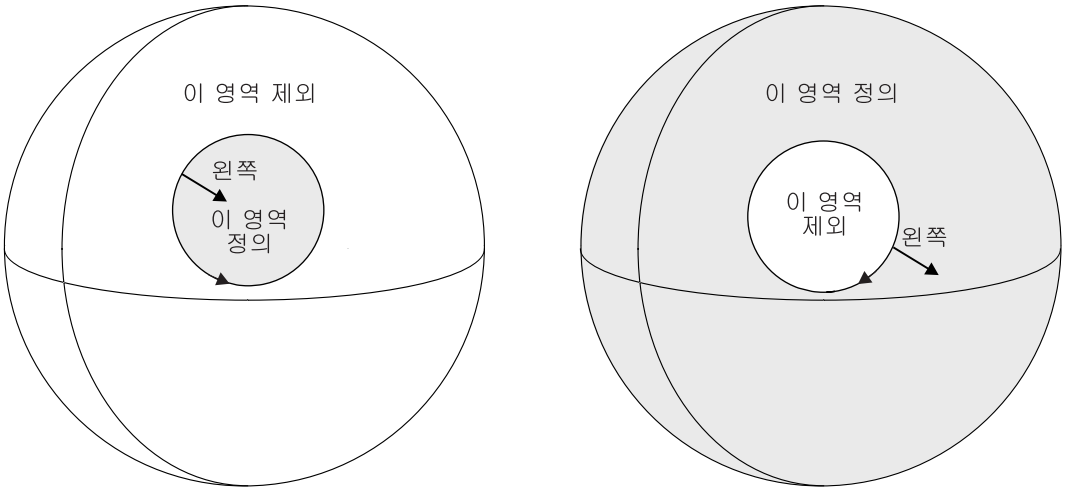


그림 19. 영역 정의 및 제외

링은 지구 표면을 두 영역 즉, 다각형 내부 영역과 다각형 외부 영역으로 나눕니다. 그림 19의 왼쪽에는 링의 정점이 시계 반대 방향으로 지정되어 왼쪽의 모든 점이 링 내부에 있는 링이 표시됩니다. 그림 오른쪽에는 링의 정점이 시계 방향으로 지정되어 왼쪽의 모든 점이 링 외부에 있는 링이 표시됩니다.

다각형으로 영역을 정의하려면 링을 지날 때 다각형 내부가 왼쪽에 오도록 각 링의 정점 순서를 지정해야 합니다. 제외된 영역을 정의하려면 정반대의 순서로 링 정점을 지정해야 합니다(153 페이지의 그림 20 참조). 다각형 내부는 항상 왼쪽이어야 합니다. 153 페이지의 그림 20에서는 한 링이 다른 링 안에 있는 두 개 링을 표시합니다. 큰 링이 다각형의 바깥쪽 바운더리를 정의하며 시계 반대 방향으로 그려집니다. 작은 링은 내부 바운더리를 정의하며 시계 방향으로 그려집니다.

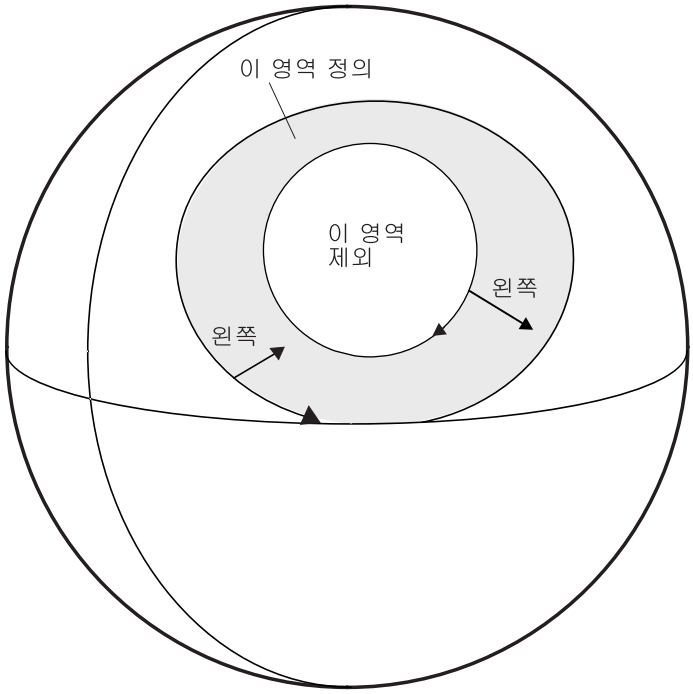


그림 20. 다중 링이 있는 영역 정의

반구보다 큰 다각형을 작성하는 경우, 다음 경고 메시지가 리턴됩니다. 이러한 큰 다각형이 실제로 필요할 수 있지만 이 경고는 작은 다각형이 필요할 때 실수로 정점 순서를 잘못 지정하여 큰 다각형을 작성하는 경우를 대비한 것입니다.

```
GSE3733W "다각형이 지구의 절반 이상을 덮고 있습니다.
         최고점의 반시계 방향을 확인하십시오."
```

제 17 장 DB2 Geodetic Data Management Feature 설정

이 섹션에서는 DB2 Geodetic Data Management Feature 설정, Informix Geodetic DataBlade에서 이주 및 측지 데이터로 공간 컬럼 채우기에 대한 지시사항을 제공합니다.

DB2 Geodetic Data Management Feature 설정 및 사용

DB2 Geodetic Data Management Feature를 사용하려면 다음을 수행해야 합니다.

- DB2 버전 9.1을 설치 및 구성하십시오.

  DB2 Spatial Extender 및 DB2 Geodetic Data Management Feature를 설치하기 *전에* 시스템에 DB2 데이터베이스를 설치해야 합니다. DB2 제어 센터를 사용할 계획이면 DB2 Administration Server(DAS)를 작성 및 구성하십시오. DAS 작성 및 구성에 대한 자세한 정보는 *IBM DB2 관리 안내서: 구현*을 참조하십시오.

- DB2 Spatial Extender를 설치 및 구성하십시오.

  DB2 Geodetic Data Management Feature는 DB2 Spatial Extender와 동일한 라이브러리 코드로 통합됩니다. 따라서 Spatial Extender용 설치 CD에는 Geodetic Data Management Feature가 포함됩니다. Spatial Extender의 디스크 스페이스 요구사항에는 Geodetic Data Management Feature가 포함됩니다. 그러나 Geodetic Data Management Feature가 라이센스를 구입하여 사용 가능할 때까지 Geodetic Data Management Feature를 사용할 수 없습니다.

- DB2 Geodetic Data Management Feature 라이센스를 구입하십시오.

  DB2 Geodetic Data Management Feature 라이센스를 구입하면 측지 라이센스 키를 사용할 수 있습니다. DB2 Geodetic Data Management Feature를 구입하려면 영업 담당자에게 문의하십시오.

제한사항:

- DB2 Geodetic Data Management Feature는 DB2 버전 9.1 Enterprise Server Edition에 대해서만 사용이 허가됩니다.
- DB2 Geodetic Data Management Feature는 지구를 구로 간주하는 반면 Spatial Extender는 지구의 곡면을 평면 지도로 간주합니다. Geodetic Data Management Feature를 설치하는 경우 평면 지도보다 정확하게 공간 데이터를 분석할 수 있습니다.

© Copyright IBM Corp. 1998, 2006

- DB2 Geodetic Data Management 시스템은 DB2 데이터베이스 시스템, DB2 Spatial Extender, DB2 Geodetic Data Management Feature 및 geobrowser(대부분의 응용프로그램에서)로 구성됩니다.

추가 또는 변경된 정보를 DB2 Geodetic Data Management Feature에서 사용할 수 있도록 하려면 *DB2 릴리스 정보*를 참조하십시오.

DB2 Geodetic Data Management Feature 라이센스를 사용 가능하게 한 후 공간 컬럼을 측지 데이터로 채웁니다.

이 태스크를 수행하려면

다음 중 한 가지 방법으로 DB2 Geodetic Data Management Feature를 사용 가능하게 하십시오.

- DB2 제어 센터의 라이센스 센터를 사용하십시오. Geodetic 라이센스 활성화 방법에 대한 자세한 정보는 DB2 라이센스 센터에 대한 온라인 도움말을 참조하십시오.
- **db2licm** 명령을 실행하십시오.

Informix Geodetic DataBlade에서 DB2 Geodetic Data Management Feature로 이주

전제조건

DB2 Geodetic Data Management Feature 데이터 유형 및 함수를 사용하려면 Geodetic DataBlade 응용프로그램을 포트해야 합니다.

제한사항

현재 Informix Geodetic DataBlade를 사용하는 경우, 다음 기준에 맞으면 DB2 Geodetic Data Management Feature로 이주할 수 있습니다.

- GeoPoint, GeoLineseg, GeoString, GeoRing 및 GeoPolygon 데이터 유형만 사용하십시오.
- 아래 테이블에서 설명한 바와 같이 DB2 Geodetic Data Management Feature에 동일하거나 유사한 해당 함수가 있는 Geodetic DataBlade 함수만 사용하십시오.
- GeoObjects의 공간 구성요소만 인덱스하십시오. 즉, 시간 범위 또는 높이 범위는 인덱스하지 마십시오.

IBM Informix Geodetic DataBlade를 사용하여 데이터베이스에 geospatial 오브젝트를 저장 및 조작하는 경우, 데이터 및 응용프로그램을 IBM DB2 Geodetic Data Management Feature로 이주하는 데 몇 가지 제한사항이 있습니다.

이 태스크를 수행하려면

1. IBM Informix Geodetic DataBlade에서 IBM DB2 Geodetic Data Management Feature로 이주하려면 다음을 수행하십시오.

표 13. Informix Geodetic DataBlade 및 Geodetic Data Management Feature의 해당 데이터 유형

Informix Geodetic DataBlade의 데이터 유형	**DB2 Geodetic Data Management Feature의 해당 데이터 유형**	**유사 데이터 유형에 대한 주석**
GeoEllipse		먼저 GeoPolygon으로 변환한 다음 ST_Polygon으로 이주
GeoLineseg	ST_LineString	
GeoObject	ST_Geometry	ST_Geometry 및 해당 부속 유형은 GeoAltRange 및 GeoTimeRange 데이터 유형을 지원하지 않음
GeoPoint	ST_Point	
GeoPolygon	ST_MultiPolygon, ST_Polygon	ST_MultiPolygon의 각 링에는 명시적인 닫힘 점이 있어야 합니다. GeoPolygon에 외부 링이 있는 경우 ST_Polygon으로 맵핑할 수 있습니다.
GeoRing	ST_LineString	
GeoString	ST_LineString	

다음 Geodetic DataBlade 데이터 유형은 Geodetic Data Management Feature에 해당 데이터 유형이 없습니다.

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

a. DB2 Geodetic Data Management Feature 데이터 유형 및 함수를 사용하도록 SQL문을 다시 작성하십시오.

b. 데이터를 DB2 Geodetic Data Management Feature로 로드 또는 임포트하십시오.

c. Informix ODBC, ESQL/C 및 JDBC를 사용하여 응용프로그램을 다시 작성하십시오. 162 페이지의 표 23은 Geodetic DataBlade 및 Geodetic Data Management Feature의 해당 클라이언트 연결을 보여줍니다.

표 14. Informix Geodetic DataBlade 및 Geodetic Data Management Feature의 해당 술어 함수

Informix Geodetic DataBlade의 함수	DB2 Geodetic Data Management Feature의 해당 함수
Contains	ST_Contains
Inside	ST_Within
Intersect	ST_Intersects
Outside	ST_Disjoint
Within	ST_Distance

2. 다음 Geodetic DataBlade 술어 함수는 Geodetic Data Management Feature에 해당 함수가 없습니다.
   - Beyond
   - Equal
   - Nearest

표 15. Informix Geodetic DataBlade 및 Geodetic Data Management Feature의 해당 프로덕션 함수

Informix Geodetic DataBlade의 함수	DB2 Geodetic Data Management Feature의 해당 함수	유사 함수에 대한 주석
Difference	ST_Difference	ST_Difference는 다각형 및 점을 지원함
Generalize	ST_Generalize	
Intersection	ST_Intersection	ST_Intersection(line,line)을 실행하면 다중점이 생성될 수 있습니다. ST_Intersection(line,poly)을 실행하면 다중 선스트링이 생성될 수 있습니다. 해체 오브젝트에는 Empty가 리턴됩니다.
SymDifference	ST_SymDifference	ST_SymDifference는 다각형 및 점을 지원함
Union	ST_Union	ST_Union은 다각형과 함께 점 및 선을 지원함

표 16. Informix Geodetic DataBlade 및 Geodetic Data Management Feature의 해당 액세서 함수

Informix Geodetic DataBlade의 함수	DB2 Geodetic Data Management Feature의 해당 함수	유사 함수에 대한 주석
Center	ST_MidPoint, ST_PointOnSurface	ST_MidPoint는 선의 유사한 대체 함수입니다. ST_PointOnSurface는 다각형의 유사한 대체 함수입니다.
Coords	ST_PointN	
차원	ST_Dimension	
HasZValue	ST_Is3d	
IsGeoBox	IS OF 표현식 또는 ST_GeometryType 사용	
IsGeoCircle	IS OF 표현식 또는 ST_GeometryType 사용	

표 16. Informix Geodetic DataBlade 및 Geodetic Data Management Feature의 해당 액세서 함수 (계속)

Informix Geodetic DataBlade의 함수	**DB2 Geodetic Data Management Feature의 해당 함수**	**유사 함수에 대한 주석**
IsGeoEllipse	IS OF 표현식 또는 ST_GeometryType 사용	
IsGeoLineseg	IS OF 표현식 또는 ST_GeometryType 사용	
IsGeoPoint	IS OF 표현식 또는 ST_GeometryType 사용	
IsGeoPolygon	IS OF 표현식 또는 ST_GeometryType 사용	
IsGeoRing	IS OF 표현식 또는 ST_GeometryType 사용	
IsGeoString	IS OF 표현식 또는 ST_GeometryType 사용	
위도	ST_Y	
경도	ST_X	
NPoints	ST_NumPoints	
NRings	ST_NumGeometries, ST_NumInteriorRing	ST_NumGeometries를 사용하여 총 외부 링 수를 얻고 다중 다각형 세트의 각 다각형에 대한 ST_NumInteriorRings를 더하십시오.
Ring	ST_GeometryN, ST_ExteriorRing, ST_InteriorRingN	ST_ExteriorRing 및 ST_InteriorRingN과 함께 ST_GeometryN 사용
SRID	ST_SRID	
Zvalue	ST_Z	

3. 다음 Geodetic DataBlade 액세서 함수는 Geodetic Data Management Feature에 해당 함수가 없습니다.
   - IsLarge
   - IsSmallArea

표 17. Informix Geodetic DataBlade 및 Geodetic Data Management Feature의 해당 수정자 함수

Informix Geodetic DataBlade의 함수	**DB2 Geodetic Data Management Feature의 해당 함수**
SetSRID	ST_SRID

4. 다음 Geodetic DataBlade 수정자 함수는 Geodetic Data Management Feature에 해당 함수가 없습니다.
   - SetAltRange
   - SetAltRangeZ
   - SetDist

- SetTimeRange

표 18. Informix Geodetic DataBlade 및 Geodetic Data Management Feature의 해당 측정 함수

Informix Geodetic DataBlade의 함수	**DB2 Geodetic Data Management Feature의 해당 함수**
Area	ST_Area
Distance	ST_Distance
Length	ST_Length, ST_Perimeter

5. VoronoiResolution 측정 함수는 Geodetic Data Management Feature에 해당 함수가 없습니다.

표 19. Informix Geodetic DataBlade 및 Geodetic Data Management Feature의 해당 다운캐스트 함수

Informix Geodetic DataBlade의 함수	**DB2 Geodetic Data Management Feature의 해당 함수**
GeoBox	SQL TREAT 표현식 사용
GeoCircle	SQL TREAT 표현식 사용
GeoEllipse	SQL TREAT 표현식 사용
GeoLineseg	SQL TREAT 표현식 사용
GeoPoint	SQL TREAT 표현식 사용
GeoPolygon	SQL TREAT 표현식 사용
GeoRing	SQL TREAT 표현식 사용
GeoString	SQL TREAT 표현식 사용

표 20. Informix Geodetic DataBlade 및 Geodetic Data Management Feature의 해당 컨스트럭터 함수

Informix Geodetic DataBlade의 함수	**DB2 Geodetic Data Management Feature의 해당 함수**
GeoCoords	ST_Point
GeoPoint	ST_Point

6. 다음 Geodetic DataBlade 컨스트럭터 함수는 Geodetic Data Management Feature에 해당 함수가 없습니다.
   - GeoBox
   - GeoCircle
   - GeoEllipse
   - GeoLineseg

표 21. Informix Geodetic DataBlade 및 Geodetic Data Management Feature의 해당 진단 함수

Informix Geodetic DataBlade의 함수	**DB2 Geodetic Data Management Feature의 해당 함수**
GeoTraceLevel	DB2 추적 기능
IsValidGeometry	ST_IsValid

7. 다음 Geodetic DataBlade 진단 함수는 Geodetic Data Management Feature에 해당 함수가 없습니다.
   - GeoInRowSize

- GeoOutOfRowSize
- GeoRelease
- GeoTotalSize
- GeoTraceLevelSet
- GeoWarningLevel
- GeoWarningLevelSet
- IsValidSDTS

표 22. Informix Geodetic DataBlade 및 Geodetic Data Management Feature의 해당 시스템 카탈로그 테이블

Informix Geodetic DataBlade의 시스템 카탈로그 테이블	**DB2 Geodetic Data Management Feature의 해당 카탈로그 뷰**
GeoLenUnit	DB2GSE.ST_UNITS_OF_MEASURE
GeoSpatialRef	DB2GSE.SPATIAL_REF_SYS

8. 다음 Geodetic DataBlade 시스템 카탈로그 테이블은 Geodetic Data Management Feature에 해당하는 테이블 또는 뷰가 없습니다.
   - GeoEllipsoid
   - GeoParam
   - GeoVoronoi
9. 다음 Geodetic DataBlade 사용자 설정 가능 매개변수 함수는 Geodetic Data Management Feature에 해당 함수가 없습니다.
   - GeoParamSessionGet
   - GeoParamSessionSet
10. 다음 Geodetic DataBlade AltRange 함수는 Geodetic Data Management Feature에 해당 함수가 없습니다.
    - AltRange
    - Bottom
    - Contains
    - Equal
    - Inside
    - Intersect
    - IsAny
    - Outside
    - Top
11. 다음 Geodetic DataBlade TimeRange 함수는 Geodetic Data Management Feature에 해당 함수가 없습니다.

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

12. 다음 Geodetic DataBlade ellipse 함수는 Geodetic Data Management Feature에 해당 함수가 없습니다.
    - Azimuth
    - Coords
    - Major
    - Minor
13. 다음 Geodetic DataBlade circle 함수는 Geodetic Data Management Feature에 해당 함수가 없습니다.
    - Coords
    - Radius
14. 다음 Geodetic DataBlade 각도 산술 함수는 Geodetic Data Management Feature에 해당 함수가 없습니다.
    - Divide
    - Minus
    - Negate
    - Plus
    - Times
15. 다음 Geodetic DataBlade 클라이언트 연결성은 Geodetic Data Management Feature에 해당 클라이언트 연결성이 없습니다.
    - Java API
    - LIBMI

표 23. Geodetic DataBlade 및 DB2 Geodetic Data Management Feature의 해당 클라이언트 연결성 제품

Informix Geodetic DataBlade의 클라이언트 연결성	**DB2 Geodetic Data Management Feature의 해당 클라이언트 연결성**
ESQLC	SQC

표 23. Geodetic DataBlade 및 DB2 Geodetic Data Management Feature의 해당 클라이언트 연결성 제품 (계속)

Informix Geodetic DataBlade의 클라이언트 연결성	**DB2 Geodetic Data Management Feature의 해당 클라이언트 연결성**
ODBC	ODBC
JDBC	JDBC

측지 데이터로 공간 컬럼 채우기

공간 컬럼을 작성하고 공간 인덱스를 작성할 컬럼을 등록하고 나면 측지 데이터로 컬럼을 채울 수 있습니다. 측지 데이터는 다음과 같은 방법으로 제공할 수 있습니다.

- 새 테이블 또는 기존 테이블로 다음 데이터 형식 임포트
  - 모양
  - SDE
- 다음 데이터 형식으로 값 삽입 또는 갱신
  - 모양
  - SDE
  - WKT(Well-Known Text)
  - WKB(Well-Known Binary)
  - GML(Geography Markup Language)

제한사항:

- Spatial Extender의 경우 지오코더(Geocoder) 명령이나 스토어드 프로시저를 사용하여 데이터를 측지 데이터로 변환할 수 없습니다.
- 측지 동작의 경우 SRID 범위가 2,000,000,000 - 2,000,001,000인 공간 참조 시스템을 사용하십시오.
- 모양 데이터 및 SDE 전송 데이터는 지리 좌표 시스템에 있어야 합니다.

측지 데이터를 임포트하는 프로시저는 공간 데이터와 동일합니다.

제 18 장 측지 인덱스

측지 데이터를 쿼리할 때 성능을 향상시킬 수 있는 측지 Voronoi 인덱스를 작성할 수 있습니다. 이 장에서는 다음 사항에 대해 다룹니다.

- 측지 Voronoi 인덱스에 대해 설명합니다.
- Voronoi 셀 구조 및 대체 구조 선택 시기에 대해 설명합니다.
- 측지 Voronoi 인덱스 작성 방법에 대해 설명합니다.

측지 Voronoi 인덱스

DB2® Geodetic Data Management Feature에서는 측지 데이터에 보다 빨리 액세스하는 측지 Voronoi 인덱스를 제공합니다. 이 인덱스는 지구 표면의 Voronoi 모자이크 세공을 사용하여 측지 데이터에 대한 액세스를 구성합니다.

Geodetic Data Management Feature에서는 각 Geometry에 대한 MBC를 계산합니다. MBC는 측지 Geometry를 감싸는 원입니다. Voronoi 인덱스에서는 이 MBC 정보를 사용하여 셀 구조로 데이터를 구성합니다. Voronoi 인덱스를 사용하여 검색하면 구성된 데이터를 내림차순으로 빠르게 검색하여 원하는 일반 영역에서 오브젝트를 찾은 다음 해당 오브젝트에 대한 보다 정확한 테스트를 수행할 수 있습니다. Voronoi 인덱스는 필요한 영역 외부의 오브젝트를 검사하지 않아도 되므로 성능을 향상시킬 수 있습니다. Voronoi 인덱스를 사용하지 않는 경우 쿼리 수행시 쿼리 기준과 일치하는 오브젝트를 찾기 위해 모든 오브젝트를 검사해야 합니다.

옵티마이저는 WHERE절에 다음 함수를 포함하는 모든 쿼리의 측지 Voronoi 인덱스 사용을 고려합니다.

- EnvelopesIntersect
- ST_Contains
- ST_Distance
- ST_EnvIntersects
- ST_Intersects
- ST_MBRIntersects
- ST_Within

측지 Voronoi 인덱스 작성 시 대체 Voronoi 셀 구조를 선택할 수 있습니다.

© Copyright IBM Corp. 1998, 2006

Voronoi 셀 구조

DB2® Geodetic Data Management Feature에서는 계산을 효율적으로 수행하기 위해 지구 표면을 보다 작고 관리가 용이한 벌집 모양의 셀로 세분화합니다. 이러한 세분화를 *Voronoi 모자이크 세공*이라고 하며 이를 설명하는 데이터 구조를 *Voronoi 셀 구조*라고 합니다. Voronoi 모자이크 세공은 각 셀의 내부가 다른 격자 점보다 특정 격자점에 가까운 모든 점으로 구성되는 셀 구조입니다. Voronoi 셀 구조에서 셀은 *볼록한 선체*입니다. 한 세트의 점으로 구성되는 볼록한 선체는 점을 포함하는 가장 작은 돌출형 모양(또는 점 집합의 "외부"를 정의하는 가장 작은 다각형)입니다. Voronoi 셀 구조는 불규칙한 모양의 다각형인 경우가 많습니다. 셀 수와 위치는 공간 데이터의 밀도와 위치에 적합하도록 조정할 수 있습니다.

예를 들어, Voronoi 셀 구조는 지구를 인구 기준의 다각형으로 나눌 수 있습니다. 인구(및 데이터) 밀도가 높은 지역은 작은 다각형으로 구성됩니다. 인구 밀도가 낮은 지역은 큰 다각형으로 구성됩니다.

167 페이지의 그림 21에서는 세계 인구 밀도를 기준으로 하는 Voronoi 구조를 보여줍니다. Geodetic Data Management Feature에서는 이 셀 구조를 공간 계산에 사용합니다.

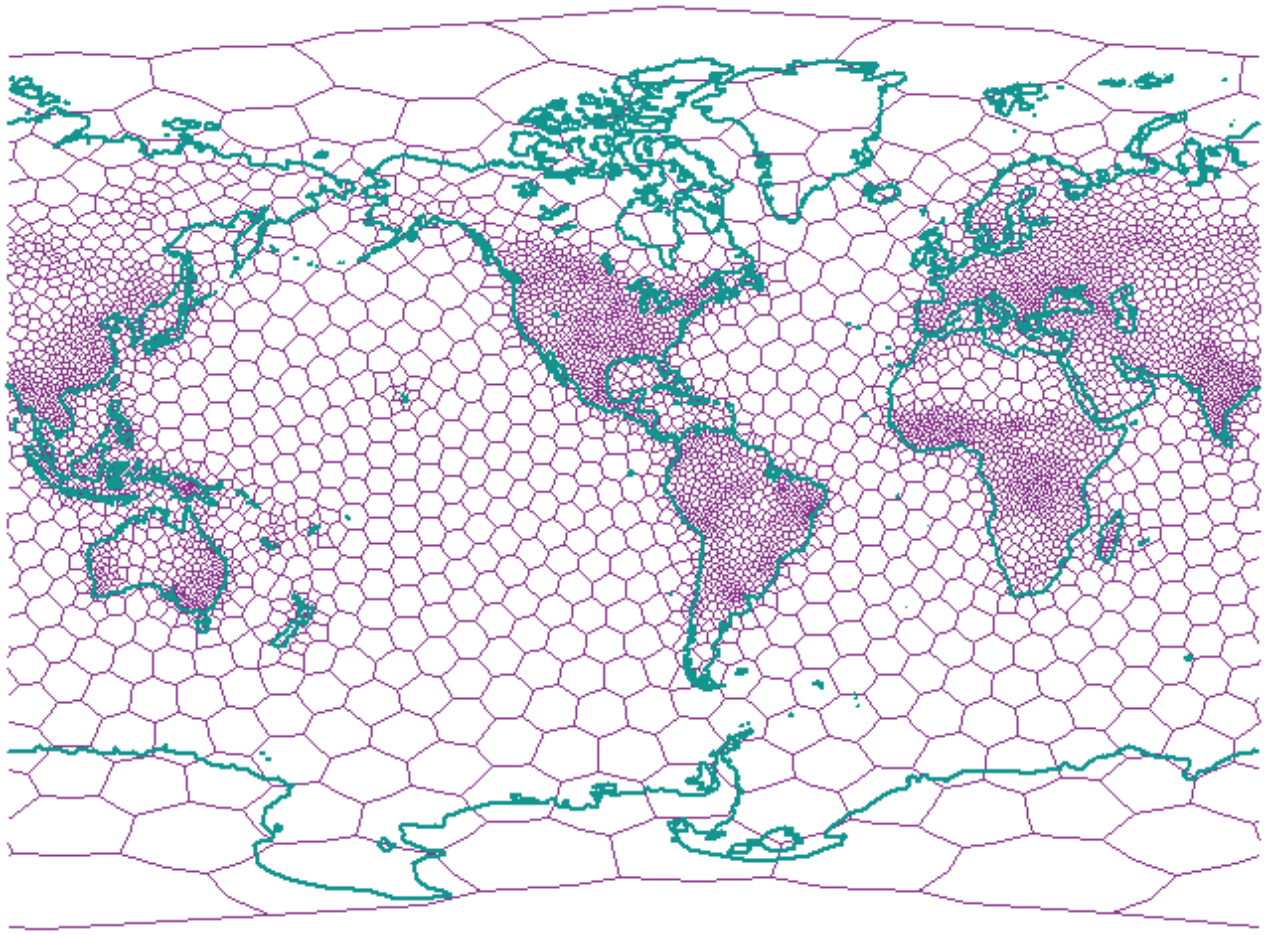


그림 21. 세계 인구 밀도를 기준으로 하는 Voronoi 구조

대체 Voronoi 셀 구조 선택 시 고려사항

측지 Geometry에 대한 모든 조작에서는 세계 인구 밀도를 기초로 Voronoi 셀 구조를 지정하는 Voronoi ID 1을 사용합니다. 인덱스를 작성할 때 데이터가 하나 이상의 지구 영역으로 클러스터되는 경우(예: 1개국 이상의 국가에 대한 거리 데이터), 데이터가 위치하는 영역에 더 작은 셀이 있는 대체 Voronoi 셀 구조(해상도는 셀 크기에 반비례하기 때문)를 선택할 수 있습니다. DB2® Geodetic Data Management Feature에서는 사용자 데이터에 보다 적합한 인덱싱을 위해 여러 Voronoi 셀 구조를 제공합니다.

제한사항:
측지 Voronoi 인덱스 작성 시에만 대체 Voronoi 셀 구조를 선택할 수 있습니다.

Satellite 이미지와 같이 지구의 전체 표면에 균등하게 분산되는 데이터에는 dodeca04 구조(Voronoi ID 12)가 가장 적합합니다. 모든 셀의 크기는 매우 불규칙하며 가장 낮

은 해상도는 10cm입니다. 다음 조건이 데이터 또는 응용프로그램에 적용되는 경우 디폴트 세계 인구 구조(Voronoi ID 1) 또는 dodeca04 구조 이외의 다른 Voronoi 셀 구조 사용을 고려하십시오.

고해상도

오브젝트가 10cm 미만 간격으로 교차하는지 여부를 판별해야 하는 경우 데이터가 있는 영역의 셀이 더 작은 Voronoi 셀 구조를 사용해야 합니다. 해상도는 셀 *크기*에 반비례합니다.

정점이 많은 다각형

데이터가 상대적으로 정점이 많아 영역에서 그 *크기*가 작은 다각형으로 구성되는 경우, 원하는 영역에 셀이 더 많은 Voronoi 셀 구조로 전환할 수 있습니다. 대부분의 다각형의 정점이 50개 이하인 경우, 전환하지 않아도 됩니다. 정점이 많은 데이터 세트의 다각형만 *크기*가 일정한 경우에도 전환하지 않아도 됩니다.

미국 카운티 *크기*에 해당하는 3000개의 정점으로 이루어진 다각형이 많이 있고 특히 응용프로그램이 여러 가지 다각형-교차-다각형 쿼리를 수행하는 경우, 다른 Voronoi 셀 구조로 전환하면 쿼리 성능을 현저히 향상시킬 수 있습니다.

조밀한 데이터

데이터가 매우 작은 영역에 밀집되어 있는 경우(예를 들어, 제곱 km당 수백 개의 오브젝트가 있는 경우), 셀 밀도가 데이터 밀도와 일치하는 Voronoi 셀 구조를 사용하여 쿼리 성능을 향상시킬 수 있습니다.

측지 Voronoi 인덱스 작성

전제조건

측지 Voronoi 인덱스를 작성하기 전에 사용자 ID에는 공간 격자 인덱스를 작성할 때와 동일한 권한 부여 및 특권이 있어야 합니다.

제한사항:

측지 Voronoi 인덱스를 작성할 때에는 CREATE INDEX문을 사용하여 인덱스를 작성할 때와 동일한 제한사항이 적용됩니다. 즉, 인덱스를 작성하는 컬럼은 뷰 컬럼 또는 별칭 컬럼이 아닌 기본 테이블 컬럼이어야 합니다. DB2 데이터베이스 시스템은 이 프로세스에서 별명을 분석합니다.

DB2 Geodetic Data Management Feature에서는 측지 데이터가 있는 컬럼에서 인덱스를 작성할 수 있는 새로운 공간 액세스 메소드를 제공합니다. 인덱스를 사용하는 쿼리는 보다 빠르게 실행할 수 있습니다.

다음 중 한 가지 방법으로 측지 Voronoi 인덱스를 작성할 수 있습니다.

- DB2 제어 센터의 색인 작성 창을 사용하십시오.

- EXTEND USING절에서 SQL CREATE INDEX문과 db2gse.spatial_index 확장자를 함께 사용하십시오.

DB2 제어 센터 또는 명령행 처리기를 사용하여 측지 Voronoi 인덱스를 작성하십시오.

- 제어 센터를 사용하여 측지 Voronoi 인덱스를 작성하려면 측지 Voronoi 인덱스를 작성하려는 공간 컬럼이 있는 테이블을 마우스 오른쪽 단추로 누른 다음 메뉴에서 **Spatial Extender → 공간 인덱스**를 누르십시오. 공간 인덱스 창이 열립니다. 프롬프트에 따라 태스크를 완료하십시오.
- SQL CREATE INDEX문을 사용하여 측지 Voronoi 인덱스를 작성하려면 EXTEND USING절과 db2gse.spatial_index 격자 인덱스 확장자를 사용하여 CREATE INDEX 문을 발행하십시오.

다음 예에서 CREATE INDEX문은 CUSTOMERS 테이블의 공간 컬럼 LOCATION에 STORESX1 측지 인덱스를 작성합니다.

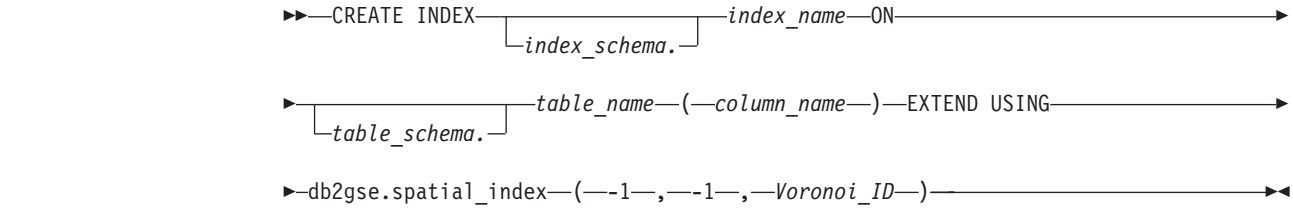
```
CREATE INDEX storesx1
      ON customers (location)
      EXTEND USING db2gse.spatial_index (-1, -1, 1)
```

측지 Voronoi 인덱스의 경우 USING db2gse.spatial_index절의 처음 두 개 매개변수에 -1 값을 지정해야 합니다.

측지 Voronoi 인덱스에 대한 CREATE INDEX문

CREATE INDEX문에 EXTEND USING절을 사용하여 측지 Voronoi 인덱스를 작성하십시오.

구문:



여기서

index_schema

작성 중인 인덱스가 속할 스키마의 이름. 이름을 지정하지 않으면 DB2 데이터베이스 시스템에서 CURRENT SCHEMA 특수 레지스터에 저장된 스키마 이름을 사용합니다.

index_name

작성 중인 측지 인덱스의 규정되지 않은 이름

table_schema

*column_name*을 포함하는 테이블이 속하는 스키마의 이름. 이름을 지정하지 않으면 DB2 데이터베이스 시스템에서 CURRENT SCHEMA 특수 레지스터에 저장된 스키마 이름을 사용합니다.

table_name

*column_name*을 포함하는 테이블의 규정되지 않은 이름

column_name

측지 Voronoi 인덱스가 작성되는 공간 컬럼의 이름

Voronoi_ID

Voronoi 셀 구조 ID를 식별하는 정수. 14개의 Voronoi 셀 구조를 사용할 수 있습니다. Voronoi ID 1은 DB2 Geodetic Data Management Feature에서 모든 공간 작업에 사용하는 세계 인구 밀도를 기초로 하는 Voronoi 셀 구조를 지정합니다.

예

다음 예 CREATE INDEX문은 CUSTOMERS 테이블의 공간 컬럼 LOCATION에 STORESX1 측지 인덱스를 작성합니다.

```
CREATE INDEX storesx1
      ON customers (location)
      EXTEND USING db2gse.spatial_index (-1, -1, 1)
```

옵티마이저는 WHERE절에 다음 함수를 포함하는 모든 쿼리의 Voronoi 인덱스 사용을 고려합니다.

- ST_Contains
- ST_Distance
- ST_Intersects
- ST_MBRIntersects
- ST_EnvIntersects
- EnvelopesIntersect
- ST_Within

다음 명령문은 Voronoi 인덱스 사용 예를 보여줍니다. 먼저 CUSTOMER 테이블에 데이터를 삽입하십시오. 이 첫 번째 INSERT문에 표시되어 있는 대로 직접 값을 입력할 수 있습니다.

```
INSERT INTO customer
(id, last_name, first_name, address, city, state, zip,
location)
  VALUES
```

```
('123-456789', 'Duck', 'Donald',
'123 Mallard Way', 'Wetland Marsh', 'ND', '55555-5555',
db2gse.ST_GeomFromWKT('POINT(123.123, 45.67)', 2000000000))
```

또는 다음 쿼리와 같이 응용프로그램의 변수를 사용하여 테이블에 값을 삽입할 수 있습니다.

```
INSERT INTO customer
(id, last_name, first_name,
address, city, state, zip,
location)
  VALUES
(:mid, :mlast, :mfirst,
:maddress, :mcity, :mstate, :mzip,
db2gse.ST_GeomFromWKB(:mlocation))
```

다음 UPDATE문은 삽입된 데이터를 수정합니다. 이 명령문은 WHERE절에 ST_Contains, ST_Distance, ST_Intersects, ST_MBRIntersects, ST_EnvIntersects, EnvelopesIntersect 또는 ST_Within 함수를 사용하지 않으므로 STORESX1 인덱스를 사용하지 않습니다.

```
UPDATE customer
SET location = db2gse.ST_GeomFromWKT('POINT(123.123, 45.67)',
2000000000)
WHERE id = '123-456789';
```

DELETE문이 WHERE절에서 각각 ST_Within 함수 및 ST_Intersects 함수를 사용하여 인덱스가 성능을 향상시키는 것으로 옵티마이저가 판별하는 경우, 다음 명령문은 STORESX1 인덱스를 사용할 수 있습니다.

```
DELETE FROM customers
WHERE db2gse.ST_Within(location, :BayArea) = 1;
```

```
DELETE FROM customers
WHERE db2gse.ST_Intersects(c.location, :BayArea) = 1
```

다음 두 SELECT문은 STORESX1 인덱스도 사용할 수 있습니다.

```
SELECT s.id, AVG(c.location..ST_Distance(s.location))
FROM customers c, stores s
WHERE db2gse.ST_Within(c.location, s.zone) = 1
GROUP BY s.id;
SELECT c.location..ST_AsText()
FROM customers c
WHERE db2gse.ST_Within(c.location, :BayArea) = 1
```

DB2 Geodetic Data Management Feature에 제공된 Voronoi 셀 구조

각 Voronoi 셀 구조는 지구 전체를 포함합니다. 아래 설명에는 해당 Voronoi 셀 구조에 대한 셀이 조밀한 영역 내 해당 지구 부분만 표시됩니다. Voronoi 셀 구조를 선택하는 경우, 표시된 영역 외부의 셀은 해당 저해상도로 넓게 표시됨에 유의하십시오. 데이터가 이렇게 성긴 영역에 있는 경우, 쿼리 성능이 저하될 수 있습니다.

다음 테이블에서는 DB2 Geodetic Data Management Feature에서 제공하는 Voronoi 셀 구조가 나열됩니다. 이러한 Voronoi 셀 구조는 Geodyssey Ltd.에서 제공합니다.

표 24. Voronoi 셀 구조

설명	Voronoi ID	설명
세계, 인구 밀도 기반	1	173 페이지의 그림 22
미국	2	174 페이지의 그림 23
	3	175 페이지의 그림 24
인도	4	176 페이지의 그림 25
일본	5	177 페이지의 그림 26
아프리카	6	178 페이지의 그림 27
호주	7	179 페이지의 그림 28
유럽	8	180 페이지의 그림 29
북아메리카	9	181 페이지의 그림 30
남아메리카	10	182 페이지의 그림 31
지중해 지역	11	183 페이지의 그림 32
세계, 단일 데이터 분산, 중해상도 (dodeca04)	12	184 페이지의 그림 33
세계, 산업 규모 기반(G7 국가)	13	185 페이지의 그림 34
세계, 단일 데이터 분산, 저해상도(아이소타입)	14	186 페이지의 그림 35

세계, 인구 밀도 기반(Voronoi ID: 1)

Voronoi 셀은 세계 인구 밀도를 기초로 지구를 세분화합니다.

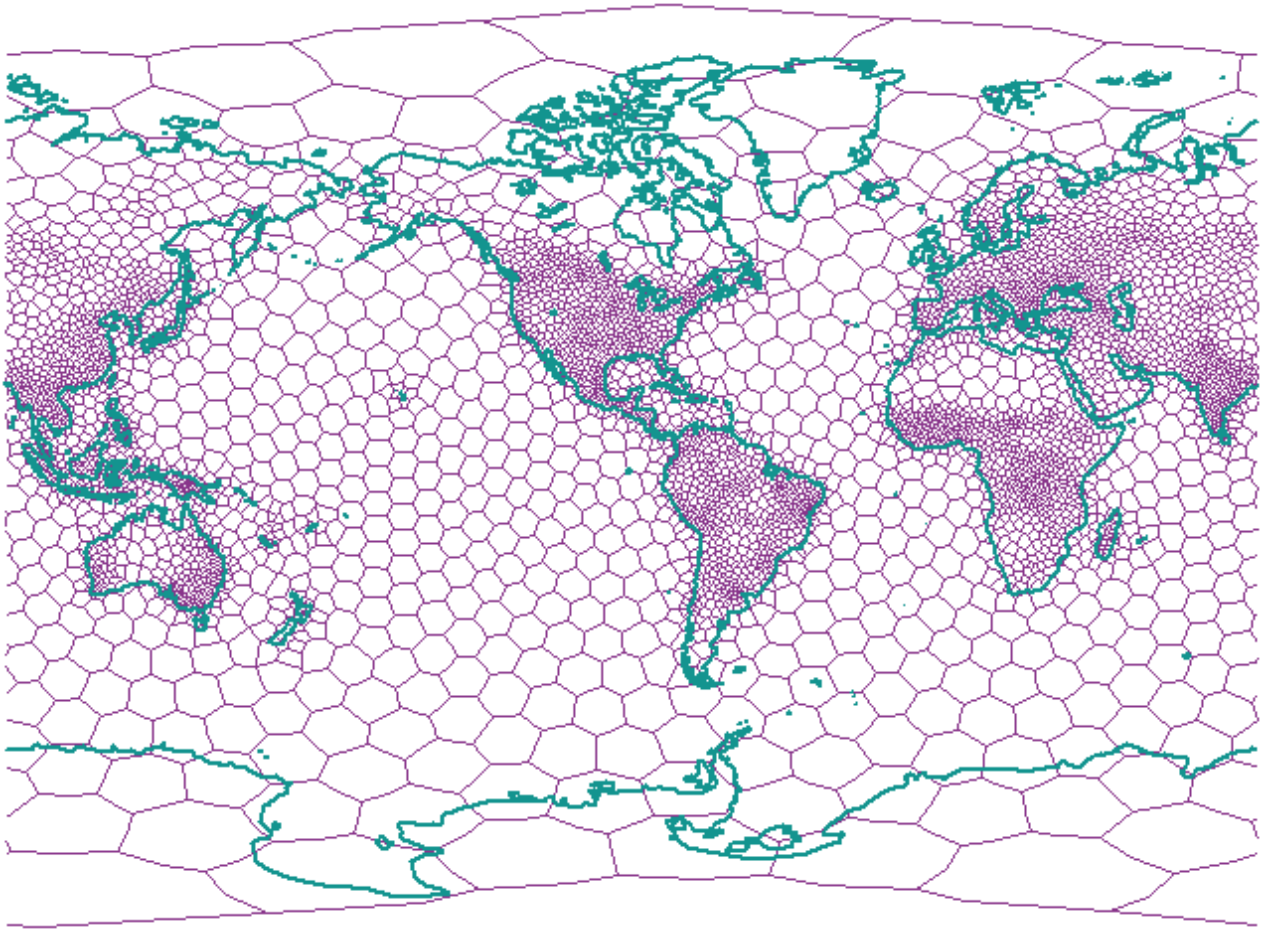


그림 22. 세계(인구) Voronoi 셀 구조

미국(Voronoi ID: 2)

Voronoi 셀은 인구 밀도를 기초로 미국을 세분화합니다.

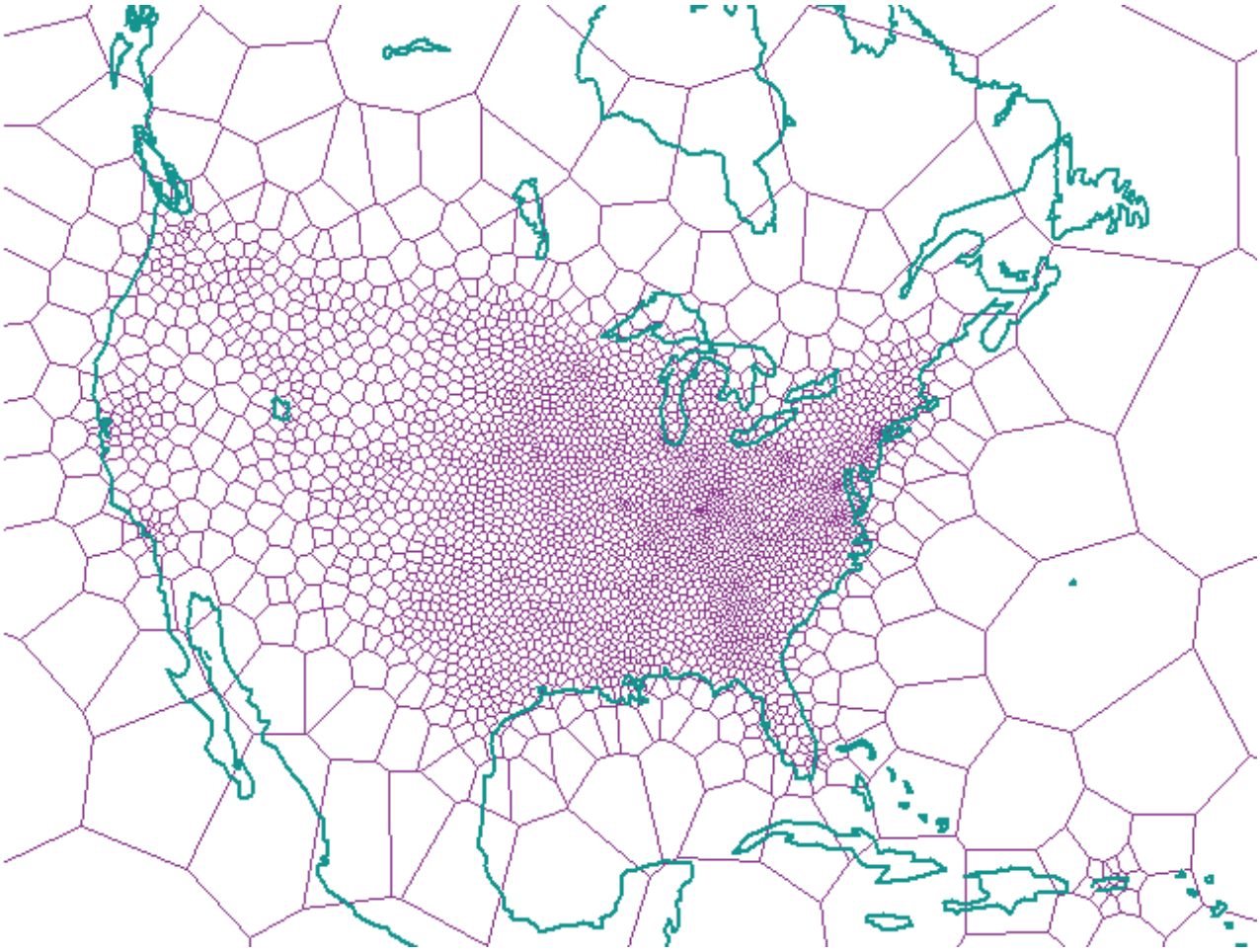


그림 23. 미국 Voronoi 셀 구조

캐나다(Voronoi ID: 3)

Voronoi 셀은 인구 밀도를 기초로 캐나다를 세분화합니다.

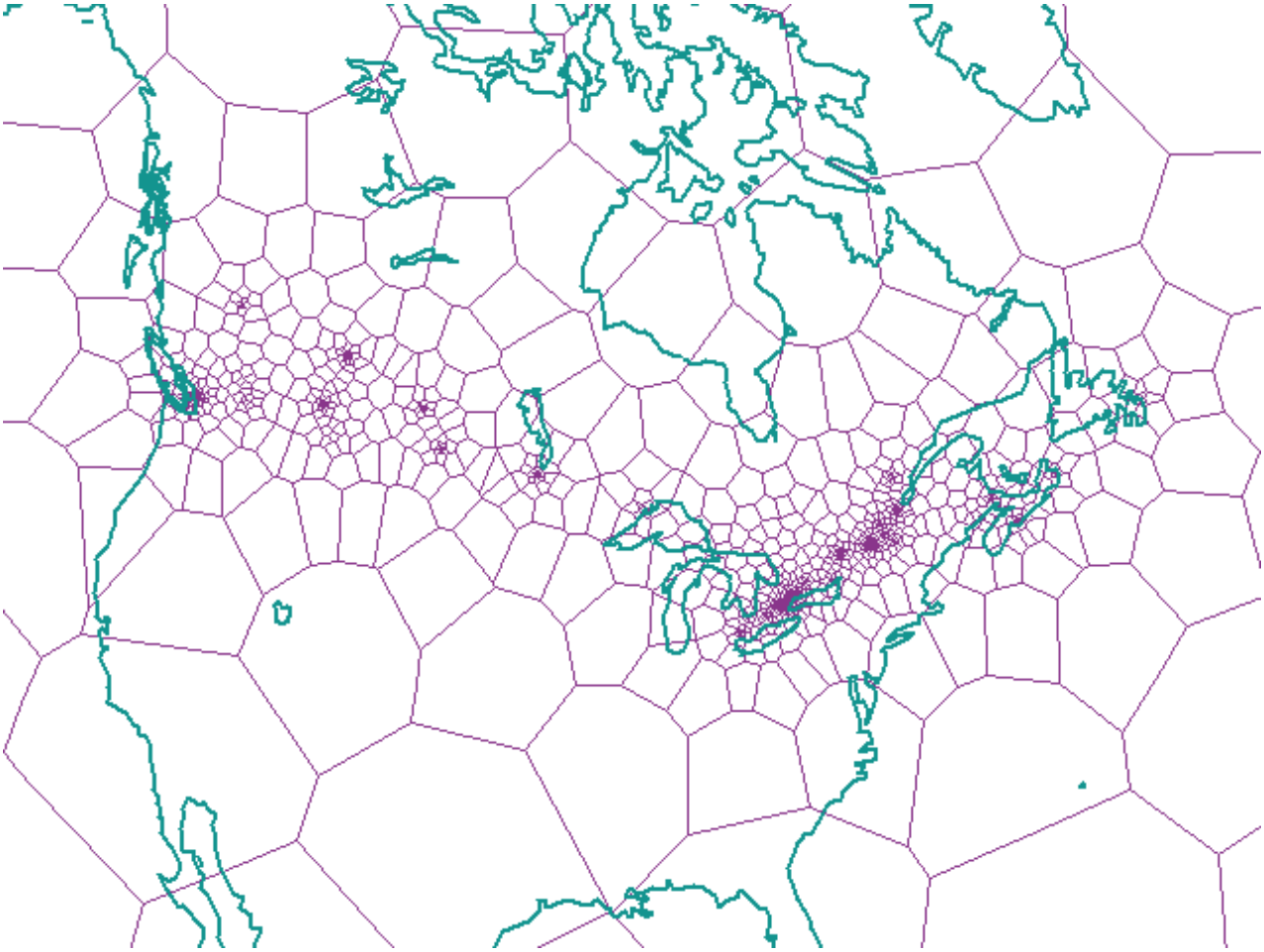


그림 24. 캐나다 Voronoi 셀 구조

인도(Voronoi ID: 4)

Voronoi 셀은 인구 밀도를 기초로 인도를 세분화합니다.

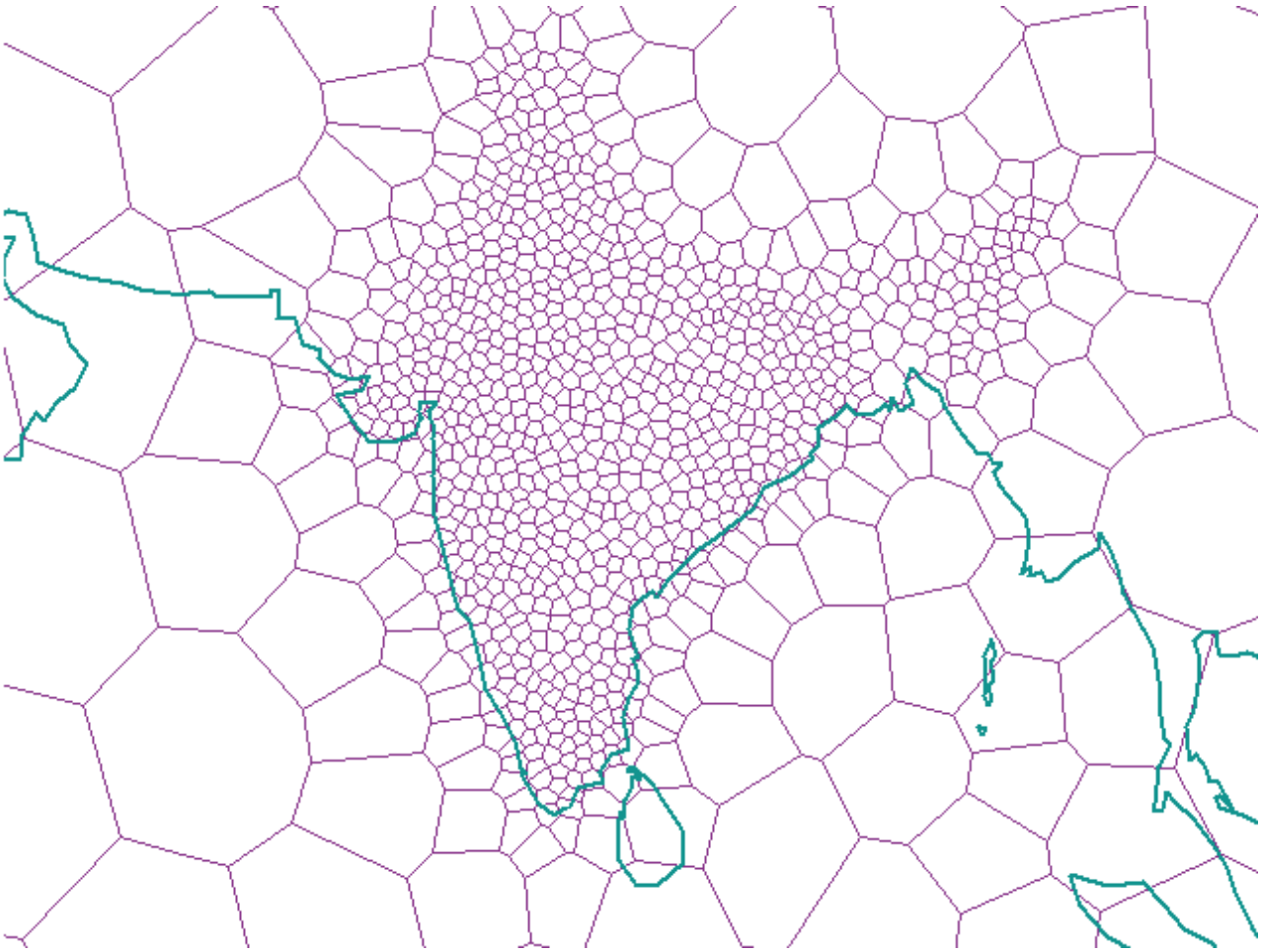


그림 25. 인도 Voronoi 셀 구조

일본(Voronoi ID: 5)

Voronoi 셀은 인구 밀도를 기초로 일본을 세분화합니다.

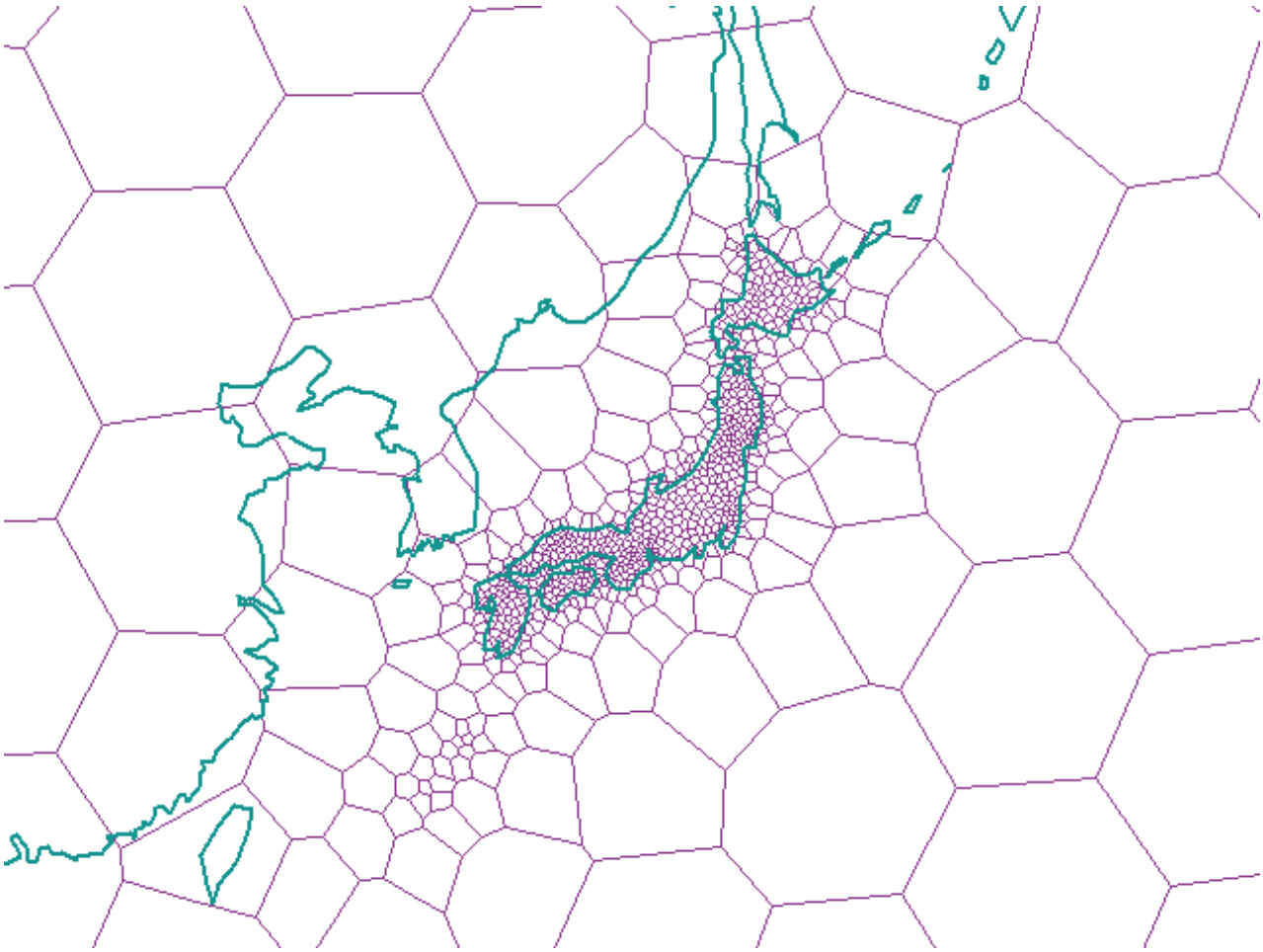


그림 26. 일본 Voronoi 셀 구조

아프리카(Voronoi ID: 6)

Voronoi 셀은 인구 밀도를 기초로 아프리카를 세분화합니다.

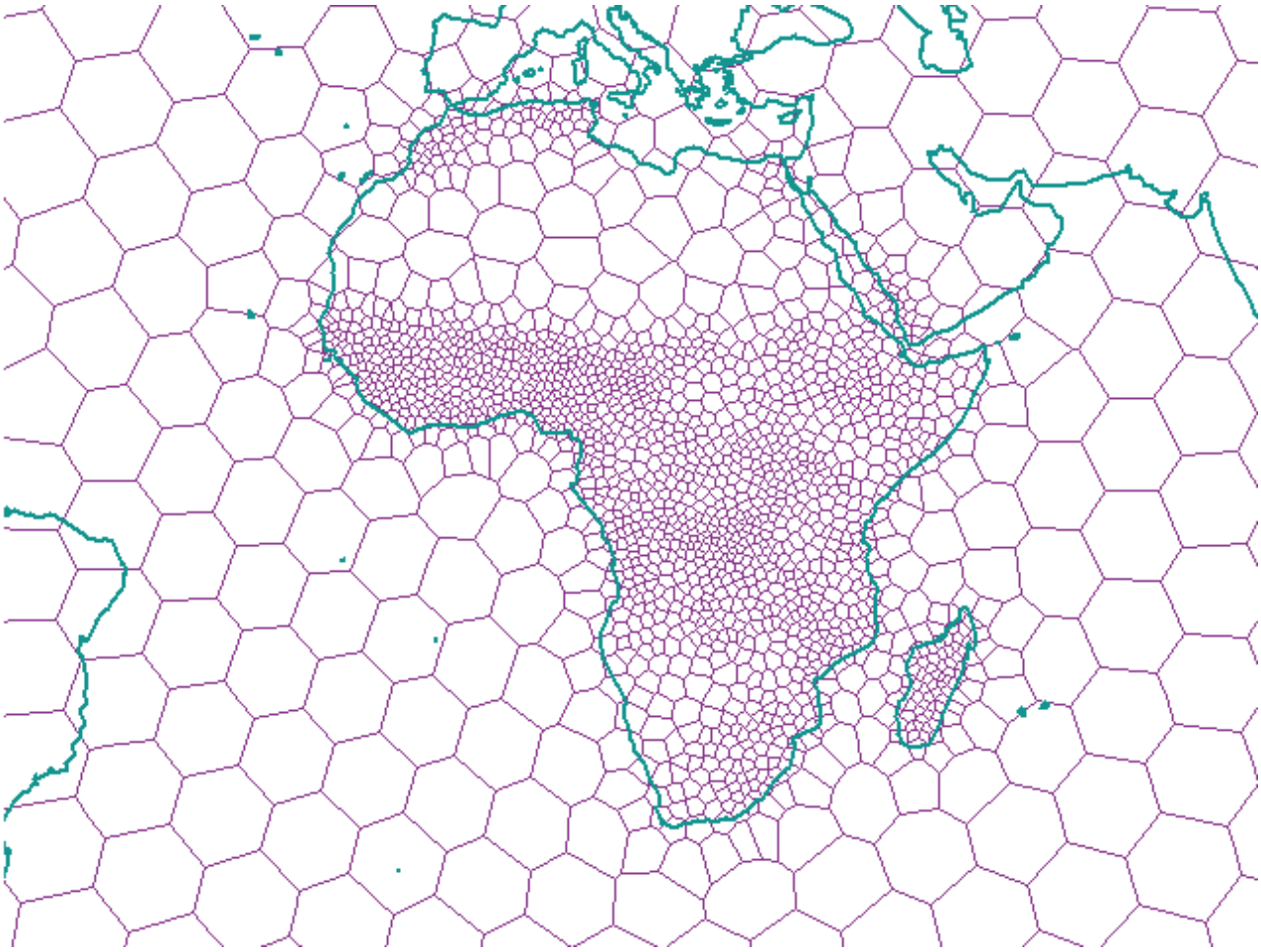


그림 27. 아프리카 Voronoi 셀 구조

호주(Voronoi ID: 7)

Voronoi 셀은 인구 밀도를 기초로 호주를 세분화합니다.

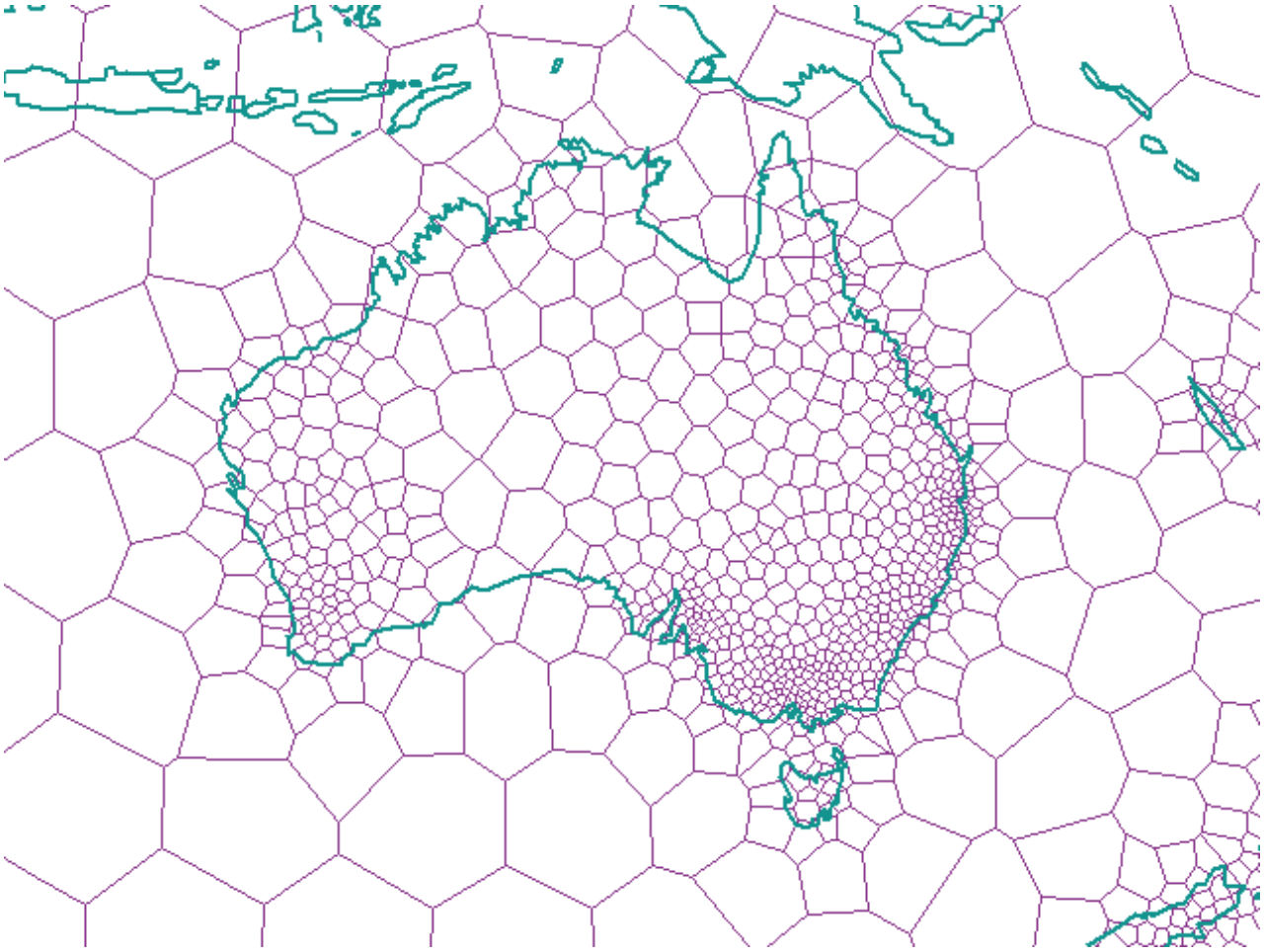


그림 28. 호주 Voronoi 셀 구조

유럽(Voronoi ID: 8)

Voronoi 셀은 인구 밀도를 기초로 유럽을 세분화합니다.

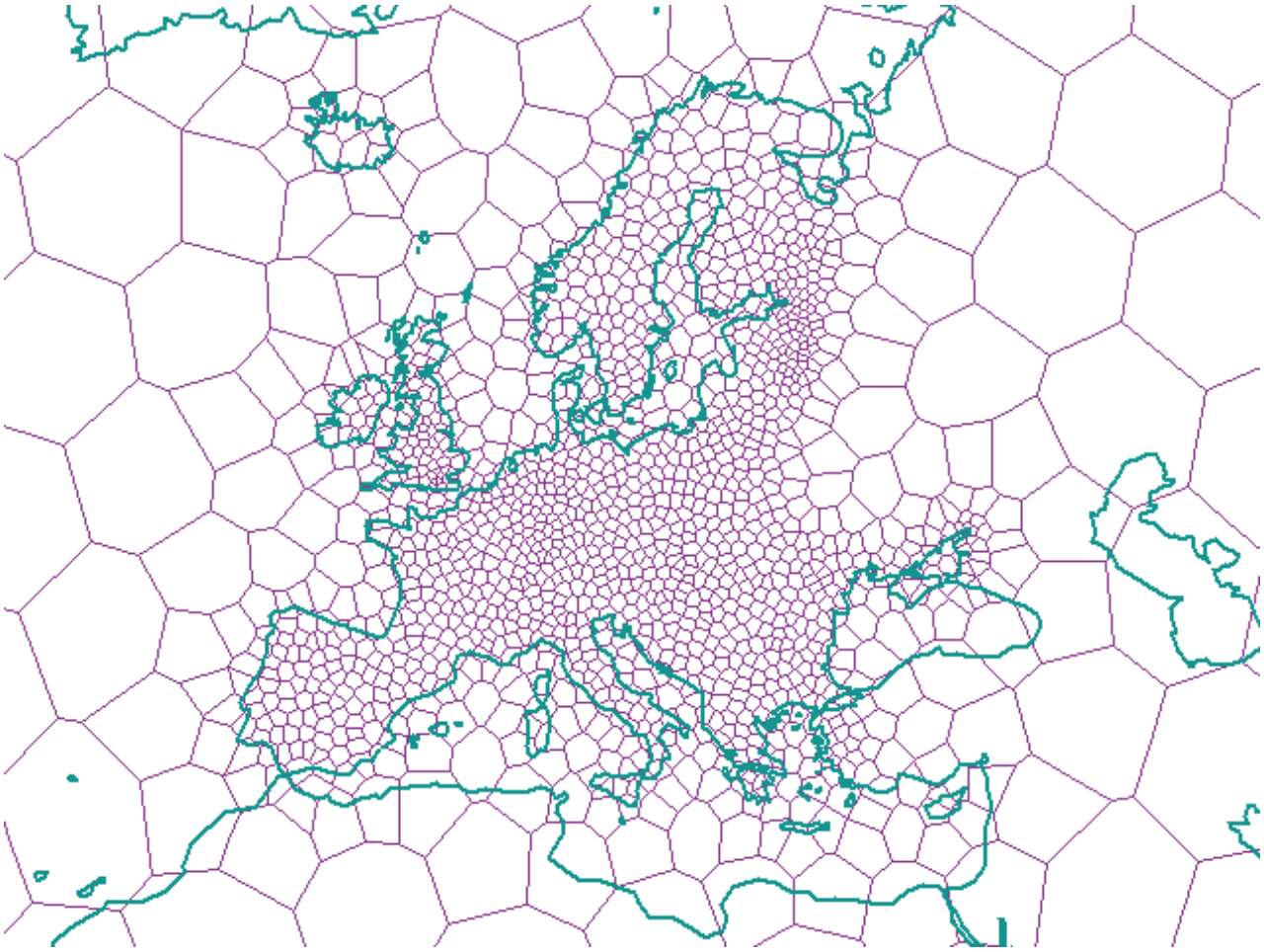


그림 29. 유럽 Voronoi 셀 구조

북아메리카(Voronoi ID: 9)

Voronoi 셀은 인구 밀도를 기초로 북아메리카를 세분화합니다.

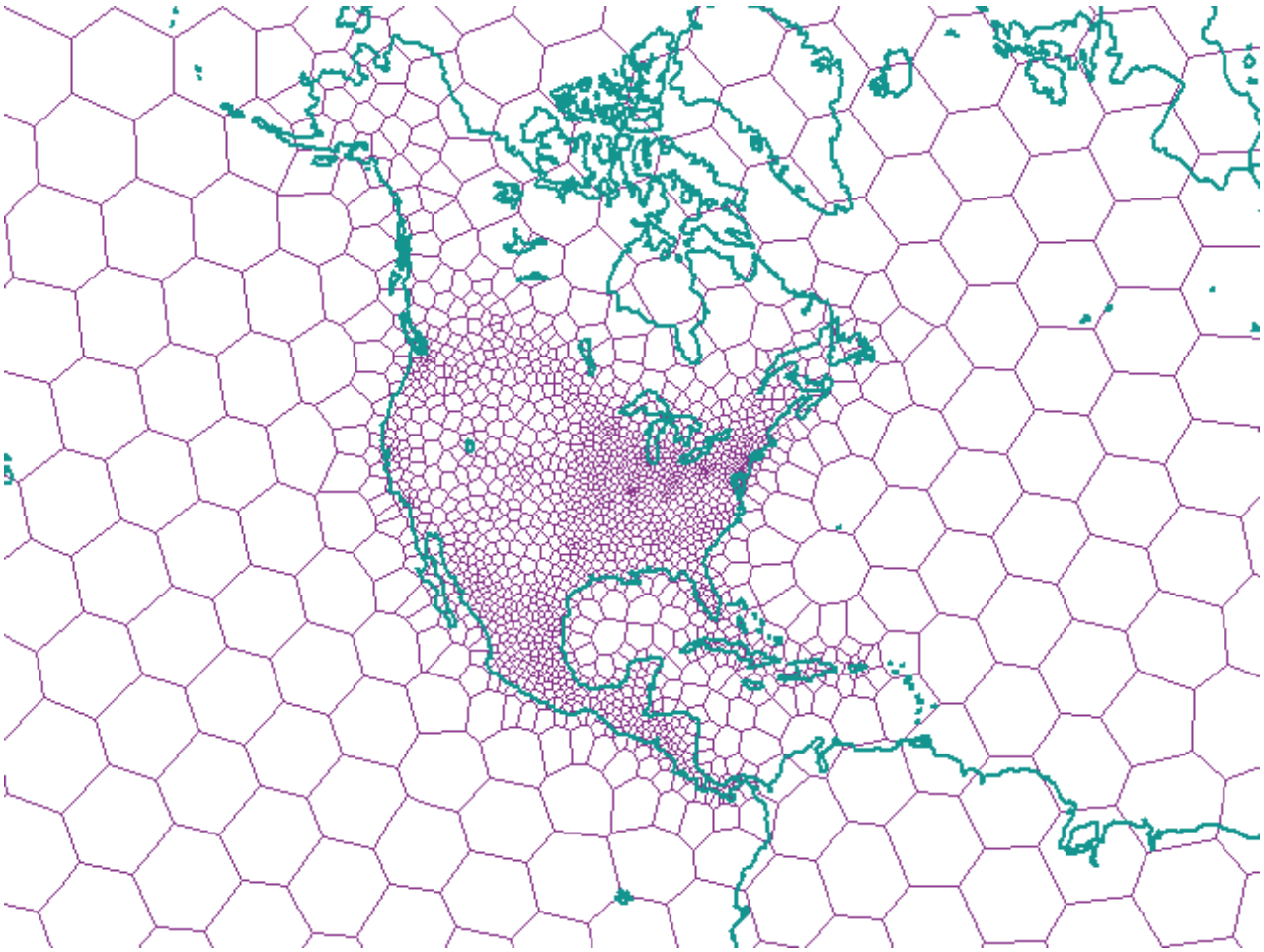


그림 30. 북아메리카 Voronoi 셀 구조

남아프리카(Voronoi ID: 10)

Voronoi 셀은 인구 밀도를 기초로 남아프리카를 세분화합니다.

그림 31. 남아메리카 Voronoi 셀 구조

지중해 지역(Voronoi ID: 11)

Voronoi 셀은 인구 밀도를 기초로 지중해 지역을 세분화합니다.

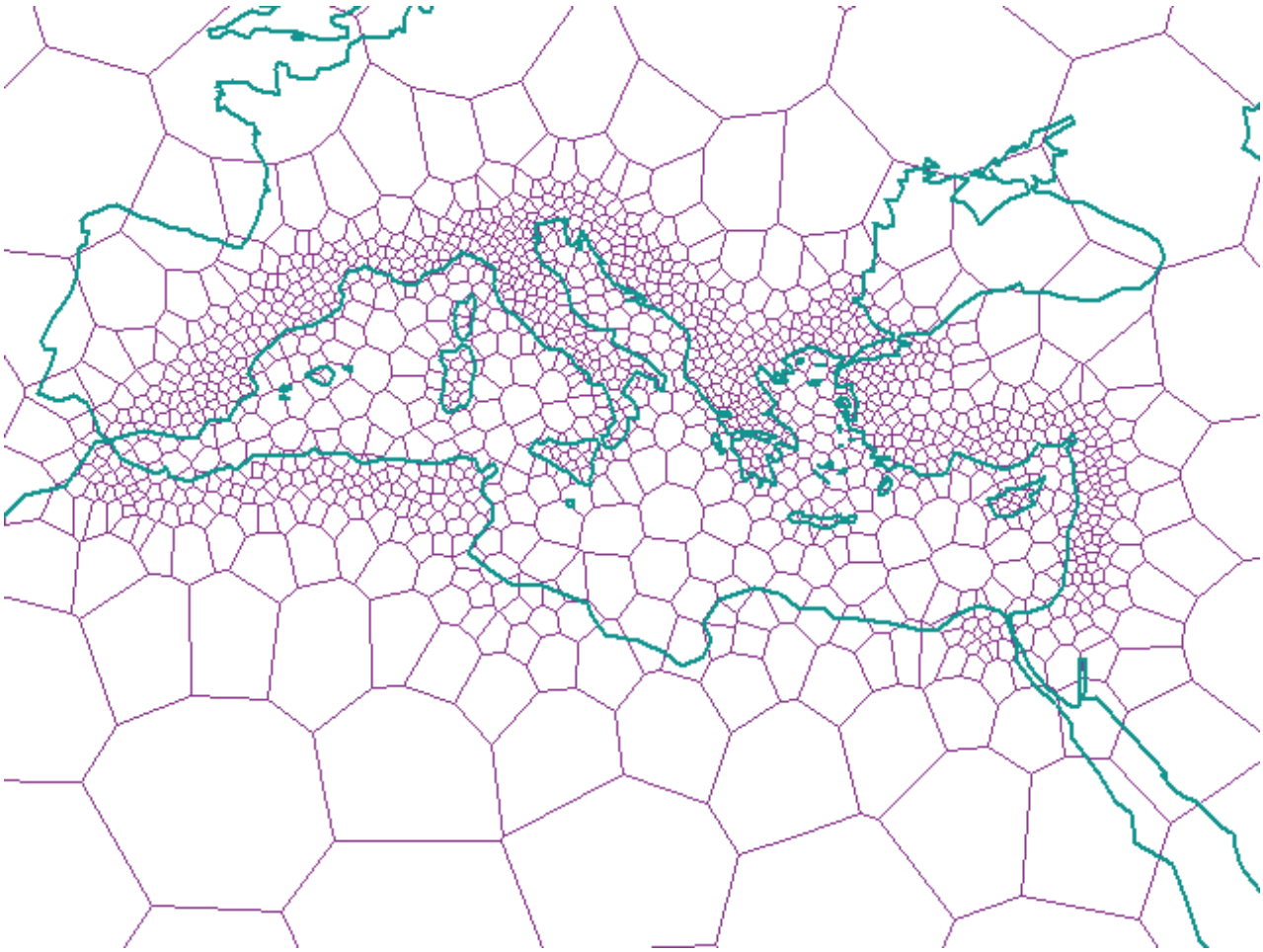


그림 32. 지중해 지역 Voronoi 셀 구조

세계, 단일 데이터 분산, 중간 해상도 – dodeca04 (Voronoi ID: 12)

Voronoi 셀은 단일 데이터 분산 및 중간 해상도로 세계를 세분화합니다.

그림 33. 세계(dodeca04) Voronoi 셀 구조

세계, 산업 국가 - G7 국가(Voronoi ID: 13)

Voronoi 셀은 국가의 산업 생산량을 기초로 세계를 세분화합니다.

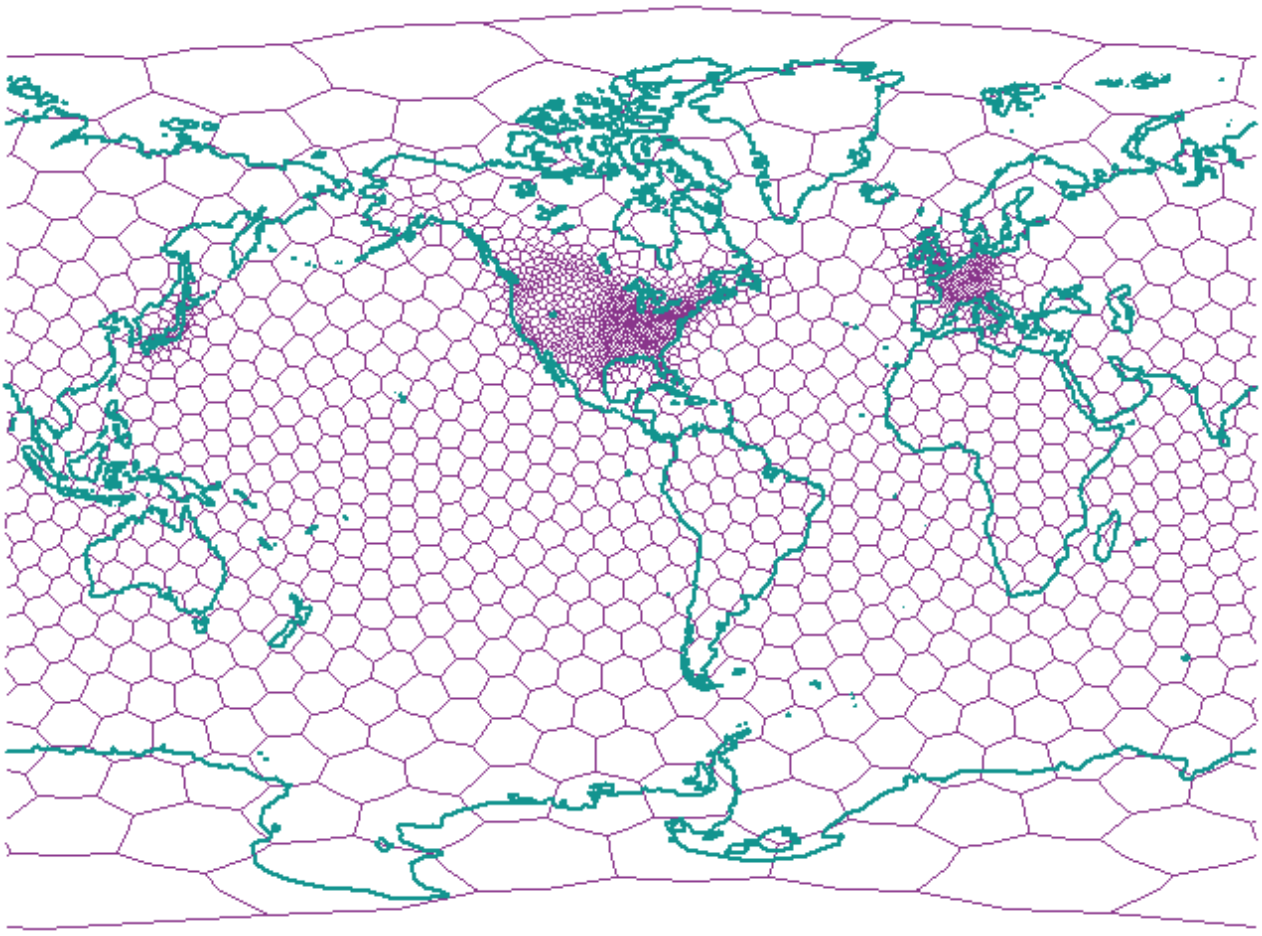


그림 34. G7 국가 Voronoi 셀 구조

세계, 단일 데이터 분산, 저해상도 - 아이소타입(Voronoi ID: 14)

Voronoi 셀은 단일 데이터 분산 및 저해상도로 세계를 세분화합니다.

그림 35. 세계(아이소타입) Voronoi 셀 구조

제 19 장 측지 데이터 사용과 공간 데이터 사용의 차이점

이 장에서는 측지 데이터를 사용하는 경우와 공간 데이터를 사용하는 경우의 다음과 같은 차이점에 대해 설명합니다.

- ST_Geometry 데이터 유형에 대한 최소 및 최대 x, y 속성
- 평면형 지구 표현과 구형 지구 표현 방법의 차이점
- DB2 Geodetic Data Management Feature에서 지원하는 공간 함수 및 함수 동작의 차이점
- DB2 Geodetic Data Management Feature에서 지원하는 스토어드 프로시저 및 카탈로그 뷰
- 추가 측지 공간 참조 시스템(기준면) 및 측지 ellipsoid

최소 및 최대 x 및 y 속성

DB2® Geodetic Data Management Feature에서는 최소 바운드 직사각형 대신 MBC(Minimum Bounding Circle)를 사용하여 데이터를 측지 Voronoi 인덱스의 셀 구조로 구성합니다.

측지 Geometry의 경우, MBC는 Geometry를 둘러싸는 원이며 최소 및 최대 x와 y의 내부 값은 다음과 같습니다.

xmin 바운드 원 중심의 방향 코사인의 *i* 용어

xmax 바운드 원 중심의 방향 코사인의 *j* 용어

ymin 바운드 원 중심의 방향 코사인의 *k* 용어

ymax 바운드 원의 *arc_radius*

측지 Geometry의 경우, ST_MinX, ST_MaxX, ST_MinY 및 ST_MaxY 함수는 MBC 상에서 점을 표시합니다. 이 함수의 결과는 공간 Geometry와 유사한 경도 및 위도 값을 생성하지만 결과는 다음과 같이 측지 Geometry와 다를 수 있습니다.

- MBC가 날짜 변경선을 지나는 경우, ST_MinX 값은 ST_MaxX 값보다 큽니다. 예를 들어, MBC의 중심이 날짜 변경선에 있고 반지름이 5도인 경우, ST_MinX 값은 175이고 ST_MaxX 값은 -175입니다.
- MBC에 북극이나 남극이 포함되는 경우, ST_MinX는 -180이고 ST_MaxX은 180입니다.
- MBC에 북극이 포함되는 경우, ST_MaxY 값은 90입니다.
- MBC에 남극이 포함되는 경우 ST_MinY 값은 -90입니다.

© Copyright IBM Corp. 1998, 2006

평면형 지구 표현과 구형 지구 표현 방법의 차이점

DB2 Spatial Extender 및 DB2 Geodetic Data Management Feature에서는 다른 핵심 기술을 사용합니다.

- Spatial Extender는 투영 좌표를 기반으로 하는 평면 지도를 사용합니다. 그러나 모든 지도에는 가장자리가 있는 반면 지구에는 가장자리가 없으므로 맵 투영으로 전체 지구를 올바르게 표현할 수는 없습니다.
- Geodetic Data Management Feature에서는 타원체를 모델로 사용하여 지구를 극 지방이나 180도 경선 끝에 왜곡 현상이 발생하지 않는 완전한 구로 간주합니다.

이 절에서 평면형 지구는 전체 지구를 표현하기 위해 투영 방법을 사용함을 의미합니다. 구형 지구는 타원체를 지구 모델로 사용하는 참조 시스템을 사용함을 의미합니다.

서로 다른 기술로 인해 이 주제에서 설명한 것과 같은 특정 상황에서 Geometry를 처리하는 방법이 달라집니다.

- 180도 경선을 지나는 선 세그먼트(및 측정 거리)
- 180도 경선을 지나는 다각형
- 180도 경선을 지나는 최소 바운드 직사각형
- 극을 감싸는 다각형
- 반구, 적도 지역 또는 전체 지구를 표현하는 다각형

180도 경선을 교차하거나 극에 가까운 Geometry에 대해 작업할 때 Spatial Extender에서 사용하는 평면 지구 표현에 한계가 있는 경우 Geodetic Data Management Feature가 특히 유용합니다.

180도 경선을 지나는 선 세그먼트

구형 지구 표현을 사용하면 평면 지구 프로젝션보다 경로가 짧아집니다.

189 페이지의 그림 36에서는 Spatial Extender와 Geodetic Data Management Feature가 180도 경선을 지나는 선 세그먼트를 처리하는 다른 방법을 보여줍니다. 이 예에서, 선 세그먼트는 앵커리지와 동경 사이의 거리를 측정하는 데 사용됩니다. Geodetic Data Management Feature는 타원체에 있는 두 점 사이의 최단 경로인 측지선 상의 거리를 측정합니다. 두 점은 지구 상의 어느 위치에도 가능하며 Geodetic Data Management Feature는 구형 지구 표현을 사용하므로 앵커리지에서 동경까지 서쪽으로 이동하는 선 세그먼트를 올바르게 선택합니다. 그러나 Spatial Extender의 경우, 평면 지도 프로젝션을 사용하므로 앵커리지에서 동경까지 선 세그먼트로 위와 같이 연결할 수 있음을 알지 못하여 동경까지 동쪽 방향으로 이동하는 더 긴 선 세그먼트를 선택합니다. 평면 지도 프로젝션에서는 왼쪽 끝이 -180도 경선이고 오른쪽 끝이 180도 경선입니다.

Spatial Extender를 사용하여 올바른 결과를 얻으려면 다음 중 한 가지 조치를 수행해야 합니다.

- 선 세그먼트를 두 개의 선 세그먼트 즉, 180도 경선의 동쪽과 서쪽으로 나누십시오.
- 180도 경선이 끝에 위치하지 않도록 데이터를 다시 투영하십시오.

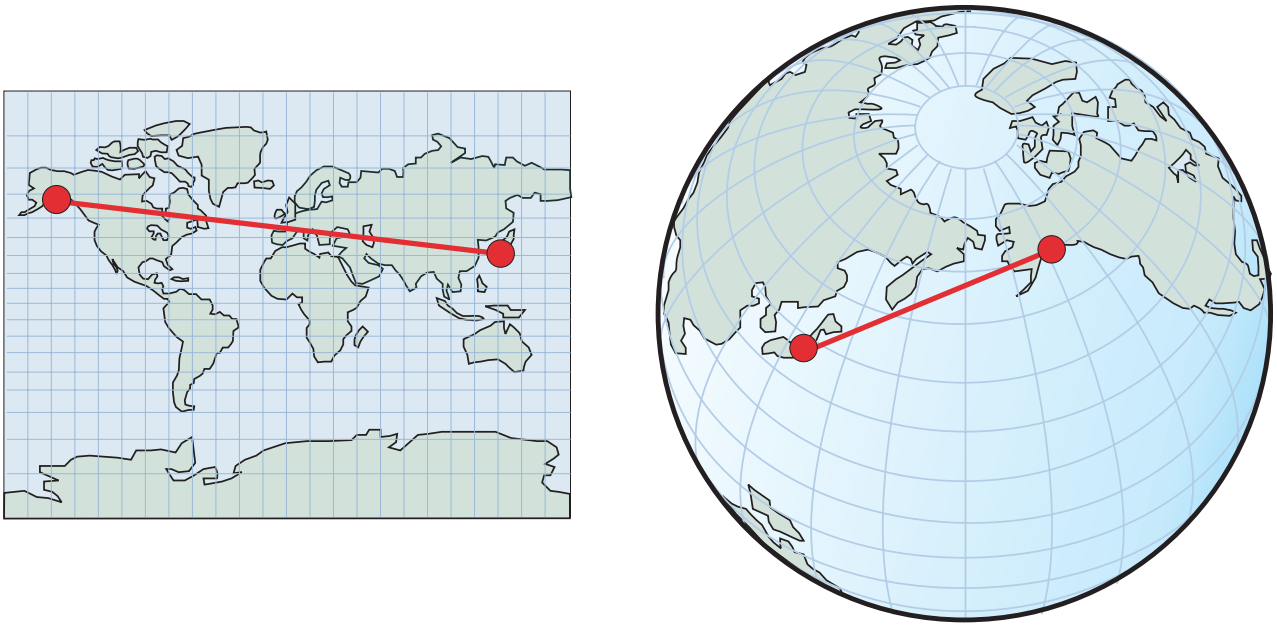


그림 36. 180도 경선을 지나는 선

180도 경선을 지나는 다각형

180도 경선을 지나는 다각형을 처리하기 위해 평면 지구 표현(Spatial Extender)에서는 다각형을 두 개 부분으로 나누어야 합니다.

다음 예에서는 180도 경선의 동쪽 부분에 대한 다각형과 경선 서쪽 부분에 대한 다각형을 표시합니다.

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

190 페이지의 그림 37에 표시된 것처럼 구형 지구 표현(Geodetic Data Management Feature)에서는 이러한 분할이 필요하지 않으므로 변경되지 않은 단일 다각형을 사용할 수 있습니다.

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

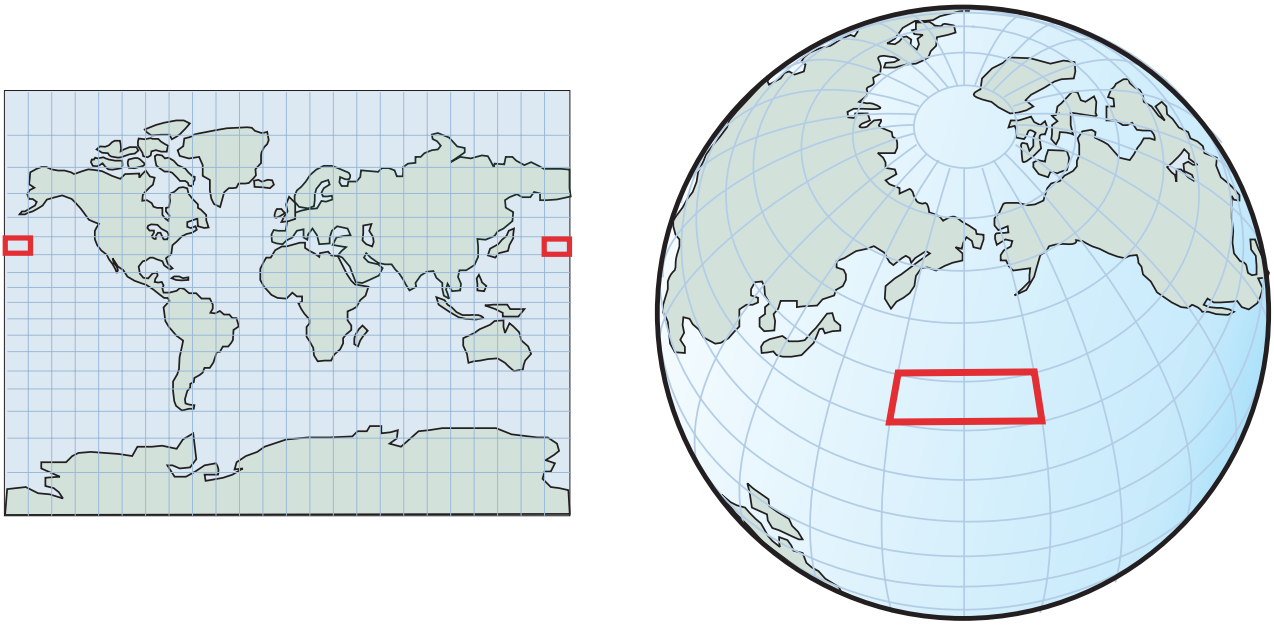


그림 37. 180도 경선을 지나는 다각형 - 두 개의 개별 다각형 작성

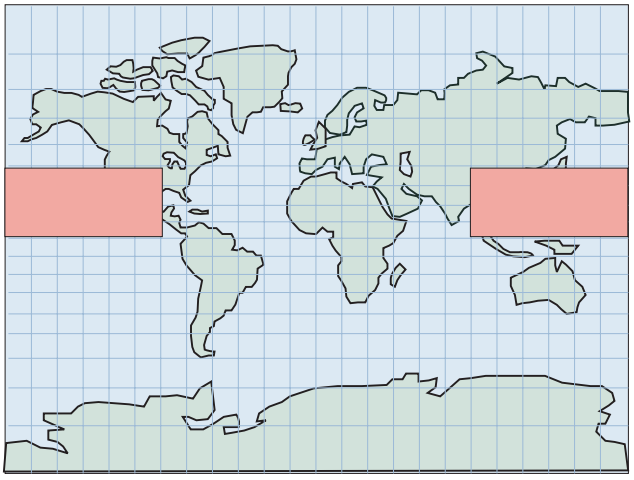
Spatial Extender 사용시 두 개의 개별 다각형을 작성하지 않은 경우, 191 페이지의 그림 38에서와 같이 다른 영역을 정의하도록 다각형의 정점을 실제로 다시 정렬합니다. 191 페이지의 그림 38의 위쪽에는 180도 경선을 지나는 다각형의 올바른 정점이 표시됩니다.

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

191 페이지의 그림 38의 맨 아래에는 정점을 다시 정렬하여 이제 180도 경선이 아닌 0도 경선을 지나는 다각형이 표시됩니다.

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

180도 경선을 지나는 다각형
Polygon ((90 0, -90 0, -90 40, 90 40))



Spatial Extender가 정점을 다시 정렬하여
결과 다각형이 다른 영역을 정의
Polygon ((-90 0, 90 0, 90 40, -90 40))

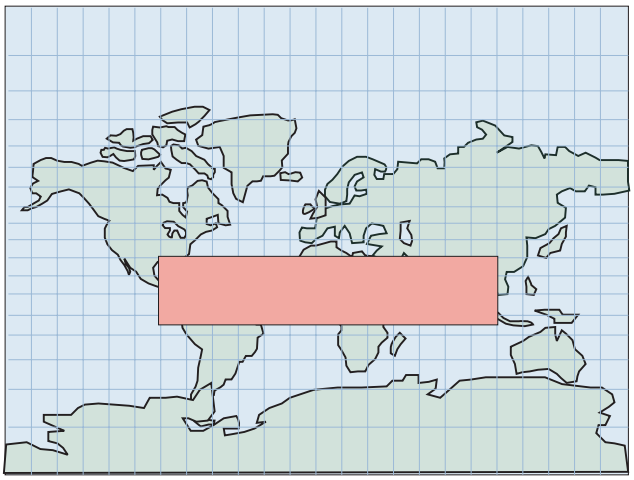


그림 38. 180도 경선을 지나는 다각형 - 정점 다시 정렬

정의된 영역은 192 페이지의 그림 39에 표시되는 것처럼 의도한 영역이 아닌 지구의 보완 영역이 됩니다. 위의 선 세그먼트 예와 유사하게 이러한 상황을 처리하는 다른 방법은 180도 경선이 끝에 위치하지 않도록 데이터를 다시 투영하는 것입니다.

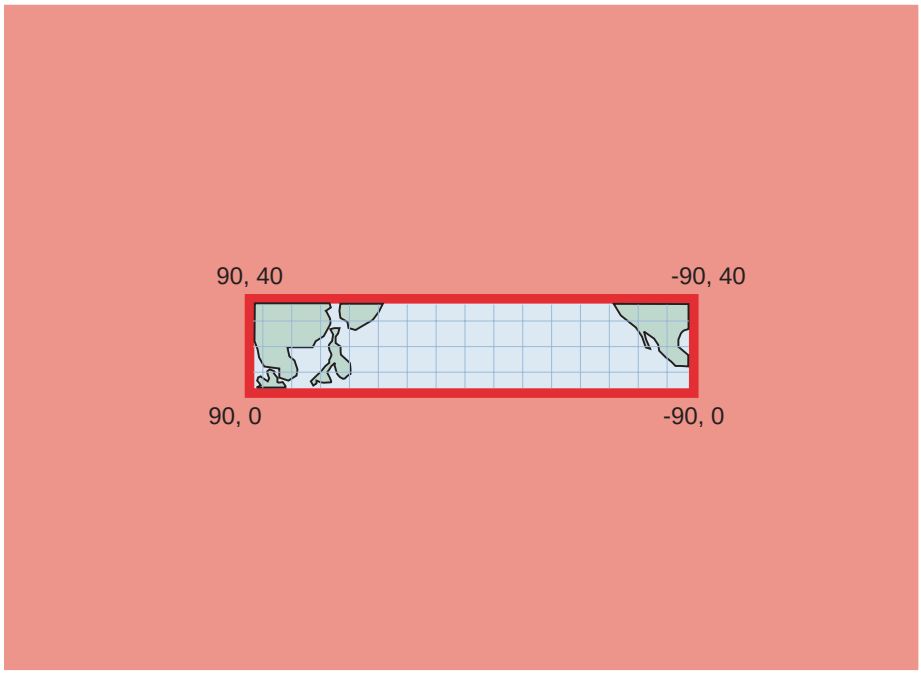


그림 39. 180도 경선을 지나는 다각형 - 보완 영역

극을 감싸는 다각형

Spatial Extender를 사용하여 평면 지도 프로젝션의 끝에서 작업을 수행하므로 지구 표면의 지도 왜곡으로 인해 다각형 내부에 극을 표현하는 끝과 정점을 추가해야 합니다.

193 페이지의 그림 40에서는 Spatial Extender 또는 Geodetic Data Management Feature를 사용하여 남극을 감싸는 다각형에 대한 작업을 수행할 수 있는 방법을 보여줍니다.

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

구형 지구 표현(Geodetic Data Management Feature)에서는 남극 주위의 다각형이 남위 60°상의 원으로 표시됩니다.

```
POLYGON((0 –60, –1 –60, –2 –60, ..., –179 –60, 180 –60, 179 –60, ..., 1 –60, 0 –60))
```

이 원을 표현하기 위한 보다 나은 방법은 남극 전체와 주변 지역이 지도에 표시될 수 있도록 데이터를 다시 투영하는 것입니다.

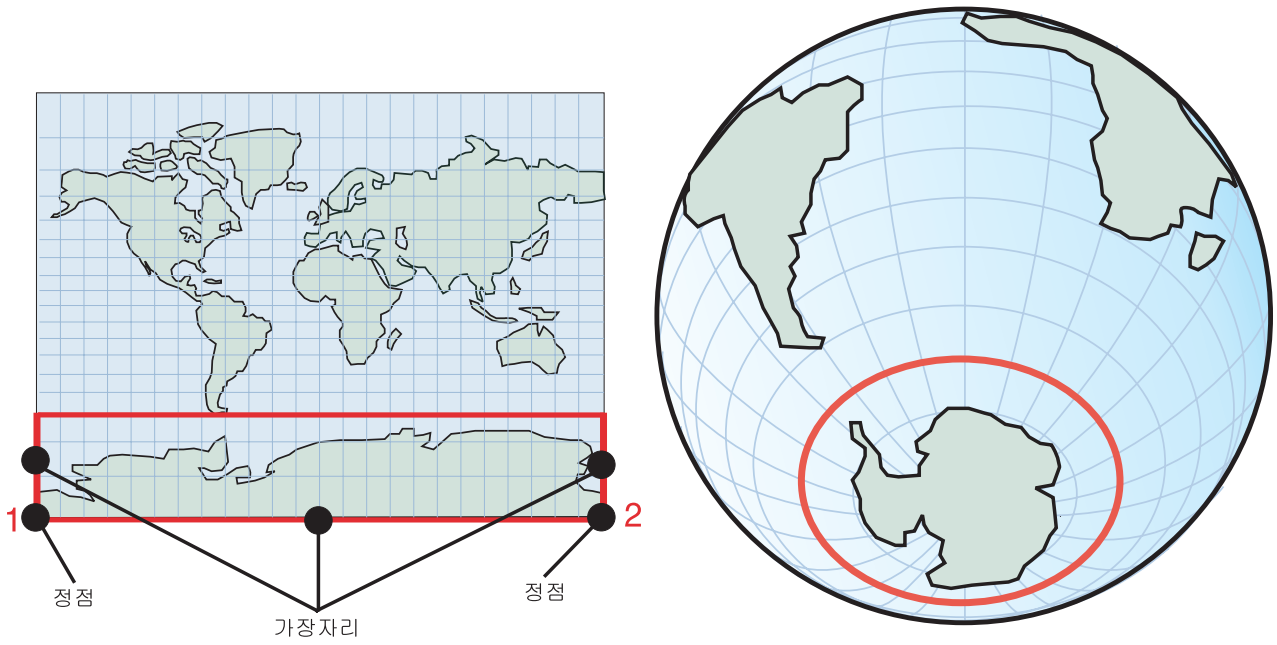


그림 40. 극을 감싸는 다각형

위의 예에서 올바른 투영 공간 참조 시스템을 선택하면 정확한 결과를 얻을 수 있습니다. 그러나 투영 방법을 사용하여 모든 문제를 동시에 해결할 수는 없습니다. 예를 들어, 가장자리에 180도 경선이 없는 투영의 경우 다른 위치에 가장자리를 넣어서 문제 영역을 바꿉니다.

반구, 적도 지역 및 지구 전체를 표현하는 다각형

지구를 둥글게 표현하면 지구 표면의 많은 영역에서 거리 및 면적 계산에 보다 좋은 결과를 얻을 수 있습니다.

다각형을 사용하여 지구 표면의 넓은 영역(예: 반구, 적도 지역 또는 지구 전체)을 표현해야 하는 경우, Spatial Extender 및 Geodetic Data Management Feature에서 이 경우를 다양한 방법으로 처리한다는 것을 명심하십시오. 이러한 경우 지구를 둥글게 표현하면 거리와 면적 계산 시 정확한 결과를 얻을 수 있지만 프로젝션 방법의 경우는 그렇지 않습니다.

예를 들어, 195 페이지의 그림 41에서는 서반구를 평면 지구 표현(Spatial Extender)과 구형 지구 표현(Geodetic Data Management Feature)으로 정의하는 다각형을 표시합니다.

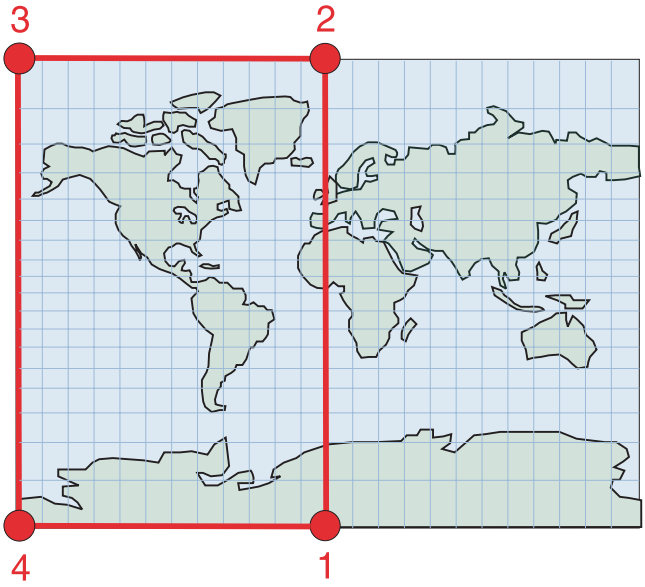
- 195 페이지의 그림 41 위쪽 부분에 있는 평면 지구 표현의 경우, 네 개의 좌표가 'POLYGON((0 -90, 0 90, -180 90, 180 -90, 0 -90))'과 같은 Well-Known Text 형식으로 서반구를 표현합니다.

- 구형 지구 표현에서 네 개 좌표는 'POLYGON((0 0, 0 90, 180 0, 0 -90, 0 0))' 과 같이 Well-Known Text 형식으로 서반구를 표현합니다. 이 네 개 좌표는 0도 경선과 반대쪽 선, 180도 경선을 따라 지구 주위의 링을 정의합니다.

반대 순서로 동일한 네 점을 지정하면 다음과 같이 동반구가 정의됩니다.

- 평면 지구 표현의 경우, 동반구는 'POLYGON((0 -90, 180 -90, 180 90, 0 90, 0 -90))'입니다.
- 구형 지구 표현의 경우, 동반구는 'POLYGON((0 -90, 180 0, 0 90, 0 0, 0 -90))' 입니다.

평면형 지구 표현 방법의 서반구
Polygon ((0 -90, 0 90, -180 90, 180 -90, 0 -90))



구형 지구 표현 방법의 서반구
Polygon ((0 0, 0 90, 180 0, 0 -90, 0 0))

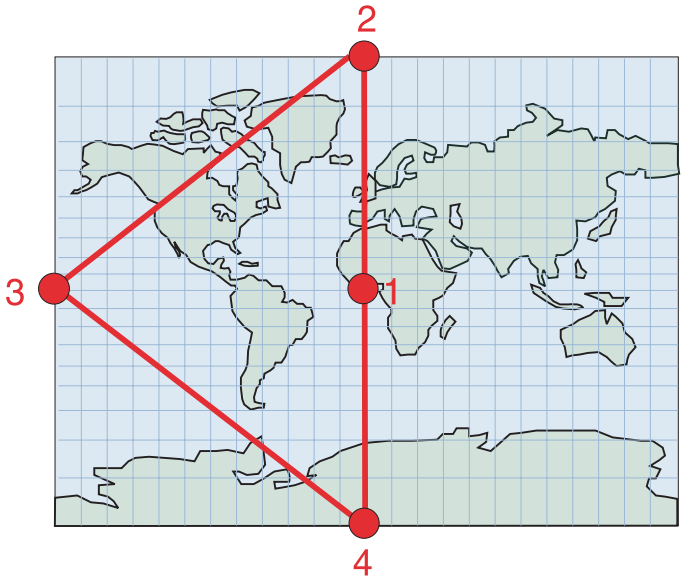


그림 41. 반구를 표현하는 다각형

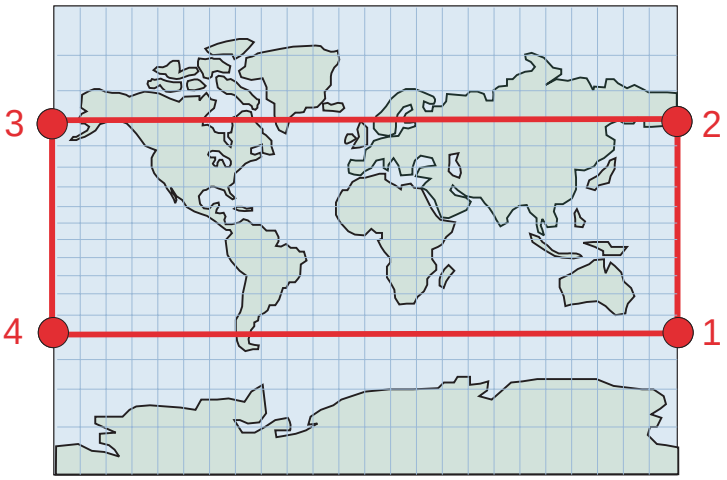
196 페이지의 그림 42에서는 평면 지구 표현(Spatial Extender)과 구형 지구 표현(Geodetic Data Management Feature)으로 적도 지역을 정의하는 다각형의 좌표를 표시합니다.

- 196 페이지의 그림 42 위쪽 부분에서는 'POLYGON((180 –60, 180 60, –180 60, –180 –60, 180 –60))'과 같이 Well-Known Text 형식의 좌표로 적도 지역의 평면 지구 표현을 표시합니다.

- 그림 42 맨 아래의 구형 지구 표현 방법에서는 두 개 링으로 구성되는 제외 영역을 정의하여 적도 지역을 표현합니다.

```
'MULTIPOLYGON(((0 60, -120 60, 120 60, 0 60)),
((0 -60, 120 -60, -120 -60, 0 -60)))'
```

  잘 알아볼 수 있도록 각 링에서 세 점만 표시됩니다. 실제로 링을 위선 60 또는 -60과 가깝게 이동하려면 보다 많은 중간 점을 추가해야 합니다. 첫 번째 링((0 60, -120 60, 120 60, 0 60))은 60도 위선 남쪽의 영역을 정의하는 순서대로 정점을 지정합니다. 두 번째 링((0 -60, 120 -60, -120 -60, 0 -60))에서는 -60 위선의 북쪽 영역을 지정합니다.



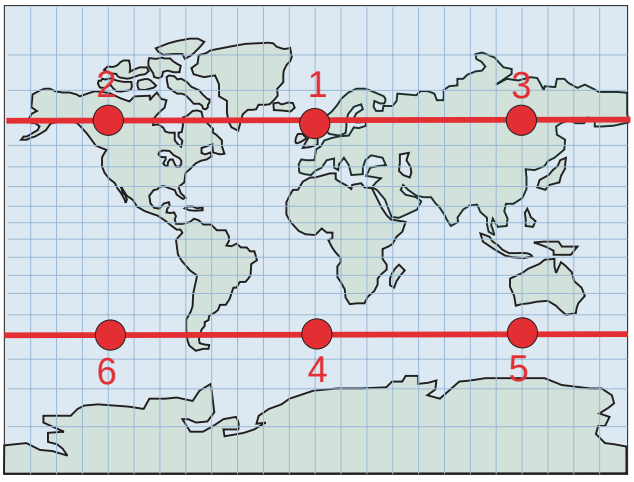
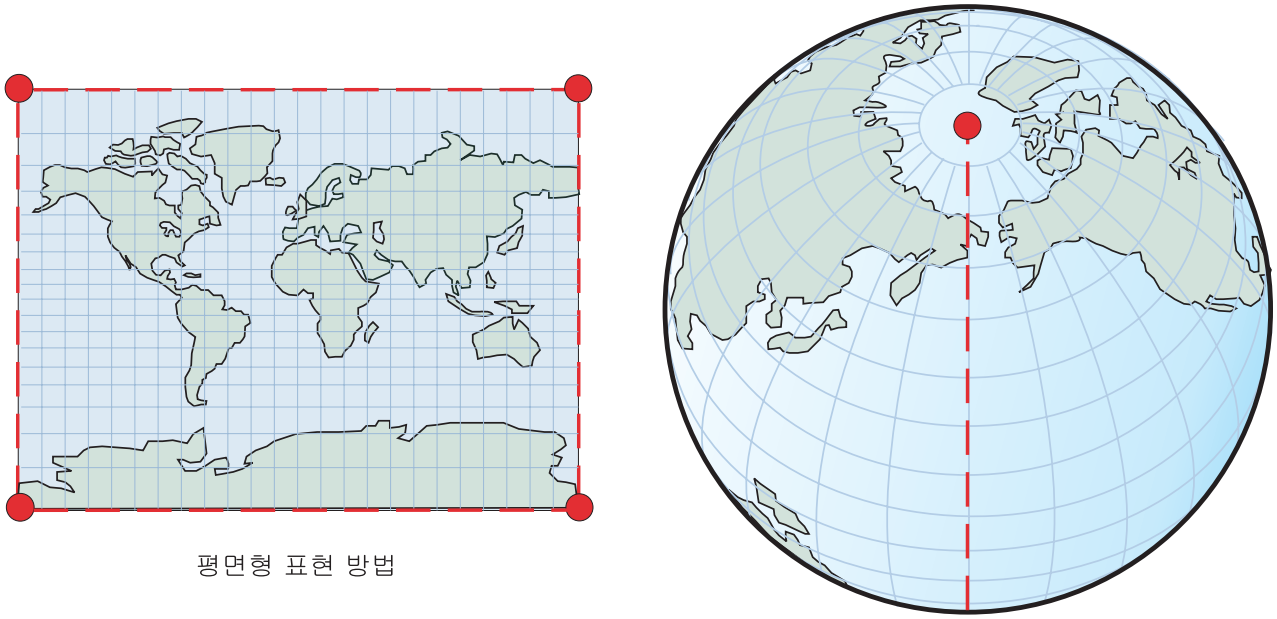


그림 42. 적도 지역을 표현하는 다각형

그림 43에서는 전체 지구를 평면 지구 표현(Spatial Extender) 및 구형 지구 표현(Geodetic Data Management Feature)으로 정의하는 다각형을 표시합니다. 두 표현 모두는 지구 전체를 'POLYGON((-180 -90, 180 -90, 180 90, -180 90, -180 -90))'과 같은 Well-Known Text 형식의 동일한 다각형으로 표현합니다.



평면형 표현 방법

타원체 표현 방법:
경계가 없는 다각형으로
특별 표기법이 필요

그림 43. 전체 지구를 표현하는 다각형

DB2 Geodetic Data Management Feature에서 지원하는 공간 함수

DB2 Spatial Extender는 ESRI에서 제공하는 함수 라이브러리에 빌드되며 DB2 Geodetic Data Management Feature는 Hipparchus 함수 라이브러리에 빌드됩니다. ESRI 라이브러리와 Hipparchus 라이브러리의 기능 차이로 인해 일부 함수 작동에 약간의 차이가 있습니다. 다음 테이블은 Geodetic Data Management Feature에서 지원하는 Spatial Extender 함수 및 작동 차이점에 대해 설명합니다. 공간 함수 사용법 및 구문에 대한 정보는 해당 공간 함수 주제를 참조하십시오.

표 25. Geodetic Data Management Feature를 위한 함수 지원

함수	**DB2 Geodetic Data Management Feature의 지원 여부**	**DB2 Geodetic Data Management Feature의 동작 차이점**
EnvelopesIntersect	예	없음
MBR Aggregate	아니오	적용할 수 없음

표 25. *Geodetic Data Management Feature*를 위한 함수 지원 (계속)

함수	**DB2 Geodetic Data Management Feature**의 지원 여부	**DB2 Geodetic Data Management Feature**의 동작 차이점
ST_AppendPoint	아니오	적용할 수 없음
ST_Area	예	디폴트 측정 단위는 미터입니다.
ST_AsBinary	예	없음
ST_AsGML	예	없음
ST_AsShape	예	없음
ST_AsText	예	없음
ST_Boundary	아니오	적용할 수 없음
ST_Buffer	예	점 및 다중점에서만 지원됩니다. 음수 거리 값이 허용됩니다. 디폴트 측정 단위는 미터입니다.
ST_Centroid	아니오	적용할 수 없음
ST_ChangePoint	아니오	적용할 수 없음
ST_Contains	예	두 Geometry 모두 동일한 측지 공간 참조 시스템(SRS) 내에 있어야 합니다.
ST_ConvexHull	아니오	적용할 수 없음
ST_CoordDim	예	없음
ST_Crosses	아니오	적용할 수 없음
ST_Difference	예	선스트링 및 다중 선스트링에서는 지원하지 않습니다. 두 Geometry는 동일한 측지 SRS 내에 있어야 합니다. 리턴된 Geometry의 차원은 입력 Geometry의 차원과 동일합니다.
ST_Dimension	예	없음
ST_Disjoint	예	없음
ST_Distance	예	*측지 거리*를 리턴합니다. 두 Geometry는 동일한 측지 SRS 내에 있어야 합니다. 디폴트 측정 단위는 미터입니다.
ST_Edge_GC_USA	예	없음
ST_Endpoint	예	없음
ST_Envelope	예	인벨로프는 Geometry의 MBC(Minimum Bounding Circle)를 감싸는 다각형입니다.
ST_EnvIntersects	예	없음
ST_EqualCoordsys	예	없음
ST_Equals	아니오	적용할 수 없음
ST_EqualSRS	예	없음
ST_ExteriorRing	예	없음
ST_FindMeasure 또는 ST_LocateAlong	아니오	적용할 수 없음
ST_Generalize	예	임계값 단위는 미터입니다.
ST_GeomCollection	아니오	적용할 수 없음
ST_GeomCollFromTxt	아니오	적용할 수 없음
ST_GeomCollFromWKB	아니오	적용할 수 없음
ST_Geometry	예	없음
ST_GeometryN	예	없음

표 25. Geodetic Data Management Feature를 위한 함수 지원 (계속)

함수	DB2 Geodetic Data Management Feature의 지원 여부	DB2 Geodetic Data Management Feature의 동작 차이점
ST_GeometryType	예	없음
ST_GeomFromText	예	없음
ST_GeomFromWKB	예	없음
ST_GetIndexParms	아니오	적용할 수 없음
ST_InteriorRingN	예	없음
ST_Intersection	예	리턴된 Geometry의 차원은 차원이 낮은 입력의 차원입니다 (두 선스트링 교차 차원이 0인 경우 제외).
ST_Intersects	예	두 Geometry는 동일한 측지 SRS 내에 있어야 합니다.
ST_Is3d	예	없음
ST_IsClosed	예	없음
ST_IsEmpty	예	없음
ST_IsMeasured	예	없음
ST_IsRing	아니오	적용할 수 없음
ST_IsSimple	아니오	적용할 수 없음
ST_IsValid	예	없음
ST_Length	예	디폴트 측정 단위는 미터입니다.
ST_LineFromText	예	없음
ST_LineFromWKB	예	없음
ST_LineString	예	없음
ST_LineStringN	예	없음
ST_M	예	없음
ST_MaxM	예	없음
ST_MaxX	예	MBC(Minimum Bounding Circle)의 최대값 X를 리턴합니다. **주:** MBC가 날짜 변경선을 지나는 경우, ST_MaxX 값은 ST_MinX 미만입니다. MBC에 북극이나 남극이 포함되는 경우, ST_MinX는 -180이고 ST_MaxX은 180입니다.
ST_MaxY	예	MBC의 최대 Y 값을 리턴합니다. **주:** MBC에 북극이 포함되는 경우, ST_MaxY 값은 90입니다.
ST_MaxZ	예	없음
ST_MBR	예	MBR은 Geometry의 MBC를 감싸는 Geometry입니다.
ST_MBRIntersects	예	없음
ST_MeasureBetween 또는 ST_LocateBetween	아니오	적용할 수 없음
ST_MidPoint	예	없음
ST_MinM	예	없음

표 25. Geodetic Data Management Feature를 위한 함수 지원 (계속)

함수	DB2 Geodetic Data Management Feature의 지원 여부	DB2 Geodetic Data Management Feature의 동작 차이점
ST_MinX	예	MBC의 최소 X 값을 리턴합니다. **주:** MBC가 날짜 변경선을 지나는 경우, ST_MinX 값은 ST_MaxX 값보다 큽니다. MBC에 북극이나 남극이 포함되는 경우, ST_MinX는 -180이고 ST_MaxX은 180입니다.
ST_MinY	예	MBC의 최소 Y 값을 리턴합니다. **주:** MBC에 남극이 포함되는 경우 ST_MinY 값은 -90입니다.
ST_MinZ	예	없음
ST_MLineFromText	예	없음
ST_MLineFromWKB	예	없음
ST_MPointFromText	예	없음
ST_MPointFromWKB	예	없음
ST_MPolyFromText	예	없음
ST_MPolyFromWKB	예	없음
ST_MultiLineString	예	없음
ST_MultiPoint	예	없음
ST_MultiPolygon	예	없음
ST_NumGeometries	예	없음
ST_NumInteriorRing	예	없음
ST_NumLineStrings	예	없음
ST_NumPoints	예	없음
ST_NumPolygons	예	없음
ST_Overlaps	아니오	적용할 수 없음
ST_Perimeter	예	디폴트 측정 단위는 미터입니다.
ST_PerpPoints	아니오	적용할 수 없음
ST_Point	예	없음
ST_PointFromText	예	없음
ST_PointFromWKB	예	없음
ST_PointN	예	없음
ST_PolyFromText	예	없음
ST_PolyFromWKB	예	없음
ST_PointOnSurface	예	없음
ST_Polygon	예	없음
ST_PolygonN	예	없음
ST_Relate	아니오	적용할 수 없음
ST_RemovePoint	아니오	적용할 수 없음
ST_SrsId 또는 ST_SRID	예	없음
ST_SrsName	예	없음
ST_StartPoint	예	없음

표 25. Geodetic Data Management Feature를 위한 함수 지원 (계속)

함수	DB2 Geodetic Data Management Feature의 지원 여부	DB2 Geodetic Data Management Feature의 동작 차이점
ST_SymDifference	예	선스트링 및 다중 선스트링에서는 지원하지 않습니다. 리턴된 Geometry의 차원은 입력 Geometry의 차원과 동일합니다. 두 Geometry는 동일한 측지 SRS 내에 있어야 합니다.
ST_ToGeomColl	아니오	적용할 수 없음
ST_ToLineString	예	없음
ST_ToMultiLine	예	없음
ST_ToMultiPoint	예	없음
ST_ToPoint	예	없음
ST_ToPolygon	예	없음
ST_Touches	아니오	적용할 수 없음
ST_Transform	예	없음. **주:** 좌표 변환은 점 단위로 수행됩니다. 측지 좌표 시스템과 비투영 평면 좌표 시스템 사이를 변환하는 경우, 180도 자오선을 지나는 모든 다각형 및 선스트링을 주의 깊게 점검하거나 한 극 또는 두 극을 모두 포함하십시오. Spatial Extender 및 Geodetic Data Management Feature는 이러한 경우를 다르게 처리하므로 평면 지구 좌표 시스템에서 유효한 Geometry가 구형 지구 시스템에서 유효하지 않을 수 있고 그 반대의 경우도 적용됩니다. 자세한 정보는 188 페이지의 『평면형 지구 표현과 구형 지구 표현 방법의 차이점』을 참조하십시오.
ST_Union	예	두 Geometry는 동일한 측지 SRS 내에 있어야 합니다.
ST_Within	예	두 Geometry는 동일한 측지 SRS 내에 있어야 합니다.
ST_WKBToSQL	예	없음
ST_WKTToSQL	예	없음
ST_X	예	없음
ST_Y	예	없음
ST_Z	예	없음
Union Aggregate	아니오	적용할 수 없음

DB2 Geodetic Data Management Feature 스토어드 프로시저 및 카탈로그 뷰

DB2 Geodetic Data Management Feature에서는 DB2 Spatial Extender와 동일한 카탈로그 뷰를 지원하며 공간 스토어드 프로시저의 서브세트를 지원합니다.

Geodetic Data Management Feature는 다음 스토어드 프로시저를 지원하지 않습니다.

- ST_disable_autogeocoding
- ST_enable_autogeocoding
- ST_register_geocoder

- ST_remove_geocoding_setup
- ST_run_geocoding
- ST_setup_geocoding
- ST_unregister_geocoder

Geodetic Data Management Feature에서는 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 표시되는 318개의 사전 정의된 측지 공간 참조 시스템을 제공합니다. 전체 목록을 참조하십시오.

DB2 Geodetic Data Management Feature에서 지원하는 기준면

설명한 바와 같이 *기준면*은 지구 중심에 상대적인 타원체의 위치를 정의하는 값 세트입니다. 공간 참조 시스템(SRS)은 기준면과 타원체를 연관시키는 매개변수 세트로서 공간 참조 시스템 ID(SRID)로 식별합니다. 203 페이지의 표 27에는 DB2 Geodetic Data Management Feature에서 제공하는 사전 정의된 기준면이 나열되어 있습니다. 사전 정의된 모든 측지 SRS의 오프셋 값 및 스케일 인수는 동일하며 해당 값은 다음 테이블을 참조하십시오.

표 26. 사전 정의된 측지 SRS의 오프셋 및 스케일 값

SRS 매개변수	**값**
xOffset	–180
yOffset	–90
zOffset	–50000
mOffset	–1000
xScale	5965232
yScale	5965232
zScale	1000
mScale	1000

yScale은 항상 xScale과 동일합니다.

203 페이지의 표 27에 나열되어 있는 모든 데이텀을 공간 참조 시스템에 선택할 수 있습니다. 해당 데이터에 가장 적합한 기준면을 선택하는 것이 바람직합니다. 예를 들어, 가장 일반적으로 사용되는 WGS 1984(World Geodetic System 1984)는 지구 중심을 원점으로 사용하며 지구 전체를 맵핑합니다. 이를 지구 중심 기준면이라고 합니다. 반대로, North American 1927 datum과 같은 로컬 기준면은 지면상의 한 점부터 북미 지역을 맵핑합니다. 로컬 기준면은 모델링하려는 지역에 대해서는 정확하지만, 지구 전체의 위치를 표현하려면 지구 중심 측지 기준면을 사용해야 합니다.

표 27. 기준면 및 타원체와 연관되는 SRID

SRID	기준면 이름	참조 타원체
2000000000	WGS 1984	WGS 1984
2000000001	Abidjan 1987	Clarke 1880 (RGS)
2000000002	Accra	War Office
2000000003	Adindan	Clarke 1880 (RGS)
2000000004	Afgooye	Krasovsky 1940
2000000005	Agadez	Clarke 1880 (IGN)
2000000006	Australian Geodetic Datum 1966	Australian
2000000007	Australian Geodetic Datum 1984	Australian
2000000008	Ain el Abd 1970	International 1924
2000000009	Airy 1830	Airy 1830
2000000010	Airy Modified	Airy Modified
2000000011	Alaskan Islands	Clarke 1866
2000000012	Amersfoort	Bessel 1841
2000000013	Anguilla 1957	Clarke 1880 (RGS)
2000000014	Anna 1 Astro 1965	Australian
2000000015	Antigua Astro 1943	Clarke 1880 (RGS)
2000000016	Aratu	International 1924
2000000017	Arc 1950	Clarke 1880 (Arc)
2000000018	Arc 1960	Clarke 1880 (RGS)
2000000019	Ascension Island 1958	International 1924
2000000020	Assumed Geographic(PRJ가 없는 모양 파일의 경우 NAD27)	Clarke 1866
2000000021	Astronomical Station 1952	International 1924
2000000022	ATF (Paris)	Plessis 1817
2000000023	Average Terrestrial System 1977	ATS 1977
2000000024	Australian National	Australian
2000000025	Ayabelle Lighthouse	Clarke 1880 (RGS)
2000000026	Bab South Astro(Bablethuap Is, Republic of Palau)	Clarke 1866
2000000027	Barbados 1938	Clarke 1880 (RGS)
2000000028	Batavia	Bessel 1841
2000000029	Batavia (Jakarta)	Bessel 1841
2000000030	Astro Beacon E 1945	International 1924
2000000031	Beduaram	Clarke 1880 (IGN)
2000000032	Beijing 1954	Krasovsky 1940
2000000033	Reseau National Belge 1950	International 1924
2000000034	Belge 1950(브뤼셀)	International 1924
2000000035	Reseau National Belge 1972	International 1924
2000000036	Bellevue (IGN)	International 1924
2000000037	Bermuda 1957	Clarke 1866
2000000038	Bern 1898	Bessel 1841
2000000039	Bern 1898(베른)	Bessel 1841
2000000040	Bern 1938	Bessel 1841
2000000041	Bessel 1841	Bessel 1841
2000000042	Bessel 수정	Bessel 수정
2000000043	Bessel Namibia	Bessel Namibia
2000000044	Bissau	International 1924
2000000045	Bogota	International 1924
2000000046	Bogota (Bogota)	International 1924
2000000047	Bukit Rimpah	Bessel 1841
2000000048	Camacupa	Clarke 1880 (RGS)
2000000049	Campo Inchauspe	International 1924
2000000050	Camp Area Astro	International 1924

표 27. 기준면 및 타원체와 연관되는 SRID (계속)

SRID	기준면 이름	참조 타원체
2000000051	Canton Astro 1966	International 1924
2000000052	Cape	Clarke 1880 (Arc)
2000000053	Cape Canaveral	Clarke 1866
2000000054	Carthage	Clarke 1880 (IGN)
2000000055	Carthage(도)	Clarke 1880 (IGN)
2000000056	Carthage (Paris)	Clarke 1880 (IGN)
2000000057	CH 1903	Bessel 1841
2000000058	CH 1903+	Bessel 1841
2000000059	Chatham Island Astro 1971	International 1924
2000000060	Chos Malal 1914	International 1924
2000000061	Swiss Terrestrial Ref. Frame 1995	GRS 1980
2000000062	Chua	International 1924
2000000063	Clarke 1858	Clarke 1858
2000000064	Clarke 1866	Clarke 1866
2000000065	Clarke 1866 (Michigan)	Clarke 1866 (Michigan)
2000000066	Clarke 1880	Clarke 1880
2000000067	Clarke 1880 (Arc)	Clarke 1880 (Arc)
2000000068	Clarke 1880 (Benoit)	Clarke 1880 (Benoit)
2000000069	Clarke 1880 (IGN)	Clarke 1880 (IGN)
2000000070	Clarke 1880 (RGS)	Clarke 1880 (RGS)
2000000071	Clarke 1880 (SGA)	Clarke 1880 (SGA)
2000000072	Conakry 1905	Clarke 1880 (IGN)
2000000073	Corrego Alegre	International 1924
2000000074	Cote d'Ivoire	Clarke 1880 (IGN)
2000000075	Dabola 1981	Clarke 1880 (RGS)
2000000076	Datum 73	International 1924
2000000077	Dealul Piscului 1933 (Romania)	International 1924
2000000078	Dealul Piscului 1970 (Romania)	Krasovsky 1940
2000000079	Deception Island	Clarke 1880 (RGS)
2000000080	Deir ez Zor	Clarke 1880 (IGN)
2000000081	Deutsche Hauptdreiecksnetz	Bessel 1841
2000000082	Dominica 1945	Clarke 1880 (RGS)
2000000083	DOS 1968	International 1924
2000000084	Astro DOS 71/4	International 1924
2000000085	Douala	Clarke 1880 (IGN)
2000000086	Easter Island 1967	International 1924
2000000087	European Datum 1950	International 1924
2000000088	European Datum 1950 (ED77)	International 1924
2000000089	European Datum 1987	International 1924
2000000090	Egypt 1907	Helmert 1906
2000000091	Estonia 1937	Bessel 1841
2000000092	Estonia 1992	GRS 1980
2000000093	European Terrestrial Ref. Frame 1989	WGS 1984
2000000094	European 1979	International 1924
2000000095	European Libyan Datum 1979	International 1924
2000000096	Everest 1830	Everest 1830
2000000097	Everest (Bangladesh)	Everest Adjustment 1937
2000000098	Everest (Definition 1962)	Everest (Definition 1962)
2000000099	Everest (Definition 1967)	Everest (Definition 1967)
2000000100	Everest (Definition 1975)	Everest (Definition 1975)
2000000101	Everest (India 및 Nepal)	Everest (Definition 1962)
2000000102	Everest 1830 Modified	Everest 1830 Modified
2000000103	Everest Modified 1969	Everest Modified 1969

표 27. 기준면 및 타원체와 연관되는 SRID (계속)

SRID	기준면 이름	참조 타원체
2000000104	Fahud	Clarke 1880 (RGS)
2000000105	Final Datum 1958	Clarke 1880 (RGS)
2000000106	Fischer 1960	Fischer 1960
2000000107	Fischer 1968	Fischer 1968
2000000108	Fischer Modified	Fischer Modified
2000000109	Fort Thomas 1955	Clarke 1880 (RGS)
2000000110	Gandajika 1970	International 1924
2000000111	Gan 1970	International 1924
2000000112	Garoua	Clarke 1880 (IGN)
2000000113	Geocentric Datum of Australia 1994	GRS 1980
2000000114	GEM 10C Gravity Potential Model	GEM 10C
2000000115	Greek Geodetic Ref. System 1987	GRS 1980
2000000116	Graciosa Base SW 1948	International 1924
2000000117	Greek	Bessel 1841
2000000118	Greek (Athens)	Bessel 1841
2000000119	Grenada 1953	Clarke 1880 (RGS)
2000000120	GRS 1967	GRS 1967
2000000121	GRS 1980	GRS 1980
2000000122	Guam 1963	Clarke 1866
2000000123	Gunung Segara	Bessel 1841
2000000124	GUX 1 Astro	International 1924
2000000125	Guyane Francaise	International 1924
2000000126	Hanoi 1972	Krasovsky 1940
2000000127	Hartebeesthoek 1994	WGS 1984
2000000128	Helmert 1906	Helmert 1906
2000000129	Herat North	International 1924
2000000130	Hermannskogel	Bessel 1841
2000000131	Hito XVIII 1963	International 1924
2000000132	Hjorsey 1955	International 1924
2000000133	Hong Kong 1963	International 1924
2000000134	Hong Kong 1980	International 1924
2000000135	Hough 1960	Hough 1960
2000000136	Hungarian Datum 1972	GRS 1967
2000000137	Hu Tzu Shan	International 1924
2000000138	Indian 1954	Everest Adjustment 1937
2000000139	Indian 1960	Everest Adjustment 1937
2000000140	Indian 1975	Everest Adjustment 1937
2000000141	Indonesian National	Indonesian National
2000000142	Indonesian Datum 1974	Indonesian
2000000143	International 1927	International 1924
2000000144	International 1967	International 1967
2000000145	IRENET95	GRS 1980
2000000146	Israel	GRS 1980
2000000147	ISTS 061 Astro 1968	International 1924
2000000148	ISTS 073 Astro 1969	International 1924
2000000149	Jamaica 1875	Clarke 1880
2000000150	Jamaica 1969	Clarke 1866
2000000151	Japan Geodetic Datum 2000	GRS 1980
2000000152	Johnston Island 1961	International 1924
2000000153	Kalianpur 1880	Everest 1830
2000000154	Kalianpur 1937	Everest Adjustment 1937
2000000155	Kalianpur 1962	Everest (Definition 1962)
2000000156	Kalianpur 1975	Everest (Definition 1975)

표 27. 기준면 및 타원체와 연관되는 SRID (계속)

SRID	기준면 이름	참조 타원체
2000000157	Kandawala	Everest Adjustment 1937
2000000158	Kerguelen Island 1949	International 1924
2000000159	Kertau	Everest 1830 Modified
2000000160	Kartastokoordinaattijarjestelma	International 1924
2000000161	Kuwait Oil Company	Clarke 1880 (RGS)
2000000162	Korean Datum 1985	Bessel 1841
2000000163	Korean Datum 1995	WGS 1984
2000000164	Krasovsky 1940	Krasovsky 1940
2000000165	Kuwait Utility	GRS 1980
2000000166	Kusaie Astro 1951	International 1924
2000000167	Lake	International 1924
2000000168	La Canoa	International 1924
2000000169	L.C. 5 Astro 1961	Clarke 1866
2000000170	Leigon	Clarke 1880 (RGS)
2000000171	Liberia 1964	Clarke 1880 (RGS)
2000000172	Datum Lisboa Bessel	Bessel 1841
2000000173	Datum Lisboa Hayford	International 1924
2000000174	Lisbon	International 1924
2000000175	Lisbon (Lisbon)	International 1924
2000000176	LKS 1994	GRS 1980
2000000177	Locodjo 1965	Clarke 1880 (RGS)
2000000178	Loma Quintana	International 1924
2000000179	Lome	Clarke 1880 (IGN)
2000000180	Luzon 1911	Clarke 1866
2000000181	Madrid 1870 (Madrid Prime Merid.)	Struve 1860
2000000182	Madzansua	Clarke 1866
2000000183	Mahe 1971	Clarke 1880 (RGS)
2000000184	Majuro (Republic of Marshall Is.)	Clarke 1866
2000000185	Makassar	Bessel 1841
2000000186	Makassar (Jakarta)	Bessel 1841
2000000187	Malongo 1987	International 1924
2000000188	Manoca	Clarke 1880 (RGS)
2000000189	Massawa	Bessel 1841
2000000190	Merchich	Clarke 1880 (IGN)
2000000191	Merchich(도)	Clarke 1880 (IGN)
2000000192	Militar-Geographische Institut	Bessel 1841
2000000193	MGI (Ferro)	Bessel 1841
2000000194	Mhast	International 1924
2000000195	Midway Astro 1961	International 1924
2000000196	Minna	Clarke 1880 (RGS)
2000000197	Monte Mario	International 1924
2000000198	Monte Mario (Rome)	International 1924
2000000199	Montserrat Astro 1958	Clarke 1880 (RGS)
2000000200	Mount Dillon	Clarke 1858
2000000201	Moznet	WGS 1984
2000000202	M'poraloko	Clarke 1880 (IGN)
2000000203	North American Datum 1927	Clarke 1866
2000000204	NAD 1927 CGQ77	Clarke 1866
2000000205	NAD 1927 (1976)	Clarke 1866
2000000206	North American Datum 1983	GRS 1980
2000000207	NAD 1983 (Canadian Spatial Ref. System)	GRS 1980
2000000208	North American Datum 1983 (HARN)	GRS 1980
2000000209	NAD Michigan	Clarke 1866 (Michigan)

표 27. 기준면 및 타원체와 연관되는 SRID (계속)

SRID	기준면 이름	참조 타원체
2000000210	Nahrwan 1967	Clarke 1880 (RGS)
2000000211	Naparima 1955	International 1924
2000000212	Naparima 1972	International 1924
2000000213	Nord de Guerre (Paris)	Plessis 1817
2000000214	National Geodetic Network (Kuwait)	WGS 1984
2000000215	NGO 1948	Bessel 수정
2000000216	NGO 1948 (Oslo)	Bessel 수정
2000000217	Nord Sahara 1959	Clarke 1880 (RGS)
2000000218	NSWC 9Z-2	NWL 9D
2000000219	Nouvelle Triangulation Francaise(도)	Clarke 1880 (IGN)
2000000220	NTF (Paris) (grads)	Clarke 1880 (IGN)
2000000221	NWL 9D Transit Precise Ephemeris	NWL 9D
2000000222	New Zealand Geodetic Datum 1949	International 1924
2000000223	New Zealand Geodetic Datum 2000	GRS 1980
2000000224	Observatario	Clarke 1866
2000000225	Observ. Meteorologico 1939	International 1924
2000000226	Old Hawaiian	Clarke 1866
2000000227	Oman	Clarke 1880 (RGS)
2000000228	OSGB 1936	Airy 1830
2000000229	OSGB 1970 (SN)	Airy 1830
2000000230	OSU 1986 Geoidal Model	OSU 86F
2000000231	OSU 1991 Geoidal Model	OSU 91A
2000000232	OS (SN) 1980	Airy 1830
2000000233	Padang 1884	Bessel 1841
2000000234	Padang 1884 (Jakarta)	Bessel 1841
2000000235	Palestine 1923	Clarke 1880 (Benoit)
2000000236	Pampa del Castillo	International 1924
2000000237	PDO Survey Datum 1993	Clarke 1880 (RGS)
2000000238	Pico de Las Nieves	International 1924
2000000239	Pitcairn Astro 1967	International 1924
2000000240	Plessis 1817	Plessis 1817
2000000241	Pohnpei (Fed. States of Micronesia)	Clarke 1866
2000000242	Point 58	Clarke 1880 (RGS)
2000000243	Pointe Noire	Clarke 1880 (IGN)
2000000244	Porto Santo 1936	International 1924
2000000245	POSGAR	GRS 1980
2000000246	Provisional South Amer. Datum 1956	International 1924
2000000247	Puerto Rico	Clarke 1866
2000000248	Pulkovo 1942	Krasovsky 1940
2000000249	Pulkovo 1995	Krasovsky 1940
2000000250	Qatar 1974	International 1924
2000000251	Qatar 1948	Helmert 1906
2000000252	Qornoq	International 1924
2000000253	Rassadiran	International 1924
2000000254	REGVEN	GRS 1980
2000000255	Reunion	International 1924
2000000256	Reseau Geodesique Francais 1993	GRS 1980
2000000257	RT38	Bessel 1841
2000000258	RT38 (Stockholm)	Bessel 1841
2000000259	RT 1990	Bessel 1841
2000000260	S-42 Hungary	Krasovsky 1940
2000000261	South American Datum 1969	GRS 1967 Truncated
2000000262	Samboja	Bessel 1841

표 27. 기준면 및 타원체와 연관되는 SRID (계속)

SRID	기준면 이름	참조 타원체
2000000263	American Samoa 1962	Clarke 1866
2000000264	Santo DOS 1965	International 1924
2000000265	Sao Braz	International 1924
2000000266	Sapper Hill 1943	International 1924
2000000267	Schwarzeck	Bessel Namibia
2000000268	Segora	Bessel 1841
2000000269	Selvagem Grande 1938	International 1924
2000000270	Serindung	Bessel 1841
2000000271	Sierra Leone 1924	War Office
2000000272	Sierra Leone 1960	Clarke 1880 (RGS)
2000000273	Sierra Leone 1968	Clarke 1880 (RGS)
2000000274	SIRGAS	GRS 1980
2000000275	South Yemen	Krasovsky 1940
2000000276	Authalic sphere	Sphere
2000000277	Authalic sphere (ARC/INFO)	Sphere ARC INFO
2000000278	Struve 1860	Struve 1860
2000000279	St. George Island (Alaska)	Clarke 1866
2000000280	St. Kitts 1955	Clarke 1880 (RGS)
2000000281	St. Lawrence Island (Alaska)	Clarke 1866
2000000282	St. Lucia 1955	Clarke 1880 (RGS)
2000000283	St. Paul Island (Alaska)	Clarke 1866
2000000284	St. Vincent 1945	Clarke 1880 (RGS)
2000000285	Sudan	Clarke 1880 (IGN)
2000000286	South Asia Singapore	Fischer Modified
2000000287	S-JTSK	Bessel 1841
2000000288	S-JTSK (Ferro)	Bessel 1841
2000000289	Tananarive 1925	International 1924
2000000290	Tananarive 1925 (Paris)	International 1924
2000000291	Tern Island Astro 1961	International 1924
2000000292	Tete	Clarke 1866
2000000293	Timbalai 1948	Everest (Definition 1967)
2000000294	TM65	Airy Modified
2000000295	TM75	Airy Modified
2000000296	Tokyo	Bessel 1841
2000000297	Trinidad 1903	Clarke 1858
2000000298	Tristan Astro 1968	International 1924
2000000299	Trucial Coast 1948	Helmert 1906
2000000300	Viti Levu 1916	Clarke 1880 (RGS)
2000000301	Voirol 1875	Clarke 1880 (IGN)
2000000302	Voirol 1875(도)	Clarke 1880 (IGN)
2000000303	Voirol 1875 (Paris)	Clarke 1880 (IGN)
2000000304	Voirol Unifie 1960	Clarke 1880 (RGS)
2000000305	Voirol Unifie 1960(도)	Clarke 1880 (RGS)
2000000306	Voirol Unifie 1960 (Paris)	Clarke 1880 (RGS)
2000000307	Wake-Eniwetok 1960	Hough 1960
2000000308	Wake Island Astro 1952	International 1924
2000000309	Walbeck	Walbeck
2000000310	War Office	War Office
2000000311	WGS 1966	WGS 1966
2000000312	WGS 1972	WGS 1972
2000000313	WGS 1972 Transit Broadcast Ephemeris	WGS 1972
2000000314	Yacare	International 1924
2000000315	Yemen Nat'l Geodetic Network 1996	WGS 1984

표 27. 기준면 및 타원체와 연관되는 SRID (계속)

SRID	기준면 이름	참조 타원체
2000000316	Yoff	Clarke 1880 (IGN)
2000000317	Zanderij	International 1924

측지 회전 타원체

회전 타원체(타원체라고도 함)는 특정 위치에서 지구 표면의 모양을 정의하는 지리 좌표 시스템의 일부입니다.

좌표 시스템의 정의에는 다음 예와 같이 DATUM 정의의 일부분인 SPHEROID 정의의 타원체 정의가 포함됩니다.

```
GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",
SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199432955]]
```

DB2GSE.ST_COORDINATE_SYSTEMS 카탈로그 뷰를 사용하면 이 정보를 검색할 수 있습니다. DB2GSE.ST_COORDINATE_SYSTEMS 카탈로그 뷰의 **DEFINITION** 컬럼에는 테이블의 **이름**, **반장축** 및 **평면화** 컬럼의 값이 포함됩니다.

제 20 장 스토어드 프로시저

이 절에서는 DB2 Spatial Extender을 설정하고 공간 데이터를 사용하는 프로젝트를 작성하는 데 사용할 수 있는 스토어드 프로시저에 대한 참조 정보를 제공합니다.

DB2 Spatial Extender를 설정하거나 DB2 제어 센터 또는 DB2 명령행 처리기에서 프로젝트를 작성할 때, 이러한 스토어드 프로시저를 암시적으로 호출합니다. 예를 들어, DB2 제어 센터에 있는 DB2 Spatial Extender 창에서 **확인**을 누르면 DB2는 해당 창과 연관된 스토어드 프로시저를 호출합니다.

또는 응용프로그램에서 스토어드 프로시저를 호출할 수 있습니다.

데이터베이스에서 대부분의 DB2 Spatial Extender 스토어드 프로시저를 호출하기 전에, ST_enable_db 스토어드 프로시저를 직접 호출하거나 DB2 제어 센터를 사용하여 공간 데이터 관련 작업에 해당 데이터베이스를 사용해야 합니다. (이 절에서는 나중에 ST_enable_db에 대한 주제에서 이 스토어드 프로시저 호출에 대해 읽을 수 있습니다.)

데이터베이스를 공간 조작에 사용한 후에는, 해당 데이터베이스에 연결한 경우 해당 데이터베이스에서 암시적으로 또는 명시적으로 DB2 Spatial Extender 스토어드 프로시저를 호출할 수 있습니다.

이 장에는 다음과 같이 모든 DB2 Spatial Extender 스토어드 프로시저에 대한 주제가 나와 있습니다.

- 212 페이지의 『GSE_export_sde』
- 214 페이지의 『GSE_import_sde』
- 216 페이지의 『ST_alter_coordsys』
- 218 페이지의 『ST_alter_srs』
- 222 페이지의 『ST_create_coordsys』
- 225 페이지의 『ST_create_srs』
- 232 페이지의 『ST_disable_autogeocoding』
- 234 페이지의 『ST_disable_db』
- 235 페이지의 『ST_drop_coordsys』
- 236 페이지의 『ST_drop_srs』
- 238 페이지의 『ST_enable_autogeocoding』
- 240 페이지의 『ST_enable_db』
- 242 페이지의 『ST_export_shape』
- 246 페이지의 『ST_import_shape』

© Copyright IBM Corp. 1998, 2006

- 254 페이지의『ST_register_geocoder』
- 259 페이지의『ST_register_spatial_column』
- 261 페이지의『ST_remove_geocoding_setup』
- 263 페이지의『ST_run_geocoding』
- 267 페이지의『ST_setup_geocoding』
- 270 페이지의『ST_unregister_geocoder』
- 272 페이지의『ST_unregister_spatial_ column』

스토어드 프로시저의 구현은 DB2 Spatial Extender 서버의 db2gse 라이브러리에 아카이브됩니다.

GSE_export_sde

이 스토어드 프로시저를 사용하여 공간 컬럼 및 그의 관련 테이블을 SDE 전송 파일로 익스포트하십시오.

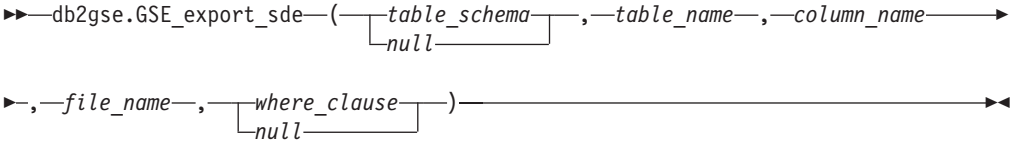
제한사항:

- 정확히 하나의 공간 컬럼이 테이블 또는 뷰에 있어야 합니다.
- 공간 컬럼은 등록되어야 합니다.
- 기존 SDE 파일에 추가할 수 없습니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다. 또한, 사용자 ID는 익스포트될 테이블에 대해 SELECT 특권을 보유하고 있어야 합니다.

구문



매개변수 설명

table_schema

table_name 매개변수에 지정된 테이블이 속해 있는 스키마 이름. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, CURRENT SCHEMA 특수 레지스터는 테이블 또는 뷰에 대한 스키마 이름으로 사용됩니다.

table_schema 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

table_name

익스포트할 테이블의 규정되지 않은 이름을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

table_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

column_name

익스포트하는 등록된 공간 컬럼 이름. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

column_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

file_name

지정된 공간 컬럼 및 해당 연관 테이블이 익스포트될 SDE 전송 파일의 이름을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

이 매개변수의 데이터 유형은 VARCHAR(256)입니다.

where_clause

익스포트할 레코드 세트에 대한 제한사항을 정의하는 SQL WHERE 절의 본문을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, WHERE 절에 제한사항이 정의되지 않습니다.

이 매개변수가 지정되면, 값은 익스포트할 테이블에 있는 모든 속성 컬럼을 참조할 수 있습니다.

이 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 GSE_export_sde 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 CUSTOMERS라는 테이블에서 SDE 파일로 데이터를 익스포트합니다.

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

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

GSE_import_sde

이 스토어드 프로시저를 사용하여 SDE 전송 파일을 공간 데이터 조작에 사용된 데이터베이스로 임포트하십시오.

스토어드 프로시저는 다음 두 가지 방법으로 작동할 수 있습니다.

- 등록된 공간 컬럼이 있는 기존 테이블의 목표가 SDE 전송 파일인 경우에는, DB2 Spatial Extender가 파일 데이터가 있는 테이블을 로드합니다.
- 그렇지 않은 경우에는 DB2 Spatial Extender가 공간 컬럼이 있는 테이블을 작성하고 이 컬럼을 등록하며 공간과 파일 데이터가 들어 있는 테이블의 다른 컬럼들을 로드합니다.

SDE 전송 파일에 지정된 공간 참조 시스템은 DB2 Spatial Extender로 등록된 공간 참조 시스템과 비교됩니다. 지정된 시스템이 등록된 시스템과 일치하면, 로드될 때 등록된 시스템이 지정한 방법으로 전송 데이터의 데이터 값이 수정됩니다. 지정된 시스템이 등록된 시스템 중 어느 것과도 일치하지 않으면, DB2 Spatial Extender가 새 공간 참조 시스템을 작성하여 수정사항을 지정합니다.

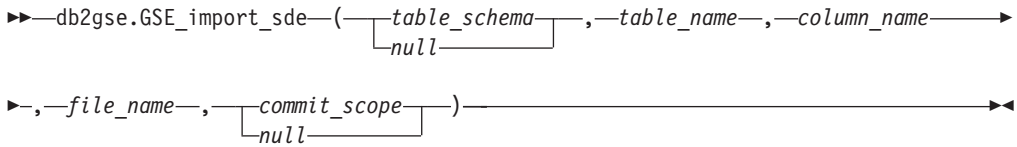
권한 부여

기존 테이블로 데이터를 임포트할 때, 이 스토어드 프로시저를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.

- 데이터가 임포트될 테이블이 들어 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한
- 이 테이블에 대한 CONTROL 특권

데이터를 임포트하려는 테이블이 작성되어 있을 때 이 스토어드 프로시저를 호출하는 사용자 ID는 작성될 테이블이 들어 있는 데이터베이스에 대해 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다.

구문



매개변수 설명

table_schema

table_name 매개변수에 지정된 테이블 또는 뷰가 속해 있는 스키마의 이름. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, CURRENT SCHEMA 특수 레지스터는 테이블 또는 뷰에 대한 스키마 이름으로 사용됩니다.

table_schema 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

table_name

SDE 전송된 데이터가 로드될 테이블의 규정되지 않은 이름을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

table_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

column_name

SDE 전송 파일의 공간 데이터가 로드될 등록된 컬럼을 이름 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

column_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

file_name

임포트될 SDE 전송 파일의 이름. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

이 매개변수의 데이터 유형은 VARCHAR(256)입니다.

commit_scope

COMMIT가 발행되기 전에 임포트될 레코드 수를 지정합니다. 이 매개변수에 값

을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 0값이 사용되며 레코드가 커미트되지 않습니다.

이 매개변수의 데이터 유형은 INTEGER입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 GSE_import_sde 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 CUSTOMERS라는 테이블에 tmp/customerSDE라는 SDE 파일을 임포트합니다. 이 CALL 명령은 COMMIT가 매 5개의 레코드가 임포트된 후에 수행될 것임을 지정합니다.

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

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_alter_coordsys

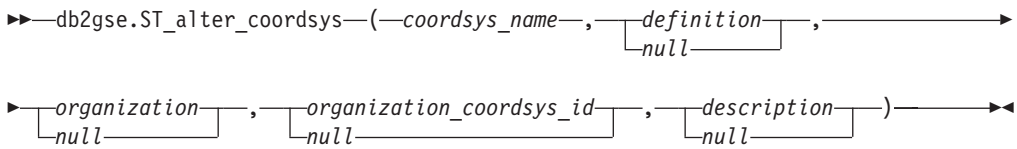
이 스토어드 프로시저를 사용하여 데이터베이스의 좌표 시스템 정의를 갱신하십시오. 이 스토어드 프로시저가 처리될 때, 좌표 시스템에 대한 정보가 DB2GSE.ST_COORDINATE_SYSTEMS 카탈로그 뷰에서 갱신됩니다.

경고: 이 스토어드 프로시저를 주의하여 사용하십시오. 이 스토어드 프로시저를 사용하여 좌표 시스템 정의를 변경하고 이 좌표 시스템을 따르는 공간 참조 시스템과 연관된 기존의 공간 데이터가 있으면, 부주의하여 공간 데이터를 변경할 수 있습니다. 공간 데이터가 영향을 받는 경우, 변경된 공간 데이터가 여전히 정확하고 유효한지 확인해야 합니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다.

구문



매개변수 설명

coordsys_name

좌표 시스템을 고유하게 식별합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

coordsys_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

definition

좌표 시스템을 정의합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 좌표 시스템의 정의가 변경되지 않습니다.

이 매개변수의 데이터 유형은 VARCHAR(2048)입니다.

organization

좌표 시스템을 정의하고 이에 대한 정의를 제공하는 소속의 이름(예: ″EPSG(European Petroleum Survey Group)″). 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다.

이 매개변수가 널(NULL)이면, 좌표 시스템의 소속이 변경되지 않습니다. 이 매개변수가 널(NULL)이 아니면, *organization_coordsys_id* 매개변수가 널(NULL)일 수 없습니다. 이 경우, *organization* 및 *organization_coordsys_id* 매개변수의 조합은 좌표 시스템을 고유하게 식별합니다.

이 매개변수의 데이터 유형은 VARCHAR(128)입니다.

organization_coordsys_id

organization 매개변수에 나열된 엔티티에 의해 이 좌표 시스템에 지정된 숫자 ID를 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다.

이 매개변수가 널(NULL)이면, *organization* 매개변수도 널(NULL) 이어야 합니다. 이 경우, 소속의 좌표 시스템 ID는 변경되지 않습니다. 이 매개변수가 널(NULL)

이 아니면, *organization* 매개변수가 널(NULL) 일 수 없습니다. 이 경우, *organization* 및 *organization_coordsys_id* 매개변수의 조합은 좌표 시스템을 고유하게 식별합니다.

이 매개변수의 데이터 유형은 INTEGER입니다.

description

해당 응용프로그램을 설명하는 좌표 시스템의 설명. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 좌표 시스템에 대한 설명 정보는 변경되지 않습니다.

이 매개변수의 데이터 유형은 VARCHAR(256)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 ST_alter_coordsys 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 NORTH_AMERICAN_TEST라는 좌표 시스템을 갱신합니다. 이 명령은 1002 값을 *coordsys_id* 매개변수에 지정합니다.

```
call db2gse.ST_alter_coordsys('NORTH_AMERICAN_TEST',NULL,NULL,1002,NULL,?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_alter_srs

이 스토어드 프로시저를 사용하여 데이터베이스의 공간 참조 시스템 정의를 갱신하십시오. 이 스토어드 프로시저가 처리될 때 공간 참조 시스템에 대한 정보가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에서 갱신됩니다.

내부적으로 DB2 Spatial Extender는 좌표 값을 양수로 저장합니다. 따라서 계산시 반올림 오류(부동 소수점 연산의 실제 값에 크게 의존하는)의 영향이 줄어들 수 있습니다. 공간 데이터 관련 작업의 성능은 현저하게 향상될 수도 있습니다.

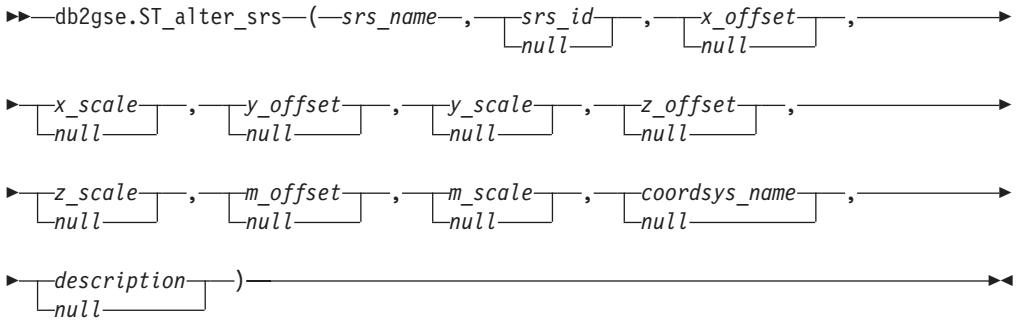
제한사항: 등록된 공간 컬럼이 해당 공간 참조 시스템을 사용하는 경우 공간 참조 시스템을 변경할 수 없습니다.

경고: 이 스토어드 프로시저를 주의하여 사용하십시오. 이 스토어드 프로시저를 사용하여 공간 참조 시스템의 오프셋, 스케일 또는 *coordsys_name* 매개변수를 사용하거나 공간 참조 시스템과 연관된 기존의 공간 데이터가 있는 경우, 공간 데이터를 부주의하게 변경할 수 있습니다. 공간 데이터가 영향을 받는 경우, 변경된 공간 데이터가 여전히 정확하고 유효한지 확인해야 합니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다.

구문



매개변수 설명

srs_name

공간 참조 시스템을 식별합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

srs_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

srs_id

공간 참조 시스템을 고유하게 식별합니다. 이 ID는 다양한 공간 함수에 대한 입력 매개변수로 사용됩니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 공간 참조 시스템의 숫자 ID는 변경되지 않습니다.

이 매개변수의 데이터 유형은 INTEGER입니다.

x_offset

이 공간 참조 시스템에 표시된 Geometry의 모든 X축에 대한 오프셋을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 공간 참조 시스템 정의에 있는 이 매개변수의 값은 변경되지 않습니다.

Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 스케일 인수 *x_scale*이 적용되기 전에 오프셋을 뺍니다. (*WKT*는 Well-Known Text이며, *WKB*는 Well-Known Binary입니다.)

이 매개변수의 데이터 유형은 DOUBLE입니다.

x_scale

이 공간 참조 시스템에 표시된 Geometry의 모든 X축에 대한 스케일 인수를 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 공간 참조 시스템 정의에 있는 이 매개변수의 값은 변경되지 않습니다.

Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 오프셋 *x_offset*을 뺀 후 스케일 인수가 적용됩니다(곱셈).

이 매개변수의 데이터 유형은 DOUBLE입니다.

y_offset

이 공간 참조 시스템에 표시된 Geometry의 모든 Y축에 대한 오프셋을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 공간 참조 시스템 정의에 있는 이 매개변수의 값은 변경되지 않습니다.

Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 스케일 인수 *y_scale*이 적용되기 전에 오프셋을 뺍니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

y_scale

이 공간 참조 시스템에 표시된 Geometry의 모든 Y축에 대한 스케일 인수를 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 공간 참조 시스템 정의에 있는 이 매개변수의 값은 변경되지 않습니다.

Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 오프셋 *y_offset*을 뺀 후 스케일 인수가 적용됩니다(곱셈). 이 스케일 인수는 *x_scale*과 동일해야 합니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

z_offset

이 공간 참조 시스템에 표시된 Geometry의 모든 Z축에 대한 오프셋을 지정합니

다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 공간 참조 시스템 정의에 있는 이 매개변수의 값은 변경되지 않습니다.

Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 스케일 인수 *z_scale*이 적용되기 전에 오프셋을 뺍니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

z_scale

이 공간 참조 시스템에 표시된 Geometry의 모든 Z축에 대한 스케일 인수를 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 공간 참조 시스템 정의에 있는 이 매개변수의 값은 변경되지 않습니다.

Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 오프셋 *z_offset*을 뺀 후 스케일 인수가 적용됩니다(곱셈).

이 매개변수의 데이터 유형은 DOUBLE입니다.

m_offset

이 공간 참조 시스템에 표시된 Geometry의 모든 M축에 대한 오프셋을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 공간 참조 시스템 정의에 있는 이 매개변수의 값은 변경되지 않습니다.

Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 스케일 인수 *m_scale*이 적용되기 전에 오프셋을 뺍니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

m_scale

이 공간 참조 시스템에 표시된 Geometry의 모든 M축에 대한 스케일 인수를 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 공간 참조 시스템 정의에 있는 이 매개변수의 값은 변경되지 않습니다.

Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 오프셋 *m_offset*을 뺀 후 스케일 인수가 적용됩니다(곱셈).

이 매개변수의 데이터 유형은 DOUBLE입니다.

coordsys_name

이 공간 참조 시스템이 기초가 되는 좌표 시스템을 고유하게 식별합니다. 좌표 시스템은 뷰 ST_COORDINATE_SYSTEMS에 나열되어야 합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 이 공간 참조 시스템에 사용되는 좌표 시스템은 변경되지 않습니다.

coordsys_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

description

해당 응용프로그램을 설명하는 공간 참조 시스템의 설명. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 공간 참조 시스템에 대한 설명 정보는 변경되지 않습니다.

이 매개변수의 데이터 유형은 VARCHAR(256)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 ST_alter_srs 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 SRSDEMO라는 공간 참조 시스템의 *description* 매개변수 값을 변경합니다.

```
call db2gse.ST_alter_srs('SRSDEMO',NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
     NULL,NULL,'SRS for GSE Demo Program: offices table',?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

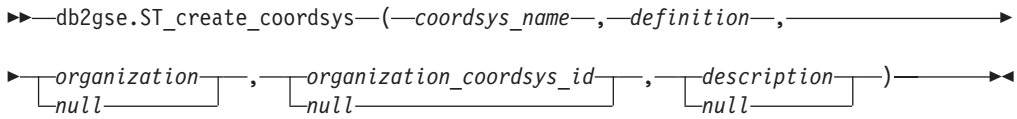
ST_create_coordsys

이 스토어드 프로시저를 사용하여 데이터베이스에서 새 좌표 시스템에 대한 정보를 저장하십시오. 이 스토어드 프로시저가 처리될 때 좌표 시스템에 대한 정보가 DB2GSE.ST_COORDINATE_SYSTEMS 카탈로그 뷰에 추가됩니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다.

구문



매개변수 설명

coordsys_name

좌표 시스템을 고유하게 식별합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

coordsys_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

definition

좌표 시스템을 정의합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다. 좌표 시스템을 제공하는 벤더는 대개 이 매개변수에 대한 정보를 제공합니다.

이 매개변수의 데이터 유형은 VARCHAR(2048)입니다.

organization

좌표 시스템을 정의하고 이에 대한 정의를 제공하는 소속의 이름(예: ″EPSG(European Petroleum Survey Group)″). 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다.

이 매개변수가 널(NULL)이면, *organization_coordsys_id* 매개변수도 널(NULL)이어야 합니다. 이 매개변수가 널(NULL)이 아니면, *organization_coordsys_id* 매개변수가 널(NULL)일 수 없습니다. 이 경우, *organization* 및 *organization_coordsys_id* 매개변수의 조합은 좌표 시스템을 고유하게 식별합니다.

이 매개변수의 데이터 유형은 VARCHAR(128)입니다.

organization_coordsys_id

숫자 ID를 지정합니다. *organization* 매개변수에 지정된 엔티티는 이 값을 지정합니다. 이 값은 모든 좌표 시스템을 통해 반드시 고유할 필요는 없습니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다.

이 매개변수가 널(NULL)이면, *organization* 매개변수도 널(NULL)이어야 합니다. 이 매개변수가 널(NULL)이 아니면, *organization* 매개변수가 널(NULL) 일 수 없습니다. 이 경우, *organization* 및 *organization_coordsys_id* 매개변수의 조합은 좌표 시스템을 고유하게 식별합니다.

이 매개변수의 데이터 유형은 INTEGER입니다.

description

해당 응용프로그램을 설명하는 좌표 시스템의 설명. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 좌표 시스템에 대한 설명 정보가 기록되지 않습니다.

이 매개변수의 데이터 유형은 VARCHAR(256)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 ST_create_coordsys 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 다음 매개변수 값으로 좌표 시스템을 작성합니다.

- *coordsys_name* 매개변수: NORTH_AMERICAN_TEST
- *definition* 매개변수:

```
GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",
     SPHEROID["GRS_1980",6378137.0,298.257222101]],
PRIMEM["Greenwich",0.0],
UNIT["Degree",0.0174532925199433]]
```

- *organization* 매개변수: EPSG
- *organization_coordsys_id* 매개변수: 1001
- *description* 매개변수: 좌표 시스템 테스트

```
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',?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_create_srs

스토어드 프로시저를 사용하여 공간 참조 시스템을 작성하십시오.

공간 참조 시스템은 좌표 시스템, 정밀도 및 이 공간 참조 시스템에서 표시되는 좌표의 범위로 정의됩니다. 범위는 X, Y, Z 및 M 좌표의 가능한 최소 및 최대 좌표 값입니다.

내부적으로 DB2 Spatial Extender는 좌표 값을 양수로 저장합니다. 따라서 계산시 반올림 오류(부동 소수점 연산의 실제 값에 크게 의존하는)의 영향이 줄어들 수 있습니다. 공간 데이터 관련 작업의 성능은 현저하게 향상될 수도 있습니다.

이 스토어드 프로시저에는 다음의 두 가지 변화가 있습니다.

- 첫 번째 변화는 변환 인수(오프셋 및 스케일 인수)를 입력 매개변수로 가져옵니다.
- 두 번째 변화는 범위와 정밀도를 입력 매개변수로 가져와 내부적으로 변환 인수를 계산합니다.

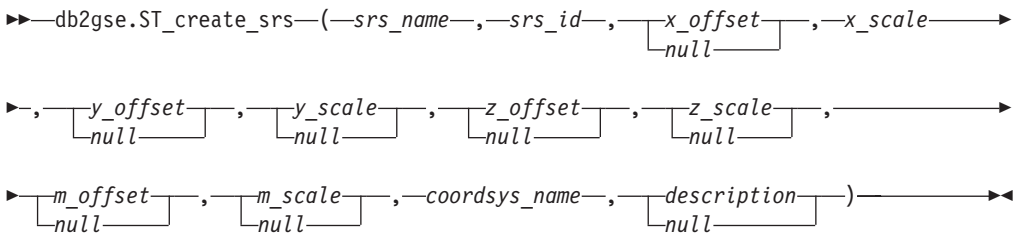
이 스토어드 프로시저는 db2gse.gse_enable_sref를 대체합니다.

권한 부여

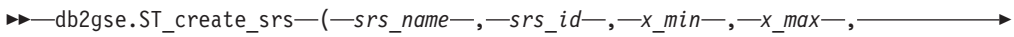
아무 것도 필요하지 않습니다.

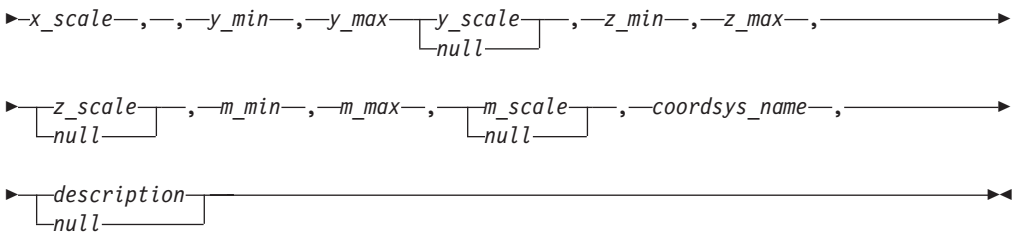
구문

변환 인수 사용(버전 1)



가능한 최대 범위 사용(버전 2)





매개변수 설명

변환 인수 사용(버전 1)

srs_name

공간 참조 시스템을 식별합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

srs_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

srs_id

공간 참조 시스템을 고유하게 식별합니다. 이 숫자 ID는 다양한 공간 함수에 대한 입력으로 사용됩니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

측지 공간 참조 시스템의 경우, *srs_id* 값은 2000000318 - 2000001000 범위 내에 있어야 합니다. DB2 Geodetic Data Management Feature에서는 *srs_id*값 2000000000 - 2000000317 범위 내에 있는 사전 정의된 측지 공간 참조 시스템을 제공합니다.

이 매개변수의 데이터 유형은 INTEGER입니다.

x_offset

이 공간 참조 시스템에 표시된 Geometry의 모든 X축에 대한 오프셋을 지정합니다. Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 스케일 인수 *x_scale*이 적용되기 전에 오프셋을 뺍니다. (*WKT*는 Well-Known Text이며, *WKB*는 Well-Known Binary입니다.) 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 0값이 사용됩니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

x_scale

이 공간 참조 시스템에 표시된 Geometry의 모든 X축에 대한 스케일 인수를 지정합니다. Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 오프셋 *x_offset*을 뺀 후 스케일 인수가 적용됩니다(곱셈).

x_offset 값은 명시적으로 지정되거나 디폴트 *x_offset* 값 0이 사용됩니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

y_offset

이 공간 참조 시스템에 표시된 Geometry의 모든 Y축에 대한 오프셋을 지정합니다. Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 스케일 인수 *y_scale*이 적용되기 전에 오프셋을 뺍니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL) 값이면, 0값이 사용됩니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

y_scale

이 공간 참조 시스템에 표시된 Geometry의 모든 Y축에 대한 스케일 인수를 지정합니다. Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 오프셋 *y_offset*을 뺀 후 스케일 인수가 적용됩니다(곱셈). *y_offset* 값은 명시적으로 지정되거나 디폴트 *y_offset* 값 0이 사용됩니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, *x_scale* 매개변수 값이 사용됩니다. 이 매개변수에 널(NULL)이 아닌 값을 지정하면, 지정한 값이 *x_scale* 매개변수 값과 일치해야 합니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

z_offset

이 공간 참조 시스템에 표시된 Geometry의 모든 Z축에 대한 오프셋을 지정합니다. Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 스케일 인수 *z_scale*이 적용되기 전에 오프셋을 뺍니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 0값이 사용됩니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

z_scale

이 공간 참조 시스템에 표시된 Geometry의 모든 Z축에 대한 스케일 인수를 지정합니다. Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 오프셋 *z_offset*을 뺀 후 스케일 인수가 적용됩니다(곱셈). *z_offset* 값은 명시적으로 지정되거나 디폴트 *z_offset* 값 0이 사용됩니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 1값이 사용됩니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

m_offset

이 공간 참조 시스템에 표시된 Geometry의 모든 M축에 대한 오프셋을 지정합니다. Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부

표현으로 변환될 때 스케일 인수 *m_scale*이 적용되기 전에 오프셋을 뺍니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 0값이 사용됩니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

m_scale

이 공간 참조 시스템에 표시된 Geometry의 모든 M축에 대한 스케일 인수를 지정합니다. Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 오프셋 *m_offset*을 뺀 후 스케일 인수가 적용됩니다(곱셈). *m_offset* 값은 명시적으로 지정되거나, 디폴트 *m_offset* 값 0이 사용됩니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 1값이 사용됩니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

coordsys_name

이 공간 참조 시스템이 기초가 되는 좌표 시스템을 고유하게 식별합니다. 좌표 시스템은 뷰 ST_COORDINATE_SYSTEMS에 나열되어야 합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 제공해야 합니다.

coordsys_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

description

응용프로그램의 목적을 기술함으로써 공간 참조 시스템을 설명합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 설명 정보가 기록되지 않습니다.

이 매개변수의 데이터 유형은 VARCHAR(256)입니다.

가능한 최대 범위 사용(버전 2)

srs_name

공간 참조 시스템을 식별합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

srs_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

srs_id

공간 참조 시스템을 고유하게 식별합니다. 이 숫자 ID는 다양한 공간 함수에 대한 입력으로 사용됩니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

이 매개변수의 데이터 유형은 INTEGER입니다.

x_min

이 공간 참조 시스템을 사용하는 모든 Geometry에 대해 가능한 최소 X 좌표 값을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

x_max

이 공간 참조 시스템을 사용하는 모든 Geometry에 대해 가능한 최대 X 좌표 값을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

x_scale 값에 따라 뷰 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS에 표시되는 값이 여기에 지정된 값보다 클 수 있습니다. 뷰의 값이 올바릅니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

x_scale

이 공간 참조 시스템에 표시된 Geometry의 모든 X축에 대한 스케일 인수를 지정합니다. Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 오프셋 *x_offset*을 뺀 후 스케일 인수가 적용됩니다(곱셈). 오프셋 *x_offset*의 계산은 *x_min* 값에 따라 다릅니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 제공해야 합니다.

x_scale 및 *y_scale* 매개변수가 모두 지정되면, 값이 일치해야 합니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

y_min

이 공간 참조 시스템을 사용하는 모든 Geometry에 대해 가능한 최소 Y 좌표 값을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 제공해야 합니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

y_max

이 공간 참조 시스템을 사용하는 모든 Geometry에 대해 가능한 최대 Y 좌표 값을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 제공해야 합니다.

y_scale 값에 따라 뷰 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS에 표시되는 값이 여기에 지정된 값보다 클 수 있습니다. 뷰의 값이 올바릅니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

y_scale

이 공간 참조 시스템에 표시된 Geometry의 모든 Y축에 대한 스케일 인수를 지정합니다. Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 오프셋 *y_offset*을 뺀 후 스케일 인수가 적용됩니다(곱셈). 오프셋 *y_offset*의 계산은 *y_min* 값에 따라 다릅니다. 이 매개변수에 값을 지정해

야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, *x_scale* 매개변수 값이 사용됩니다. *y_scale* 및 *x_scale* 매개변수 모두가 지정되면 값이 일치해야 합니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

z_min

이 공간 참조 시스템을 사용하는 모든 Geometry에 대해 가능한 최소 Z 좌표 값을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

z_max

이 공간 참조 시스템을 사용하는 모든 Geometry에 대해 가능한 최대 Z 좌표 값을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

z_scale 값에 따라 뷰 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS에 표시되는 값이 여기에 지정된 값보다 클 수 있습니다. 뷰의 값이 올바릅니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

z_scale

이 공간 참조 시스템에 표시된 Geometry의 모든 Z축에 대한 스케일 인수를 지정합니다. Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender 내부 표현으로 변환될 때 오프셋 *z_offset*을 뺀 후 스케일 인수가 적용됩니다(곱셈). 오프셋 *z_offset*의 계산은 *z_min* 값에 따라 다릅니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 1 값이 사용됩니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

m_min

이 공간 참조 시스템을 사용하는 모든 Geometry에 대해 가능한 최소 M 좌표 값을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

m_max

이 공간 참조 시스템을 사용하는 모든 Geometry에 대해 가능한 최대 M 좌표 값을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

m_scale 값에 따라 뷰 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS에 표시되는 값이 여기에 지정된 값보다 클 수 있습니다. 뷰의 값이 올바릅니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

m_scale

이 공간 참조 시스템에 표시된 Geometry의 모든 M축에 대한 스케일 인수를 지정합니다. Geometry가 외부 표현(WKT, WKB, 모양)에서 DB2 Spatial Extender

내부 표현으로 변환될 때 오프셋 *m_offset*을 뺀 후 스케일 인수가 적용됩니다(곱셈). 오프셋 *m_offset*의 계산은 *m_min* 값에 따라 다릅니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 1값이 사용됩니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

coordsys_name

이 공간 참조 시스템이 기초가 되는 좌표 시스템을 고유하게 식별합니다. 좌표 시스템은 뷰 ST_COORDINATE_SYSTEMS에 나열되어야 합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

coordsys_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

description

응용프로그램의 목적을 기술함으로써 공간 참조 시스템을 설명합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 설명 정보가 기록되지 않습니다.

이 매개변수의 데이터 유형은 VARCHAR(256)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 ST_create_srs 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 다음 매개변수 값으로 SRSDEMO라는 공간 참조 시스템을 작성합니다.

- *srs_id*: 1000000
- *x_offset*: -180

- *x_scale*: 1000000
- *y_offset*: -90
- *y_scale*: 1000000

```
call db2gse.ST_create_srs('SRSDEMO',1000000,
                     -180,1000000, -90, 1000000,
                     0, 1, 0, 1,'NORTH_AMERICAN',
                     'SRS for GSE Demo Program: customer table',?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_disable_autogeocoding

이 스토어드 프로시저를 사용하여 DB2 Spatial Extender가 지역코드화된 컬럼과 그와 연관된 지오코딩(Geocoding) 컬럼 또는 컬럼의 동기화를 중지하도록 지정합니다.

*지오코딩 컬럼*은 지오코더(Geocoder)로 입력할 때 사용됩니다.

이 스토어드 프로시저는 db2gse.gse_disable_autogc를 대체합니다.

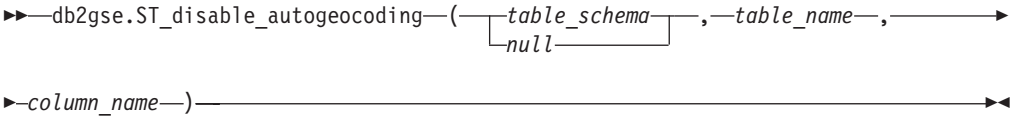
권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.

- 트리거가 삭제되거나 정의된 테이블이 들어 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한.
- 이 테이블에 대한 CONTROL 특권
- 이 테이블에서의 ALTER 및 UPDATE 특권.

주: CONTROL 및 ALTER 특권의 경우, DB2GSE 스키마에 대한 DROPIN 권한이 있어야 합니다.

구문



매개변수 설명

table_schema

table_name 매개변수에 지정된 테이블 또는 뷰가 속해 있는 스키마의 이름. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, CURRENT SCHEMA 특수 레지스터는 테이블 또는 뷰에 대한 스키마 이름으로 사용됩니다.

table_schema 값은 따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

table_name

삭제할 트리거가 정의된 테이블의 규정되지 않은 이름을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

table_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

column_name

삭제할 트리거가 유지보수하는 지역코드화된 컬럼의 이름. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

column_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 ST_disable_autogeocoding 스토어드 프로시저를 호출하기 위해 DB2 명령행 처리기를 사용하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 CUSTOMERS라는 테이블의 LOCATION 컬럼에서 자동 지오코딩(Geocoding)을 사용하지 않게 합니다.

```
call db2gse.ST_disable_autogeocoding(NULL,'CUSTOMERS','LOCATION',?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_disable_db

이 스토어드 프로시저를 사용하여 DB2 Spatial Extender가 공간 데이터를 저장 및 지원할 수 있도록 자원을 제거하십시오.

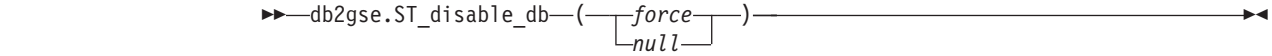
이 스토어드 프로시저는 공간 데이터 조작에 대해 데이터베이스를 사용할 수 있게 된 후에 발생되는 문제점을 해결하는 데 도움을 주는 것입니다. 예를 들면, 공간 데이터 관련 작업에 대해 데이터베이스를 사용한 후 대신 DB2 Spatial Extender에 또 다른 데이터베이스를 사용하도록 결정할 수 있습니다. 공간 컬럼을 정의하지 않았거나 공간 데이터를 임포트하지 않은 경우, 이 스토어드 프로시저를 호출하여 첫 번째 데이터베이스에서 모든 공간 자원을 제거할 수 있습니다. 공간 컬럼과 유형 정의 사이의 상호 의존성으로 인해, 그러한 유형의 컬럼이 존재할 때 유형 정의를 삭제할 수 없습니다. 공간 컬럼을 이미 정의했으나 공간 데이터 관련 작업에 대해 여전히 데이터베이스를 사용하지 않게 하려면, 이들에 대해 다른 종속성이 없는 데이터베이스의 모든 공간 자원을 제거하기 위해 *force* 매개변수에 대해 0이 아닌 값을 지정해야 합니다.

이 스토어드 프로시저는 db2gse.gse_disable_db를 대체합니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 DB2 Spatial Extender 자원을 제거할 데이터베이스에 대해 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다.

구문



매개변수 설명

force

공간 유형 또는 공간 함수에 종속하는 데이터베이스 오브젝트를 가질 수 있더라도, 공간 데이터 관련 작업에 대해 데이터베이스를 사용하지 않도록 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. *force* 매개변수에 0 또는 널(NULL)이 아닌 값을 지정하면, 데이터베이스를 사용할 수 없으며 DB2 Spatial Extender의 모든 자원이 제거됩니다(가능한 경우). 0 또는 널(NULL)을 지정하면, 데이터베이스가 공간 유형 또는 공간 함수에 종속하는 경우 데이터베이스는 사용 가능합니다. 그러한 종속성을 가질 수 있는 데이터베이스 오브젝트에는 테이블, 뷰, 제한조건, 트리거, 생성된 컬럼, 메소드, 함수, 프로시저 및 기타 데이터 유형(공간 속성이 있는 부속 유형 또는 구조화된 유형)이 포함됩니다.

이 매개변수의 데이터 유형은 SMALLINT입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 ST_disable_db 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 *force* 매개변수 값이 1인 공간 데이터 관련 작업에 대해 데이터베이스를 사용하지 않게 합니다.

```
call db2gse.ST_disable_db(1,?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_drop_coordsys

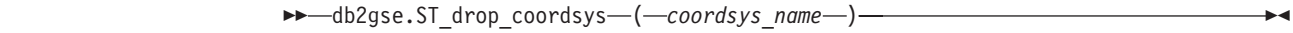
이 스토어드 프로시저를 사용하여 데이터베이스에서 좌표 시스템에 대한 정보를 삭제하십시오. 이 스토어드 프로시저가 처리될 때 좌표 시스템에 대한 정보가 DB2GSE.ST_COORDINATE_SYSTEMS 카탈로그 뷰에서 제거됩니다.

제한사항:
공간 참조 시스템의 기초가 되는 좌표 시스템을 삭제할 수 없습니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다.

구문



매개변수 설명

coordsys_name

좌표 시스템을 고유하게 식별합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

coordsys_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 ST_drop_coordsys 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 데이터베이스에서 NORTH_AMERICAN_TEST라는 좌표 시스템을 삭제합니다.

```
call db2gse.ST_drop_coordsys('NORTH_AMERICAN_TEST',?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_drop_srs

이 스토어드 프로시저를 사용하여 공간 참조 시스템을 삭제하십시오.

이 스토어드 프로시저가 처리될 때 공간 참조 시스템에 대한 정보가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에서 제거됩니다.

제한사항: 공간 참조 시스템을 사용하는 공간 컬럼이 등록된 경우 해당 공간 참조 시스템을 삭제할 수 없습니다.

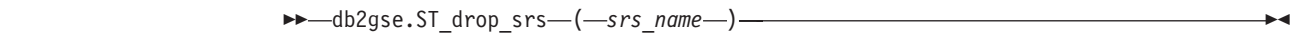
중요사항:
이 스토어드 프로시저를 사용할 때 주의하십시오. 이 스토어드 프로시저를 사용하여 공간 참조 시스템을 삭제할 경우와 공간 데이터가 해당 공간 참조 시스템과 연관된 경우, 더 이상 공간 데이터에서 공간 데이터 관련 작업을 수행할 수 없습니다.

이 스토어드 프로시저는 db2gse.gse_disable_sref를 대체합니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다.

구문



매개변수 설명

srs_name

공간 참조 시스템을 식별합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

srs_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 ST_drop_srs 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 SRSDEMO라는 공간 참조 시스템을 삭제하기 위해 DB2 CALL 명령을 사용합니다.

```
call db2gse.ST_drop_srs('SRSDEMO',?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_enable_autogeocoding

이 스토어드 프로시저를 사용하여 DB2 Spatial Extender가 지오코딩(Geocoding)된 컬럼과 그와 연관된 지오코딩(Geocoding) 컬럼 또는 컬럼들을 동기화하도록 지정합니다.

*지오코딩 컬럼*은 지오코더(Geocoder)로 입력할 때 사용됩니다. 값이 삽입되거나 갱신될 때마다 지오코딩(Geocoding) 컬럼 트리거가 활성화됩니다. 이러한 트리거는 연관된 지오코더(Geocoder)를 호출하여 삽입되거나 갱신된 값을 지오코딩(Geocoding)하고 지역코드화된 컬럼에 결과 데이터를 배치합니다.

제한사항: INSERT 및 UPDATE 트리거가 작성될 수 있는 테이블에서만 autogeocoding을 사용할 수 있습니다. 따라서 뷰 또는 별칭에서 자동 지오코딩(Geocoding)을 사용할 수 없습니다.

전제조건: 자동 지오코딩을 사용하기 전에, ST_setup_geocoding 스토어드 프로시저를 호출하여 지오코딩 설정을 수행해야 합니다. 지오코딩(Geocoding) 설정 단계는 지오코더(Geocoder)와 지오코딩(Geocoding) 매개변수 값을 지정합니다. 지역코드화된 컬럼과 동기화될 지오코딩(Geocoding) 컬럼도 식별합니다.

이 스토어드 프로시저는 db2gse.gse_enable_autogc를 대체합니다.

권한 부여

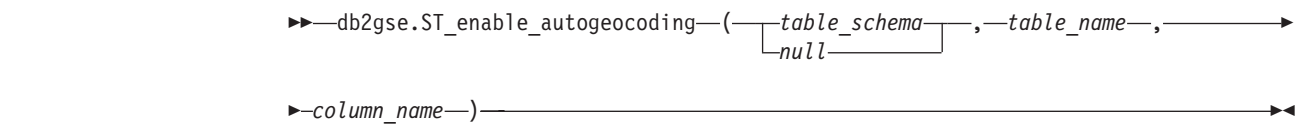
이 스토어드 프로시저를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.

- 이 스토어드 프로시저가 작성한 트리거가 정의되는 테이블이 들어 있는 데이터베이스에 대한 SYSADM 또는 DBADM 권한
- 테이블에 대한 CONTROL 특권
- 테이블에 대한 ALTER 특권

명령문의 권한 부여 ID에 SYSADM 또는 DBADM 권한이 없으면, 명령문의 권한 부여 ID가 보유하는 특권(PUBLIC 또는 그룹 특권을 고려하지 않음)은 트리거가 존재하는 한 다음 특권을 모두 포함해야 합니다.

- 자동 지오코딩(Geocoding)이 사용 가능한 테이블의 SELECT 특권
- 지오코딩(Geocoding) 설정에서 매개변수에 지정된 SQL 표현식을 평가하는 데 필요한 특권

구문



매개변수 설명

table_schema

table_name 매개변수에 지정된 테이블이 속해 있는 스키마를 식별합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, CURRENT SCHEMA 특수 레지스터는 테이블에 대한 스키마 이름으로 사용됩니다.

table_schema 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

table_name

지역코드화된 데이터가 삽입되거나 갱신될 컬럼이 들어 있는 테이블의 규정되지 않은 이름을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

table_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

column_name

지역코드화된 데이터가 삽입되거나 갱신될 컬럼을 식별합니다. 이 컬럼은 지역코드화된 컬럼으로 언급됩니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

column_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 ST_enable_autogeocoding을 호출하기 위해 DB2 명령행 처리기를 사용하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 CUSTOMER라는 테이블의 LOCATION 컬럼에서 자동 지오코딩(Geocoding)을 사용합니다.

```
call db2gse.ST_enable_autogeocoding(NULL,'CUSTOMERS','LOCATION',?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_enable_db

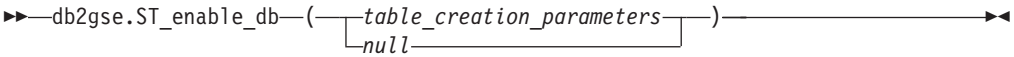
이 스토어드 프로시저를 사용하여 공간 데이터를 저장하고 공간 조작들을 지원하는 데 필요한 자원을 데이터베이스에 제공하십시오. 이 자원에는 공간 데이터 유형, 공간 인덱스 유형, 카탈로그 뷰, 제공된 함수 및 다른 스토어드 프로시저가 포함됩니다.

이 스토어드 프로시저는 db2gse.gse_enable_db를 대체합니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 사용될 데이터베이스에 대해 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다.

구문



매개변수 설명

table_creation_parameters

DB2 Spatial Extender 카탈로그 테이블에 대해 CREATE TABLE문에 추가할 옵션을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, CREATE TABLE문에 옵션이 추가되지 않습니다.

이 옵션을 지정하려면, DB2 CREATE TABLE문의 구문을 사용하십시오. 예를 들어, 테이블을 작성할 테이블 스페이스를 지정하려면, 다음을 사용하십시오.

```
IN tsName INDEX IN indexTsName
```

이 매개변수의 데이터 유형은 VARCHAR(32K)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

다음 예에서는 콜 레벨 인터페이스(CLI)를 사용하여 ST_enable_db 스토어드 프로시저를 호출하는 방법을 보여줍니다.

```
SQLHANDLE henv;
  SQLHANDLE hdbc;
  SQLHANDLE hstmt;
  SQLCHAR uid[MAX_UID_LENGTH + 1];
  SQLCHAR pwd[MAX_PWD_LENGTH + 1];
  SQLINTEGER ind[3];
  SQLINTEGER msg_code = 0;
    char       msg_text[1024] = "";
  SQLRETURN rc;
  char *table_creation_parameters = NULL;

  /* Allocate environment handle */
  rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);

  /* Allocate database handle */
  rc = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);

    /* Establish a connection to database "testdb" */
    rc = SQLConnect(hdbc, (SQLCHAR *)"testdb", SQL_NTS, (SQLCHAR *)uid,SQL_NTS,
                  (SQLCHAR *)pwd, SQL_NTS);


  /* Allocate statement handle */
  rc = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt) ;

  /* Associate SQL statement to call the ST_enable_db stored procedure */
  /* with statement handle and send the statement to DBMS to be prepared. */
rc = SQLPrepare(hstmt, "call db2gse!ST_enable_db(?,?,?)", SQL_NTS);

  /* Bind 1st parameter marker in the SQL call statement, the input */
  /* parameter for table creation parameters, to variable */
  /* table_creation_parameters. */
```

```
ind[0] = SQL_NULL_DATA;
  rc = SQLBindParameter(hstmt, 1, SQL_PARAM_OUTPUT, SQL_C_CHAR,
       SQL_VARCHAR, 255, 0, table_creation_parameters, 256, &ind[0]);

  /* Bind 2nd parameter marker in the SQL call statement, the output */
  /* parameter for returned message code, to variable msg_code. */
ind[1] = 0;
  rc = SQLBindParameter(hstmt, 2, SQL_PARAM_OUTPUT, SQL_C_LONG,
                        SQL_INTEGER, 0, 0, &msg_code, 4, &ind[1]);

  /* Bind 3rd parameter marker in the SQL call statement, the output */
  /* parameter returned message text, to variable msg_text. */
ind[2] = 0;
  rc = SQLBindParameter(hstmt, 3, SQL_PARAM_OUTPUT, SQL_C_CHAR,
                        SQL_VARCHAR, (sizeof(msg_text)-1), 0, msg_text,
                         sizeof(msg_text), &ind[2]);
  rc = SQLExecute(hstmt);
```

ST_export_shape

이 스토어드 프로시저를 사용하여 공간 컬럼 및 그의 관련 테이블을 모양 파일로 익스포트하십시오.

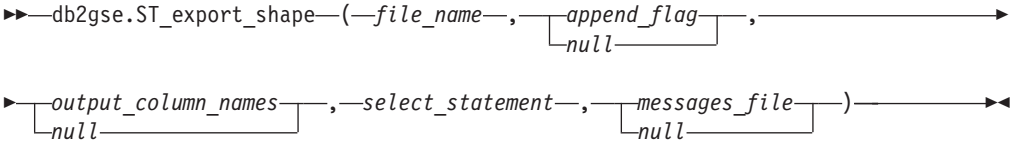
이 스토어드 프로시저는 db2gse.gse_export_shape를 대체합니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID에는 데이터가 익스포트될 SELECT문을 실행하는 데 필요한 특권이 있어야 합니다.

DB2 인스턴스 소유자가 소유하는 프로세스로 실행되는 스토어드 프로시저에는 모양 파일을 작성하거나 기록하는 데 필요한 서버 머신의 특권이 있어야 합니다.

구문



매개변수 설명

file_name

지정된 데이터가 익스포트될 모양 파일의 전체 경로명을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

ST_export_shape 스토어드 프로시저를 사용하여 새 파일을 익스포트하거나 기존 파일에 익스포트된 데이터를 추가하여 기존 파일에 익스포트할 수 있습니다.

- 새 파일에 익스포트할 경우, 선택적 파일 확장자를 .shp 또는 SHP로 지정할 수 있습니다. 파일 확장자에 .shp 또는 .SHP를 지정하면, DB2 Spatial Extender

는 지정된 *file_name* 값으로 파일을 작성합니다. 선택적 파일 확장자를 지정하지 않으면, DB2 Spatial Extender는 이름이 지정한 *file_name* 값이고 확장자가 .shp인 파일을 작성합니다.

- 기존 파일에 데이터를 추가하여 데이터를 익스포트할 경우, DB2 Spatial Extender는 먼저 *file_name* 매개변수에 지정한 완전 일치하는 이름을 찾습니다. DB2 Spatial Extender가 완전 일치하는 이름을 찾지 못하면, 먼저 확장자가 .shp인 파일을 찾은 후 확장자가 .SHP인 파일을 찾습니다.

append_flag 매개변수 값이 기존 파일에 추가하지는 않는다는 것을 나타내지만 *file_name* 매개변수에서 이름 지정한 파일이 이미 있는 경우, DB2 Spatial Extender는 오류를 리턴하며 파일 위에 겹쳐쓰지 않습니다.

서버 머신에 기록된 파일 목록은 사용 참고를 참조하십시오. DB2 인스턴스 소유자가 소유하는 프로세스로 실행되는 스토어드 프로시저에는 파일을 작성하거나 기록하는 데 필요한 서버 머신의 특권이 있어야 합니다.

이 매개변수의 데이터 유형은 VARCHAR(256)입니다.

append_flag

익스포트될 데이터가 기존 모양 파일에 추가될 것인지 여부를 나타냅니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 다음과 같이 기존 모양 파일에 추가할 것인지 여부를 나타냅니다.

- 기존 모양 파일에 데이터를 추가하려면, 0과 널(NULL)이 아닌 값을 지정하십시오. 이 경우, 파일 구조는 익스포트된 데이터와 일치해야 합니다. 그렇지 않으면 오류가 리턴됩니다.
- 새 파일로 익스포트하려면, 0 또는 널(NULL)을 지정하십시오. 이 경우, DB2 Spatial Extender는 기존 파일 위에 겹쳐쓰지 않습니다.

이 매개변수의 데이터 유형은 SMALLINT입니다.

output_column_names

출력 dBASE 파일에서 비 공간 컬럼에 사용할 하나 이상의 컬럼 이름(쉼표로 구분된)을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, SELECT문에서 파생된 이름이 사용됩니다.

이 매개변수를 지정하나 컬럼 이름을 큰따옴표로 묶지 않으면, 컬럼 이름은 대문자로 변환됩니다. 공간 컬럼을 제외하고 *select_statement* 매개변수에 지정된 대로, 지정된 컬럼 수가 SELECT문에서 리턴된 컬럼 수와 일치해야 합니다.

이 매개변수의 데이터 유형은 VARCHAR(32K)입니다.

select_statement

익스포트될 데이터를 리턴하는 subselect를 지정합니다. subselect는 하나의 공간 컬럼과 임의의 수의 속성 컬럼을 정확히 참조해야 합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

이 매개변수의 데이터 유형은 VARCHAR(32K)입니다.

messages_file

익스포트 조작에 대한 메시지를 포함할 파일의 전체 경로 이름(서버 머신의)을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, DB2 Spatial Extender 메시지에 대한 파일이 작성되지 않습니다.

이 메시지 파일로 보낸 메시지는 다음과 같을 수 있습니다.

- 익스포트 조작 요약과 같은 정보 메시지
- 익스포트될 수 없는 데이터에 대한 오류 메시지(예: 다른 좌표 시스템으로 인해)

DB2 인스턴스 소유자가 소유하는 프로세스로 실행되는 스토어드 프로시저에는 파일을 작성하는 데 필요한 서버의 특권이 있어야 합니다.

이 매개변수의 데이터 유형은 VARCHAR(256)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

사용 참고

한번에 하나의 공간 컬럼만 익스포트할 수 있습니다.

ST_export_shape 스토어드 프로시저는 다음의 네 가지 파일을 작성하거나 이 파일에 기록합니다.

- 기본 모양 파일(.shp 확장자).
- 모양 인덱스 파일(.shx 확장자).

- 비 공간 컬러에 대한 데이터가 들어 있는 dBASE 파일(.dbf 확장자). 이 파일은 속성 컬럼이 실제로 익스포트되어야 하는 경우에만 작성됩니다.
- 좌표 시스템이 "UNSPECIFIED"와 동일하지 않은 경우 공간 데이터와 연관된 좌표 시스템을 지정하는 프로젝션 파일(.prj extension). 좌표 시스템은 처음 공간 레코드로부터 얻어집니다. 후속 레코드에 다른 좌표 시스템이 있는 경우 오류가 발생합니다.

다음 테이블에서는 DB2 데이터 유형이 dBASE 속성 파일에 저장되는 방법을 설명합니다. 기타 모든 DB2 데이터 유형은 지원되지 않습니다.

표 28. 속성 파일에 있는 DB2 데이터 유형의 스토리지

SQL 유형	**.dbf 유형**	**.dbf 길이**	**.dbf 10진수**	**주석**
SMALLINT	N	6	0	
INTEGER	N	11	0	
BIGINT	N	20	0	
DECIMAL	N	정밀도+2	스케일	
REAL FLOAT(1) through FLOAT(24)	F	14	6	
DOUBLE FLOAT(25) through FLOAT(53)	F	19	9	
CHARACTER, VARCHAR, LONG VARCHAR, and DATALINK	C	*len*	0	길이 ≤ 255
DATE	D	8	0	
TIME	C	8	0	
TIMESTAMP	C	26	0	

앞의 테이블에 나열되어 있는 유형을 따르는 데이터 유형과 구별 유형에 대한 모든 동의어가 지원됩니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 ST_export_shape 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 `/tmp/export_file`을 작성하고 이름 지정할 모양 파일에 CUSTOMERS 테이블의 모든 행을 익스포트합니다.

```
call db2gse.ST_export_shape('/tmp/export_file',0,NULL,
     'select * from customers','/tmp/export_msg',?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_import_shape

이 스토어드 프로시저를 사용하여 모양 파일을 공간 데이터 조작에 사용된 데이터베이스로 임포트하십시오.

스토어드 프로시저는 *create_table_flag* 매개변수에 기초한 다음 두 가지 방법으로 작동할 수 있습니다.

- DB2 Spatial Extender는 공간 컬럼 및 속성 컬럼이 있는 테이블을 작성할 수 있으며, 파일 데이터로 테이블의 컬럼을 로드할 수 있습니다.
- 그렇지 않으면, 모양 및 속성 데이터는 파일 데이터와 일치하는 공간 컬럼 및 속성 컬럼이 있는 기존 테이블에 로드될 수 있습니다.

이 스토어드 프로시저는 db2gse.gse_import_shape를 대체합니다.

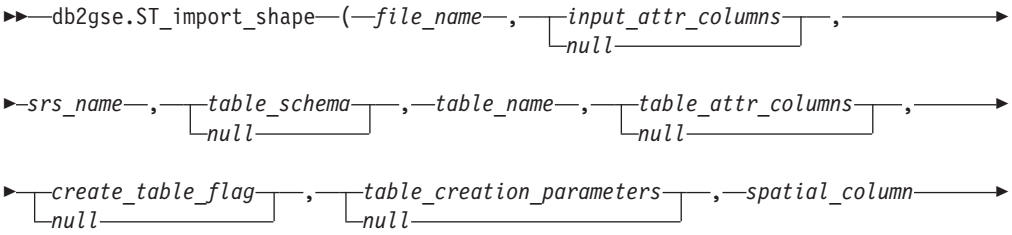
권한 부여

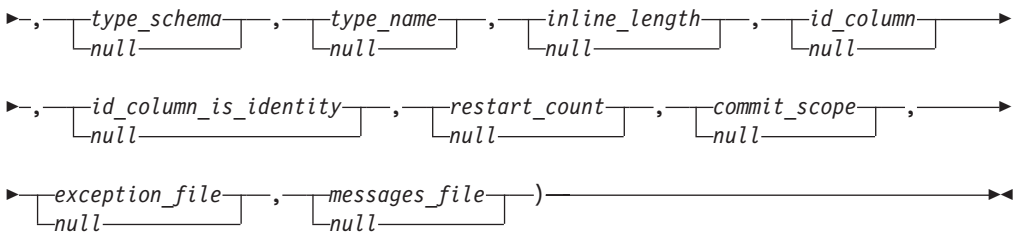
DB2 인스턴스의 소유자에게는 입력 파일을 읽고 선택적으로 오류 파일을 기록하는 데 필요한 서버 머신의 특권이 있어야 합니다. 추가 권한 부여 요구사항은 기존 테이블이나 새 테이블로 임포트할 것인지 여부에 따라 달라집니다.

- **기존 테이블로 임포트할 때**, 이 스토어드 프로시저를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.
  - SYSADM 또는 DBADM
  - 테이블 또는 뷰에 대한 CONTROL 특권
  - 테이블 또는 뷰에 대한 INSERT 및 SELECT 특권
- **새 테이블로 임포트할 때**, 이 스토어드 프로시저를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.
  - SYSADM 또는 DBADM
  - 데이터베이스의 CREATETAB 권한

  사용자 ID에는 다음 권한 중 하나가 있어야 합니다.
  - 테이블의 스키마 이름이 없는 경우 데이터베이스의 IMPLICIT_SCHEMA 권한
  - 테이블의 스키마가 존재하는 경우 스키마의 CREATEIN 특권

구문





매개변수 설명

file_name

임포트될 모양 파일의 전체 경로 이름을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

선택적 파일 확장자를 지정할 경우, .shp 또는 .SHP를 지정하십시오. DB2 Spatial Extender는 먼저 완전 일치하는 지정된 파일 이름을 찾습니다. DB2 Spatial Extender가 완전 일치하는 이름을 찾지 못하면, 먼저 확장자가 .shp인 파일을 찾은 후 확장자가 .SHP인 파일을 찾습니다.

서버 머신에 상주해야 하는 필수 파일 목록은 사용 참고를 참조하십시오. DB2 인스턴스 소유자가 소유하는 프로세스로 실행되는 스토어드 프로시저에는 파일을 읽는 데 필요한 서버의 특권이 있어야 합니다.

이 매개변수의 데이터 유형은 VARCHAR(256)입니다.

input_attr_columns

dBASE 파일에서 임포트할 속성 컬럼의 목록을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 모든 컬럼이 임포트됩니다. dBASE 파일이 없으면, 이 매개변수가 빈 문자열이거나 널(NULL)이어야 합니다.

이 매개변수에 대한 널(NULL)이 아닌 값을 지정하려면, 다음 권장 스펙 중 하나를 사용하십시오.

- **속성 컬럼 이름을 나열하십시오.** 다음 예에서는 dBASE 파일에서 임포트될 속성 컬럼의 이름 목록을 지정하는 방법을 보여줍니다.

  ```
  N(COLUMN1,COLUMN5,COLUMN3,COLUMN7)
  ```

  컬럼 이름이 큰따옴표로 묶여 있지 않으면, 대문자로 변환됩니다. 목록의 각 이름의 쉼표로 구분되어야 합니다. 결과 이름은 dBASE 파일의 컬럼 이름과 완전히 일치해야 합니다.

- **속성 컬럼 번호를 나열하십시오.** 다음 예에서는 dBASE 파일에서 임포트될 속성 컬럼의 수 목록을 지정하는 방법을 보여줍니다.

  ```
  P(1,5,3,7)
  ```

  컬럼은 1로 시작하여 번호가 지정됩니다. 목록의 각 번호는 쉼표로 구분되어야 합니다.

- **임포트될 속성 데이터가 없다는 것을 표시하십시오.** DB2 Spatial Extender가 임포트할 속성 데이터가 없음을 명시적으로 지정하는 빈 문자열에 ""을 지정하십시오.

이 매개변수의 데이터 유형은 VARCHAR(32K)입니다.

srs_name

공간 컬럼으로 임포트되는 Geometry에 사용될 공간 참조 시스템을 식별합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

공간 컬럼은 등록되지 않을 것입니다. 공간 참조 시스템(SRS)은 데이터가 임포트되기 전에 존재해야 합니다. 임포트 프로세스는 SRS를 암시적으로 작성하지 않으나, SRS의 좌표 시스템을 .prj 파일에 지정된 좌표 시스템과 비교합니다(모양 파일에 사용 가능한 경우). 임포트 프로세스는 모양 파일의 데이터 범위가 제공된 공간 참조 시스템에 표시될 수 있다는 것도 확인합니다. 즉, 임포트 프로세스는 범위가 SRS의 가능한 최소 및 최대 X, Y, Z 및 M 좌표 내에 있다는 것을 확인합니다.

srs_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

table_schema

table_name 매개변수에 지정된 테이블이 속해 있는 스키마 이름. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, CURRENT SCHEMA 특수 레지스터는 테이블 또는 뷰에 대한 스키마 이름으로 사용됩니다.

table_schema 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

table_name

임포트된 모양 파일이 로드될 테이블의 규정되지 않은 이름을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

table_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

table_attr_columns

dBASE 파일의 속성 데이터가 저장될 테이블 컬럼 이름을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, dBASE 파일의 컬럼 이름이 사용됩니다.

이 매개변수가 지정되면, 이름 수는 dBASE 파일에서 임포트된 컬럼 수와 일치해야 합니다. 테이블이 존재하면, 컬럼 정의는 들어오는 데이터와 일치해야 합니다. 속성 데이터 유형이 DB2 데이터 유형에 맵핑되는 방법에 대한 설명은 사용 참고를 참조하십시오.

이 매개변수의 데이터 유형은 VARCHAR(32K)입니다.

create_table_flag

임포트 프로세스가 새 테이블을 작성할지 여부를 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이거나 0이 아닌 다른 값이면, 새 테이블이 작성됩니다. (테이블이 이미 존재하면, 오류가 리턴됩니다.) 이 매개변수가 0이면, 테이블이 작성되지 않고 테이블이 이미 존재해야 합니다.

이 매개변수의 데이터 유형은 INTEGER입니다.

table_creation_parameters

데이터가 임포트될 테이블을 작성하는 CREATE TABLE문에 추가할 옵션을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, CREATE TABLE문에 옵션이 추가되지 않습니다.

CREATE TABLE 옵션을 지정하려면, DB2 CREATE TABLE문의 구문을 사용하십시오. 예를 들어, 테이블을 작성할 테이블 스페이스를 지정하려면, 다음을 지정하십시오.

```
IN tsName INDEX IN indexTsName LONG IN longTsName
```

이 매개변수의 데이터 유형은 VARCHAR(32K)입니다.

spatial_column

모양 데이터가 로드될 테이블에서 공간 컬럼의 이름. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

새 테이블의 경우, 이 매개변수는 작성될 새 공간 컬럼의 이름을 지정합니다. 그렇지 않으면, 이 매개변수는 테이블의 기존 공간 컬럼 이름을 지정합니다.

spatial_column 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

type_schema

새 테이블에 공간 컬럼 작성시 사용할 공간 데이터 유형의 스키마 이름(*type_name* 매개변수로 지정된)을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, DB2GSE 값이 사용됩니다.

type_schema 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

type_name

공간 값에 사용할 데이터 유형 이름을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 데이터 유형은 모양 파일로 판별되며 다음 유형 중 하나입니다.

- ST_Point
- ST_MultiPoint
- ST_MultiLineString
- ST_MultiPolygon

정의별로 모양 파일을 사용하여 점과 다중점 사이만 구별할 수 있으며, 다각형과 다중 다각형 사이 또는 선스트링과 다중 선스트링 사이는 구별할 수 없습니다.

아직 존재하지 않는 테이블에 임포트할 경우, 이 데이터 유형은 공간 컬럼의 데이터 유형에도 사용됩니다. 이 경우, 데이터 유형은 ST_Point, ST_MultiPoint, ST_MultiLineString 또는 ST_MultiPolygon의 슈퍼 유형일 수도 있습니다.

type_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

inline_length

새 테이블의 경우 테이블 내에서 공간 컬럼에 할당될 최대 바이트 수를 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, CREATE TABLE문에 명시적인 INLINE LENGTH 옵션이 사용되지 않으며 DB2 디폴트가 암시적으로 사용됩니다.

이 *크기*를 초과하는 공간 레코드는 더 느리게 액세스할 수도 있는 LOB 테이블 스페이스에 별도로 저장됩니다.

다양한 공간 유형에 필요한 일반적인 *크기*는 다음과 같습니다.

- **하나의 점:** 292.
- **다중점, 행 또는 다각형:** 가능한 큰 값으로. 한 행의 총 바이트 수가 테이블이 작성된 테이블 스페이스의 페이지 *크기* 한계를 초과해서는 안된다는 점을 고려하십시오.

이 값에 대한 완전한 설명은 CREATE TABLE SQL문에 대한 DB2 문서를 참조하십시오. 기존 테이블에 대한 inline Geometry 수와 inline length를 변경할 수 있는 능력을 판별하려면, db2dart 유틸리티도 참조하십시오.

이 매개변수의 데이터 유형은 INTEGER입니다.

id_column

각 데이터 행에 대한 고유 번호가 들어 있도록 작성할 컬럼을 이름 지정합니다. (ESRI 도구에는 SE_ROW_ID라는 컬럼이 필요합니다.) 해당 컬럼에 대한 고유 값은 임포트 프로세스시 자동으로 생성됩니다. 이 매개변수에 값을 지정해야 하더라도, 테이블에 컬럼(각 행에 고유 ID가 있는)이 없거나 새로 작성된 테이블에 그러한 컬럼을 추가하지 않을 경우 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 컬럼이 작성되지 않으며 고유 번호로 채워지지 않습니다.

제한사항: dBASE 파일의 컬럼 이름과 일치하는 *id_column* 이름을 지정할 수 없습니다.

이 매개변수의 요구사항 및 영향은 테이블이 이미 존재하는지 여부에 따라 달라집니다.

- **기존 테이블의 경우**, *id_column* 매개변수의 데이터 유형은 임의의 정수 유형(INTEGER, SMALLINT 또는 BIGINT)일 수 있습니다.
- **작성될 새 테이블의 경우**, 컬럼은 스토어드 프로시저가 컬럼을 작성할 때 테이블에 추가됩니다. 컬럼은 다음과 같이 정의됩니다.

  ```
  INTEGER NOT NULL PRIMARY KEY
  ```

  id_column_is_identity 매개변수의 값이 널(NULL)이 아니고 0이 아니면, 정의는 다음과 같이 확장됩니다.

  ```
  INTEGER NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY
  ( START WITH 1 INCREMENT BY 1 )
  ```

id_column 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

id_column_is_identity

지정된 *id_column*이 IDENTITY 절을 사용하여 작성될 것인지 여부를 나타냅니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 0 또는 널(NULL)이면, 컬럼이 식별 컬럼으로 작성되지 않습니다. 매개변수가 0 또는 널(NULL) 이외의 값이면, 컬럼은 식별 컬럼으로 작성됩니다. 이미 존재하는 테이블의 경우 이 매개변수는 무시됩니다.

이 매개변수의 데이터 유형은 SMALLINT입니다.

restart_count

임포트 조작이 레코드 *n* + 1에서 시작되도록 지정합니다. 처음 *n* 레코드를 건너뜁니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 모든 레코드(레코드 번호 1로 시작하는)가 임포트됩니다.

이 매개변수의 데이터 유형은 INTEGER입니다.

commit_scope

COMMIT가 최소한 *n* 레코드가 임포트된 후에 수행될 것임을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 0값이 사용되며 레코드가 커미트되지 않습니다.

이 매개변수의 데이터 유형은 INTEGER입니다.

exception_file

임포트될 수 없는 모양 파일이 저장되는 모양 파일의 전체 경로 이름을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 파일이 작성되지 않습니다.

매개변수에 대해 값을 지정하고 선택적 파일 확장자를 포함하면, .shp 또는 .SHP를 지정하십시오. 확장자가 널(NULL)이면, 확장자 .shp가 추가됩니다.

예외 파일은 하나의 INSERT문이 실패한 행의 전체 블록을 보류합니다. 예를 들면, 모양 데이터가 잘못 인코드되었으므로 한 행을 임포트할 수 없다고 가정합시다. 하나의 INSERT문에서 오류가 있는 행을 포함하여 20개의 행을 임포트하려고 합니다. 하나의 행에 있는 문제점으로 인해, 20개 행의 전체 블록이 예외 파일에 기록됩니다.

레코드는 모양 레코드 유형이 유효하지 않은 경우에서와 마찬가지로 그러한 레코드가 올바로 식별될 수 있는 경우에만 예외 파일에 기록됩니다. 모양 데이터(.shp 파일) 및 모양 인덱스(.shx 파일)에 대한 일부 손상 유형으로 해당 레코드가 식별될 수 없습니다. 이 경우, 레코드가 예외 파일에 기록되지 않으며, 문제점을 보고하기 위해 오류 메시지가 발행됩니다.

이 매개변수에 값을 지정하면, 네 개의 파일이 서버 머신에 작성됩니다. 이러한 파일에 대한 설명은 사용 참고를 참조하십시오. DB2 인스턴스 소유자가 소유하는 프로세스로 실행되는 스토어드 프로시저에는 파일을 작성하는 데 필요한 서버의 특권이 있어야 합니다. 파일이 이미 존재하면, 스토어드 프로시저가 오류를 리턴합니다.

이 매개변수의 데이터 유형은 VARCHAR(256)입니다.

messages_file

임포트 조작에 대한 메시지를 포함할 파일의 전체 경로 이름(서버 머신의)을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, DB2 Spatial Extender 메시지에 대한 파일이 작성되지 않습니다.

메시지 파일에 기록된 메시지는 다음과 같을 수 있습니다.

- 임포트 조작 요약과 같은 정보 메시지
- 임포트될 수 없는 데이터에 대한 오류 메시지(예: 다른 좌표 시스템으로 인해)

이러한 메시지는 예외 파일에 저장된 모양 데이터에 해당됩니다(*exception_file* 매개변수로 식별되는).

DB2 인스턴스 소유자가 소유하는 프로세스로 실행되는 스토어드 프로시저에는 파일을 작성하는 데 필요한 서버의 특권이 있어야 합니다. 파일이 이미 존재하면, 스토어드 프로시저가 오류를 리턴합니다.

이 매개변수의 데이터 유형은 VARCHAR(256)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

사용 참고

ST_import_shape 스토어드 프로시저는 한 파일에서 네 개 파일까지 사용합니다.

- 기본 모양 파일(.shp 확장자). 이 파일은 필수입니다.
- 모양 인덱스 파일(.shx 확장자). 이 파일은 선택적입니다. 이 파일이 존재하면, 임포트 조작의 성능이 향상될 수 있습니다.
- 속성 데이터가 들어 있는 dBASE 파일(.dbf 확장자). 이 파일은 속성 데이터가 임포트될 경우에만 필수입니다.
- 모양 데이터의 좌표 시스템을 지정하는 프로젝션 파일(.proj 확장자). 이 파일은 선택적입니다. 이 파일이 존재하면, 정의된 좌표 시스템이 *srs_id* 매개변수에 지정된 공간 참조 시스템의 좌표 시스템과 비교됩니다.

다음 테이블은 dBASE 속성 데이터 유형이 DB2 데이터 유형에 맵핑되는 방법에 대해 설명합니다. 기타 모든 속성 데이터 유형은 지원되지 않습니다.

표 29. DB2 데이터 유형과 dBASE 속성 데이터 유형간의 관계

.dbf 유형	**.dbf 길이(주 참고)**	**.dbf 10진수(주 참고)**	**SQL 유형**	**주석**
N	< 5	0	SMALLINT	
N	< 10	0	INTEGER	
N	< 20	0	BIGINT	
N	*len*	*dec*	DECIMAL(*len*,*dec*)	*len*<32
F	*len*	*dec*	REAL	*len* + *dec* < 7
F	*len*	*dec*	DOUBLE	

표 29. DB2 데이터 유형과 dBASE 속성 데이터 유형간의 관계 (계속)

.dbf 유형	.dbf 길이(주 참고)	.dbf 10진수(주 참고)	SQL 유형	주석
C	*len*		CHAR(*len*)	
L			CHAR(1)	
D			DATE	

주: 이 테이블에는 dBASE 파일의 헤더에 정의된 다음 변수가 둘 모두 포함됩니다.

- dBASE 파일에서 총 컬럼의 길이를 표시하는 *len*. DB2 Spatial Extender는 다음의 두 가지 목적으로 이 값을 사용합니다.
  - SQL 데이터 유형 DECIMAL의 정밀도와 SQL 데이터 유형 CHAR의 길이를 정의하기 위해
  - 사용될 정수 또는 부동 소수점 유형을 판별하기 위해
- dBASE 파일에서 컬럼에서 소수점 오른쪽 최대 자릿수를 표시하는 *dec*. DB2 Spatial Extender는 이 값을 사용하여 SQL 데이터 유형 DECIMAL의 스케일을 정의합니다.

예를 들어, dBASE 파일에 길이(*len*)가 20으로 정의된 데이터의 컬럼이 들어 있다고 가정합시다. 소수점 오른쪽 자릿수(*dec*)가 5로 정의된다고 가정합시다. DB2 Spatial Extender가 해당 컬럼에서 데이터를 임포트할 때, *len* 및 *dec* 값을 사용하여 SQL 데이터 유형 DECIMAL(20,5)를 가져옵니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 ST_import_shape 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 다음 DB2 CALL 명령을 사용하여 /tmp/officesShape라는 모양 파일을 OFFICES라는 테이블로 임포트합니다.

```
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',?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_register_geocoder

이 스토어드 프로시저를 사용하여 DB2 Spatial Extender와 함께 제공된 DB2SE_USA_GEOCODER 지오코더(Geocoder) 이외의 다른 지오코더(Geocoder)를 등록하십시오. DB2SE_USA_GEOCODER 지오코더(Geocoder)는 데이터베이스 사용 가능시 DB2 Spatial Extender가 등록합니다.

전제조건: 지오코더를 등록하기 전에 다음을 수행하십시오.

- 지오코더(Geocoder)를 구현하는 함수가 이미 작성되었는지 확인하십시오. 각 지오코더(Geocoder) 함수가 지오코더(Geocoder) 이름을 고유하게 식별하는 지오코더(Geocoder)로 등록될 수 있습니다.
- 지오코더(Geocoder) 벤더로부터 다음과 같은 정보를 확보하십시오.
  - 함수를 작성하는 SQL문
  - Geometry 데이터를 지원할 수 있도록 ST_create_srs 매개변수에 사용할 값
  - 다음과 같은 지오코더(Geocoder) 등록을 위한 정보.
    - 지오코더(Geocoder) 설명
    - 지오코더(Geocoder)용 매개변수 설명
    - 지오코더(Geocoder) 매개변수의 디폴트값

지오코더(Geocoder) 함수의 리턴 유형은 지역코드화된 컬럼의 데이터 유형과 일치해야 합니다. 지오코딩(Geocoding) 매개변수는 지오코더가 필요로 하는 데이터가 들어 있는 컬럼 이름(*지오코딩 컬럼*이라고 함)이 될 수 있습니다. 예를 들어, 지오코더(Geocoder) 매개변수는 주소나 최소 일치 스코어와 같은 지오코더(Geocoder)에 대한 특정 의미 값을 식별할 수 있습니다. 지오코딩(Geocoding) 매개변수가 컬럼 이름이면, 컬럼은 지역코드화된 컬럼과 동일한 테이블 또는 뷰에 있어야 합니다.

지오코더(Geocoder) 함수의 리턴 유형은 지역코드화된 컬럼의 데이터 유형으로 제공됩니다. 리턴 유형은 DB2 데이터 유형, 사용자 정의 유형 또는 구조화된 유형일 수 있습니다. 사용자 정의 유형 또는 구조화된 유형이 리턴되면, 지오코더(Geocoder) 함수는 각각의 데이터 유형의 올바른 값을 리턴해야 합니다. 지오코더(Geocoder) 함수가 ST_Geometry 또는 해당 부속 유형 중 하나인 공간 유형의 값을 리턴하면, 지오코더(Geocoder) 함수는 유효한 Geometry를 구성해야 합니다. Geometry는 기존 공간 참조 시스템을 사용하여 표시되어야 합니다. Geometry는 Geometry에서 ST_IsValid 공간 함수를 호출하여 1값이 리턴되는 경우 유효합니다. 지오코더(Geocoder) 함수에서 리턴된 데이터는 지역코드화된 값이 생성되는 조작(INSERT 또는 UPDATE)에 따라 지역코드화된 컬럼에서 갱신되거나 이 컬럼에 삽입됩니다.

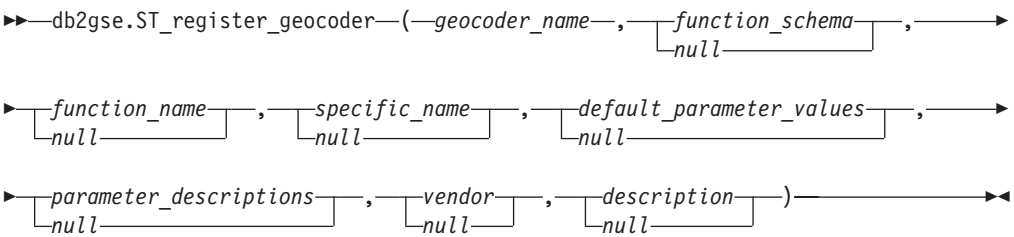
지오코더(Geocoder)가 이미 등록되었는지 여부를 알려면, DB2GSE.ST_GEOCODERS 카탈로그 뷰를 검사하십시오.

이 스토어드 프로시저는 db2gse.gse_register_gc를 대체합니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 이 스토어드 프로시저가 등록하는 지오코더(Geocoder)가 들어 있는 데이터베이스에 대해 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다.

구문



매개변수 설명

geocoder_name

지오코더(Geocoder)를 고유하게 식별합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

geocoder_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

function_schema

이 지오코더(Geocoder)를 구현하는 함수의 스키마 이름을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, CURRENT SCHEMA 특수 레지스터의 값이 함수에 대한 스키마 이름으로 사용됩니다.

function_schema 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

function_name

지오코더(Geocoder)를 구현하는 함수의 규정되지 않은 이름을 지정합니다. 함수는 이미 작성되어 SYSCAT.ROUTINES에 나열되어 있어야 합니다.

이 매개변수의 경우, *specific_name* 매개변수가 지정되면 널(NULL)을 지정할 수 있습니다. *specific_name* 매개변수가 지정되지 않으면, 암시적으로 또는 명시적으로 정의된 *function_schema* 값과 함께 *function_name* 값은 함수를 고유하게 식별해야 합니다. *function_name* 매개변수가 지정되지 않으면, DB2 Spatial Extender는 SYSCAT.ROUTINES 카탈로그 뷰에서 *function_name* 값을 검색합니다.

function_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

specific_name

지오코더(Geocoder)를 구현하는 함수의 특정 이름을 식별합니다. 함수는 이미 작성되어 SYSCAT.ROUTINES에 나열되어 있어야 합니다.

이 매개변수의 경우, *function_name* 매개변수가 지정되고 *function_schema* 및 *function_name*의 조합이 지오코더 함수를 고유하게 식별하는 경우 널(NULL)을 지정할 수 있습니다. 지오코더(Geocoder) 함수 이름이 과부하되면, *specific_name* 매개변수는 널(NULL)일 수 없습니다. (함수 이름에 동일한 이름이 있으나 하나 이상의 다른 함수와 동일한 매개변수 또는 매개변수 데이터 유형이 없는 경우 함수 이름은 *과부하*됩니다.)

specific_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

default_parameter_values

지오코더(Geocoder) 함수에 대한 디폴트 지오코딩(Geocoding) 매개변수 값의 목록을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 전체 *default_parameter_values* 매개변수가 널(NULL)이면, 모든 매개변수 디폴트 값은 널(NULL)입니다.

매개변수 값을 지정할 경우 함수가 이들을 정의한 순서로 지정한 후 쉼표로 구분하십시오. 예를 들면, 다음과 같습니다.

```
default_parm1_value,default_parm2_value,...
```

각각의 매개변수는 SQL 표현식입니다. 다음 지침을 따르십시오.

- 값이 문자열이면, 작은따옴표로 묶으십시오.
- 매개변수 값이 숫자이면, 작은따옴표로 묶지 마십시오.
- 매개변수 값이 널(NULL)이면, 올바른 유형으로 캐스트하십시오. 예를 들면, 널(NULL)을 지정하는 대신에 다음을 지정하십시오.

  ```
  CAST(NULL AS INTEGER)
  ```

- 지오코딩(Geocoding) 매개변수가 지오코딩(Geocoding) 컬럼이 될 경우, 디폴트 매개변수 값을 지정하지 마십시오.

매개변수 값이 지정되지 않으면(즉, 두 개의 연속적인 쉼표(`...,,...`)를 지정하는 경우), 이 매개변수는 지오코딩이 설정되거나 지오코딩이 각각의 스토어드 프로시저의 *parameter_values* 매개변수를 사용하여 일괄처리 모드로 실행될 때 지정되어야 합니다.

이 매개변수의 데이터 유형은 VARCHAR(32K)입니다.

parameter_descriptions

지오코더(Geocoder) 함수에 대한 지오코딩(Geocoding) 매개변수 설명 목록을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다.

전체 *parameter_descriptions* 매개변수가 널(NULL)이면, 모든 매개변수 설명이 널(NULL)입니다. 지정한 각 매개변수 설명은 매개변수의 의미와 사용을 설명하며 최

대 256자까지 가능합니다. 매개변수 설명은 쉼표로 구분되어야 하며, 함수에 의해 정의된 대로 매개변수 순으로 나타나야 합니다. 쉼표가 매개변수 설명 내에서 사용될 경우, 문자열을 작은따옴표나 큰따옴표로 묶으십시오. 예를 들면, 다음과 같습니다.

```
description,'description2, which contains a comma',description3
```

이 매개변수의 데이터 유형은 VARCHAR(32K)입니다.

vendor

지오코더(Geocoder)를 구현한 벤더 이름. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 지오코더(Geocoder)를 구현한 벤더에 대한 정보가 기록되지 않습니다.

이 매개변수의 데이터 유형은 VARCHAR(128)입니다.

description

해당 응용프로그램을 설명하는 지오코더(Geocoder)의 설명. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 지오코더(Geocoder)에 대한 설명 정보가 기록되지 않습니다.

권장사항: 다음 정보를 포함시키십시오.

- Well-Known Text 또는 Well-Known Binary와 같은 공간 데이터가 리턴될 경우 좌표 시스템 이름
- ST_Geometry 또는 해당 부속 유형이 리턴될 경우 공간 참조 시스템
- 이 지오코더(Geocoder)가 적용되는 지리적인 영역의 이름
- 사용자가 알아야 하는 지오코더(Geocoder)에 대한 또 다른 정보

이 매개변수의 데이터 유형은 VARCHAR(256)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 입력으로 위도와 경도를 가져오는 지오코더(Geocoder)를 작성하여 ST_Point 공간 데이터로 지오코딩(Geocoding)하고자 한다고 가정합니다. 이렇게 하려면, 먼저 lat_long_gc_func라는 함수를 작성합니다. 그런 후 함수 lat_long_gc_func를 사용하는 SAMPLEGC라는 지오코더(Geocoder)를 등록합니다.

다음은 ST_Point를 리턴하는 함수 lat_long_gc_func를 작성하는 SQL문의 예입니다.

```
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)
```

함수를 작성한 후에, 지오코더(Geocoder)로 등록할 수 있습니다. 이 예에서는 DB2 명령행 처리기 CALL 명령을 사용하여 함수 lat_long_gc_func를 사용하여 SAMPLEGC라는 지오코더(Geocoder)를 등록할 ST_register_geocoder 스토어드 프로시저를 호출하는 방법을 보여줍니다.

```
call db2gse.ST_register_geocoder ('SAMPLEGC',NULL,'LAT_LONG_GC_FUNC',',,1'
            ,NULL,'My Company','Latitude/Longitude to
            ST_Point Geocoder'?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_register_spatial_column

이 스토어드 프로시저를 사용하여 공간 컬럼을 등록하고 공간 참조 시스템(SRS)을 여기에 연관시키십시오.

이 스토어드 프로시저가 처리될 때 등록 중인 공간 컬럼에 대한 정보가 DB2GSE.ST_GEOMETRY_COLUMNS 카탈로그 뷰에 추가됩니다. 공간 컬럼을 등록하면 테이블에 제한조건이 작성되어 가능한 경우 모든 Geometry가 지정된 SRS를 사용하는지 확인합니다.

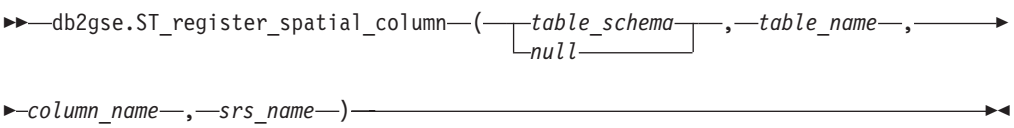
이 스토어드 프로시저는 db2gse.gse_register_ layer를 대체합니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.

- 등록되고 있는 공간 컬럼이 속하는 테이블이 들어 있는 데이터베이스의 SYSADM 또는 DBADM 권한
- 이 테이블에 대한 CONTROL 또는 ALTER 특권

구문



매개변수 설명

table_schema

table_name 매개변수에 지정된 테이블 또는 뷰가 속해 있는 스키마의 이름. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, CURRENT SCHEMA 특수 레지스터는 테이블 또는 뷰에 대한 스키마 이름으로 사용됩니다.

table_schema 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

table_name

등록될 컬럼이 들어있는 테이블 또는 뷰의 규정되지 않은 이름을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

table_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

column_name

등록될 컬럼의 이름. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

column_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

srs_name

이 공간 컬럼에 사용될 공간 참조 시스템의 이름을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

srs_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값

이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 ST_register_spatial_column 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 CUSTOMERS라는 테이블에서 LOCATION으로 이름 지정된 공간 컬럼을 등록합니다. 이 CALL 명령은 *srs_name* 매개변수 값을 USA_SRS_1로 지정합니다.

```
call db2gse.ST_register_spatial_column(NULL,'CUSTOMERS','LOCATION',
     'USA_SRS_1',?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_remove_geocoding_setup

이 스토어드 프로시저를 사용하여 지역코드화된 컬럼에 대한 모든 지오코딩(Geocoding) 설정 정보를 제거하십시오.

이 스토어드 프로시저는 DB2GSE.ST_GEOCODING and DB2GSE.ST_GEOCODING_PARAMETERS 카탈로그 뷰에서 지정된 지역코드화된 컬럼과 연관된 정보를 제거합니다.

제한사항:

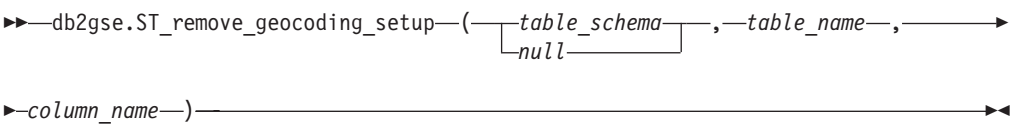
autogeocoding을 지역코드화된 컬럼에 사용 가능하면 지오코딩(Geocoding) 설정을 제거할 수 없습니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.

- 지정된 지오코더(Geocoder)가 운영되는 테이블을 포함하는 데이터베이스의 SYSADM 또는 DBADM 권한.
- 이 테이블에 대한 CONTROL 또는 UPDATE 특권

구문



매개변수 설명

table_schema

table_name 매개변수에 지정된 테이블 또는 뷰가 속해 있는 스키마의 이름. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, CURRENT SCHEMA 특수 레지스터는 테이블 또는 뷰에 대한 스키마 이름으로 사용됩니다.

table_schema 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

table_name

지역코드화된 데이터가 삽입되거나 갱신될 컬럼이 들어 있는 테이블 또는 뷰의 규정되지 않은 이름을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

table_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

column_name

지역코드화된 데이터가 삽입되거나 갱신될 컬럼 이름. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

column_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 ST_remove_geocoding_setup 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 CUSTOMER라는 테이블과 LOCATION이라는 컬럼에 대한 지오코딩(Geocoding) 설정을 제거합니다.

```
call db2gse.ST_remove_geocoding_setup(NULL, 'CUSTOMERS', 'LOCATION',?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_run_geocoding

이 스토어드 프로시저를 사용하여 지오코드 컬럼에서 지오코더(Geocoder)를 일괄처리 모드로 실행하십시오.

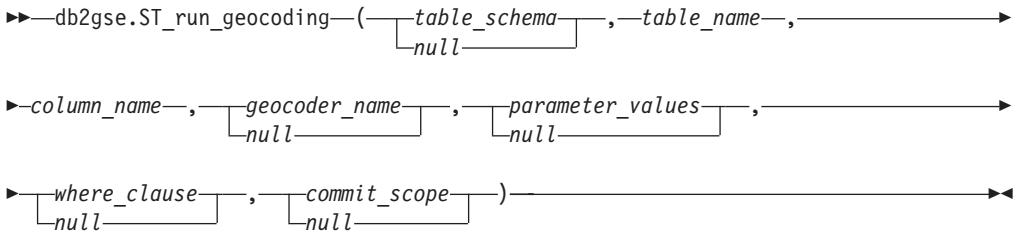
이 스토어드 프로시저는 db2gse.gse_enable_autogc를 대체합니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.

- 지정된 지오코더(Geocoder)가 운영되는 테이블을 포함하는 데이터베이스의 SYSADM 또는 DBADM 권한.
- 이 테이블에 대한 CONTROL 또는 UPDATE 특권

구문



매개변수 설명

table_schema

table_name 매개변수에 지정된 테이블 또는 뷰가 속해 있는 스키마의 이름. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, CURRENT SCHEMA 특수 레지스터는 테이블 또는 뷰에 대한 스키마 이름으로 사용됩니다.

table_schema 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

table_name

지역코드화된 데이터가 삽입되거나 갱신될 컬럼이 들어 있는 테이블 또는 뷰의 규정되지 않은 이름을 지정합니다. 뷰 이름이 지정되면, 뷰는 갱신 가능한 뷰여야 합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

table_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

column_name

지역코드화된 데이터가 삽입되거나 갱신될 컬럼 이름. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

column_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

geocoder_name

지오코딩(Geocoding)을 수행하는 지오코더(Geocoder) 이름. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 지오코딩(Geocoding)이 설정되었을 때 지정된 지오코더(Geocoder)가 지오코딩(Geocoding)을 수행합니다.

geocoder_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

parameter_values

지오코더(Geocoder) 함수에 대한 지오코딩(Geocoding) 매개변수 값 목록을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 전체 *parameter_values* 매개변수가 널(NULL)이면, 사용되는 값은 지오코더가 설정되었을 때 지정된 매개변수 값이거나 지오코더가 설정되지 않은 경우 지오코더에 대한 디폴트 매개변수 값입니다.

매개변수 값을 지정할 경우 함수가 이들을 정의한 순서로 지정한 후 쉼표로 구분하십시오. 예를 들면, 다음과 같습니다.

```
parameter1-value,parameter2-value,...
```

각 매개변수 값은 컬럼 이름, 문자열, 숫자 값 또는 널(NULL)일 수 있습니다.

각각의 매개변수는 SQL 표현식입니다. 다음 지침을 따르십시오.

- 매개변수 값이 지오코딩(Geocoding) 컬럼 이름이면, 컬럼은 지역코드화된 컬럼이 상주하는 동일한 테이블 또는 뷰에 있어야 합니다.
- 매개변수 값이 문자열이면, 작은따옴표로 묶으십시오.
- 매개변수 값이 숫자이면, 작은따옴표로 묶지 마십시오.
- 매개변수 값이 널(NULL)이면, 올바른 유형으로 캐스트하십시오. 예를 들면, 널(NULL)을 지정하는 대신에 다음을 지정하십시오.

  ```
  CAST(NULL AS INTEGER)
  ```

매개변수 값이 지정되지 않으면(즉, 두 개의 연속적인 쉼표(`...,,...`)를 지정하는 경우), 이 매개변수는 지오코딩이 설정되거나 지오코딩이 각각의 스토어드 프로시저의 *parameter_values* 매개변수를 사용하여 일괄처리 모드로 실행될 때 지정되어야 합니다.

이 매개변수의 데이터 유형은 VARCHAR(32K)입니다.

where_clause

지오코딩(Geocoding)될 레코드 세트에 대한 제한사항을 정의하는 WHERE 절의 본문을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다.

where_clause 매개변수가 널(NULL)이면, 결과 동작은 스토어드 프로시저가 실행되기 전에 컬럼(*column_name* 매개변수에 지정된)에 지오코딩이 설정되었는지 여부에 따라 달라집니다. *where_clause* 매개변수가 널(NULL)이고,

- 지오코딩이 설정되었을 때 값이 지정되었으면, 해당 값이 *where_clause* 매개변수에 사용됩니다.
- 지오코딩(Geocoding)이 설정되지 않았거나 지오코딩(Geocoding)이 설정되었을 때 값이 지정되지 않았으면, Where절이 사용되지 않습니다.

지오코더(Geocoder)가 작동될 테이블 또는 뷰에 있는 모든 컬럼을 참조하는 절을 지정할 수 있습니다. 키워드 WHERE는 지정하지 마십시오.

이 매개변수의 데이터 유형은 VARCHAR(32K)입니다.

commit_scope

지오코딩된 모든 *n* 레코드 다음에 COMMT가 수행될 것임을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다.

commit_scope 매개변수가 널(NULL)이면, 결과 동작은 스토어드 프로시저가 실행되기 전에 컬럼(*column_name* 매개변수에 지정된)에 지오코딩이 설정되었는지 여부에 따라 달라집니다. *commit_scope* 매개변수가 널(NULL)이고,

- 컬럼에 지오코딩이 설정되었을 때 값이 지정되었으면, 해당 값이 *commit_scope* 매개변수에 사용됩니다.
- 지오코딩(Geocoding)이 설정되지 않았거나 설정되었으나 값이 지정되지 않았으면, 디폴트 값 0이 사용되고 COMMIT가 수행되지 않습니다.

이 매개변수의 데이터 유형은 INTEGER입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 ST_run_geocoding 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 CUSTOMER라는 테이블의 LOCATION 컬럼을 지오코딩(Geocoding)합니다. 이 CALL 명령은 *geocoder_name* 매개변수 값을 DB2SE_USA_GEOCODER로, *commit_scope* 매개변수 값을 10으로 지정합니다. COMMIT는 매 10개의 레코드가 지역코드화된 후에 수행됩니다.

```
call db2gse.ST_run_geocoding(NULL, 'CUSTOMERS', 'LOCATION',
     'DB2SE_USA_GEOCODER',NULL,NULL,10,?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_setup_geocoding

지오코딩(Geocoding)될 컬럼을 지오코더(Geocoder)와 연관시키고 해당 지오코딩 매개변수를 설정하려면, 이 스토어드 프로시저를 사용하십시오. 여기에 설정된 정보는 DB2GSE.ST_GEOCODING 및 DB2GSE.ST_GEOCODING_PARAMETERS 카탈로그 뷰에 기록됩니다.

이 스토어드 프로시저는 지오코딩(Geocoding)을 호출하지 않습니다. 지오코딩(Geocoding)될 컬럼에 매개변수 설정값을 지정하는 방법을 제공합니다. 이러한 설정값으로 후속적인 일괄처리 지오코딩(Geocoding) 또는 자동 지오코딩(Geocoding)의 호출이 더 단순한 인터페이스로 실행될 수 있습니다. 이 설정 단계에 지정된 매개변수 설정값은 지오코더(Geocoder)가 등록되었을 때 지정된 지오코더(Geocoder)의 디폴트 매개변수 값 위에 겹쳐씁니다. 또한, ST_run_geocoding 스토어드 프로시저를 일괄처리 모드로 실행하여 이러한 매개변수 설정 위에 겹쳐쓸 수도 있습니다.

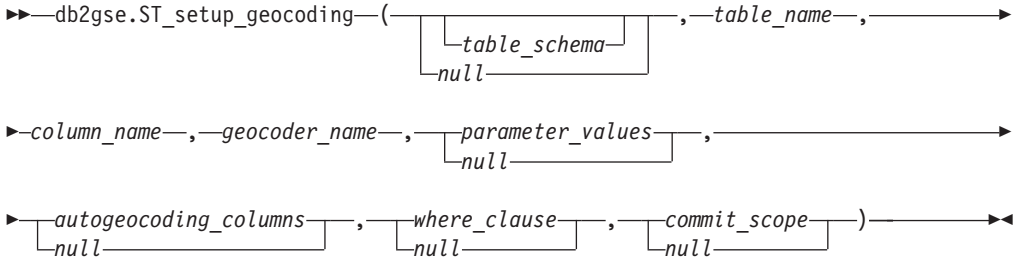
이 단계는 자동 지오코딩(Geocoding)의 전제조건입니다. 처음에 지오코딩(Geocoding) 매개변수를 설정하지 않고는 자동 지오코딩(Geocoding)을 사용할 수 없습니다. 이 단계는 일괄처리 지오코딩(Geocoding)의 전제조건이 아닙니다. 설정 단계를 수행하거나 수행하지 않고 일괄처리 모드로 지오코딩(Geocoding)을 실행할 수 있습니다. 그러나 설정 단계가 일괄처리 지오코딩(Geocoding) 이전에 수행되면, 매개변수 값은 런타임시 지정되지 않은 경우 설정시 얻어집니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.

- 지정된 지오코더(Geocoder)가 운영되는 테이블을 포함하는 데이터베이스의 SYSADM 또는 DBADM 권한.
- 이 테이블에 대한 CONTROL 또는 UPDATE 특권

구문



매개변수 설명

table_schema

table_name 매개변수에 지정된 테이블 또는 뷰가 속해 있는 스키마의 이름. 이 매

개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, CURRENT SCHEMA 특수 레지스터는 테이블 또는 뷰에 대한 스키마 이름으로 사용됩니다.

table_schema 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

table_name

지역코드화된 데이터가 삽입되거나 갱신될 컬럼이 들어 있는 테이블 또는 뷰의 규정되지 않은 이름을 지정합니다. 뷰 이름이 지정되면, 뷰는 갱신 가능해야 합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

table_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

column_name

지역코드화된 데이터가 삽입되거나 갱신될 컬럼 이름. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

column_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

geocoder_name

지오코딩(Geocoding)을 수행하는 지오코더(Geocoder) 이름. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

geocoder_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

parameter_values

지오코더(Geocoder) 함수에 대한 지오코딩(Geocoding) 매개변수 값 목록을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 전체 *parameter_values* 매개변수가 널(NULL)이면, 사용되는 값은 지오코더가 등록되었을 때 디폴트 매개변수에서 가져옵니다.

매개변수 값을 지정할 경우, 함수가 이들을 정의한 순서로 지정한 후 쉼표로 구분하십시오. 예를 들면, 다음과 같습니다.

```
parameter1-value,parameter2-value,...
```

각 매개변수 값은 SQL 표현식이며, 컬럼 이름, 문자열, 숫자값 또는 널(NULL)일 수 있습니다. 다음 지침을 따르십시오.

- 매개변수 값이 지오코딩(Geocoding) 컬럼 이름이면, 컬럼은 지역코드화된 컬럼이 상주하는 동일한 테이블 또는 뷰에 있어야 합니다.
- 매개변수 값이 문자열이면, 작은따옴표로 묶으십시오.
- 매개변수 값이 숫자이면, 작은따옴표로 묶지 마십시오.
- 매개변수 값이 널(NULL) 값으로 지정되면, 올바른 유형으로 캐스트하십시오. 예를 들면, 널(NULL)을 지정하는 대신에 다음을 지정하십시오.

  ```
  CAST(NULL AS INTEGER)
  ```

매개변수 값이 지정되지 않으면(즉, 두 개의 연속적인 쉼표(`...,,...`)를 지정하는 경우), 이 매개변수는 지오코딩이 설정되거나 지오코딩이 각각의 스토어드 프로시저의 *parameter_values* 매개변수를 사용하여 일괄처리 모드로 실행될 때 지정되어야 합니다.

이 매개변수의 데이터 유형은 VARCHAR(32K)입니다.

autogeocoding_columns

트리거가 작성될 컬럼 이름의 목록을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이고 자동 지오코딩(Geocoding)이 사용 가능하면, 테이블의 컬럼 갱신으로 인해 트리거가 활성화됩니다.

autogeocoding_columns 매개변수에 대해 값을 지정하면, 임의의 순서로 컬럼 이름을 지정한 후 컬럼 이름을 쉼표로 구분하십시오. 컬럼 이름은 지역코드화된 컬럼이 상주하는 동일한 테이블에 있어야 합니다.

이 매개변수 설정은 후속 자동 지오코딩(Geocoding)에만 적용됩니다.

이 매개변수의 데이터 유형은 VARCHAR(32K)입니다.

where_clause

지오코딩(Geocoding)될 레코드 세트에 대한 제한사항을 정의하는 WHERE 절의 본문을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, WHERE 절에 제한사항이 정의되지 않습니다.

절에서는 지오코더(Geocoder)가 운용될 테이블 또는 뷰에 있는 모든 컬럼을 참조할 수 있습니다. 키워드 WHERE는 지정하지 마십시오.

이 매개변수 설정은 후속 일괄처리 모드 지오코딩(Geocoding)에만 적용됩니다.

이 매개변수의 데이터 유형은 VARCHAR(32K)입니다.

commit_scope

지오코딩된 매 *n* 레코드마다 COMMIT가 수행될 것임을 지정합니다. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, 모든 레코드가 지역코드화된 후에 COMMIT가 수행됩니다.

이 매개변수 설정은 후속 일괄처리 모드 지오코딩(Geocoding)에만 적용됩니다.

이 매개변수의 데이터 유형은 INTEGER입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 ST_setup_geocoding 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 CUSTOMER라는 테이블에서 LOCATION으로 이름 지정된 지역코드화된 컬럼에 대해 지오코딩(Geocoding) 프로세스를 설정합니다. 이 CALL 명령은 *geocoder_name* 매개변수 값을 DB2SE_USA_GEOCODER로 지정합니다.

```
call db2gse.ST_setup_geocoding(NULL, 'CUSTOMERS', 'LOCATION',
        'DB2SE_USA_GEOCODER','ADDRESS,CITY,STATE,ZIP,1,100,80,,,,"$HOME/sqllib/
     gse/refdata/ky.edg","$HOME/sqllib/samples/extenders/spatial/EDGESample.loc"',
          'ADDRESS,CITY,STATE,ZIP',NULL,10,?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_unregister_geocoder

이 스토어드 프로시저를 사용하여 DB2 Spatial Extender와 함께 제공된 DB2SE_USA_GEOCODER 지오코더(Geocoder) 이외의 다른 지오코더(Geocoder)를 등록 해제하십시오.

제한사항:

컬럼에 대한 지오코딩(Geocoding) 설정에서 지정된 경우 지오코더(Geocoder)를 등록 해제할 수 없습니다.

지오코더(Geocoder)가 컬럼에 대해 지오코딩(Geocoding) 설정에 지정되었는지 여부를 판별하려면, DB2GSE.ST_GEOCODING 및 DB2GSE.ST_GEOCODING_PARAMETERS 카탈로그 뷰를 점검하십시오. 등록 해제하려는 지오코더(Geocoder)에 대한 정보는 DB2GSE.ST_GEOCODERS 카탈로그 뷰를 보십시오.

이 스토어드 프로시저는 db2gse.gse_unregist_gc를 대체합니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 등록 해제될 지오코더(Geocoder)가 들어 있는 데이터베이스에 대해 SYSADM 또는 DBADM 권한을 가지고 있어야 합니다.

구문

매개변수 설명

geocoder_name

지오코더(Geocoder)를 고유하게 식별합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

geocoder_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 ST_unregister_geocoder 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 SAMPLEGC라는 지오코더(Geocoder)를 등록 해제합니다.

```
call db2gse.ST_unregister_geocoder('SAMPLEGC',?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

ST_unregister_spatial_ column

이 스토어드 프로시저를 사용하여 공간 컬럼의 등록을 제거하십시오.

스토어드 프로시저는 다음을 수행하여 등록을 제거합니다.

- 공간 컬럼과 공간 참조 시스템의 연관 제거. ST_GEOMETRY_COLUMNS 카탈로그 뷰는 공간 컬럼을 계속 포함하나 컬럼은 더 이상 공간 참조 시스템과 연관되지 않습니다.
- 기본 테이블의 경우, 이 공간 컬럼의 Geometry 값이 동일한 공간 참조 시스템에 모두 표시되도록 DB2 Spatial Extender가 이 테이블에 배치한 제한조건 삭제

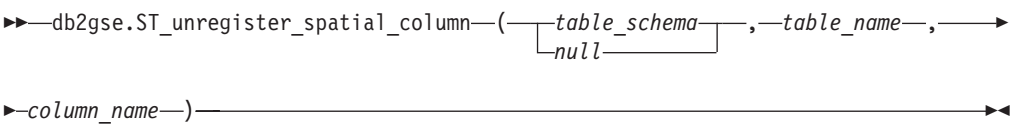
이 스토어드 프로시저는 db2gse.gse_unregist_layer를 대체합니다.

권한 부여

이 스토어드 프로시저를 호출하는 사용자 ID는 다음의 권한이나 특권 중 하나를 보유하고 있어야 합니다.

- SYSADM 또는 DBADM 권한
- 이 테이블에 대한 CONTROL 또는 ALTER 특권

구문



매개변수 설명

table_schema

table_name 매개변수에 지정된 테이블이 속해 있는 스키마 이름. 이 매개변수에 값을 지정해야 할 경우에도, 값은 널(NULL)일 수 있습니다. 이 매개변수가 널(NULL)이면, CURRENT SCHEMA 특수 레지스터는 테이블 또는 뷰에 대한 스키마 이름으로 사용됩니다.

table_schema 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

table_name

column_name 매개변수에서 지정된 컬럼이 들어있는 테이블의 규정되지 않은 이름을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

table_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

column_name

등록 해제할 공간 컬럼 이름. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

column_name 값은 큰따옴표로 묶인 경우를 제외하고는 대문자로 변환됩니다.

이 매개변수의 데이터 유형은 VARCHAR(128)이거나, 큰따옴표로 둘러싸여 있으면 VARCHAR(130)입니다.

출력 매개변수

msg_code

스토어드 프로시저에서 리턴된 메시지 코드를 지정합니다. 이 출력 매개변수 값은 프로시저 처리시 발생한 오류, 성공 또는 경고 조건을 식별합니다. 이 매개변수 값이 성공 또는 경고 조건에 대한 것이면, 프로시저가 해당 태스크를 완료합니다. 매개변수 값이 오류 조건에 대한 것이면, 데이터베이스가 변경되지 않습니다.

이 출력 매개변수의 데이터 유형은 INTEGER입니다.

msg_text

스토어드 프로시저에서 리턴된 메시지 코드와 연관된 실제 메시지 텍스트를 지정합니다. 메시지 텍스트에는 오류가 발생한 위치와 같은 성공, 경고 또는 오류 조건에 대한 추가 정보가 포함될 수 있습니다.

이 출력 매개변수의 데이터 유형은 VARCHAR(1024)입니다.

예

이 예에서는 DB2 명령행 처리기를 사용하여 ST_unregister_spatial_column 스토어드 프로시저를 호출하는 방법을 보여줍니다. 이 예에서는 DB2 CALL 명령을 사용하여 CUSTOMERS라는 테이블에서 LOCATION이라는 공간 컬럼을 등록 해제합니다.

```
call db2gse.ST_unregister_spatial_column(NULL,'CUSTOMERS','LOCATION',?,?)
```

이 CALL 명령 끝의 두 개의 물음표는 *msg_code* 및 *msg_text* 출력 매개변수를 나타냅니다. 이러한 출력 매개변수 값은 스토어드 프로시저가 실행된 후 표시됩니다.

제 21 장 카탈로그 뷰

공간 및 Geodetic Data Management Feature의 카탈로그 뷰에서는 유용한 정보를 제공합니다.

Spatial Extender의 카탈로그 뷰에는 다음에 대한 정보가 들어 있습니다.

277 페이지의 『DB2GSE.ST_COORDINATE_SYSTEMS 카탈로그 뷰』
사용할 수 있는 좌표 시스템

『DB2GSE.ST_GEOMETRY_COLUMNS 카탈로그 뷰』
데이터를 채우거나 갱신할 수 있는 공간 컬럼

280 페이지의 『DB2GSE.ST_GEOCODERS 카탈로그 뷰』 및 282 페이지의 『DB2GSE.ST_GEOCODING_PARAMETERS 카탈로그 뷰』
사용할 수 있는 지오코더(Geocoder)

281 페이지의 『DB2GSE.ST_GEOCODING 카탈로그 뷰』 및 282 페이지의 『DB2GSE.ST_GEOCODING_PARAMETERS 카탈로그 뷰』
자동으로 실행할 지오코더(Geocoder) 설정 및 일괄처리 지오코딩(Geocoding) 시 수행할 조작 설정에 대한 권장 스펙

283 페이지의 『DB2GSE.ST_SIZINGS 카탈로그 뷰』
변수에 지정할 수 있는 값의 허용 가능한 최대 길이

284 페이지의 『DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰』
사용할 수 있는 공간 참조 시스템

286 페이지의 『DB2GSE.ST_UNITS_OF_MEASURE 카탈로그 뷰』
공간 함수로 생성된 거리를 표시할 수 있는 측정 단위(미터, 마일, 피트 등)

DB2GSE.ST_GEOMETRY_COLUMNS 카탈로그 뷰

데이터베이스에 있는 공간 데이터를 포함하는 모든 테이블의 모든 컬럼에 대한 정보를 찾으려면 DB2GSE.ST_GEOMETRY_COLUMNS 카탈로그 뷰를 사용하십시오.

공간 컬럼이 공간 참조 시스템과 연관되어 등록되었으면, 이 뷰를 사용하여 공간 참조 시스템의 이름 및 숫자 ID도 찾을 수 있습니다. 공간 컬럼에 대한 추가 정보를 보려면 DB2의 SYSCAT.COLUMN 카탈로그 뷰를 쿼리하십시오.

DB2GSE.ST_GEOMETRY_COLUMNS에 대한 설명은 다음 표를 참조하십시오.

© Copyright IBM Corp. 1998, 2006

표 30. DB2GSE.ST_GEOMETRY_COLUMNS 카탈로그 뷰의 컬럼

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
TABLE_SCHEMA	VARCHAR(128)	아니오	이 공간 컬럼이 들어 있는 테이블이 속하는 스키마의 이름
TABLE_NAME	VARCHAR(128)	아니오	이 공간 컬럼이 들어 있는 테이블의 규정되지 않은 이름
COLUMN_NAME	VARCHAR(128)	아니오	이 공간 컬럼의 이름 TABLE_SCHEMA, TABLE_NAME 및 COLUMN_NAME의 조합이 컬럼을 고유하게 식별합니다.
TYPE_SCHEMA	VARCHAR(128)	아니오	이 공간 컬럼의 선언된 데이터 유형이 속하는 스키마의 이름. 이 이름은 DB2 카탈로그에서 획득됩니다.
TYPE_NAME	VARCHAR(128)	아니오	이 공간 컬럼의 선언된 데이터 유형의 규정되지 않은 이름. 이 이름은 DB2 카탈로그에서 획득됩니다.
SRS_NAME	VARCHAR(128)	예	이 공간 컬럼과 연관된 공간 참조 시스템의 이름. 컬럼과 연관된 공간 참조 시스템이 없으면 SRS_NAME은 널(NULL)입니다.
SRS_ID	INTEGER	예	이 공간 컬럼과 연관된 공간 참조 시스템의 숫자 ID. 컬럼과 연관된 공간 참조 시스템이 없으면 SRS_ID는 널(NULL)입니다.

DB2GSE.SPATIAL_REF_SYS 카탈로그 뷰

공간 참조 시스템을 작성할 때, DB2 Spatial Extender는 카탈로그 테이블에 공간 참조 시스템의 ID와 그에 관한 정보를 기록하여 등록합니다. 이 테이블에서 선택된 컬럼은 다음 테이블에 설명되어 있는 DB2GSE.SPATIAL_REF_SYS 카탈로그 뷰를 구성합니다.

표 31. DB2GSE.SPATIAL_REF_SYS 카탈로그 뷰의 컬럼

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
SRID	INTEGER	아니오	이 공간 참조 시스템에 대한 사용자 정의 ID
SR_NAME	VARCHAR(64)	아니오	이 공간 참조 시스템의 이름
CSID	INTEGER	아니오	이 공간 참조 시스템의 기초가 되는 좌표 시스템에 대한 숫자 ID
CS_NAME	VARCHAR(64)	아니오	이 공간 참조 시스템의 기초가 되는 좌표 시스템의 이름
AUTH_NAME	VARCHAR(256)	예	이 공간 참조 시스템에 대한 표준을 설정하는 소속의 이름
AUTH_SRID	INTEGER	예	AUTH_NAME 컬럼에 지정된 소속에서 이 공간 참조 시스템에 할당하는 ID
SRTEXT	VARCHAR(2048)	아니오	이 공간 참조 시스템에 대한 주석 텍스트
FALSEX	FLOAT	아니오	음의 X 좌표값에서 뺄 때 음수가 아닌 숫자(즉, 양수 또는 0)가 남는 숫자
FALSEY	FLOAT	아니오	음의 Y 좌표값에서 뺄 때 음수가 아닌 숫자(즉, 양수 또는 0)가 남는 숫자
XYUNITS	FLOAT	아니오	10진 X 좌표 또는 10진 Y 좌표로 곱할 때 32비트 데이터 항목으로 저장될 수 있는 정수를 산출하는 숫자.
FALSEZ	FLOAT	아니오	음의 Z 좌표값에서 뺄 때 음수가 아닌 숫자(즉, 양수 또는 0)가 남는 숫자
ZUNITS	FLOAT	아니오	10진 Z 좌표로 곱할 때 32비트 데이터 항목으로 저장될 수 있는 정수를 산출하는 숫자.

표 31. DB2GSE.SPATIAL_REF_SYS 카탈로그 뷰의 컬럼 (계속)

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
FALSEM	FLOAT	아니오	음의 치수에서 뺄 때 음수가 아닌 숫자(즉, 양수 또는 0)가 남는 숫자
MUNITS	FLOAT	아니오	10진 치수로 곱할 때 32비트 데이터 항목으로 저장될 수 있는 정수를 산출하는 숫자.

DB2GSE.ST_COORDINATE_SYSTEMS 카탈로그 뷰

등록된 좌표 시스템에 대한 정보를 검색하려면 DB2GSE.ST_COORDINATE_SYSTEMS 카탈로그 뷰를 쿼리하십시오.

Spatial Extender는 다음과 같은 경우 Spatial Extender 카탈로그에 좌표 시스템을 자동으로 등록합니다.

- 공간 데이터 관련 작업에 데이터베이스를 사용하는 경우
- 사용자가 데이터베이스에 추가 좌표 시스템을 정의하는 경우

이 뷰에 있는 컬럼에 대한 설명은 다음 표를 참조하십시오.

표 32. DB2GSE.ST_COORDINATE_SYSTEMS 카탈로그 뷰의 컬럼

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
COORDSYS_NAME	VARCHAR(128)	아니오	이 좌표 시스템의 이름 이름은 데이터베이스에서 고유합니다.
COORDSYS_TYPE	VARCHAR(128)	아니오	이 좌표 시스템의 유형: **PROJECTED** 2차원 **GEOGRAPHIC** 3차원. X 및 Y 좌표를 사용합니다. **GEOCENTRIC** 3차원. X, Y 및 Z 좌표를 사용합니다. **UNSPECIFIED** 추상 또는 비실세계(non-real world) 좌표 시스템. 이 컬럼의 값은 DEFINITION 컬럼에서 획득됩니다.
DEFINITION	VARCHAR(2048)	아니오	이 좌표 시스템 정의의 Well-Known Text 표현.
ORGANIZATION	VARCHAR(128)	예	이 좌표 시스템을 정의한 조직의 이름(예: European Petrol Survey Group 또는 ESPG와 같은 표준 단체). ORGANIZATION_COORDSYS_ID 컬럼이 널(NULL)이면 이 컬럼도 컬럼입니다.

표 32. DB2GSE.ST_COORDINATE_SYSTEMS 카탈로그 뷰의 컬럼 (계속)

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
ORGANIZATION_COORDSYS_ID	INTEGER	예	좌표 시스템을 정의한 조직에 의해 좌표 시스템에 지정된 숫자 ID. 이 ID와 ORGANIZATION 컬럼의 값은 둘 다 널(NULL)이 아닌 한 좌표 시스템을 고유하게 식별합니다. ORGANIZATION 컬럼이 널(NULL)이면 ORGANIZATION_COORDSYS_ID 컬럼도 널(NULL)입니다.
DESCRIPTION	VARCHAR(256)	예	해당 적용을 표시하는 좌표 시스템의 설명

DB2GSE.ST_GEOMETRY_COLUMNS 카탈로그 뷰

데이터베이스에 있는 공간 데이터를 포함하는 모든 테이블의 모든 컬럼에 대한 정보를 찾으려면 DB2GSE.ST_GEOMETRY_COLUMNS 카탈로그 뷰를 사용하십시오.

공간 컬럼이 공간 참조 시스템과 연관되어 등록되었으면, 이 뷰를 사용하여 공간 참조 시스템의 이름 및 숫자 ID도 찾을 수 있습니다. 공간 컬럼에 대한 추가 정보를 보려면 DB2의 SYSCAT.COLUMN 카탈로그 뷰를 쿼리하십시오.

DB2GSE.ST_GEOMETRY_COLUMNS에 대한 설명은 다음 표를 참조하십시오.

표 33. DB2GSE.ST_GEOMETRY_COLUMNS 카탈로그 뷰의 컬럼

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
TABLE_SCHEMA	VARCHAR(128)	아니오	이 공간 컬럼이 들어 있는 테이블이 속하는 스키마의 이름
TABLE_NAME	VARCHAR(128)	아니오	이 공간 컬럼이 들어 있는 테이블의 규정되지 않은 이름
COLUMN_NAME	VARCHAR(128)	아니오	이 공간 컬럼의 이름 TABLE_SCHEMA, TABLE_NAME 및 COLUMN_NAME의 조합이 컬럼을 고유하게 식별합니다.
TYPE_SCHEMA	VARCHAR(128)	아니오	이 공간 컬럼의 선언된 데이터 유형이 속하는 스키마의 이름. 이 이름은 DB2 카탈로그에서 획득됩니다.
TYPE_NAME	VARCHAR(128)	아니오	이 공간 컬럼의 선언된 데이터 유형의 규정되지 않은 이름. 이 이름은 DB2 카탈로그에서 획득됩니다.
SRS_NAME	VARCHAR(128)	예	이 공간 컬럼과 연관된 공간 참조 시스템의 이름. 컬럼과 연관된 공간 참조 시스템이 없으면 SRS_NAME은 널(NULL)입니다.
SRS_ID	INTEGER	예	이 공간 컬럼과 연관된 공간 참조 시스템의 숫자 ID. 컬럼과 연관된 공간 참조 시스템이 없으면 SRS_ID는 널(NULL)입니다.

DB2GSE.ST_GEOCODER_PARAMETERS 카탈로그 뷰

공간 데이터 관련 작업에 대해 데이터베이스를 사용 가능화하면 제공되는 지오코더(Geocoder) DB2GSE_USA_GEOCODER의 매개변수에 대한 정보가 자동으로 DB2 Spatial Extender 카탈로그에 기록됩니다. 추가 지오코더(Geocoder)를 등록하면 이들의 매개변수에 대한 정보도 카탈로그에 기록됩니다. 카탈로그에서 지오코더(Geocoder)의 매개변수에 대한 정보를 검색하려면 DB2GSE.ST_GEOCODER_PARAMETERS 카탈로그 뷰를 쿼리하십시오. 이 뷰에 있는 컬럼에 대한 설명은 다음 표를 참조하십시오.

지오코더(Geocoder)의 매개변수에 대한 자세한 정보를 보려면 DB2의 SYSCAT.ROUTINEPARMS 카탈로그 뷰를 쿼리하십시오. 이 뷰에 대한 설명은 SQL 참조서를 참조하십시오.

표 34. DB2GSE.ST_GEOCODER_PARAMETERS의 컬럼

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
GEOCODER_NAME	VARCHAR(128)	아니오	이 매개변수가 속하는 지오코더(Geocoder)의 이름.
ORDINAL	SMALLINT	아니오	GEOCODER_NAME 컬럼에 지정된 지오코더(Geocoder)로서 기능하는 함수의 서명에서 이 매개변수(즉, PARAMETER_NAME 컬럼에 지정된 매개변수)의 위치. GEOCODER_NAME 및 ORDINAL 컬럼의 값 조합이 이 매개변수를 고유하게 식별합니다. DB2의 SYSCAT.ROUTINEPARMS 카탈로그 뷰에 있는 레코드에도 이 매개변수에 대한 정보가 포함됩니다. 이 레코드에는 SYSCAT.ROUTINEPARMS의 ORDINAL 컬럼에 나타나는 값이 포함됩니다. 이 값은 DB2GSE.ST_GEOCODER_PARAMETERS 뷰의 ORDINAL 컬럼에 나타나는 값과 동일합니다.
PARAMETER_NAME	VARCHAR(128)	예	이 매개변수의 이름. 이 매개변수가 속하는 함수를 작성할 때 이름을 지정하지 않으면 PARAMETER_NAME 컬럼은 널(NULL)입니다. PARAMETER_NAME 컬럼의 컨텐츠는 DB2 카탈로그에서 획득됩니다.
TYPE_SCHEMA	VARCHAR(128)	아니오	이 매개변수가 속하는 스키마의 이름. 이 이름은 DB2 카탈로그에서 획득됩니다.
TYPE_NAME	VARCHAR(128)	아니오	이 매개변수에 지정된 값의 데이터 유형의 규정되지 않은 이름. 이 이름은 DB2 카탈로그에서 획득됩니다.

표 34. DB2GSE.ST_GEOCODER_PARAMETERS의 컬럼 (계속)

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
PARAMETER_DEFAULT	VARCHAR(2048)	예	이 매개변수에 지정되는 디폴트값. DB2가 이 값을 SQL 표현식으로서 해석합니다. 이 값이 따옴표로 둘러져 있으면 이 값은 문자열로서 지오코더(Geocoder)로 패스됩니다. 그렇지 않은 경우에는, 지오코더(Geocoder)로 패스될 때 SQL 표현식 평가에 의해 매개변수의 데이터 유형이 판별됩니다. PARAMETER_VALUE 컬럼이 널(NULL)을 포함하면 이 널(NULL) 값이 지오코더(Geocoder)로 패스됩니다. 디폴트값은 DB2GSE.ST_GEOCODING_PARAMETERS 카탈로그 뷰에 그와 연관된 값이 있을 수 있습니다. ST_run_geocoding 스토어드 프로시저에 대한 입력에도 그와 연관된 값이 있을 수 있습니다. 연관된 값이 디폴트값과 다르면 연관된 값이 디폴트값을 겹쳐씁니다.
DESCRIPTION	VARCHAR(256)	예	해당 적용을 표시하는 매개변수의 설명

DB2GSE.ST_GEOCODERS 카탈로그 뷰

공간 데이터 관련 작업에 대해 데이터베이스를 사용 가능화하면 제공되는 지오코더(Geocoder)인 DB2GSE_USA_GEOCODER가 자동으로 DB2 Spatial Extender 카탈로그에 등록됩니다. 사용자가 추가 지오코더(Geocoder)를 사용할 수 있도록 하려면 추가 지오코더(Geocoder)도 등록해야 합니다. 등록된 지오코더(Geocoder)에 대한 정보를 검색하려면 DB2GSE.ST_GEOCODERS 카탈로그 뷰를 쿼리하십시오. 이 뷰에 있는 컬럼에 대한 설명은 다음 표를 참조하십시오.

지오코더(Geocoder)의 매개변수에 대해서는 DB2 Spatial Extender의 DB2GSE.ST_GEOCODER_PARAMETERS 카탈로그 뷰 및 DB2의 SYSCAT.ROUTINEPARMS 카탈로그 뷰를 쿼리하십시오. 지오코더(Geocoder)로서 사용되는 함수에 대해서는 DB2의 SYSCAT.ROUTINES 카탈로그 뷰를 쿼리하십시오.

표 35. DB2GSE.ST_GEOCODERS 카탈로그 뷰의 컬럼

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
GEOCODER_NAME	VARCHAR(128)	아니오	이 지오코더(Geocoder)의 이름. 데이터베이스에서 고유해야 합니다.
FUNCTION_SCHEMA	VARCHAR(128)	아니오	이 지오코더(Geocoder)로서 사용되는 함수가 속하는 스키마의 이름
FUNCTION_NAME	VARCHAR(128)	아니오	이 지오코더(Geocoder)로서 사용되는 함수의 규정되지 않은 이름
SPECIFIC_NAME	VARCHAR(128)	아니오	이 지오코더(Geocoder)로서 사용되는 함수의 특정 이름 FUNCTION_SCHEMA 및 SPECIFIC_NAME의 값 조합이 이 지오코더(Geocoder)로서 사용되는 함수를 고유하게 식별합니다.

표 35. DB2GSE.ST_GEOCODERS 카탈로그 뷰의 컬럼 (계속)

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
RETURN_TYPE_SCHEMA	VARCHAR(128)	아니오	이 지오코더(Geocoder)의 출력 매개변수의 데이터 유형이 속하는 스키마의 이름. 이 이름은 DB2 카탈로그에서 획득됩니다.
RETURN_TYPE_NAME	VARCHAR(128)	아니오	이 지오코더(Geocoder)의 출력 매개변수 데이터 유형의 규정되지 않은 이름. 이 이름은 DB2 카탈로그에서 획득됩니다.
VENDOR	VARCHAR(256)	예	이 지오코더(Geocoder)를 작성한 벤더의 이름
DESCRIPTION	VARCHAR(256)	예	해당 적용을 표시하는 지오코더(Geocoder)의 설명

DB2GSE.ST_GEOCODING 카탈로그 뷰

지오코딩(Geocoding) 조작을 설정할 때, 설정의 상세 정보가 자동으로 DB2 Spatial Extender 카탈로그에 기록됩니다. 이러한 상세 정보를 보려면 DB2GSE.ST_GEOCODING 및 DB2GSE.ST_GEOCODING_PARAMETERS 카탈로그 뷰를 쿼리하십시오. 다음 테이블에 설명된 DB2GSE.ST_GEOCODING 카탈로그 뷰에는 모든 설정의 상세 정보가 포함되어 있습니다(예: 지오코더(Geocoder)가 커미트 전에 처리하는 레코드 수). DB2GSE.ST_GEOCODING_PARAMETERS 카탈로그 뷰에는 각 지오코더(Geocoder)에 고유한 상세 정보가 들어 있습니다. 예를 들어, 제공되는 지오코더(Geocoder)인 DB2GSE_USA_GEOCODER에 대한 설정에는 지오코더(Geocoder)가 입력을 지오코딩(Geocoding)하기 위해 입력으로서 제공된 주소와 실제 주소가 반드시 일치해야 하는 최소 등급이 포함됩니다. 최소 일치 스코어라고 하는 이 최소 요구사항이 DB2GSE.ST_GEOCODING_PARAMETERS 카탈로그 뷰에 기록됩니다.

표 36. DB2GSE.ST_GEOCODING 카탈로그 뷰의 컬럼

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
TABLE_SCHEMA	VARCHAR(128)	아니오	COLUMN_NAME 컬럼에 식별된 컬럼이 있는 테이블이 들어 있는 스키마의 이름
TABLE_NAME	VARCHAR(128)	아니오	COLUMN_NAME 컬럼에 식별된 컬럼이 있는 테이블의 규정되지 않은 이름
COLUMN_NAME	VARCHAR(128)	아니오	이 카탈로그 뷰에 표시된 스펙에 따라 채워지는 공간 컬럼의 이름 TABLE_SCHEMA, TABLE_NAME 및 COLUMN_NAME 컬럼의 값 조합이 공간 컬럼을 고유하게 식별합니다.
GEOCODER_NAME	VARCHAR(128)	아니오	COLUMN_NAME 컬럼에 지정된 공간 컬럼의 데이터를 생성하는 지오코더(Geocoder)의 이름. 공간 컬럼에 하나의 지오코더(Geocoder)만 지정할 수 있습니다.

표 36. DB2GSE.ST_GEOCODING 카탈로그 뷰의 컬럼 (계속)

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
MODE	VARCHAR(128)	아니오	지오코딩(Geocoding) 프로세스의 모드: **일괄처리** 일괄처리 지오코딩(Geocoding)만이 사용 가능화됩니다. **자동** 자동 지오코딩(Geocoding)이 설정되고 활성화됩니다. **유효하지 않음** 공간 카탈로그 테이블에서 불일치가 발견되었습니다. 지오코딩(Geocoding) 항목이 유효하지 않습니다.
SOURCE_COLUMNS	VARCHAR(10000)	예	자동 지오코딩(Geocoding)용으로 설정된 테이블 컬럼의 이름. 이러한 컬럼이 갱신될 때마다 트리거가 지오코더(Geocoder)에 갱신된 데이터를 지오코딩(Geocoding)하도록 요구합니다.
WHERE_CLAUSE	VARCHAR(10000)	예	WHERE절 내의 검색 조건. 이 조건은 지오코더(Geocoder)가 일괄처리 모드로 실행될 때 지정된 레코드 서브세트 내의 데이터만을 지오코딩(Geocoding)함을 표시합니다.
COMMIT_COUNT	INTEGER	예	커미트가 발행되기 전 일괄처리 지오코딩(Geocoding) 동안에 처리되는 행 수. COMMIT_COUNT 컬럼의 값이 0 또는 널(NULL)이면 커미트가 발행되지 않습니다.

DB2GSE.ST_GEOCODING_PARAMETERS 카탈로그 뷰

특정 지오코더(Geocoder)에 대해 지오코딩(Geocoding) 조작을 설정할 때, 이러한 설정 중에서 지오코더(Geocoder)에 고유한 조작은 자동으로 Spatial Extender 카탈로그에 기록됩니다. 예를 들어, 제공되는 지오코더(Geocoder) DB2GSE_USA_GEOCODER에 고유한 조작은 입력으로서 제공된 주소와 참조 데이터를 비교하고 이들이 지정된 등급 이상으로 일치할 경우 입력으로서 제공된 주소를 지오코딩(Geocoding)하는 것입니다. 이 지오코더에 대해 조작을 설정할 때, 사용자는 최소 일치 스코어라는 등급을 지정하며 이러한 사용자 스펙은 카탈로그에 기록됩니다.

지오코딩(Geocoding) 조작 설정 중 지오코더(Geocoder) 고유 조작을 알아 보려면 DB2GSE.ST_GEOCODING_PARAMETERS 카탈로그 뷰를 쿼리하십시오. 다음 테이블에 이 뷰에 대한 설명이 나와 있습니다.

DB2GSE.ST_GEOCODER_PARAMETERS 카탈로그 뷰에서는 지오코딩(Geocoding) 조작 설정에 대한 특정 디폴트값을 제공합니다. DB2GSE.ST_GEOCODING_PARAMETERS 뷰에 있는 값은 이러한 디폴트값을 겹쳐씁니다.

표 37. DB2GSE.ST_GEOCODING_PARAMETERS 카탈로그 뷰의 컬럼

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
TABLE_SCHEMA	VARCHAR(128)	아니오	COLUMN_NAME 컬럼에 식별된 컬럼이 있는 테이블이 들어 있는 스키마의 이름
TABLE_NAME	VARCHAR(128)	아니오	공간 컬럼이 들어 있는 테이블의 규정되지 않은 이름
COLUMN_NAME	VARCHAR(128)	아니오	이 카탈로그 뷰에 표시된 스펙에 따라 채워지는 공간 컬럼의 이름 TABLE_SCHEMA, TABLE_NAME 및 COLUMN_NAME 컬럼의 값 조합이 이 공간 컬럼을 고유하게 식별합니다.
ORDINAL	SMALLINT	아니오	COLUMN_NAME 컬럼에 식별된 컬럼에 대한 지오코더(Geocoder)로서 기능하는 함수의 서명에서 이 매개변수(즉, PARAMETER_NAME 컬럼에 지정된 매개변수)의 위치. DB2의 SYSCAT.ROUTINEPARMS 카탈로그 뷰에 있는 레코드에도 이 매개변수에 대한 정보가 포함됩니다. 이 레코드에는 SYSCAT.ROUTINEPARMS의 ORDINAL 컬럼에 나타나는 값이 포함됩니다. 이 값은 DB2GSE.ST_GEOCODING_PARAMETERS 뷰의 ORDINAL 컬럼에 나타나는 값과 동일합니다.
PARAMETER_NAME	VARCHAR(128)	예	지오코더(Geocoder) 정의에 있는 매개변수의 이름. 지오코더(Geocoder)를 정의할 때 이름을 지정하지 않은 경우에는 PARAMETER_NAME이 널(NULL)입니다. PARAMETER_NAME 컬럼의 이 컨텐츠는 DB2 카탈로그에서 획득됩니다.
PARAMETER_VALUE	VARCHAR(2048)	예	이 매개변수에 지정되는 값. DB2가 이 값을 SQL 표현식으로서 해석합니다. 이 값이 따옴표로 둘러져 있으면 이 값은 문자열로서 지오코더(Geocoder)로 패스됩니다. 그렇지 않은 경우에는, 지오코더(Geocoder)로 패스될 때 SQL 표현식 평가에 의해 매개변수의 데이터 유형이 판별됩니다. PARAMETER_VALUE 컬럼이 널(NULL)을 포함하면 이 널(NULL) 값이 지오코더(Geocoder)로 패스됩니다. PARAMETER_VALUE 컬럼은 DB2GSE.ST_GEOCODER_PARAMETERS 카탈로그 뷰의 PARAMETER_DEFAULT 컬럼과 연관됩니다. PARAMETER_VALUE 컬럼에 값이 포함될 경우, 이 값은 PARAMETER_DEFAULT 컬럼에 있는 디폴트값을 겹쳐씁니다. PARAMETER_VALUE 컬럼이 널(NULL)이면 디폴트값이 사용됩니다.

DB2GSE.ST_SIZINGS 카탈로그 뷰

DB2GSE.ST_SIZINGS 카탈로그 뷰를 사용하여 다음을 검색하십시오.

- Spatial Extender에서 지원하는 모든 변수. 예를 들면, 좌표 시스템 이름, 지오코더 이름 그리고 공간 데이터의 Well-Known Text 표현을 지정할 수 있는 변수

- 이러한 변수에 지정된 값의 허용 가능한 최대 길이(알려진 경우). 예를 들면, 좌표 시스템 이름, 지오코더(Geocoder) 이름 및 공간 데이터의 Well-Known Text 표현의 최대 허용 가능 길이.

이 뷰에 있는 컬럼에 대한 설명은 다음 테이블을 참조하십시오.

표 38. DB2GSE.ST_SIZINGS 카탈로그 뷰의 컬럼

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
VARIABLE_NAME	VARCHAR(128)	아니오	변수를 나타내는 용어. 이 용어는 데이터베이스에서 고유합니다.
SUPPORTED_VALUE	INTEGER	예	VARIABLE_NAME 컬럼에 표시된 변수에 지정된 값의 허용 가능 최대 길이. SUPPORTED_VALUE 컬럼에서 가능한 값은 다음과 같습니다. **0보다 큰 숫자 값** 이 변수에 지정된 값의 허용 가능 최대 길이. **0** 어떤 길이라도 허용되거나 또는 허용 가능한 길이를 판별할 수 없습니다. **NULL** Spatial Extender가 이 변수를 지원하지 않습니다.
DESCRIPTION	VARCHAR(128)	예	이 변수의 설명

DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰

등록된 공간 참조 시스템에 대한 정보를 검색하려면 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰를 쿼리하십시오. Spatial Extender는 다음과 같은 경우 Spatial Extender 카탈로그에 공간 참조 시스템을 자동으로 등록합니다.

- 공간 데이터 관련 작업, 다섯 개의 디폴트 공간 참조 시스템 및 318개의 사전 정의된 측지 공간 참조 시스템에 데이터베이스를 사용하는 경우
- 사용자가 추가 공간 참조 시스템을 작성하는 경우

DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에서 완전한 값을 가져오려면 각 공간 참조 시스템이 좌표 시스템과 연관되어 있음을 이해해야 합니다. 공간 참조 시스템은 부분적으로 좌표 시스템에서 도출된 좌표를 DB2가 최대 효율로 처리할 수 있는 값으로 변환하고 부분적으로 이러한 좌표가 참조할 수 있는 스페이스의 최대 가능 범위를 정의할 수 있도록 설계되었습니다.

제공된 공간 참조 시스템과 연관된 좌표 시스템의 이름 및 유형을 알아 보려면 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰의 COORDSYS_NAME 및 COORDSYS_TYPE 컬럼을 쿼리하십시오. 좌표 시스템에 대한 자세한 정보를 보려면 DB2GSE.ST_COORDINATE_SYSTEMS 카탈로그 뷰를 쿼리하십시오.

표 39. DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰의 컬럼

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
SRS_NAME	VARCHAR(128)	아니오	공간 참조 시스템의 이름. 이 이름은 데이터베이스에서 고유합니다.
SRS_ID	INTEGER	아니오	공간 참조 시스템의 숫자 ID. 각 공간 참조 시스템은 고유한 숫자 ID를 가집니다. 측지 공간 참조 시스템은 2000000000 - 2000001000 범위의 SRS_ID 값을 갖습니다. 공간 함수는 이름이 아닌 숫자 ID로 공간 참조 시스템을 지정합니다.
X_OFFSET	DOUBLE	아니오	Geometry의 모든 X 좌표에서 뺄 오프셋. 빼기는 Geometry의 좌표를 DB2가 최대 효율로 처리할 수 있는 값으로 변환하는 프로세스의 한 단계입니다. 다음 단계는 빼기 결과값을 X_SCALE 컬럼에 표시된 스케일 인수로 곱하는 것입니다.
X_SCALE	DOUBLE	아니오	X 좌표에서 오프셋을 뺀 결과값에 곱할 스케일 인수. 이 인수는 Y_SCALE 컬럼에 표시된 값과 동일합니다.
Y_OFFSET	DOUBLE	아니오	Geometry의 모든 Y 좌표에서 뺄 오프셋. 빼기는 Geometry의 좌표를 DB2가 최대 효율로 처리할 수 있는 값으로 변환하는 프로세스의 한 단계입니다. 다음 단계는 빼기 결과값을 Y_SCALE 컬럼에 표시된 스케일 인수로 곱하는 것입니다.
Y_SCALE	DOUBLE	아니오	Y 좌표에서 오프셋을 뺀 결과값에 곱할 스케일 인수. 이 인수는 X_SCALE 컬럼에 표시된 값과 동일합니다.
Z_OFFSET	DOUBLE	아니오	Geometry의 모든 Z 좌표에서 뺄 오프셋. 빼기는 Geometry의 좌표를 DB2가 최대 효율로 처리할 수 있는 값으로 변환하는 프로세스의 한 단계입니다. 다음 단계는 빼기 결과값을 Z_SCALE 컬럼에 표시된 스케일 인수로 곱하는 것입니다.
Z_SCALE	DOUBLE	아니오	Z 좌표에서 오프셋을 뺀 결과값에 곱할 스케일 인수.
M_OFFSET	DOUBLE	아니오	Geometry와 연관된 모든 도량 단위에서 뺄 오프셋. 빼기는 도량 단위를 DB2가 최대 효율로 처리할 수 있는 값으로 변환하는 프로세스의 한 단계입니다. 다음 단계는 빼기 결과값을 M_SCALE 컬럼에 표시된 스케일 인수로 곱하는 것입니다.
M_SCALE	DOUBLE	아니오	수치에서 오프셋을 뺀 결과값에 곱할 스케일 인수.
MIN_X	DOUBLE	아니오	이 공간 참조 시스템이 적용되는 Geometry에서 X 좌표에 가능한 최소값. 이 값은 X_OFFSET 및 X_SCALE 컬럼에 있는 값으로부터 도출됩니다.
MAX_X	DOUBLE	아니오	이 공간 참조 시스템이 적용되는 Geometry에서 X 좌표에 가능한 최대값. 이 값은 X_OFFSET 및 X_SCALE 컬럼에 있는 값으로부터 도출됩니다.
MIN_Y	DOUBLE	아니오	이 공간 참조 시스템이 적용되는 Geometry에서 Y 좌표에 가능한 최소값. 이 값은 Y_OFFSET 및 Y_SCALE 컬럼에 있는 값으로부터 도출됩니다.
MAX_Y	DOUBLE	아니오	이 공간 참조 시스템이 적용되는 Geometry에서 Y 좌표에 가능한 최대값. 이 값은 Y_OFFSET 및 Y_SCALE 컬럼에 있는 값으로부터 도출됩니다.

표 39. DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰의 컬럼 (계속)

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
MIN_Z	DOUBLE	아니오	이 공간 참조 시스템이 적용되는 Geometry에서 Z 좌표에 가능한 최소값. 이 값은 Z_OFFSET 및 Z_SCALE 컬럼에 있는 값으로부터 도출됩니다.
MAX_Z	DOUBLE	아니오	이 공간 참조 시스템이 적용되는 Geometry에서 Z 좌표에 가능한 최대값. 이 값은 Z_OFFSET 및 Z_SCALE 컬럼에 있는 값으로부터 도출됩니다.
MIN_M	DOUBLE	아니오	이 공간 참조 시스템이 적용되는 Geometry와 함께 저장할 수 있는 도량 단위에 가능한 최소값. 이 값은 M_OFFSET 및 M_SCALE 컬럼에 있는 값으로부터 도출됩니다.
MAX_M	DOUBLE	아니오	이 공간 참조 시스템이 적용되는 Geometry와 함께 저장할 수 있는 도량 단위에 가능한 최대값. 이 값은 M_OFFSET 및 M_SCALE 컬럼에 있는 값으로부터 도출됩니다.
COORDSYS_NAME	VARCHAR(128)	아니오	이 공간 참조 시스템의 기초가 되는 좌표 시스템의 식별 이름
COORDSYS_TYPE	VARCHAR(128)	아니오	이 공간 참조 시스템의 기초가 되는 좌표 시스템의 유형
ORGANIZATION	VARCHAR(128)	예	이 공간 참조 시스템의 기초가 되는 좌표 시스템을 정의한 조직(예: 표준 단체)의 이름. ORGANIZATION_COORSYS_ID가 널(NULL)이면 ORGANIZATION도 널(NULL)입니다.
ORGANIZATION_ COORDSYS_ID	INTEGER	예	이 공간 참조 시스템의 기초가 되는 좌표 시스템을 정의한 조직(예: 표준 단체)의 이름. ORGANIZATION이 널(NULL)이면 ORGANIZATION_COORDSYS_ID도 널(NULL)입니다.
DEFINITION	VARCHAR(2048)	아니오	좌표 시스템 정의의 Well-Known Text 표현
DESCRIPTION	VARCHAR(256)	예	공간 참조 시스템의 설명

DB2GSE.ST_UNITS_OF_MEASURE 카탈로그 뷰

특정 공간 함수는 특정 거리를 표시하는 값을 승인하거나 리턴합니다. 몇몇 경우에는 거리를 표현할 단위를 선택할 수 있습니다. 예를 들어, ST_Distance는 지정된 두 Geometry 사이의 최소 거리를 리턴합니다. ST_Distance가 거리를 마일 단위로 리턴해야 하는 경우가 있는 반면, 거리를 미터 단위로 표현해야 하는 경우도 있을 수 있습니다. 선택 가능한 측정 단위를 알아 보려면 DB2GSE.ST_UNITS_OF_MEASURE 카탈로그 뷰를 참조하십시오.

표 40. DB2GSE.ST_UNITS_OF_MEASURE 카탈로그 뷰의 컬럼

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
UNIT_NAME	VARCHAR(128)	아니오	측정 단위의 이름. 이 이름은 데이터베이스에서 고유합니다.

표 40. DB2GSE.ST_UNITS_OF_MEASURE 카탈로그 뷰의 컬럼 (계속)

이름	데이터 유형	널(NULL) 입력 가능 여부	내용
UNIT_TYPE	VARCHAR(128)	아니오	측정 단위의 유형. 가능한 값은 다음과 같습니다. **LINEAR** 측정 단위가 선형 측정 단위입니다. **ANGULAR** 측정 단위가 각 측정 단위입니다.
CONVERSION_FACTOR	DOUBLE	아니오	이 측정 단위를 해당 기본 단위로 변환하는 데 사용되는 숫자값. 선형 측정 단위의 기본 단위는 METER이고 각 측정 단위의 기본 단위는 RADIAN입니다. 기본 단위 자체의 변환 인수는 1.0입니다.
DESCRIPTION	VARCHAR(256)	예	측정 단위의 설명

제 22 장 공간 함수: 범주 및 사용

이 장에서는 모든 공간 함수를 범주별로 구성하여 소개합니다.

공간 함수: 범주 및 사용

DB2® Spatial Extender에서는 다음 함수를 제공합니다.

- Geometry를 데이터 교환 형식으로(또는 그 반대로) 변환. 이러한 함수를 컨스트럭터 함수라고 합니다.
- 바운더리, 교차 및 기타 정보에 대한 Geometry 비교. 이러한 함수를 비교 함수라고 합니다.
- Geometry 등록 정보(예: Geometry 내의 좌표와 수치, Geometry 간의 관계 및 바운더리와 기타 정보)에 대한 정보를 제공
- 기존 Geometry에서 새 Geometry를 생성
- Geometry의 위치 사이의 최소 거리를 측정
- 인덱스 매개변수에 대한 정보를 제공
- 다름 좌표 시스템 사이에 프로젝션과 변환을 제공

Geometry 값을 데이터 교환 형식으로 변환하는 공간 함수

DB2 Spatial Extender는 Geometry를 데이터 교환 형식으로(또는 그 반대로) 변환하는 공간 함수를 제공합니다.

- Well-Known Text 표현
- Well-Known Binary 표현
- ESRI 모양 표현
- GML(Geography Markup Language) 표현

이 형식으로부터 Geometry를 작성하는 함수를 *컨스트럭터 함수*라고 합니다.

컨스트럭터 함수

DB2 Spatial Extender는 Geometry를 데이터 교환 형식으로(또는 그 반대로) 변환하는 공간 함수를 제공합니다.

- Well-Known Text 표현
- Well-Known Binary 표현

© Copyright IBM Corp. 1998, 2006

- ESRI 모양 표현
- GML(Geography Markup Language) 표현

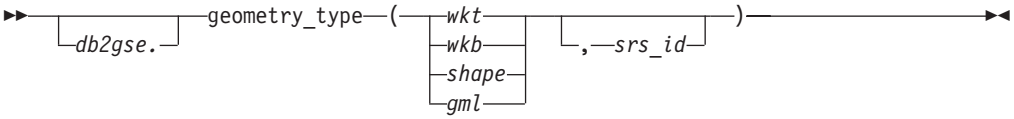
이 형식으로부터 Geometry를 작성하는 함수를 *컨스트럭터 함수*라고 합니다. 컨스트럭터 함수는 데이터를 삽입하는 컬럼의 Geometry 데이터 유형과 동일한 이름을 갖고 있습니다. 이러한 함수는 각 입력 데이터 교환 형식에 대해 일관성있게 작동합니다. 이 절에서는 다음 사항이 제공됩니다.

- 데이터 교환 형식에 대해 작동하는 함수를 호출하는 SQL 및 이 함수에서 리턴되는 Geometry 유형
- X 및 Y 좌표에서 포인트를 작성하는 함수를 호출하는 SQL 및 이 함수에서 리턴된 Geometry 유형
- 코드 예 및 결과 세트

데이터 교환 형식에 대해 수행되는 함수

이 섹션에서는 데이터 교환 형식에서 작동하는 함수를 호출하는 구문을 제공하며, 함수의 입력 매개변수를 설명하고, 이 함수를 리턴하는 Geometry 유형을 식별합니다.

구문



구문의 매개변수 및 기타 요소

db2gse

DB2® Spatial Extender가 제공하는 공간 데이터 유형이 속하는 스키마 이름.

geometry_type

다음 컨스트럭터 함수 중 하나입니다.

- ST_Point
- ST_LineString
- ₩
- ST_Polygon
- ST_MultiPoint
- ST_MultiLineString
- ST_MultiPolygon
- ST_GeomCollection
- ST_Geometry

wkt Geometry의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

wkb Geometry의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

shape Geometry의 ESRI 모양 표현이 있는 유형 BLOB(2G)의 값

gml Geometry의 GML 표현이 들어 있는 유형 CLOB(2G) 값

srs_id 결과 Geometry에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 식별자가 0인 공간 참조 시스템이 사용됩니다.

리턴 유형

geometry_type

*geometry_type*이 *ST_Geometry*이면 리턴된 Geometry 유형의 동적 유형은 입력 값에 표시되는 Geometry에 해당합니다.

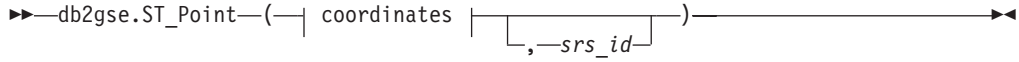
*geometry_type*이 다른 유형이면 리턴된 Geometry 유형의 동적 유형은 함수 이름에 해당합니다. 입력 값에 표시된 Geometry가 함수 이름 또는 해당 부속 유형의 이름 중 하나와 일치하지 않을 경우 오류가 리턴됩니다.

좌표에서 Geometry를 작성하는 함수

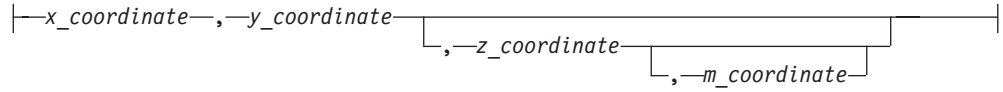
이 절에서는 ST_Point 호출 구문, 해당 매개변수에 대한 설명 및 리턴하는 Geometry 유형에 대한 정보를 제공합니다.

ST_Point 함수는 데이터 교환 형식뿐만 아니라 숫자 좌표 값(위치 데이터가 이미 데이터베이스에 저장되어 있는 경우 매우 유용함)에서도 Geometry를 작성합니다.

구문



좌표:



매개변수

x_coordinate

결과 점에 대한 X 좌표를 지정하는 DOUBLE 유형의 값

y_coordinate

결과 점에 대한 Y 좌표를 지정하는 DOUBLE 유형의 값

z_coordinate

결과 점에 대한 Z 좌표를 지정하는 DOUBLE 유형의 값

z_coordinate 매개변수를 생략하면 결과 점에 Z 좌표가 없습니다. 이러한 점에 대한 ST_Is3D 결과는 0입니다.

m_coordinate

결과 점에 대한 M 좌표를 지정하는 DOUBLE 유형의 값

m_coordinate 매개변수를 생략하면 결과 점에는 측정 단위가 없습니다. 이러한 점에 대한 ST_IsMeasured 결과는 0입니다.

srs_id 결과 점에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

*srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 않으면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_Point

예

이 섹션에서는 컨스트럭터 함수를 호출하는 코드, 컨스트럭터 함수의 출력을 포함할 테이블 작성 코드, 출력을 검색하는 코드 및 출력 그 자체에 대한 예를 제공합니다.

다음 예는 좌표 표시 및 WKT(Well-Known Text) 표시를 사용하여 ID가 100이고 X 좌표가 30, Y 좌표가 40인 포인트 값을 가진 SAMPLE_GEOMETRY 테이블 및 공간 참조 시스템 1에 행을 삽입합니다. 그런 다음 ID가 200인 행과 표시된 좌표가 있는 선스트링 값을 삽입합니다.

```
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"
```

공간 컬럼이 ST_Point 값만을 포함할 수 있는 경우, 다음 예제를 통해 두 개의 포인트를 삽입할 수 있습니다. SQL 오류의 포인트 결과가 아닌 기타 유형 또는 선스트링 삽입을 시도합니다. 첫 번째 점을 삽입하면 Well-Known Text 표현으로부터 점 Geometry가 작성됩니다. 두 번째 점을 삽입하면 숫자 좌표 값으로부터 점 Geometry가 작성됩니다. 이러한 입력 값은 기존의 테이블 컬럼으로부터 선택할 수 있습니다.

```
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"
```

다음 예는 Embedded SQL을 사용하며 응용프로그램이 데이터 영역을 적절한 값으로 채운다고 가정합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS CLOB(10000) wkt_buffer;
   SQL TYPE IS CLOB(10000) gml_buffer;
   SQL TYPE IS BLOB(10000) wkb_buffer;
   SQL TYPE IS BLOB(10000) shape_buffer;
EXEC SQL END DECLARE SECTION;

// * Application logic to read into buffers goes here */

EXEC SQL INSERT INTO sample_geometry(id, geom)
         VALUES(:id, db2gse.ST_Geometry(:wkt_buffer,1));

EXEC SQL INSERT INTO sample_geometry(id, geom)
         VALUES:id, db2gse.ST_Geometry(:wkb_buffer,1));

EXEC SQL INSERT INTO sample_geometry(id, geom)
         VALUES(:id, db2gse.ST_Geometry(:gml_buffer,1));

EXEC SQL INSERT INTO sample_geometry(id, geom)
         VALUES(:id, db2gse.ST_Geometry(:shape_buffer,1));
```

다음 샘플 Java™ 코드에서는 JDBC를 사용하여 X, Y 숫자 좌표 값으로 점 Geometry를 삽입하고 WKT 표현을 사용하여 Geometry를 지정합니다.

```
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();
```

WKT(Well-Known Text) 표현으로 변환

텍스트 표현은 ASCII 문자열을 나타내는 CLOB 값입니다. 이는 Geometry를 ASCII 텍스트 양식으로 교환할 수 있습니다.

ST_AsText 함수는 테이블에 저장된 Geometry 값을 WKT 문자열로 변환합니다. 다음 예에서는 단순한 명령행 쿼리를 사용하여 앞에서 SAMPLE_GEOMETRY 테이블에 삽입한 값을 선택합니다.

```
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)
```

다음 예에서는 Embedded SQL을 사용하여 앞에서 SAMPLE_GEOMETRY 테이블에 삽입한 값을 선택합니다.

```
 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;
```

다른 방법으로, 바인드할 때 ST_WellKnownText 변환 그룹을 사용하여 내재적으로 Geometry를 Well-Known Text 표현으로 변환할 수 있습니다. 다음 샘플 예에서는 변환 그룹을 사용하는 방법을 보여줍니다.

```
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;
```

Geometry를 변환하기 위한 SELECT문에 공간 함수가 사용되지 않았습니다.

이 섹션에서 설명한 함수 외에도 DB2® Spatial Extender는 Geometry와 Well-Known Text 표현 사이를 변환하는 기타 함수도 제공합니다. DB2 Spatial Extender는 OGC 『Simple Features for SQL』 스펙 및 ISO SQL/MM Part 3: Spatial 표준을 구현하는 기타 함수를 제공합니다. 함수는 다음과 같습니다.

- **ST_WKTToSQL**
- **ST_GeomFromText**
- **ST_GeomCollFromTxt**
- **ST_PointFromText**
- **ST_LineFromText**
- **ST_PolyFromText**
- **ST_MPointFromText**
- **ST_MLineFromText**
- **ST_MPolyFromText**

WKB(Well-Known Binary) 표현으로 변환

WKB 표현은 2진 데이터 구조로 구성되며, 이는 BLOB 값이어야 합니다. 이러한 BLOB 값은 DB2®에서 지원하고 DB2에 언어 바인딩 기능이 있는 프로그래밍 언어로 작성된 응용프로그램에서 관리해야 하는 실행 파일 데이터 구조를 나타냅니다.

ST_AsBinary 함수는 테이블에 저장된 Geometry 값을 프로그램 스토리지의 BLOB 변수에 페치될 수 있는 WKB(Well-Known Binary) 표현으로 변환합니다. 다음 예에서는 Embedded SQL을 사용하여 앞에서 SAMPLE_GEOMETRY 테이블에 삽입한 값을 선택합니다.

```
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;
```

다른 방법으로, 바인드할 때 ST_WellKnownBinary 변환 그룹을 사용하여 내재적으로 Geometry를 Well-Known Binary 표현으로 변환할 수 있습니다. 다음 샘플 예에서는 이 변환 그룹을 사용하는 방법을 보여줍니다.

```
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;
```

Geometry를 변환하기 위한 SELECT문에 공간 함수가 사용되지 않았습니다.

이 절에 설명된 함수 외에도 Geometry를 Well-Known Binary 표현으로 변환하는 다른 함수가 있습니다. DB2 Spatial Extender는 OGC『Simple Features for SQL』스펙 및 ISO SQL/MM Part 3: Spatial 표준을 구현하는 기타 함수를 제공합니다. 함수는 다음과 같습니다.

- **ST_WKBToSQL**
- **ST_GeomFromWKB**
- **ST_GeomCollFromWKB**
- **ST_PointFromWKB**
- **ST_LineFromWKB**
- **ST_PolyFromWKB**
- **ST_MPointFromWKB**
- **ST_MLineFromWKB**
- **ST_MPolyFromWKB**

ESRI 모양 표현으로 변환

ESRI 모양 표현은 지원되는 언어로 작성된 응용프로그램에 의해 관리되어야 하는 2진 데이터 구조로 구성됩니다.

ST_AsShape 함수는 테이블에 저장된 Geometry 값을 프로그램 스토리지의 BLOB 변수에 페치될 수 있는 ERSI 모양 표현으로 변환합니다. 다음 예에서는 Embedded SQL을 사용하여 앞에서 SAMPLE_GEOMETRY 테이블에 삽입한 값을 선택합니다.

```
 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;
```

다른 방법으로, 바인드할 때 ST_Shape 변환 그룹을 사용하여 내재적으로 Geometry를 Well-Known Text 표현으로 변환할 수 있습니다. 다음 샘플 예에서는 변환 그룹을 사용하는 방법을 보여줍니다.

```
 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;
```

Geometry를 변환하기 위한 SELECT문에 공간 함수가 사용되지 않았습니다.

GML(Geography Markup Language) 표현으로 변환

GML(Geography Markup Language) 표현은 ASCII 문자열입니다. 이는 Geometry를 ASCII 텍스트 양식으로 교환할 수 있습니다.

ST_AsGML 함수는 테이블에 저장된 Geometry 값을 GML 텍스트 문자열로 변환합니다. 다음 예에서는 앞에서 SAMPLE_GEOMETRY 테이블에 삽입한 값을 선택합니다. 예에서 결과는 읽기 쉽도록 다시 형식화된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변할 수 있습니다.

```
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>
```

다른 방법으로, 바인드할 때 ST_GML 변환 그룹을 사용하여 내재적으로 Geometry를 HTML 표현으로 변환할 수 있습니다.

```
  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>
```

Geometry를 변환하기 위한 SELECT문에 공간 함수가 사용되지 않았습니다.

지리 대상물을 비교하는 함수

여러 공간 함수는 지리 대상물 상호간의 관계나 서로를 비교하는 방법에 대한 정보를 리턴합니다. 다른 공간 함수는 좌표 시스템의 두 정의 또는 두 공간 참조 시스템이 동일한지에 대한 정보를 리턴합니다. 모든 경우에서, 리턴된 정보는 Geometry간, 좌표 시스템의 정의간 또는 공간 참조 시스템간을 비교한 결과입니다. 이 정보를 제공하는 함수를 비교 함수라고 합니다.

표 41. 목적별 비교 함수

목적	함수
한 Geometry의 내부가 다른 Geometry의 내부와 교차하는지 판별합니다.	• ST_Contains • ST_Within
Geometry 교차에 대한 정보를 리턴합니다.	• ST_Crosses • ST_Intersects • ST_Overlaps • ST_Touches
한 Geometry를 둘러싸는 가장 작은 직사각형이 다른 Geometry를 둘러싸는 가장 작은 직사각형과 교차하는지 판별합니다.	• ST_EnvIntersects • ST_MBRIntersects
두 개의 오브젝트가 동일한지 여부를 판별합니다.	• ST_Equals • ST_EqualCoordsys • ST_EqualSRS
비교하는 Geometry가 DE-9IM 패턴 행렬 문자열의 조건을 충족시키는지 여부를 판별합니다.	• ST_Relate
두 개의 Geometry 간에 교차되는지 점검합니다.	• ST_Disjoint

비교 함수

DB2® Spatial Extender의 비교 함수에서는 비교가 특정 기준을 충족하면 값 1을 리턴하고 비교가 특정 기준을 충족하지 않으면 값 0을 리턴하며 비교를 수행할 수 없는 경우 널(NULL) 값을 리턴합니다. 입력 매개변수에 대한 비교 조작이 정의되어 있지 않거나 매개변수 중 하나가 널(NULL)이면 비교를 수행할 수 없습니다. 매개변수에 다른 데이터 유형 또는 차원을 가진 Geometry를 지정할 경우에는 비교를 수행할 수 있습니다.

(DE-9IM(Dimensionally Extended 9 Intersection Model)은 다른 유형과 차원을 가진 Geometry 간에 쌍으로 된 공간 관계를 정의하는 수학적 접근 방식입니다. 이 모델은 결과로 발생되는 교차의 차원을 고려하여 모든 유형의 Geometry 사이의 공간 관계를 그의 내부, 바운더리 및 외부의 교차쌍으로서 나타냅니다.

제공된 Geometry *a* 및 *b*: I(*a*), B(*a*) 및 E(*a*)는 각각 *a*의 내부, 바운더리 및 외부를 나타냅니다. 그리고 I(b), B(b) 및 E(b)는 *b*의 내부, 바운더리 및 외부를 나타냅니다. I(b), B(b) 및 E(b)가 있는 I(a), B(a) 및 E(a)의 교차점에서는 3*3 행렬을 생성합니다. 각각을 교차하면 서로 다른 차원의 Geometry가 생성될 수 있습니다. 예를 들어, 두 다각형 바운더리의 교차는 점과 선스트링으로 구성되는데, 이러한 경우에는 dim 함수가 최대 차원 1을 리턴합니다.

dim 함수는 -1, 0, 1 또는 2 값을 리턴합니다. -1은 널(NULL) 세트나 dim(null)에 해당하며 교차가 없을 경우 리턴됩니다.

비교 함수에서 리턴된 결과는 비교 함수에서 리턴된 결과를 DE-9IM에 대해 허용 가능한 값을 나타내는 패턴 행렬과 비교하여 이해하거나 검증할 수 있습니다.

패턴 행렬에는 각각의 교차 행렬 셀에 대한 허용 가능한 값이 들어 있습니다. 가능한 패턴값은 다음과 같습니다.

T 교차가 존재해야 함. dim = 0, 1 또는 2

F 교차가 존재하지 않아야 함. dim = -1

***** 교차가 존재하는지 여부에 상관하지 않음. dim = -1, 0, 1 또는 2

0 교차가 존재해야 하며 해당 차원은 정확히 0이어야 함. dim = 0

1 교차가 존재해야 하고 최대 차원은 1이어야 함. dim = 1

2 교차가 존재해야 하고 최대 차원은 2이어야 함. dim = 2

예를 들어, ST_Within 함수에 대한 다음 패턴 행렬에는 T, F 및 * 값이 포함됩니다.

표 42. ST_Within 행렬. Geometry 조합에 대한 ST_Within 함수의 패턴 행렬

	Geometry b 내부	**Geometry b 바운더리**	**Geometry b 외부**
Geometry a 내부	T	*	F
Geometry a 바운더리	*	*	F
Geometry a 외부	*	*	*

ST_Within 함수는 두 Geometry 내부가 교차할 때와 *a*의 내부 또는 바운더리가 *b*의 외부와 교차하지 않을 때 값 1을 리턴합니다. 다른 모든 조건은 상관없습니다.

각 함수에는 적어도 하나의 패턴 행렬이 있지만, 일부 술어는 다양한 Geometry 유형 조합의 관계를 설명하기 위해 둘 이상의 패턴 행렬을 필요로 합니다.

DE-91M은 Egenhofer 및 Herring의 9개 교차 모델을 공간적으로 확장한 Clementini와 Felice가 개발하였습니다. DE-9IM은 ″A Small Set of Formal Topological Relationships Suitable for End-User Interaction,″ D. Abel 및 B.C. Ooi(Ed.), *Advances in Spatial Database—Third International Symposium. SSD '93*. LNCS 692. Pp. 277-295에 모델을 발표하였습니다. M. J. Egenhofer 및 J. Herring (Springer-Verlag Singapore [1993])의 9개의 교차 모델은 ″Categorizing binary topological relationships between regions, lines, and points in geographic databases,″ *Tech. Report, Department of Surveying Engineering*에 발표되었습니다.

공간 비교 함수

비교 함수는 다음과 같습니다.

- ST_Contains
- ST_Crosses
- ST_Disjoint
- ST_EnvIntersects
- ST_EqualCoordsys
- ST_Equals
- ST_EqualSRS
- ST_Intersects
- ST_MBRIntersects
- ST_Overlaps
- ST_Relate
- ST_Touches
- ST_Within

한 Geometry가 다른 Geometry를 포함하는지 여부를 점검하는 함수

ST_Contains 및 ST_Within은 모두 두 개의 Geometry를 입력으로 가져와 한 Geometry의 내부가 다른 Geometry의 내부와 교차하는지를 판별합니다. ST_Contains는 제공된 첫 번째 Geometry가 두 번째 Geometry를 둘러싸는지(첫 번째 Geometry가 두 번째 Geometry를 포함하는지)를 판별합니다. ST_Within은 첫 번째 Geometry가 완전히 두 번째 Geometry 내에 있는지(첫 번째 Geometry가 두 번째 Geometry에 포함되는지)를 판별합니다.

ST_Contains

ST_Contains를 사용하여 한 Geometry가 다른 Geometry에 완전히 포함되는지 여부를 판별하십시오.

ST_Contains는 두 번째 Geometry가 첫 번째 Geometry에 완전히 포함되는 경우 값 1을 리턴합니다. ST_Contains 함수는 ST_Within 함수의 정반대 결과를 리턴합니다.

302 페이지의 그림 44에서는 ST_Contains 예가 표시됩니다.

- 모든 점이 첫 번째 Geometry 내부에 있는 경우 다중점 Geometry는 하나의 점 또는 다중점 Geometry를 포함합니다.
- 모든 점이 다각형의 바운더리 또는 다각형 내부에 있는 경우 다각형 Geometry는 다중점 Geometry를 포함합니다.
- 모든 점이 첫 번째 Geometry 내부에 있는 경우 선스트링 Geometry는 하나의 점, 다중점 또는 선스트링 Geometry를 포함합니다.
- 두 번째 Geometry가 다각형 내부에 있는 경우 다각형 Geometry는 하나의 점, 선스트링 또는 다각형 Geometry를 포함합니다.

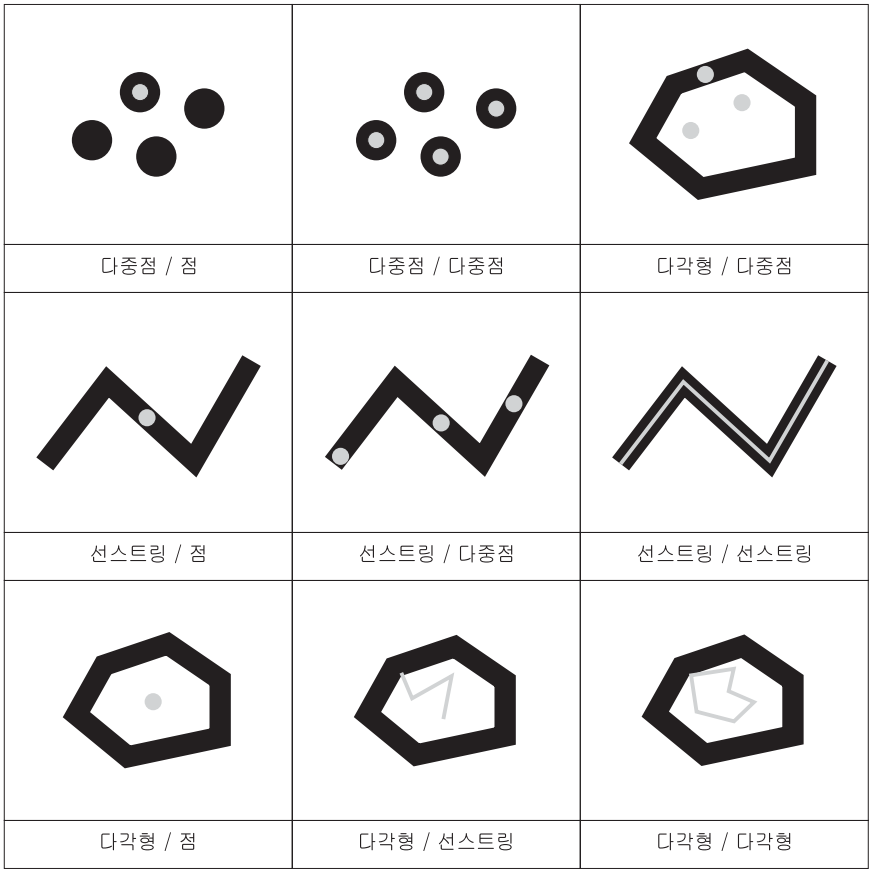


그림 44. ST_Contains. 검은색 Geometry는 Geometry a를 나타내고 회색 Geometry는 Geometry b를 나타냅니다. 모든 경우에 Geometry a는 Geometry b를 완전히 포함합니다.

ST_Contains 함수의 패턴 행렬은 두 Geometry의 내부가 교차하고 2차(Geometry*b*)의 내부 또는 바운더리가 1차(Geometry *a*)의 외부와 교차하지 않아야 함을 나타냅니다. 별표(*)는 Geometry의 일부 파트에 교차가 존재한다 하더라도 중요하지 않음을 나타냅니다.

표 43. ST_Contains 행렬

	Geometry b 내부	**Geometry b 바운더리**	**Geometry b 외부**
Geometry a 내부	T	*	*
Geometry a 바운더리	*	*	*
Geometry a 외부	F	F	*

ST_Within

ST_Within을 사용하여 한 Geometry가 다른 Geometry 내에 완전히 포함되는지 여부를 판별하십시오.

ST_Within은 첫 번째 Geometry가 두 번째 Geometry 내에 완전히 포함되면 값 1을 리턴합니다. ST_Within은 ST_Contains와 정반대의 결과를 리턴합니다.

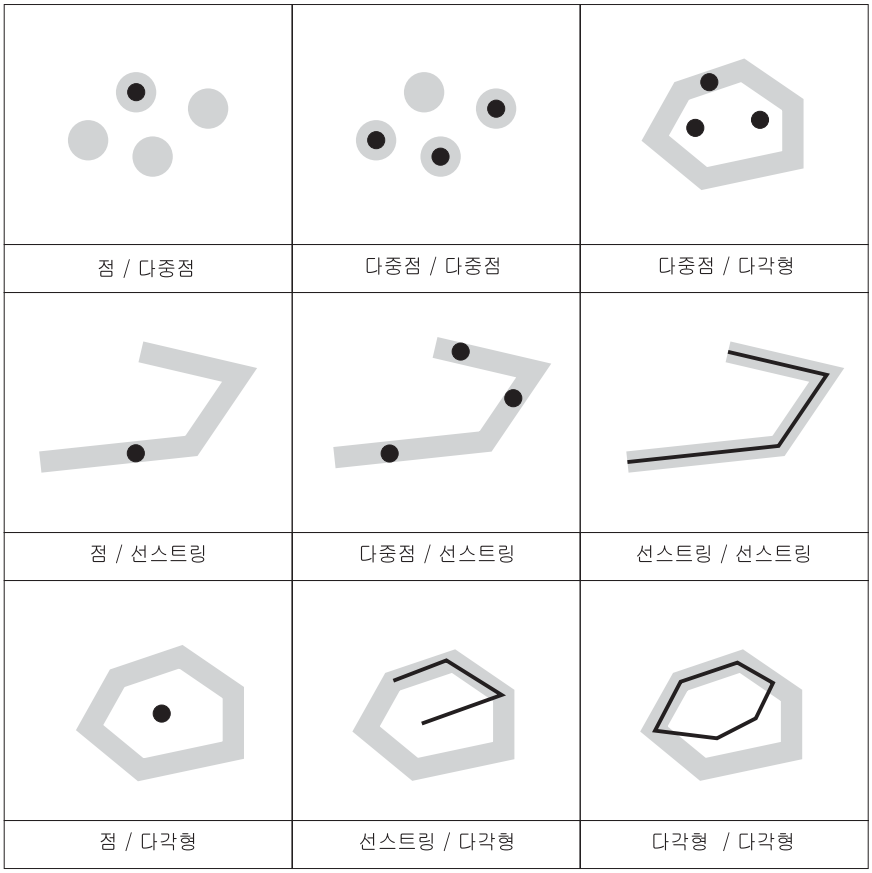


그림 45. ST_Within

ST_Within 함수 패턴 행렬은 두 Geometry 내부가 서로 교차하고 1차 Geometry(Geometry *a*)의 내부 또는 바운더리가 2차(Geometry *b*)의 외부와 교차하지 않아야 함을 나타냅니다. 별표(*)는 다른 모든 교차가 중요하지 않음을 나타냅니다.

표 44. ST_Within 행렬

	Geometry b 내부	**Geometry b 바운더리**	**Geometry b 외부**
Geometry a 내부	T	*	F
Geometry a 바운더리	*	*	F
Geometry a 외부	*	*	*

그림 45에서는 ST_Within 예가 표시됩니다.

- 내부가 두 번째 Geometry 내부의 한 점과 교차하는 경우 점 Geometry는 다중점 Geometry 내부에 있습니다.

- 모든 점의 내부가 두 번째 Geometry와 교차하는 경우 다중점 Geometry는 다중점 Geometry 내부에 있습니다.
- 모든 점이 다각형의 바운더리 또는 다각형 내부에 있는 경우 다중점 Geometry는 다각형 Geometry 내부에 있습니다.
- 모든 점이 두 번째 Geometry 내부에 있는 경우 점 Geometry는 선스트링 Geometry 내부에 있습니다. 303 페이지의 그림 45의 경우, 그 내부가 선스트링과 교차하지 않으므로 점은 선스트링 내부에 없습니다. 그러나 다중점 Geometry의 모든 점은 선스트링의 내부와 교차하므로 이 Geometry는 선스트링 내부에 있습니다.
- 모든 점이 두 번째 Geometry와 교차하는 경우 선스트링 Geometry는 다른 선스트링 내부에 있습니다.
- 내부가 다각형의 바운더리 또는 내부와 교차하지 않으므로 점 Geometry는 다각형 Geometry 내부에 없습니다.
- 모든 점이 다각형의 바운더리 또는 내부와 교차하는 경우 선스트링 Geometry는 다각형 Geometry 내부에 있습니다.
- 모든 점이 다각형의 바운더리 또는 내부와 교차하는 경우 다각형 Geometry는 다각형 Geometry 내부에 있습니다.

Geometry 사이의 교차를 점검하는 함수

ST_Intersects, ST_Crosses, ST_Overlaps 및 ST_Touches는 모두 한 Geometry가 다른 Geometry와 교차되는지 여부를 판별합니다. 이 함수의 차이점은 테스트하는 범위입니다.

- ST_Intersects는 제공된 두 Geometry가 네 개의 조건(Geometry의 내부가 교차되는지, 바운더리가 교차되는지, 첫 번째 Geometry의 바운더리가 두 번째 Geometry의 내부와 교차되는지 또는 첫 번째 Geometry의 내부가 두 번째 Geometry의 바운더리와 교차되는지 여부) 중 하나를 만족하는지 판별하기 위해 테스트합니다.
- ST_Crosses는 다른 차원의 Geometry 교차를 분석하는 데 사용되며 한 가지 예외는 선스트링의 교차도 분석할 수 있다는 점입니다. 모든 경우에서 교차 위치 자체가 Geometry로 간주되며 ST_Crosses에서는 이 Geometry 차원이 교차하는 Geometry의 차원보다 작아야 합니다(또는 모두가 선스트링인 경우 교차 위치가 선스트링의 차원보다 작아야 함). 예를 들어, 선스트링 및 다각형의 차원은 각각 1과 2입니다. 이러한 두 Geometry가 교차하고 교차 위치가 선형(다각형을 따르는 선스트링의 경로)일 경우 해당 위치 자체를 선스트링으로 간주할 수 있습니다. 선스트링의 차원(1)은 다각형의 차원(2)보다 작으므로 교차를 분석한 후 ST_Crosses는 값 1을 리턴합니다.
- ST_Overlaps에 입력으로 제공된 Geometry는 차원이 동일해야 합니다. ST_Overlaps에서는 이러한 Geometry가 일부 겹쳐 동일한 차원의 새 Geometry(겹쳐지는 영역)를 형성해야 합니다.
- ST_Touches는 두 Geometry 바운더리가 교차하는지 여부를 판별합니다.

ST_Intersects

ST_Intersects를 사용하여 두 Geometry가 교차되는지 여부를 판별하십시오.

ST_Intersects는 교차 결과가 빈 세트가 아니면 값 1을 리턴합니다. ST_Intersects는 ST_Disjoint와 정반대의 결과를 리턴합니다.

ST_Intersects 함수는 다음 패턴 행렬의 조건이 TRUE를 리턴하면 1을 리턴합니다.

표 45. ST_Intersects 행렬(1). ST_Intersects 함수는 두 Geometry 내부가 교차할 경우 1을 리턴합니다.

	Geometry b 내부	**Geometry b 바운더리**	**Geometry b 외부**
Geometry a 바운더리	*	*	*
Geometry a 내부	T	*	*
Geometry a 외부	*	*	*

표 46. ST_Intersects 행렬(2). ST_Intersects 함수는 첫 번째 Geometry의 바운더리가 두 번째 Geometry의 바운더리와 교차하는 경우 1을 리턴합니다.

	Geometry b 내부	**Geometry b 바운더리**	**Geometry b 외부**
Geometry a 바운더리	*	*	*
Geometry a 내부	*	T	*
Geometry a 외부	*	*	*

표 47. ST_Intersects 행렬(3). ST_Intersects 함수는 첫 번째 Geometry의 바운더리가 두 번째 Geometry의 내부와 교차하는 경우 1을 리턴합니다.

	Geometry b 내부	**Geometry b 바운더리**	**Geometry b 외부**
Geometry a 바운더리	T	*	*
Geometry a 내부	*	*	*
Geometry a 외부	*	*	*

표 48. ST_Intersects 행렬(4). ST_Intersects 함수는 두 Geometry 바운더리가 교차하는 경우 1을 리턴합니다.

	Geometry b 내부	**Geometry b 바운더리**	**Geometry b 외부**
Geometry a 바운더리	*	T	*
Geometry a 내부	*	*	*
Geometry a 외부	*	*	*

ST_Crosses

ST_Crosses를 사용하여 한 Geometry가 다른 Geometry에 교차하는지 여부를 판별하십시오.

ST_Crosses는 두 Geometry를 사용하며 다음의 경우 값 1을 리턴합니다.

- 교차 결과, 소스 Geometry의 최대 차원보다 낮은 Geometry이 됩니다.
- 교차 세트는 두 소스 Geometry 모두의 내부입니다.

ST_Crosses는 첫 번째 Geometry가 표면 또는 다중 표면이거나 두 번째 Geometry가 점 또는 다중점인 경우 널(NULL)을 리턴합니다. 기타 모든 조합에서 ST_Crosses는 값 1(두 Geometry가 교차함을 표시) 또는 값 0(교차하지 않음을 표시)을 리턴합니다.

다음 그림은 선스트링을 지나는 다중점, 선스트링을 지나는 선스트링, 다각형을 지나는 다중점 및 다각형을 지나는 선스트링을 보여줍니다. 네 가지 경우 중 세 가지 경우에서 Geometry b는 Geometry a를 지납니다. 네 번째 경우에서 Geometry a는 선을 지나지 않는 다중점이지만 Geometry b 다각형 내부의 영역은 접촉합니다.

검은색 Geometry는 Geometry a를 나타내고 회색 Geometry는 Geometry b를 나타냅니다.

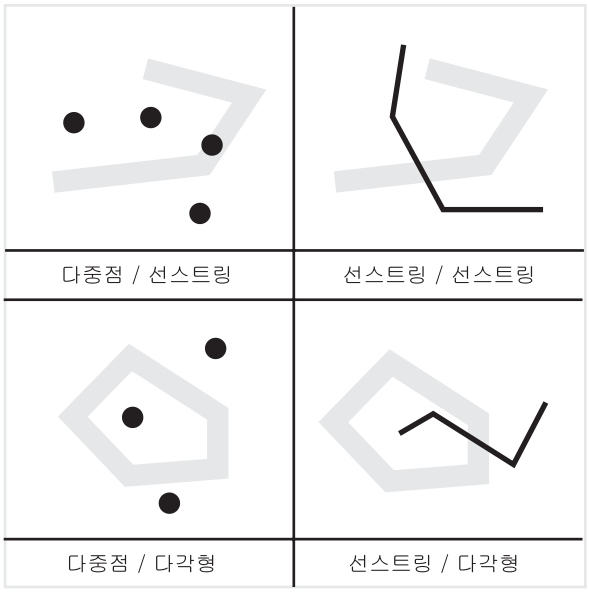


그림 46. ST_Crosses

표 49의 패턴 행렬은 첫 번째 Geometry가 점 또는 다중점이거나 첫 번째 Geometry가 곡선 또는 다중 곡선이고 두 번째 Geometry가 표면인 경우에 적용됩니다. 행렬은 내부가 교차해야 하고 1차의 내부(Geometry *a*)가 2차의 외부(Geometry *b*)와 교차해야 함을 나타냅니다.

표 49. ST_Crosses 행렬(1)

	Geometry b 내부	**Geometry b 바운더리**	**Geometry b 외부**
Geometry a 바운더리	*	*	*
Geometry a 내부	T	*	T
Geometry a 외부	*	*	*

307 페이지의 표 50의 패턴 행렬은 첫 번째 및 두 번째 Geometry가 둘 다 곡선 또는 다중 곡선일 경우에 적용됩니다. 0은 내부 교차가 점이어야 함을 나타냅니다(차원 0). 이 교차의 차원이 1(선스트링에서 교차)인 경우 ST_Crosses 함수는 값 0(Geometry가 교차되지 않음을 표시)을 리턴하지만 ST_Overlaps 함수는 값 1(Geometry가 겹침을 표

시)을 리턴합니다.

표 50. ST_Crosses 행렬(2)

	Geometry b 내부	Geometry b 바운더리	Geometry b 외부
Geometry a 바운더리	*	*	*
Geometry a 내부	0	*	*
Geometry a 외부	*	*	*

ST_Overlaps

ST_Overlaps를 사용하여 동일한 차원의 두 Geometry가 겹치는지 여부를 판별하십시오.

ST_Overlaps는 동일한 차원의 두 Geometry를 비교합니다. 교차 세트로 인해 서로 다른 Geometry가 발생하지만 차원은 동일한 경우 값 1을 리턴합니다.

검은색 Geometry는 Geometry *a*를 나타내고 회색 Geometry는 Geometry *b*를 나타냅니다. 모든 경우에 두 Geometry는 동일한 차원을 가지며 한 Geometry는 다른 Geometry와 일부 겹칩니다. 겹치는 영역은 새 Geometry입니다. 이는 Geometry *a* 및 *b*와 동일한 차원을 갖습니다.

다음 그림은 Geometry의 겹침을 보여줍니다. 세 가지 예로 점, 선스트링 및 다각형의 겹침을 보여줍니다. 점의 경우, 실제 점이 겹칩니다. 선스트링의 경우, 선의 일부가 겹칩니다. 다각형의 경우, 영역의 일부가 겹칩니다.

그림 47. ST_Overlaps

표 51의 패턴 행렬은 첫 번째 및 두 번째 Geometry가 점, 다중점, 표면 또는 다중 표면일 경우에 적용됩니다. ST_Overlaps는 각 Geometry의 내부가 다른 Geometry의 내부 및 외부와 교차할 경우에 값 1을 리턴합니다.

표 51. ST_Overlaps 행렬(1)

	Geometry b 내부	Geometry b 바운더리	Geometry b 외부
Geometry a 바운더리	*	*	*
Geometry a 내부	T	*	T
Geometry a 외부	T	*	*

표 52의 패턴 행렬은 첫 번째 및 두 번째 Geometry가 둘 다 곡선 또는 다중 곡선일 경우에 적용됩니다. 이러한 경우의 Geometry 교차 결과는 1차원 Geometry(다른 곡선)이어야 합니다. 내부 교차의 차원이 0인 경우, ST_Overlaps는 값 0(Geometry가 겹치지 않음을 표시)을 리턴하지만 ST_Crosses 함수는 값 1(Geometry가 교차됨을 표시)을 리턴합니다.

표 52. ST_Overlaps 행렬(2)

	Geometry b 내부	**Geometry b 바운더리**	**Geometry b 외부**
Geometry a 바운더리	*	*	*
Geometry a 내부	1	*	T
Geometry a 외부	T	*	*

ST_Touches

ST_Touches는 두 Geometry에 공통인 모든 점이 바운더리에만 있을 경우 값 1을 리턴합니다.

Geometry의 내부는 다른 부분과 교차하지 않아야 합니다. 적어도 하나의 Geometry가 곡선, 표면, 다중 곡선 또는 다중 표면이어야 합니다.

검은색 Geometry는 Geometry a를 나타내고 회색 Geometry는 Geometry b를 나타냅니다. 모든 경우에 Geometry b의 바운더리는 Geometry a와 교차합니다. Geometry b의 내부는 Geometry a와 분리되어 있습니다.

다음 그림은 Geometry 유형이 교차하는 예(즉, 점과 선스트링, 선스트링과 선스트링, 점과 다각형, 다중점과 다각형 및 선스트링과 다각형)을 보여줍니다.

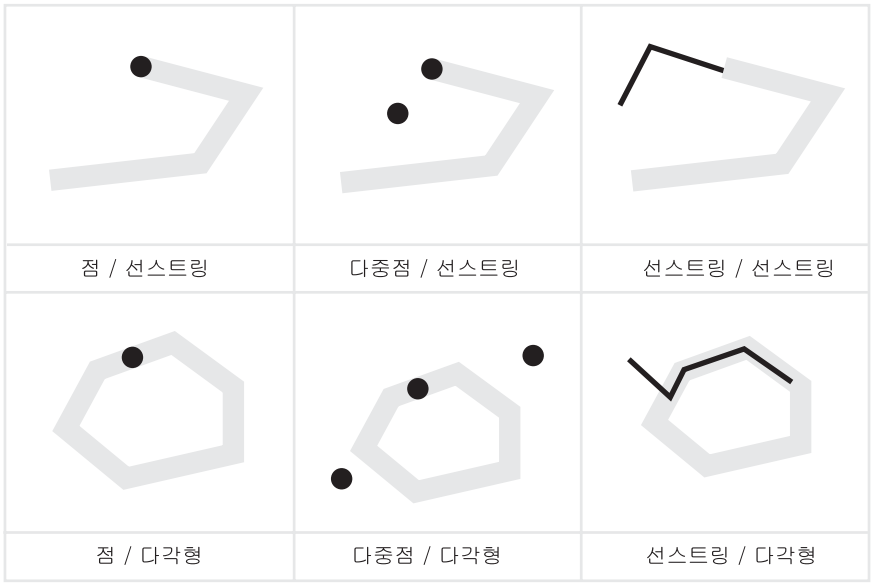


그림 48. ST_Touches

패턴 행렬에서는 Geometry의 내부가 교차하고 Geometry의 바운더리가 다른 내부 또는 바운더리와 교차할 때 ST_Touches 함수가 1를 리턴한다는 것을 보여줍니다.

표 53. ST_Touches에 대한 행렬(1)

	Geometry b 내부	**Geometry b 바운더리**	**Geometry b 외부**
Geometry a 바운더리	*	*	*
Geometry a 내부	F	T	*
Geometry a 외부	*	*	*

표 54. ST_Touches에 대한 행렬(2)

	Geometry b 내부	**Geometry b 바운더리**	**Geometry b 외부**
Geometry a 바운더리	T	*	*
Geometry a 내부	F	*	*
Geometry a 외부	*	*	*

표 55. ST_Touches에 대한 행렬(3)

	Geometry b 내부	**Geometry b 바운더리**	**Geometry b 외부**
Geometry a 바운더리	*	T	*
Geometry a 내부	F	*	*
Geometry a 외부	*	*	*

Geometry의 인벨로프를 비교하는 함수

ST_EnvIntersects 및 ST_MBRIntersects는 한 Geometry를 둘러싸는 가장 작은 직사각형이 다른 Geometry를 둘러싸는 가장 작은 직사각형과 교차하는지를 판별합니다. 이러한 직사각형을 일반적으로 인벨로프(envelope)라고 합니다. 다중 다각형, 다각형, 다중 선스트링 및 구부러진 선스트링은 해당 인벨로프의 변에 접해 있습니다. 수평 선스트링, 수직 선스트링 및 점은 인벨로프보다 약간 작습니다. ST_EnvIntersects는 테스트를 수행하여 Geometry의 인벨로프가 교차하는지를 판별합니다.

Geometry가 꼭 맞을 수 있는 가장 작은 직사각형 영역을 MBR(Minimun Bounding Rectangle)이라고 합니다. 다중 다각형, 다각형, 다중 선스트링 및 구부러진 선스트링 주위의 인벨로프는 실제로 MBR입니다. 그러나 후자 Goemetry는 꼭 맞는 최소 영역을 구성하지 않기 때문에 수평 선스트링, 수직 선스트링 및 점을 둘러싸는 인벨로프는 MBR이 아닙니다. 후자 Geometry는 정의할 수 있는 스페이스를 점유하지 않으므로 MBR을 가질 수 없습니다. 그럼에도 불구하고 이는 자체 MBR로 참조되기 때문에 규약이 채택되었습니다. 따라서 다중 다각형, 다각형, 다중 선스트링 및 구부러진 선스트링의 경우, ST_MBRIntersects는 ST_EnvIntersects가 테스트하는 것과 동일한 주변 직사각형과 교차하는지를 테스트합니다. 그러나 수평 선스트링, 수직 선스트링 및 점의 경우 ST_MBRIntersects는 이러한 Geometry 자체와 교차하는지를 테스트합니다.

ST_EnvIntersects

ST_EnvIntersects는 두 Geometry의 인벨로프(envelope)가 교차하면 값 1을 리턴합니다. 이 함수는 ST_Intersects(ST_Envelope(g1),ST_Envelope(g2))를 효율적으로 구현하는 편리한 함수입니다.

ST_MBRIntersects

ST_MBRIntersects는 두 Geometry의 MBR(Minimum Bounding Rectangle)이 교차하면 값 1을 리턴합니다.

두 개가 동일한지 점검하는 함수

ST_EqualCoordsys

ST_EqualCoordsys는 두 좌표 시스템 정의가 동일하면 값 1을 리턴합니다.

정의를 비교할 때 ST_EqualCoordsys는 대소문자, 스페이스, 괄호 및 부동 소수점 숫자 표현의 차이는 무시합니다.

ST_Equals

ST_Equals는 두 Geometry가 동일하면 값 1을 리턴합니다.

Geometry를 정의하는 데 사용되는 점의 순서는 동등 테스트에 관련되지 않습니다.

여섯 개의 예(점, 다중점, 선스트링, 다중 스트링, 다각형 및 다중 다각형)에서 Geometry a와 Geometry b는 동일합니다.

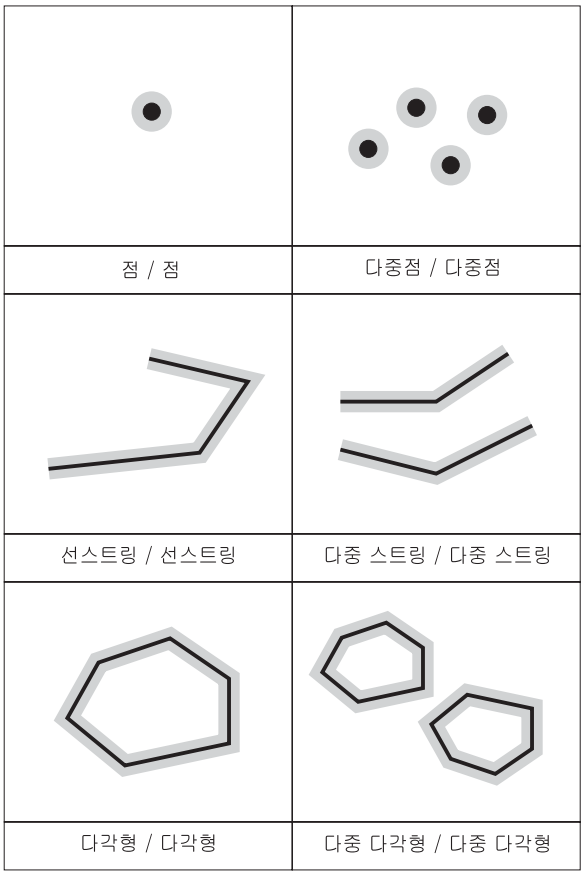


그림 49. ST_Equals. 검정색 Geometry는 Geometry a를 나타내고 회색 Geometry는 Geometry b를 나타냅니다. 모든 경우에 Geometry a는 Geometry b와 동일합니다.

표 56. 동등 행렬. 동등성에 대한 DE-9IM 패턴 행렬을 통해, 내부가 교차하고 둘 중 한 Geometry의 파트 내부 또는 바운더리 어느 것도 서로의 외부와 교차하지 않는다는 것을 확인할 수 있습니다.

	Geometry b 내부	**Geometry b 바운더리**	**Geometry b 외부**
Geometry a 바운더리	*	*	F
Geometry a 내부	T	*	F
Geometry a 외부	F	F	*

ST_EqualSRS

ST_EqualSRS를 사용하여 두 공간 참조 시스템이 동일한지 테스트하십시오.

ST_EqualSRS는 두 공간 참조 시스템이 동일하고 두 시스템 중 하나 또는 모두의 숫자 ID가 널이 아닌 경우 값 1을 리턴합니다.

두 Geometry에 교차가 없는지 점검하는 함수

ST_Disjoint는 두 Geometry의 교차가 빈 세트인 경우 값 1을 리턴합니다. 이 함수는 ST_Intersects가 리턴하는 값과 정반대 값을 리턴합니다.

이 예는 다른 Geometry 및 어떠한 위치에서도 바운더리가 교차하지 않는 이유를 보여줍니다.

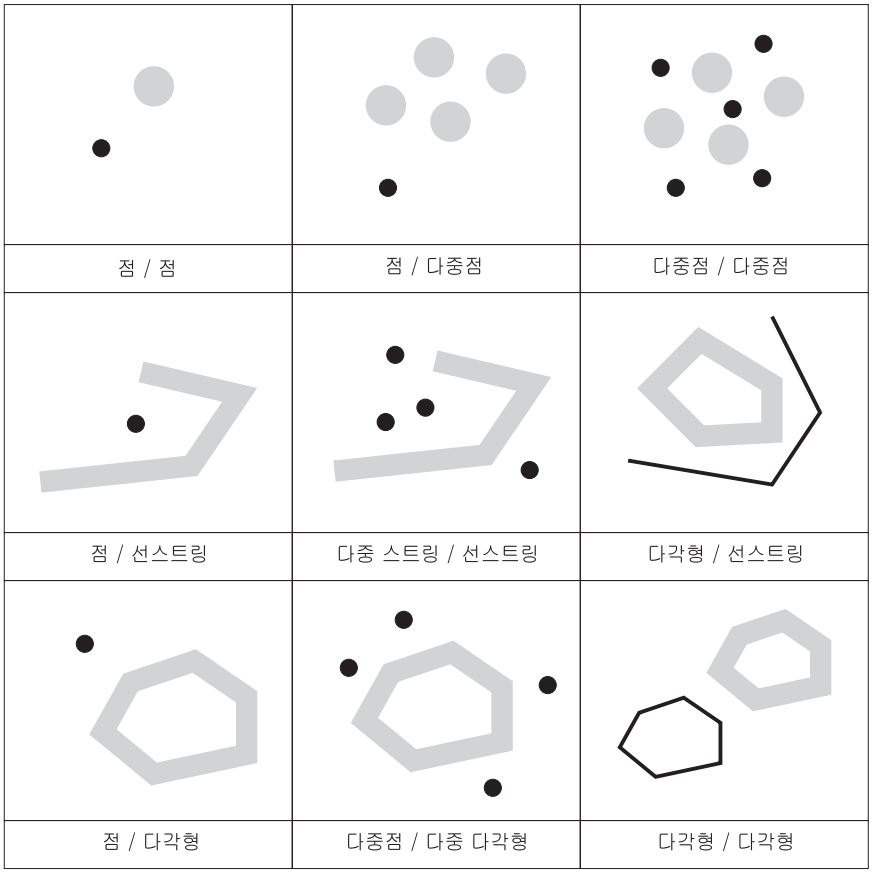


그림 50. ST_Disjoint. 검은색 Geometry는 Geometry a를 나타내고 회색 Geometry는 Geometry b를 나타냅니다. 모든 경우에 Geometry a 및 Geometry b는 교차하지 않습니다.

표 57. ST_Disjoint 행렬. 이 행렬은 단순히 Geometry의 내부 또는 바운더리가 모두 교차하지 않음을 나타냅니다.

	Geometry b 내부	**Geometry b 바운더리**	**Geometry b 외부**
Geometry a 바운더리	F	F	*
Geometry a 내부	F	F	*
Geometry a 외부	*	*	*

Geometry와 DE-9IM 패턴 행렬 문자열을 비교하는 함수

ST_Relate 함수는 두 Geometry를 비교하고 Geometry가 DE-9IM 패턴 행렬 문자열에서 지정하는 조건에 일치하면 값 1을 리턴하고 그렇지 않으면 값 0을 리턴합니다.

Geometry 등록 정보에 대한 정보를 리턴하는 함수

이 절에서는 Geometry 특성에 대한 정보를 리턴하는 공간 함수를 소개합니다. 다음에 대한 정보가 들어 있습니다.

- Geometry 데이터 유형
- Geometry 내의 좌표 및 측정 단위
- 링, 바운더리, 인벨로프(envelope) 및 MBR(Minimum Bounding Rectangle)
- 차원
- 닫혔거나 비었거나 단순한 품질
- Geometry 콜렉션 내의 기본 Geometry
- 공간 참조 시스템

일부 특성은 당연히 Geometry입니다. 예를 들어, 표면의 외부 및 내부 링 또는 곡선의 시작 및 끝점이 있습니다. 이러한 Geometry는 이 범주의 일부 함수에 의해 작성됩니다. 다른 종류의 Geometry(예: 지정된 위치를 둘러싸는 지역을 나타내는 Geometry)를 생성하는 함수는 다른 범주에 속합니다. 다른 범주(『새 범주를 생성하는 공간 함수』라고 함)에 대한 정보는 이 절의 끝에 있는 적절한 링크 또는 상호 참조를 참조하십시오.

데이터 유형 정보를 리턴하는 함수

ST_GeometryType은 입력 매개변수로 Geometry를 가져오며 해당 Geometry 동적 유형의 완전한 유형 이름을 리턴합니다.

좌표 및 측정 정보를 리턴하는 함수

다음 함수는 Geometry 내의 좌표 및 측정 단위에 대한 정보를 리턴합니다. 예를 들어, ST_X는 지정된 점 내의 X 좌표를 리턴할 수 있으며 ST_MaxX는 Geometry 내의 최대 X 좌표를 리턴하고 ST_MinX는 Geometry 내의 최소 X 좌표를 리턴합니다.

함수는 다음과 같습니다.

- ST_CoordDim
- ST_IsMeasured
- ST_IsValid

- ST_Is3D
- ST_M
- ST_MaxM
- ST_MaxX
- ST_MaxY
- ST_MaxZ
- ST_MinM
- ST_MinX
- ST_MinY
- ST_MinZ
- ST_X
- ST_Y
- ST_Z

ST_CoordDim

ST_CoordDim은 Geometry의 좌표 유형과 Geometry에 수치가 포함되는지 여부를 표시하는 값을 리턴합니다.

이 값을 *좌표 차원*이라고 합니다. 좌표 차원은 차원으로 참조되는 등록 정보와 다릅니다. 후자는 Geometry가 폭 또는 길이를 갖고 있는지 아니면 특정 유형 또는 측정 단위의 좌표를 포함하지 않는지를 나타냅니다.

ST_IsMeasured

ST_IsMeasured는 입력 매개변수로 Geometry를 가져오며 제공된 Geometry에 M 좌표(측정 단위)가 있는 경우 1을 리턴합니다. 그렇지 않은 경우에는 0(영)이 리턴됩니다.

ST_IsValid

ST_IsValid는 Geometry를 입력 매개변수로 가져오며 유효한 경우 1을 리턴합니다. 그렇지 않은 경우에는 0(영)이 리턴됩니다. Geometry는 구조화된 유형의 모든 속성이 Geometry 데이터의 내부 표현과 일치할 경우와 내부 표현이 손상되지 않은 경우에만 유효합니다.

ST_Is3D

ST_Is3d는 입력 매개변수로 Geometry를 가져오며 제공된 Geometry에 Z 좌표가 있는 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다.

ST_M

제공된 점에 측정 단위를 저장하지 않으면 ST_M은 점을 입력 매개변수로 가져오며 측정 단위를 리턴합니다.

ST_MaxM

ST_MaxM은 입력 매개변수로 Geometry를 가져오며 해당 최대 수치를 리턴합니다.

ST_MaxX

ST_MaxX는 입력 매개변수로 Geometry를 가져오며 해당 최대 X 좌표를 리턴합니다.

ST_MaxY

ST_MaxY는 입력 매개변수로 Geometry를 가져오며 해당 최대 Y 좌표를 리턴합니다.

ST_MaxZ

ST_MaxZ는 입력 매개변수로 Geometry를 가져오며 해당 최대 Z 좌표를 리턴합니다.

ST_MinM

ST_MinM은 입력 매개변수로 Geometry를 가져오며 해당 최소 수치를 리턴합니다.

ST_MinX

ST_MinX는 입력 매개변수로 Geometry를 가져오며 해당 최소 X 좌표를 리턴합니다.

ST_MinY

ST_MinY는 입력 매개변수로 Geometry를 가져오며 해당 최소 Y 좌표를 리턴합니다.

ST_MinZ

ST_MinY는 입력 매개변수로 Geometry를 가져오며 해당 최소 Z 좌표를 리턴합니다.

ST_X

ST_X는 점을 입력 매개변수로 가져오며 점의 X 좌표를 리턴합니다.

ST_Y

ST_Y는 점을 입력 매개변수로 가져오며 점의 Y 좌표를 리턴할 수 있습니다.

ST_Z

제공된 점에 Z 좌표가 저장되어 있으면 ST_Z는 점을 입력 매개변수로 가져오며 Z 좌표를 리턴할 수 있습니다.

Geometry 내의 Geometry에 대한 정보를 리턴하는 함수

다음 함수는 Geometry 내의 Geometry에 대한 정보를 리턴합니다. 일부 함수는 Geometry 내의 특정 점을 식별하고 다른 함수는 콜렉션 내의 기본 Geometry 수를 리턴합니다.

함수는 다음과 같습니다.

- ST_Centroid
- ST_EndPoint
- ST_GeometryN
- ST_LineStringN
- ST_MidPoint
- ST_NumGeometries
- ST_NumLineStrings
- ST_NumPoints
- ST_NumPolygons
- ST_PointN
- ST_PolygonN
- ST_StartPoint

ST_Centroid

ST_Centroid는 입력 매개변수로 Geometry를 가져오며 점으로 제공된 Geometry의 최소 바운딩 직사각형의 중심인 Geometry 중심을 리턴합니다.

ST_EndPoint

ST_Endpoint는 입력 매개변수로 곡선을 가져오며 곡선의 마지막 점을 리턴합니다.

ST_GeometryN

ST_GeometryN은 입력 매개변수로 Geometry 콜렉션과 인덱스를 가져오며 인덱스로 식별되는 콜렉션의 Geometry를 리턴합니다.

ST_LineStringN

ST_LineStringN은 입력 매개변수로 다중 선스트링 및 인덱스를 가져오며 인덱스에 의해 식별되는 선스트링을 리턴합니다.

ST_MidPoint

ST_MidPoint는 입력 매개변수로 곡선을 가져오며 곡선과 함께 측정된 곡선의 양쪽 끝점에서 같은 거리에 있는 곡선의 점을 리턴합니다.

ST_NumGeometries

ST_NumGeometries는 Geometry 콜렉션을 입력 매개변수로 가져오며 콜렉션 내의 Geometry 수를 리턴합니다.

ST_NumLineStrings

ST_NumLineStrings는 다중 선스트링을 입력 매개변수로 가져오며 포함하는 선스트링 수를 리턴합니다.

ST_NumPoints

ST_NumPoints는 Geometry를 입력 매개변수로 가져오며 해당 Geometry를 정의하는 데 사용된 점의 수를 리턴합니다. 예를 들어, Geometry가 다각형이고 이 다각형을 정의하는 데 다섯 개의 점이 사용된 경우 숫자 5가 리턴됩니다.

ST_NumPolygons

ST_NumPolygons는 다중 다각형을 입력 매개변수로 가져오며 포함하는 다각형의 수를 리턴합니다.

ST_PointN

ST_PointN은 입력 매개변수로 선스트링이나 다중점 및 인덱스를 가져오며 인덱스에 의해 식별되는 선스트링이나 다중점의 점을 리턴합니다.

ST_PolygonN

ST_PolygonN은 입력 매개변수로 다중 다각형 및 인덱스를 가져오며 인덱스에 의해 식별되는 다각형을 리턴합니다.

ST_StartPoint

ST_StartPoint는 입력 매개변수로 곡선을 가져오며 곡선의 시작 점을 리턴합니다.

바운더리, 인벨로프 및 링에 대한 정보를 표시하는 함수

다음 함수는 Geometry의 내부를 외부와 나누거나 Geometry 자체를 외부 스페이스와 나누는 경계에 대한 정보를 리턴합니다. 예를 들어, ST_Boundary는 곡선 양식으로 Geometry의 바운더리를 리턴합니다.

함수는 다음과 같습니다.

- ST_Boundary
- ST_Envelope
- ST_EnvIntersects
- ST_ExteriorRing

- ST_InteriorRingN
- ST_MBR
- ST_MBRIntersects
- ST_NumInteriorRing
- ST_Perimeter

ST_Envelope

ST_Envelope는 입력 매개변수로 Geometry를 가져오며 Geometry 주위에 인벨로프를 리턴합니다. 인벨로프는 다각형으로 표시되는 직사각형입니다.

ST_EnvIntersects

ST_EnvIntersect는 입력 매개변수로 Geometry를 가져오며 두 Geometry의 인벨로프가 교차하는 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다.

ST_ExteriorRing

ST_ExteriorRing은 입력 매개변수로 다각형을 가져오며 해당 외부 링을 곡선으로 리턴합니다.

ST_InteriorRingN

ST_InteriorRingN은 입력 매개변수로 다각형과 인덱스를 가져오며 제공된 인덱스로 식별되는 내부 링을 선스트링으로 리턴합니다. 내부 링은 내부 Geometry 검증 루틴에서 정의한 규칙에 따라 구성됩니다.

ST_MBR

ST_MBR은 입력 매개변수로 Geometry를 가져오며 해당 최소 바운딩 직사각형을 리턴합니다.

ST_MBRIntersects

ST_MBRIntersects는 두 Geometry의 MBR(Minimum Bounding Rectangle)이 교차하면 값 1을 리턴합니다.

ST_NumInteriorRing

ST_NumInteriorRing은 입력 매개변수로 다각형을 가져오며 내부 링의 수를 리턴합니다.

ST_Perimeter

ST_Perimeter는 표면 또는 다중 표면, 단위(선택적)를 입력 매개변수로 가져오며 제공된 단위로 측정된 표면 또는 다중 표면의 둘레(즉, 바운더리 길이)를 리턴합니다.

Geometry 차원에 대한 정보를 리턴하는 함수

다음 함수는 Geometry의 차원에 대한 정보를 리턴합니다. 예를 들어, ST_Area는 제공된 Geometry가 담당하는 영역을 보고합니다.

함수는 다음과 같습니다.

- ST_Area
- ST_Dimension
- ST_Length

ST_Area

ST_Area는 Geometry와 단위(선택적)를 입력 매개변수로 가져오며 제공된 측정 단위로 제공된 Geometry가 적용할 수 있는 영역을 리턴합니다.

ST_Dimension

ST_Dimension은 입력 매개변수로 Geometry를 가져오며 차원을 리턴합니다.

ST_Length

ST_Length는 곡선 또는 다중 곡선, 단위(선택적)를 입력 매개변수로 가져오며 제공된 측정 단위로 제공된 곡선 또는 다중 곡선의 길이를 리턴합니다.

Geometry가 닫혔는지, 비었는지 또는 단순한지 여부를 표시하는 함수

다음 함수는 다음과 같은 정보를 표시합니다.

- 제공된 곡선이 닫혀 있는지 여부(즉, 곡선 또는 다중 곡선의 시작점 또는 끝점이 동일함)
- 제공된 Geometry가 비어 있는지 여부(즉, 점이 없음)
- 곡선, 다중 곡선 또는 다중점이 단순한지 여부(즉, 이러한 Geometry가 일반적인 구성을 갖고 있는지 여부)

ST_IsClosed

ST_IsClosed는 입력 매개변수로 곡선 또는 다중 곡선을 가져오며 제공된 곡선이나 다중 곡선이 닫혀 있는 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다.

ST_IsEmpty

ST_IsEmpty는 입력 매개변수로 Geometry를 가져오며 제공된 Geometry가 비어 있는 경우 1을 리턴합니다. 그렇지 않은 경우에는 0(영)이 리턴됩니다.

ST_IsSimple

ST_IsSimple은 입력 매개변수로 Geometry를 가져오며 제공된 Geometry가 단순한 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다.

Geometry의 공간 참조 시스템을 식별하는 함수

다음 함수는 Geometry에 연관된 공간 참조 시스템을 식별하는 값을 리턴합니다. 또한, ST_SrsID 함수는 Geometry를 변경하거나 변환하지 않고 Geometry의 공간 참조 시스템을 변경할 수 있습니다.

ST_SrsId(ST_SRID라고도 함)

ST_SrsId(또는 ST_SRID)는 Geometry 및 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져옵니다. 리턴되는 이 정보는 지정된 입력 매개변수에 따라 다릅니다.

- 공간 참조 시스템 ID를 지정하면 이 함수는 지정된 공간 참조 시스템으로 변경된 공간 참조 시스템과 함께 Geometry를 리턴합니다. Geometry는 변환되지 않습니다.
- 입력 매개변수로 공간 참조 시스템 ID를 제공하지 않은 경우 제공된 Geometry의 현재 공간 참조 시스템 ID가 리턴됩니다.

ST_SrsName

ST_SrsName은 입력 매개변수로 Geometry를 가져오며 제공된 Geometry가 표시되는 공간 참조 시스템의 이름을 리턴합니다.

기존 Geometry에서 새 Geometry를 생성하는 함수

이 절에서는 기존의 Geometry에서 새 Geometry를 파생하는 함수 범주를 소개합니다. 이 범주는 다른 Geometry의 등록 정보를 나타내는 Geometry를 파생하는 함수를 포함하지는 않습니다. 그보다는 다음과 같은 함수에 대한 범주입니다.

- Geometry를 다른 Geometry로 변환
- 스페이스의 구성을 나타내는 Geometry 작성
- 다중 Geometry로부터 개별적 Geometry 파생
- 측정 단위를 기본으로 하는 Geometry 작성
- Geometry의 수정 작성

Geometry에서 다른 Geometry로 변환하는 함수

다음 함수는 슈퍼 유형의 Geometry를 부속 유형을 가진 해당 Geometry로 변환합니다. 예를 들어, ST_ToLineString 함수는 ST_Geometry 유형의 선스트링을 ST_LineString 선스트링으로 변환할 수 있습니다. 이 함수 중 일부는 기본 Geometry 및 Geometry 콜렉션을 단일 Geometry 콜렉션으로 조합할 수 있습니다. 예를 들어, ST_ToMultiLine은 선스트링 및 다중 선스트링을 단일 다중 선스트링으로 변환할 수 있습니다.

ST_Polygon

ST_Polygon은 닫힌 선스트링으로부터 다각형을 생성할 수 있습니다. 선스트링은 다각형의 외부 링을 정의합니다.

ST_ToGeomColl

ST_ToGeomColl은 입력 매개변수로 Geometry를 가져오며 이를 Geometry 콜렉션으로 변환합니다.

ST_ToLineString

ST_ToLineString은 입력 매개변수로 Geometry를 가져오며 이를 선스트링으로 변환합니다.

ST_ToMultiLine

ST_ToMultiLine은 입력 매개변수로 Geometry를 가져오며 이를 다중 선스트링으로 변환합니다.

ST_ToMultiPoint

ST_ToMultiPoint는 입력 매개변수로 Geometry를 가져오며 이를 다중점으로 변환합니다.

ST_ToMultiPolygon

ST_ToMultiPolygon은 입력 매개변수로 Geometry를 가져오며 이를 다중 다각형으로 변환합니다.

ST_ToPoint

ST_ToPoint는 입력 매개변수로 Geometry를 가져오며 이를 점으로 변환합니다.

ST_ToPolygon

ST_ToPolygon은 입력 매개변수로 Geometry를 가져오며 이를 다각형으로 변환합니다.

다른 스페이스 구성으로 새 Geometry를 작성하는 함수

다음 함수는 기존의 Geometry를 시작점으로 사용하여 순환 영역 또는 기타 스페이스의 구성을 나타내는 새 Geometry를 작성합니다. 예를 들어, 제시된 공항의 중심을 나타내는 점이 제공된 경우 ST_Buffer는 원 양식으로 공항의 Extent를 나타내는 표면을 작성할 수 있습니다.

함수는 다음과 같습니다.

- ST_Buffer
- ST_ConvexHull
- ST_Difference
- ST_Intersection
- ST_SymDifference

ST_Buffer

ST_Buffer 함수는 지정된 반지름을 사용하여 기존 Geometry에서 외부로 확장하는 새 Geometry를 생성할 수 있습니다. 새 Geometry는 기존 Geometry가 버퍼될 때 또는 콜렉션의 요소가 너무 가까워서 콜렉션의 단일 요소 주위의 버퍼가 겹쳐질 때마다 표면이 됩니다. 그러나 버퍼가 별도인 경우 개별 버퍼 표면이 발생하며, 이 경우 ST_Buffer는 다중 표면을 리턴합니다.

다음 그림은 단일 요소 및 겹친 요소 주변의 버퍼를 보여줍니다.

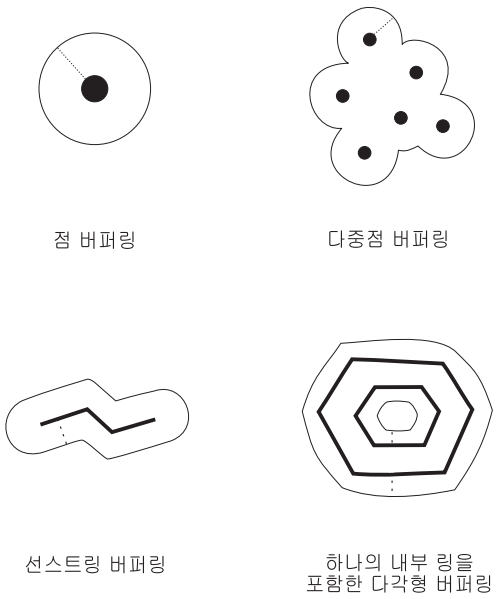


그림 51. ST_Buffer

ST_Buffer 함수는 양수와 음수 거리 모두를 승인하지만 2차원 Geometry(표면 및 다중 표면)만이 음수 버퍼에 적용됩니다. 버퍼 간격의 절대값은 소스 Geometry의 차원이 2보다 적을 때마다(표면 또는 다중 표면 이외의 모든 Geometry) 사용됩니다.

일반적으로 외부 링의 경우에 양수 버퍼 간격은 소스 Geometry의 중심에서 멀리 떨어진 표면 링을 생성하고 음수 버퍼 간격은 중심을 향하고 있는 표면 또는 다중 표면 링을 생성합니다. 표면 또는 다중 표면의 내부 링 경우에, 양수 버퍼 간격은 중심을 향하고있는 버퍼링을 생성하고 음수 버퍼 간격은 중심에서 멀리 떨어진 버퍼링을 생성합니다.

버퍼화 프로세스에서는 겹치는 표면을 병합합니다. 다각형의 최대 내부 폭의 한 배 반보다 큰 음수 간격의 경우에는 결국 빈 Geometry가 됩니다.

ST_ConvexHull

ST_ConvexHull 함수는 볼록면을 형성하는 적어도 세 개의 정점이 있는 Geometry의 볼록한 선체를 리턴합니다. 정점은 Geometry 내의 X 및 Y 좌표쌍입니다. 볼록한 선체(convex hull)는 제공된 정점 세트 내의 모든 정점을 사용하여 형성할 수 있는 가장 작은 볼록 다각형입니다.

다음 예는 네 가지 볼록한 선체의 예를 보여줍니다. 첫 번째 예에서는 문자 c와 유사한 불규칙한 모양을 그렸습니다. c는 볼록한 선체에 의해 닫힙니다. 네 번째 예에는 네 개의 점이 Z자 패턴의 선으로 연결되어 있습니다. 볼록형의 선이 한 쪽의 점 4와 점 2를 연결하고 다른 한쪽의 점 3과 점 1을 연결합니다.

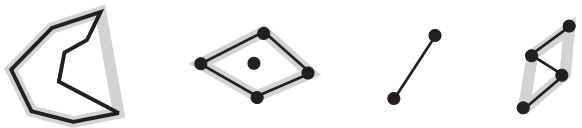


그림 52. ST_ConvexHull

ST_Difference

ST_Difference는 동일 차원의 두 Geometry를 입력으로 사용합니다. ST_Difference 함수는 두 번째 Geometry와 교차하지 않는 첫 번째 Geometry 부분을 리턴합니다. 이 조작은 논리적 AND NOT과 동등한 공간 조작입니다. ST_Difference에서 리턴한 Geometry 부분은 그 자체가 Geometry(입력으로 사용하는 Geometry와 동일한 차원의 콜렉션)입니다. 이러한 두 Geometry가 동일할 경우(즉, 동일한 스페이스를 점유하는 경우) 리턴되는 Geometry는 비어 있습니다.

각 화살표의 왼쪽에는 입력으로 ST_Difference에 제공되는 두 개의 Geometry가 있습니다. 각 화살표의 오른쪽에는 ST_Difference의 출력이 있습니다. 첫 번째 Geometry

의 부분이 두 번째 Geometry와 교차할 경우 출력은 교차되지 않는 첫 번째 Geometry의 일부입니다. 입력 매개변수로 제공된 Geometry가 동일한 경우, 비어있는 Geometry(없음(nil)이라는 용어로 표시됨)가 출력됩니다.

이 그림은 ST_Difference의 입력 및 출력에 대해 설명합니다. 예를 들어, 점을 입력하고 점 a와 점 b가 동일한 경우, 널(NULL)이 출력됩니다. 점 a와 점 b가 다른 경우, 두 점 사이의 새 점이 출력됩니다. b의 다각형과 첫 번째 내부에 있는 Geometry a에 대한 더 작지만 동일한 다각형을 입력한 경우, 널(NULL)이 출력됩니다. 다각형이 겹친 경우, 결합된 다각형의 외부 가장자리가 출력됩니다.

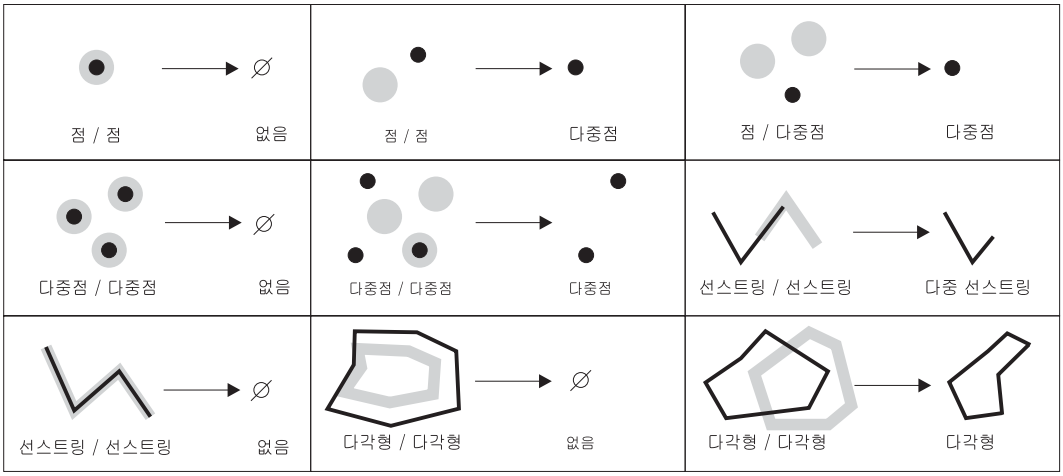


그림 53. ST_Difference

ST_Intersection

ST_Intersection 함수는 Geometry로 표현되는 점 세트를 리턴하며, 이 점 세트는 제공된 두 Geometry의 교차를 정의합니다.

ST_Intersection에 입력으로 제공된 Geometry가 교차하지 않거나 교차하더라도 교차 차원이 Geometry 차원보다 낮은 경우, ST_Intersection은 빈 Geometry를 리턴합니다.

각 화살표의 왼쪽에는 ST_Intersection에 입력으로 제공된 교차하는 두 Geometry가 있습니다. 각 화살표의 오른쪽에는 왼쪽의 Geometry에서 작성한 교차를 나타내는 Geometry인 ST_Intersection의 출력이 있습니다.

이 그림에서는 제공된 Geometry가 교차하는 위치에 대한 정보를 리턴하는 10개의 ST_Intersection 출력 예를 보여줍니다. 예를 들어, b가 선스트링이고 Geometry a가 선 위의 한 점인 경우, Geometry a와 Geometry b가 모이는 다중점이 출력됩니다.

Geometry a와 Geometry b가 겹치는 다각형인 경우, 겹쳐진 부분으로만 구성되는 새 다중 다각형이 출력됩니다.

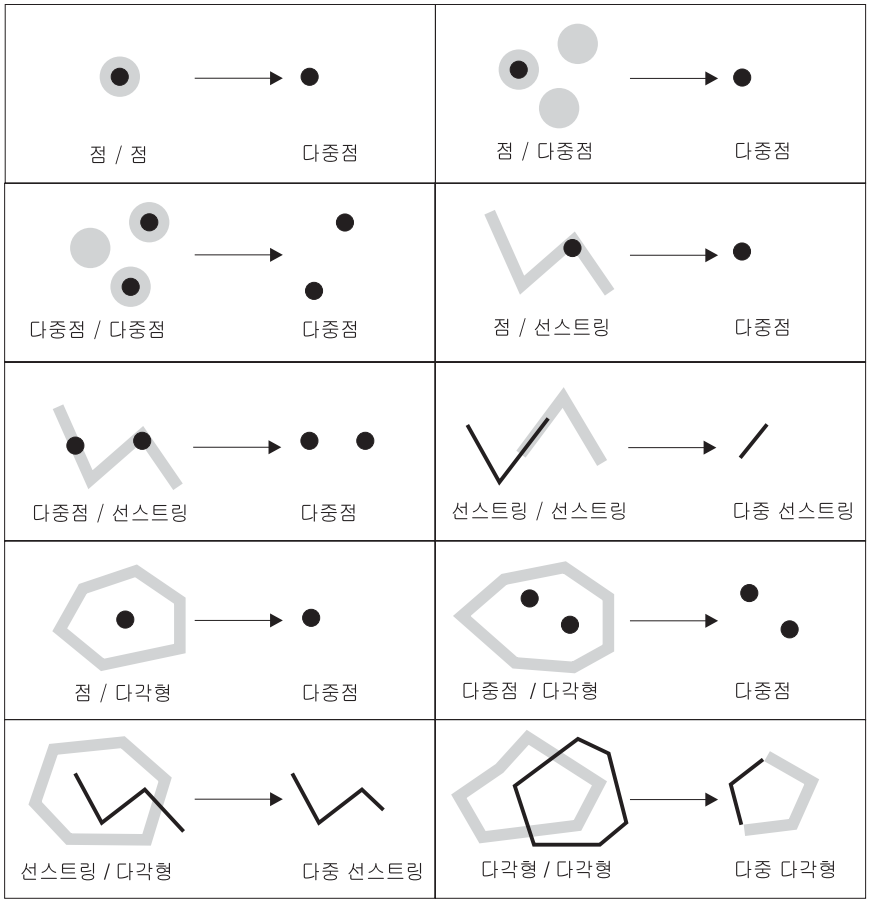


그림 54. ST_Intersection

ST_SymDifference

ST_SymDifference 함수는 동일한 차원을 가진 두 개의 교차 Geometry의 대칭 차이(논리 XOR 조작과 동일한 공간)를 리턴합니다. 이 Geometry가 동일하면 ST_SymDifference는 빈 Geometry를 리턴합니다. 동일하지 않으면 둘 중 하나 또는 둘 모두의 일부가 교차 영역 외부에 놓이게 됩니다.

다수의 Geometry에서 한 Geometry를 파생시키는 함수

다음 함수는 다중 Geometry에서 개별적 Geometry를 파생합니다. 예를 들어, ST_Union은 두 개의 Geometry를 단일 Geometry로 조합합니다.

MBR 집계

ST_BuildMBRAggr 및 ST_GetAggrResult 함수를 조합하여 컬럼의 모든 Geometry를 둘러싸는 최소 바운딩 직사각형을 나타내는 직사각형을 구성함으로써 선택된 컬럼의 Geometry 컬럼을 단일 Geometry로 집계합니다. 집계를 계산할 때 Z 및 M 좌표를 버립니다.

ST_Union

ST_Union 함수는 두 Geometry의 통합 세트를 리턴합니다.

이 조작은 논리적 OR과 동등한 공간 조작입니다. 두 Geometry는 동일한 차원을 가져야 합니다. ST_Union은 항상 결과를 콜렉션으로 리턴합니다.

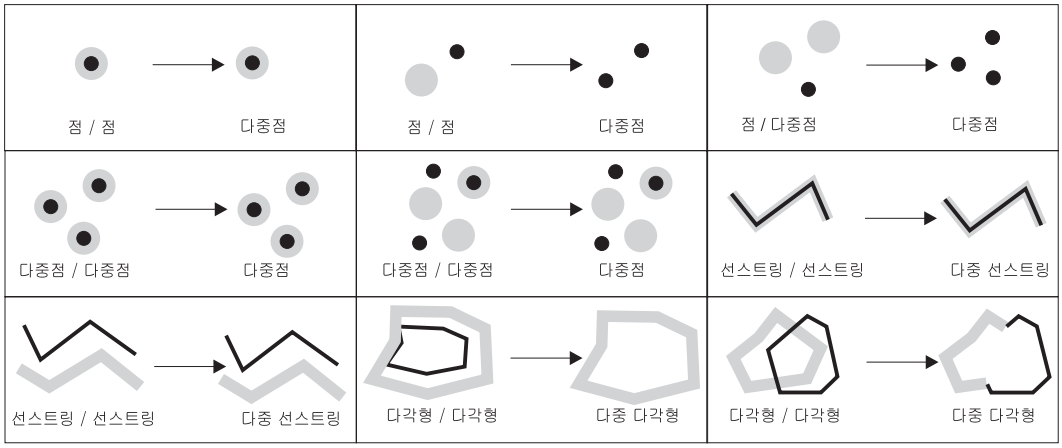


그림 55. ST_Union

총계

통합 집계는 ST_BuildUnionAggr 및 ST_GetAggrResult 함수를 조합한 것입니다. 이러한 조합은 통합 세트를 구성함으로써 테이블에 있는 Geometry 컬럼을 하나의 Geometry로 집계합니다.

도량 단위에 기초한 새 Geometry를 파생시키는 함수

이 절에 설명된 함수는 점이 특정 측정 단위 또는 특정 시퀀스를 가진 두 개의 측정 단위에 연관된 Geometry를 작성할 수 있습니다. 예를 들어, 값의 범위가 4 - 8인 측정 단위는 다중 곡선에 점으로 저장됩니다. 값이 7인 측정 단위가 저장되는 점을 알고자 할 경우, ST_FindMeasure 함수를 사용하여 단일 다중점 내의 해당 점을 리턴할 수 있습니다.

함수는 다음과 같습니다.

- ST_FindMeasure(ST_LocateAlong이라고도 함)
- ST_MeasureBetween(ST_LocateBetween이라고도 함)

ST_FindMeasure(ST_LocateAlong이라고도 함)

ST_FindMeasure(또는 ST_LocateAlong)는 Geometry 및 수치를 입력 매개변수로 가져옵니다. 이는 지정된 수치와 일치하는 제공된 Geometry의 다중점 또는 다중 곡선을 리턴합니다. 점 및 다중점의 경우, 지정된 측정 단위를 갖는 모든 점이 리턴됩니다. 곡선, 다중 곡선, 표면 및 다중 표면의 경우, 결과를 계산하기 위해 보간법(interpolation)이 수행됩니다. Geometry의 바운더리에서 표면 또는 다중 표면에 대한 계산이 수행됩니다.

ST_MeasureBetween(ST_LocateBetween이라고도 함)

ST_MeasureBetween(또는 ST_LocateBetween)은 하나의 Geometry 및 두 개의 M 좌표(측정 단위)를 입력 매개변수로 가져오며 두 개의 M 좌표 간에 연결이 끊어진 경로 또는 점 세트를 나타내는 제공된 Geometry의 일부를 리턴합니다.

곡선, 다중 곡선, 표면 및 다중 표면의 경우, 결과를 계산하기 위해 보간법(interpolation)이 수행됩니다. 그림 56에서 점 3, 4, 5, 6, 7, 8 및 9는 곡선을 표시합니다. 두 개의 M 좌표가 4와 7인 경우, ST_MeasureBetween은 점 4와 7 사이에 있는 곡선의 일부분을 리턴합니다.

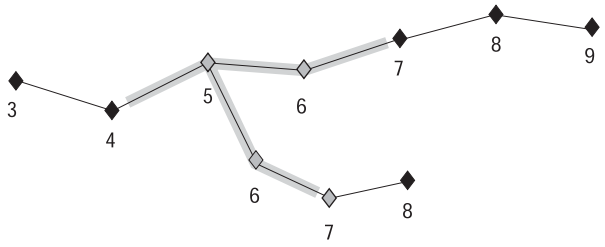


그림 56. LocateBetween

기존 Geometry의 수정된 양식을 작성하는 함수

다음 함수는 기존 Geometry의 수정된 양식을 작성합니다. 예를 들어, ST_AppendPoint 함수는 기존 곡선의 확장된 버전을 작성합니다. 각 버전은 기존 곡선의 점뿐 아니라 추가로 점을 포함합니다.

함수는 다음과 같습니다.

- ST_AppendPoint
- ST_ChangePoint

- ST_Generalize
- ST_M
- ST_PerpPoints
- ST_RemovePoint
- ST_X
- ST_Y
- ST_Z

ST_AppendPoint

ST_AppendPoint는 입력 매개변수로 곡선과 점을 가져오며 제공된 점으로 곡선을 확장합니다.

ST_ChangePoint

ST_ChangePoint는 입력 매개변수로 곡선과 두 개의 점을 가져옵니다. 제공된 곡선의 발생한 모든 첫 번째 점을 두 번째 점으로 바꾸고 결과 곡선을 리턴합니다.

ST_Generalize

Geometry의 일반 특성을 보존하는 동안 ST_Generalize는 입력 매개변수로 Geometry와 임계값을 가져오며 점 수가 줄어든 제공된 Geometry를 표시합니다. Douglas-Peucker line-simplification 알고리즘이 사용되며, 이 알고리즘에서는 Geometry를 정의하는 점 시퀀스가 연속된 점이 직선 세그먼트로 대체될 때까지 순환적으로 세분화됩니다. 이 선 세그먼트에서, 정의한 점은 제공된 임계값을 초과하여 직선 세그먼트에서 벗어나지 않습니다. Z 및 M 좌표는 단순화를 위해 고려되지 않습니다.

ST_M

제공된 점이 측정 단위에 연관되어 있지 않으면 ST_M은 점과 함께 저장할 측정 단위를 제공할 수 있습니다. 점에 연관된 측정 단위가 있으면 ST_M은 이를 다른 측정 단위로 바꿀 수 있습니다.

ST_PerpPoints

ST_PerpPoints는 곡선 또는 다중 곡선 및 점을 입력 매개변수로 가져오며 곡선 또는 다중 곡선의 제공된 점에 대해 수직인 프로젝션을 리턴합니다. 제공된 점과 수직 점 간의 거리가 가장 짧은 점이 리턴됩니다. 제공된 점에서 둘 이상의 수직 프로젝트화된 점까지의 거리가 동일하면 그 점이 모두 리턴됩니다.

ST_RemovePoint

ST_RemovePoint는 입력 매개변수로 곡선과 점을 가져오며 여기에서 제거된 지정된 점과 동일한 모든 점이 있는 제공된 곡선을 리턴합니다. 제공된 곡선에 Z 또는 M 좌표가 있는 경우, 점에도 Z 또는 M 좌표가 있어야 합니다.

ST_X

ST_X는 점의 X 좌표를 다른 X 좌표로 바꿀 수 있습니다.

ST_Y

ST_Y는 점의 Y 좌표를 다른 Y 좌표로 바꿀 수 있습니다.

ST_Z

제공된 점에 Z 좌표가 없으면 ST_Z는 점에 Z 좌표를 추가합니다. 점에 Z 좌표가 있으면 ST_Z는 이 좌표를 다른 Z 좌표로 바꿀 수 있습니다.

거리 정보를 리턴하는 함수

ST_Distance는 두 개의 Geometry와 단위(선택적)를 입력 매개변수로 가져와 첫 번째 Geometry의 점과 두 번째 Geometry의 점 사이의 최소 거리(제공된 단위로 측정됨)를 리턴합니다.

두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템에 표시되지 않으면, 다른 공간 참조 시스템으로 변환됩니다.

제공된 두 Geometry가 널(NULL)이거나 비어 있으면 널(NULL)이 리턴됩니다.

예를 들어, ST_Distance는 비행기가 두 위치 사이에서 비행해야 하는 최단 거리를 알려줄 수 있습니다. 그림 57에서는 이 정보에 대해 설명합니다.

그림은 로스엔젤레스와 시카고로 표시되는 지점 사이를 직선으로 연결한 미국 지도를 보여줍니다.

그림 57. 두 도시 사이의 최소 거리. ST_Distance는 LA와 시카고의 위치 좌표를 입력으로 사용하고 이 위치 사이의 최소 거리를 나타내는 값을 리턴할 수 있습니다.

인덱스 정보를 리턴하는 함수

ST_GetIndexParms는 공간 인덱스 또는 공간 컬럼에 대한 ID를 입력 매개변수로 가져오며 인덱스 또는 공간 컬럼에 대한 인덱스를 정의하는 데 사용되는 매개변수를 리턴합니다. 추가 매개변수 번호를 지정할 경우 숫자로 식별되는 매개변수만이 리턴됩니다.

좌표 시스템 사이의 변환

ST_Transform은 Geometry 및 공간 참조 시스템 ID를 입력 매개변수로 가져오며 제공된 공간 참조 시스템에 표시할 수 있도록 Geometry를 변환합니다. 다른 좌표 시스템 간의 프로젝션 및 변환이 수행되고 Geometry의 좌표가 그에 맞게 조정됩니다.

제 23 장 공간 함수: 구문 및 매개변수

이 절에서는 다음 절에 설명된 공간 함수를 소개합니다. 여기에는 모든 또는 대부분의 공간 함수에 공통인 특정 인수에 대해 논의되어 있습니다. 함수는 여기에서 영문자 순으로 설명되어 있습니다.

공간 함수: 고려사항 및 연관된 데이터 유형

이 절에서는 공간 함수를 코드화할 때 알아야 할 정보를 제공합니다. 다음과 같은 정보가 포함됩니다.

- 고려사항: 공간 함수가 속하는 스키마를 지정하기 위한 요구사항 및 일부 함수를 메소드로 호출할 수 있음
- 공간 함수가 다른 공간 함수에서 리턴한 Geometry 유형을 처리할 수 없는 경우의 해결 방법
- 각 공간 데이터 유형의 값을 입력 매개변수로 가져오는 함수를 표시하는 테이블

공간 함수를 사용할 때 다음과 같은 점을 유의하십시오.

- 공간 함수를 호출하기 전에 공간 함수가 속하는 스키마 이름(DB2GSE)으로 함수 이름이 규정되어야 합니다. 이를 수행하는 한 가지 방법은 함수 이름을 참조하는 SQL 문에 스키마를 명시적으로 지정하는 것입니다. 예를 들면, 다음과 같습니다.

  ```
  SELECT db2gse.ST_Relate (g1, g2, 'T*F**FFF2') EQUALS FROM relate_test
  ```

  다른 방법으로, 함수를 호출할 때마다 스키마를 지정하지 않기 위해 DB2GSE를 CURRENT FUNCTION PATH 특수 레지스터에 추가할 수 있습니다. 이 특수 레지스터에 대한 현재 설정값을 확보하려면 다음 SQL 명령을 입력하십시오.

  ```
  VALUES CURRENT FUNCTION PATH
  ```

  DB2GSE를 사용하여 CURRENT FUNCTION PATH 특수 레지스터를 갱신하려면 다음 SQL 명령을 발행하십시오.

  ```
  set CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
  ```

- 일부 공간 함수를 메소드로 호출할 수 있습니다. 예를 들어, 다음 코드에서 ST_Area는 처음에는 함수로 호출된 다음 메소드로 호출되었습니다. 두 경우 모두 ST_Area는 ID가 10인 다각형에서 작동하도록 코드화되었으며 STORES 테이블의 SALES_ZONE 컬럼에 저장됩니다. 호출 시 ST_Area는 다각형이 표시하는 실제 장소(판매 지역 번호 10) 영역을 리턴합니다.

  함수로 호출된 ST_Area:

© Copyright IBM Corp. 1998, 2006

```
SELECT ST_Area(sales_zone)
FROM   stores
  WHERE id = 10
```

메소드로 호출된 ST_Area:

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

ST_BuildMBRAggr 및 ST_BuildUnionAggr 함수는 ″MBR Aggregate″ 및 ″Union Aggregate″에서 각각 설명됩니다.

고려사항

ST_Geometry 값을 부속 유형 값으로 처리

공간 함수에서 정적 유형이 슈퍼 유형인 Geometry를 리턴할 경우와 Geometry가 이 슈퍼 유형에 종속된 Geometry 유형만을 승인하는 함수로 패스될 경우, 컴파일 시간 예외가 발생합니다.

예를 들어, ST_Union 함수의 출력 매개변수에 대한 정적 유형은 모든 공간 데이터 유형의 슈퍼 유형인 ST_Geometry입니다. ST_PointOnSurface 함수의 정적 입력 매개변수는 ST_Geometry의 두 부속 유형인 ST_Polygon 또는 ST_MultiPolygon일 수 있습니다. DB2®가 ST_Union에서 리턴한 Geometry를 ST_PointOnSurface로 패스하려고 할 경우 DB2에서는 다음과 같은 컴파일 예외가 발생합니다.

```
SQL00440N 함수 경로에서 호환되는 인수를 갖는 "ST_POINTONSURFACE"라는
함수를 함수 경로에서 찾을 수 없습니다.
          SQLSTATE=42884
```

이 메시지는 DB2가 ST_PointOnSurface라는 함수를 찾을 수 없으며 ST_Geometry 입력 매개변수를 갖고 있음을 나타냅니다.

슈퍼 유형을 가진 Geometry를 슈퍼 유형의 부속 유형만을 승인하는 함수로 패스하려면 TREAT 연산자를 사용하십시오. 앞에 표시된 대로 ST_Union은 ST_Geometry의 정적 유형을 가진 Geometry를 리턴합니다. 또한 ST_Geometry의 동적 부속 유형을 가진 Geometry도 리턴할 수 있습니다. 예를 들어, ST_MultiPolygon의 동적 유형을 가진 Geometry를 리턴한다고 가정합시다. 이 경우, TREAT 연산자에서는 정적 유형 ST_MultiPolygon을 가진 Geometry를 사용해야 합니다. 이는 ST_PointOnSurface의 입력 매개변수에 대한 데이터 유형 중 하나와 일치합니다. ST_Union이 ST_MultiPolygon 값을 리턴하지 않을 경우 DB2에서는 런타임 예외가 발생합니다.

함수에서 슈퍼 유형의 Geometry를 리턴할 경우 TREAT 연산자는 일반적으로 DB2에게 이 Geometry를 이 슈퍼 유형의 부속 유형으로 간주하도록 지시합니다. 이 조작은

부속 유형이 일치하거나 Geometry를 패스할 함수의 입력 매개변수로 정의된 정적 부속 유형에 종속되어 있을 경우에만 성공함을 알아두십시오. 이 조건이 일치하지 않으면 DB2에서 런타임 오류가 발생합니다.

다른 예로 닫힌 다각형의 바운더리에 제공된 점에 대해 수직인 점을 판별하려 한다고 가정합시다. ST_Boundary 함수를 사용하여 다각형으로부터 바운더리를 파생시킬 수 있습니다. ST_Boundary의 정적 출력 매개변수는 ST_Geometry이나 ST_PerpPoints는 ST_Curve Geometry를 승인합니다. 모든 다각형은 바운더리로 선스트링(곡선일 수도 있음)을 갖고 있고 선스트링의 데이터 유형(ST_LineString)은 ST_Curve에 종속되기 때문에 다음 조작은 ST_Boundary에서 리턴한 ST_Geometry 다각형을 ST_PerpPoints로 패스합니다.

```
SELECT ST_AsText(ST_PerpPoints(TREAT(ST_Boundary(polygon) as ST_Curve)),
             ST_Point(30.5, 65.3, 1)))
FROM   polygon_table
```

ST_Boundary 및 ST_PerpPoints를 함수로 호출하는 대신 메소드로 호출할 수 있습니다. 이를 수행하려면 다음 코드를 지정하십시오.

```
SELECT TREAT(ST_Boundary(polygon) as ST_Curve)..
             ST_PerpPoints(St_Point(30.5, 65.3, ))..ST_AsText()
FROM   polygon_table
```

입력 유형에 따라 나열된 공간 함수

334 페이지의 표 58에 승인할 수 있는 입력 유형에 따라 공간 함수가 나열되어 있습니다.

중요: 앞에서 설명한 것처럼 공간 데이터 유형은 루트가 ST_Geometry인 계층 구조를 형성합니다. DB2 Spatial Extender의 문서에 이 계층 구조의 슈퍼 유형 값이 함수의 입력 매개변수로 사용될 수 있다고 나와 있으면 이 슈퍼 유형의 부속 유형 값도 함수의 입력 매개변수로 사용될 수 있습니다.

예를 들어, 334 페이지의 표 58의 첫 번째 항목은 ST_Area 및 ST_Geometry 데이터 유형의 값을 입력으로 가져올 수 있는 다른 함수의 번호를 나타냅니다. 따라서 ST_Geometry의 부속 유형(ST_Point, ST_Curve, ST_LineString 등) 값이 이러한 함수에 대한 입력 매개변수가 될 수도 있습니다.

표 58. 입력 유형에 따라 나열된 공간 함수

입력 매개변수의 데이터 유형	함수
ST_Geometry	
	EnvelopesIntersect
	ST_Area
	ST_AsBinary
	ST_AsGML
	ST_AsShape
	ST_AsText
	ST_Boundary
	ST_Buffer
	ST_BuildMBRAggr
	ST_BuildUnionAggr
	ST_Centroid
	ST_Contains
	ST_ConvexHull
	ST_CoordDim
	ST_Crosses
	ST_Difference
	ST_Dimension
	ST_Disjoint
	ST_Distance
	ST_Envelope
	ST_EnvIntersects
	ST_Equals
	ST_FindMeasure 또는 ST_LocateAlong
	ST_Generalize
	ST_GeometryType

표 58. 입력 유형에 따라 나열된 공간 함수 (계속)

입력 매개변수의 데이터 유형	함수
ST_Geometry, continued	
	ST_Intersection
	ST_Intersects
	ST_Is3D
	ST_IsEmpty
	ST_IsMeasured
	ST_IsSimple
	ST_IsValid
	ST_MaxM
	ST_MaxX
	ST_MaxY
	ST_MaxZ
	ST_MBR
	ST_MBRIntersects
	ST_MeasureBetween 또는 ST_LocateBetween
	ST_MinM
	ST_MinX
	ST_MinY
	ST_MinZ
	ST_NumPoints
	ST_Overlaps
	ST_Relate
	ST_SRID 또는 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

표 58. 입력 유형에 따라 나열된 공간 함수 (계속)

입력 매개변수의 데이터 유형	함수
ST_Curve	
	ST_AppendPoint
	ST_ChangePoint
	ST_EndPoint
	ST_IsClosed
	ST_IsRing
	ST_Length
	ST_MidPoint
	ST_PerpPoints
	ST_RemovePoint
	ST_StartPoint
ST_LineString	
	ST_PointN
	ST_Polygon
ST_Surface	
	ST_Perimeter
	ST_PointOnSurface
ST_GeomCollection	
	ST_GeometryN
	ST_NumGeometries
ST_MultiPoint	ST_PointN
ST_MultiCurve	
	ST_IsClosed
	ST_Length
	ST_PerpPoints
ST_MultiLineString	
	ST_LineStringN
	ST_NumLineStrings
	ST_Polygon
ST_MultiSurface	
	ST_Perimeter
	ST_PointOnSurface
ST_MultiPolygon	
	ST_NumPolygons
	ST_PolygonN

EnvelopesIntersect

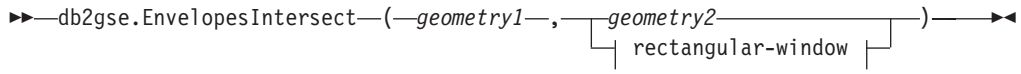
EnvelopesIntersect는 두 가지 유형의 입력 매개변수를 허용합니다.

- 두 개의 Geometry

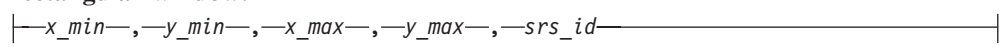
  첫 번째 Geometry의 인벨로프가 두 번째 Geometry의 인벨로프와 교차하면 EnvelopesIntersect가 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다.
- Geometry, 직사각형 창의 왼쪽 하단 구석 및 오른쪽 상단 구석을 정의하는 네 가지 유형의 DOUBLE 좌표 값과 공간 참조 시스템 ID

EnvelopesIntersect는 첫 번째 geometry의 인벨로프(envelope)가 4가지 유형의 DOUBLE 값으로 정의된 인벨로프(envelope)와 교차하면 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다.

구문



rectangular-window:



매개변수

geometry1

인벨로프가 *geometry2* 또는 4가지 유형의 DOUBLE 값으로 정의된 직사각형 창의 인벨로프와 교차되는지 테스트할 Geometry를 표시하는 ST_Geometry 유형의 값 또는 해당 부속 유형 중 하나의 값

geometry2

유형 ST_Geometry 값 또는 인벨로프가 *geometry1*의 인벨로프와의 교차에 대해 테스트할 Geometry를 표시하는 해당 부속 유형 중 하나의 값

x_min

인벨로프에 대한 최소 X 좌표값을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

측지 데이터에는 다음과 같은 조건이 적용됩니다.

- *x_min*은 –180도와 180도 사이의 경도 값이어야 합니다.
- 인벨로프가 180도 경선과 겹치는 경우 *x_min*은 *x_max*보다 큽니다.

y_min

인벨로프에 대한 최소 Y 좌표값을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

측지 데이터에는 다음과 같은 조건이 적용됩니다.

- *y_min*은 –90도와 90도 사이의 위도 값이어야 합니다.
- *y_min*은 *y_max* 값 미만이어야 합니다.

x_max

인벨로프에 대한 최대 X 좌표값을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

측지 데이터에는 다음과 같은 조건이 적용됩니다.

- *x_max*는 –180도와 180도 사이의 경도 값이어야 합니다.
- 인벨로프가 180도 경선과 겹치는 경우 *x_max*는 *x_min* 값 미만입니다.

y_max

인벨로프에 대한 최대 Y 좌표값을 지정합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

이 매개변수의 데이터 유형은 DOUBLE입니다.

측지 데이터에는 다음과 같은 조건이 적용됩니다.

- *y_max*는 –90도와 90도 사이의 위도 값이어야 합니다.
- *y_max*는 *y_min* 값보다 커야 합니다.

srs_id

공간 참조 시스템을 고유하게 식별합니다. 공간 참조 시스템 ID는 Geometry 매개변수의 공간 참조 시스템 ID와 일치해야 합니다. 이 매개변수에 대해 널(NULL)이 아닌 값을 지정해야 합니다.

이 매개변수의 데이터 유형은 INTEGER입니다.

리턴 유형

INTEGER

예

이 예제는 County를 나타내는 두 개의 다각형을 작성하고 이들 중 하나가 4가지 유형의 DOUBLE 값으로 지정된 공간 영역과 교차하는지 판별합니다.

```
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
```

결과:

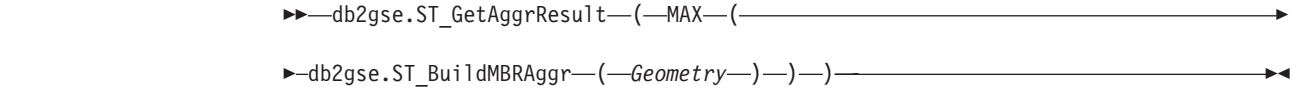
```
Name
--------------------
County_1
County_2
```

MBR 집계

ST_BuildMBRAggr 및 ST_GetAggrResult 함수를 조합하여 컬럼의 모든 Geometry를 둘러싸는 최소 바운딩 직사각형을 나타내는 직사각형을 구성함으로써 선택된 컬럼의 Geometry 컬럼을 단일 Geometry로 집계합니다. 집계를 계산할 때 Z 및 M 좌표를 버립니다.

조합할 모든 Geometry가 널(NULL)이면 널(NULL)이 리턴됩니다. 모든 Geometry가 널(NULL)이거나 비어 있으면 빈 Geometry가 리턴됩니다. 조합할 모든 Geometry의 최소 바운딩 직사각형으로부터 점이 생성될 경우 이 점이 ST_Point 값으로 리턴됩니다. 조합할 모든 Geometry의 최소 바운딩 직사각형으로부터 수평 또는 수직 선스트링이 생성될 경우, 이 선스트링이 ST_LineString 값으로 리턴됩니다. 그렇지 않으면, 최소 바운딩 직사각형이 ST_Polygon 값으로 리턴됩니다.

구문



매개변수

Geometry

ST_Geometry 또는 부속 유형 중 하나를 가지며 최소 바운딩 직사각형을 계산할 모든 Geometry를 나타내는 선택된 컬럼

리턴 유형

db2gse.ST_Geometry

제한사항

다음과 같은 상황에서는 Fullselect에 통합된 공간 컬럼 집계를 구성할 수 없습니다.

- MPP 환경
- Fullselect에서 GROUP BY절을 사용할 경우
- DB2 집계 함수 MAX가 아닌 다른 함수를 사용할 경우

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_BuildMBRAggr 함수를 사용하여 컬럼 내의 모든 Geometry에 대한 최대 바운드 직사각형을 확보하는 방법을 보여줍니다. 이 예에서는 여러 개의 점을 SAMPLE_POINTS 테이블의 GEOMETRY 컬럼에 추가합니다. 그런 다음 SQL 코드는 함께 추가된 모든 점에 대한 최대 바운드 직사각형을 판별합니다.

```
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
```

결과:

```
Aggregate_of_Points
-----------------------------------------------------------
POLYGON (( 2.00000000 2.00000000, 23.00000000 2.00000000,
23.00000000 15.00000000, 2.00000000 15.00000000, 2.00000000
2.00000000))
```

ST_AppendPoint

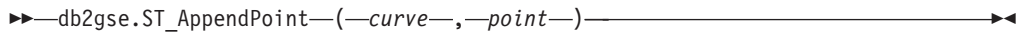
ST_AppendPoint는 입력 매개변수로 곡선과 점을 가져오며 제공된 점으로 곡선을 확장합니다. 제공된 곡선에 Z 또는 M 좌표가 있는 경우, 점에도 Z 또는 M 좌표가 있어야 합니다. 결과 곡선은 제공된 곡선의 공간 참조 시스템에 표시됩니다.

추가될 점이 곡선과 동일한 공간 참조 시스템에 표시되지 않으면, 다른 공간 참조 시스템으로 변환됩니다.

제공된 곡선이 닫혀있거나 단순하면, 결과 곡선은 닫혀있지 않거나 더 이상 단순하지 않을 수 있습니다. 제공된 곡선 또는 점이 널(NULL)이거나 곡선이 비어 있으면 널(NULL)이 리턴됩니다. 추가될 점이 비어 있으면, 제공된 곡선이 변경되지 않은 상태로 리턴되며 경고가 발생됩니다(SQLSTATE 01HS3).

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

curve 유형 ST_Curve의 값 또는 *point*가 추가된 곡선을 표시하는 해당 부속 유형 중 하나의 값

point *curve*에 추가된 점을 표시하는 유형 ST_Point의 값

리턴 유형

db2gse.ST_Curve

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 코드는 각각 세 개의 점이 있는 두 개의 선스트링을 작성합니다.

```
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) )
```

예 1

이 예는 점 (5, 5)를 선스트링의 끝에 추가합니다.

```
SELECT CAST(ST_AsText(ST_AppendPoint(line, ST_Point(5, 5)))
    AS VARCHAR(120)) New
FROM   sample_lines
WHERE  id=1
```

결과:

```
NEW
----------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 10.00000000 0.00000000,
0.00000000 0.00000000, 5.00000000 5.00000000)
```

예 2

이 예는 점 (15, 15, 7)을 Z 좌표를 가진 선스트링의 끝에 추가합니다.

```
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
```

결과:

```
NEW
------------------------------------------------------------
LINESTRING Z ( 0.00000000 0.00000000 4.00000000, 5.00000000
5.00000000 5.00000000, 10.00000000 10.00000000 6.00000000,
15.00000000 15.00000000 7.00000000)
```

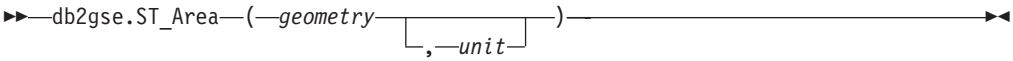
ST_Area

ST_Area는 Geometry와 단위(선택적)를 입력 매개변수로 가져오며 디폴트 또는 제공된 측정 단위로 Geometry가 포함할 수 있는 영역을 리턴합니다.

Geometry가 다각형 또는 다중 다각형이면 Geometry가 담당하는 영역이 리턴됩니다. 점, 선스트링, 다중점 및 다중 선스트링의 영역은 0입니다. Geometry가 널(NULL)이거나 빈 Geometry인 경우 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

ST_Geometry 유형 또는 영역을 판별하는 Geometry를 표시하는 해당 부속 유형 중 하나의 값

unit 영역을 측정하는 단위를 식별하는 VARCHAR(128) 값. 지원되는 측정 단위는 DB2GSE.ST_UNITS_OF_MEASURE 카탈로그 뷰에 나열되어 있습니다.

unit 매개변수를 생략하면 다음 규칙을 사용하여 영역 측정 단위를 판별합니다.

- *geometry*가 프로젝트화 또는 지구 중심 좌표 시스템에 있는 경우 이 좌표 시스템에 연관된 선형 단위가 사용됩니다.
- *Geometry*가 지리 좌표 시스템에 있지만 측지 공간 참조 시스템(SRS)에 없는 경우, 이 좌표 시스템과 연관된 각도 단위가 사용됩니다.
- *Geometry*가 측지 SRS에 있는 경우, 디폴트 측정 단위는 제곱 미터입니다.

단위 변환에 대한 제한사항: 다음과 같은 조건이 발생하는 경우 오류(SQLSTATE 38SU4)가 리턴됩니다.

- Geometry가 지정되지 않은 좌표 시스템에 있고 *unit* 매개변수가 지정됩니다.
- Geometry가 투영 좌표 시스템에 있고 각도 단위가 지정됩니다.
- Geometry는 지리 좌표 시스템에 있지만 측지 SRS에는 없으며 선형 단위가 지정됩니다.

- Geometry가 지리 좌표 시스템에 있고 측지 SRS에도 있으며 각도 단위가 지정됩니다.

리턴 유형

DOUBLE

예

예 1

공간 분석자에게는 각 판매 지역에서 담당하는 영역 목록이 필요합니다. 판매 지역 다각형은 SAMPLE_POLYGONS 테이블에 저장됩니다. 영역은 ST_Area 함수를 Geometry 컬럼에 적용하여 계산됩니다.

```
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))
```

다음 SELECT문은 판매 영역 ID 및 영역을 검색합니다.

```
SELECT id, ST_Area(geometry) AS area
FROM sample_polygons
```

결과:

```
ID        AREA
--------- ------------------------
       1   +1.00000000000000E+002
       2   +2.00000000000000E+002
       3   +2.50000000000000E+001
```

예 2

다음 SELECT문은 판매 영역 ID 및 영역을 여러 단위로 검색합니다.

```
SELECT id,
       ST_Area(geometry) square_feet,
       ST_Area(geometry, 'METER') square_meters,
       ST_Area(geometry, 'STATUTE MILE') square_miles
FROM   sample_polygons
```

결과:

```
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
```

예 3

이 예는 State Plane 좌표에 정의되어 있는 영역을 찾습니다.

ID가 3인 State Plane 공간 참조 시스템이 다음 명령으로 작성됩니다.

```
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
```

다음 SQL문은 공간 참조 시스템 3에서 다각형을 테이블에 추가하고 제곱 피트, 제곱 미터 및 제곱 마일 단위로 영역을 판별합니다.

```
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;
```

결과:

```
ID   Square Feet              Square Meters            Square Miles
--- ------------------------ ------------------------ ------------------------
  1 +7.94698788000000E+008   +7.38302286101346E+007   +2.85060106320552E+001
```

예 4

공간 분석자에게는 각 판매 지역에 해당하는 영역의 목록이 필요합니다. 판매 지역 다각형은 SAMPLE_GEODETIC_TAB 테이블에 저장되며 다음 지역이 포함됩니다.

- 북극 주변 지역
- 남극 주변 지역
- 경도 180도 주변 지역

다음 입력 파일(samp_wkt_rows.txt)의 두 번째 필드에는 이러한 지역을 나타내는 다각형이 들어 있습니다.

```
1|'polygon((5 82,15 82,25 82,35 82,45 82,55 82,65 82,75 82,85 82,95 82,
105 82,115 82,125 82,135 82,145 82,155 82,165 82,175 82,-175 82,-165 82,
-155 82,-145 82,-135 82,-125 82,-115 82,-105 82,-95 82,-85 82,-75 82,
-65 82,-55 82,-45 82,-35 82,-25 82,-15 82,-5 82,5 82))'|'North Pole region'
2|'polygon((175 -82,165 -82,155 -82,145 -82,135 -82,125 -82,115 -82,
```

```
105 -82,95 -82,85 -82,75 -82,65 -82,55 -82,45 -82,35 -82,25 -82,15 -82,
5 -82,-5 -82,-15 -82,-25 -82,-35 -82,-45 -82,-55 -82,-65 -82,-75 -82,
-85 -82,-95 -82,-105 -82,-115 -82,-125 -82,-135 -82,-145 -82,-155 -82,
-165 -82,-175 -82,175 -82))'|'South Pole region'
3|'polygon((-175 -42,-175 1,-175 42,175 42,175 -1,175 -42,-175 -42))
'|'180th meridian'
```

다음 SQL문은 측지 공간 참조 시스템 2000000000의 다각형을 SAMPLE_GEODETIC_TAB 테이블에 추가합니다.

```
SET current function path db2gse;
CREATE TABLE db2se_samp.gsege_temp_samp (
            gid INTEGER,
            g1_wkt varchar(500),
            comment varchar(255)
   ) NOT LOGGED INITIALLY;
LOAD FROM samp_wkt_rows.txt OF DEL MODIFIED BY CHARDEL'' COLDEL|
   INSERT INTO db2se_samp.gsege_temp_samp;

CREATE TABLE sample_geodetic_tab
  (gid INTEGER NOT NULL PRIMARY KEY,
   geometry ST_Geometry),
   comment varchar(255));

INSERT INTO sample_geodetic_tab
     SELECT gid, ST_GeomFromText(g1_wkt, 2000000000), comment
   FROM db2se_samp.gsege_temp_samp;
```

ST_Area 함수는 Geometry 컬럼의 다각형 영역을 계산합니다. ST_Area의 디폴트 측정 단위는 제곱 미터입니다. 다음 SELECT문은 판매 영역 ID 및 영역을 제곱 미터, 제곱 피트 및 제곱 마일 단위로 검색합니다.

```
SELECT id, ST_Area(geometry) AS SQUARE_METERS,
ST_Area(geometry,'FOOT') AS SQUARE_FEET,
ST_Area(geometry, 'STATUTE MILE') AS SQUARE_MILES
FROM sample_geodetic_tab
WHERE id BETWEEN 1 AND 9 ORDER BY id;
```

```
ID  SQUARE_METERS            SQUARE_FEET              SQUARE_MILES
--- ------------------------ ------------------------ ------------------------
  1 +2.52472719957839E+012   +2.71759374028922E+013   +9.74802621488040E+005
  2 +2.52475431563494E+012   +2.71762292776957E+013   +9.74813091056005E+005
  3 +9.43568029137069E+012   +1.01564817377028E+014   +3.64313652781464E+006
```

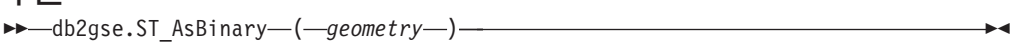
ST_AsBinary

ST_AsBinary는 입력 매개변수로 Geometry를 가져오며 Well-Known Binary 표현을 리턴합니다. Z 및 M 좌표는 버려지며 Well-Known Binary 표현으로 표시되지 않습니다.

제공된 Geometry가 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

유형 ST_Geometry 값 또는 해당 Well-Known Binary 표현으로 변환될 해당 부속 유형 중 하나의 값

리턴 유형

BLOB(2G)

예

다음 코드는 ST_AsBinary 함수를 사용하여 SAMPLE_POINTS 테이블의 Geometry 컬럼에 있는 점을 BLOB 컬럼에 Well-Known Binary 표현으로 변환하는 방법을 보여줍니다.

```
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))
```

예 1

이 예는 ID가 1100인 GEOMETRY 컬럼의 데이터로 ID가 1111인 WKB 컬럼을 채웁니다.

```
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
```

결과:

```
ID          Point
----------- -----------------------------------
       1111 POINT ( 10.00000000 20.00000000)
```

예 2

이 예는 WKB 2진 표현으로 표시됩니다.

```
SELECT id, substr(ST_AsBinary(geometry), 1, 21) AS point_wkb
FROM   sample_points
WHERE  id = 1100
```

결과:

```
ID     POINT_WKB
------ ------------------------------------------
  1100 x'0101000000000000000000244000000000000003440'
```

ST_AsGML

ST_AsGML은 Geometry를 입력 매개변수로 가져오며 GML을 사용하여 해당 표현을 리턴합니다.

제공된 Geometry가 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 값 또는 해당 해당 GML 표현으로 변환될 해당 부속 유형 중 하나의 값

리턴 유형

CLOB(2G)

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

다음 코드 조각은 ST_AsGML 함수를 사용하여 GML 조각을 보는 방법을 보여줍니다. 이 예는 ID가 2222인 Geometry 컬럼의 데이터로 GML 컬럼을 채웁니다.

```
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))
```

다음 SELECT문은 Geometry의 GML 표현과 ID를 나열합니다. Geometry는 ST_AsGML 함수에 의해 GML 조각으로 변환됩니다.

```
SELECT id, cast(ST_AsGML(geometry) AS varchar(110)) AS gml_fragment
FROM sample_points
WHERE  id = 1100
```

결과:

SELECT문은 다음의 결과 세트를 리턴합니다.

```
ID          GML_FRAGMENT

----------- ------------------------------------------------------------
       1100 <gml:Point srsName";EPSG:4269";><gml:coord>
            <gml:X>10</gml:X><gml:Y>20</gml:Y>
            </gml:coord></gml:Point>
```

ST_AsShape

St_AsShape에서는 Geometry를 입력 매개변수로 사용하고 ESRI 모양 표현을 리턴합니다.

제공된 Geometry가 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 값 또는 해당 해당 ESRI 모양 표현으로 변환될 해당 부속 유형 중 하나의 값

리턴 유형

BLOB(2G)

예

다음 코드는 ST_AsShape 함수를 사용하여 SAMPLE_POINTS 테이블의 Geometry 컬럼에 있는 점을 모양 BLOB 컬럼에 있는 모양 2진 표현으로 변환하는 방법을 보여줍니다. 이 예는 Geometry 컬럼의 데이터로 모양 컬럼을 채웁니다. 모양 2진 표현은 geobrowser에 Geometry를 표시하거나(이 경우 Geometry는 ESRI 모양 파일 형식을 준수해야 함) 모양 파일의 *.SHP 파일에 대한 Geometry를 구성하는 데 사용됩니다.

```
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
```

리턴:

```
ID     SHAPE

------ ------------------------------------------
  1100 x'0100000000000000000002440000000000000003440'
```

ST_AsText

ST_AsText는 입력 매개변수로 Geometry를 가져오며 Well-Known Text 표현을 리턴합니다.

제공된 Geometry가 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 값 또는 해당 Well-Known Text 표현으로 변환될 해당 부속 유형 중 하나의 값

리턴 유형

CLOB(2G)

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

데이터를 캡처하여 SAMPLE_GEOMETRIES 테이블에 삽입한 후, 분석자는 Geometry의 Well-Known Text 표현을 검토하여 삽입된 값이 올바른지 검증할 수 있습니다.

```
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))
```

다음 SELECT문은 Geometry의 WKT 표현과 공간 유형을 나열합니다. Geometry는 ST_AsText 함수에 의해 텍스트로 변환됩니다. 그런 다음 ST_AsText 함수의 출력이 CLOB(2G)이기 때문에 varchar(120)으로 캐스트됩니다.

```
SELECT id, spatial_type, cast(geometry..ST_AsText
   AS varchar(150)) AS wkt
FROM   sample_geometries
```

결과:

```
ID     SPATIAL_TYPE       WKT
------ ------------------ -------------------------------------------------
     1 st_point           POINT ( 50.00000000 50.00000000)

     2 st_linestring      LINESTRING ( 200.00000000 100.00000000,
                            210.00000000 130.00000000, 220.00000000
                            140.00000000)

     3 st_polygon         POLYGON (( 110.00000000 120.00000000,
                            130.00000000 120.00000000, 130.00000000
                            140.00000000, 110.00000000140.00000000,
                            110.00000000 120.00000000))
```

ST_Boundary

ST_Boundary는 입력 매개변수로 Geometry를 가져오며 새 Geometry로 바운더리를 리턴합니다. 결과 Geometry는 제공된 Geometry의 공간 참조 시스템에 표시됩니다.

제공된 Geometry가 점, 다중점, 닫힌 곡선 또는 닫힌 다중 곡선이거나 비어 있으면, 결과는 유형 ST_Point의 빈 Geometry입니다. 닫혀 있지 않은 곡선 또는 다중 곡선의 경우, 곡선의 시작점과 끝점이 짝수 곡선의 시작점 또는 끝점이 아니면 이 점은 ST_MultiPoint 값으로 리턴됩니다. 표면 및 다중 표면의 경우, 제공된 Geometry의 바운더리를 정의하는 곡선이 ST_Curve 또는 ST_MultiCurve 값으로 리턴됩니다. 제공된 Geometry가 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

가능한 경우, 리턴된 Geometry의 특정 유형은 ST_Point, ST_LineString 또는 ST_Polygon입니다. 예를 들어, 홀(hole)이 없는 다각형의 바운더리는 ST_LineString으로 표시되는 단일 선스트링입니다. 하나 이상의 홀이 있는 다각형의 바운더리는 ST_MultiLineString으로 표시되는 다중 선스트링으로 구성됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 또는 해당 부속 유형 중 하나의 값. 이 Geometry의 바운더리가 리턴됩니다.

리턴 유형

db2gse.ST_Geometry

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 여러 개의 Geometry를 작성하고 각 Geometry의 바운더리를 판별합니다.

```
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
```

결과:

```
ID      BOUNDARY
```

```
------- ------------------------------------------------------------------
      1 LINESTRING ( 40.00000000 120.00000000, 90.00000000 120.00000000,
          90.00000000 150.00000000, 40.00000000 150.00000000, 40.00000000
          120.00000000)

      2 MULTILINESTRING (( 40.00000000 120.00000000, 90.00000000 120.00000000,
          90.00000000 150.00000000, 40.00000000 150.00000000, 40.00000000
          120.00000000),( 70.00000000 130.00000000, 70.00000000 140.00000000,
          80.00000000 140.00000000, 80.00000000 130.00000000, 70.00000000
          130.00000000))

      3 MULTIPOINT ( 60.00000000 60.00000000, 70.00000000 70.00000000)

      4 MULTIPOINT ( 50.00000000 50.00000000, 70.00000000 70.00000000,
          80.00000000 80.00000000, 90.00000000 90.00000000)

      5 POINT EMPTY
```

ST_Buffer

ST_Buffer는 Geometry, 거리 및 단위(선택적)를 입력 매개변수로 가져오며 제공된 단위로 측정된 지정 거리만큼 제공된 Geometry를 둘러싸는 Geometry를 리턴합니다.

결과 Geometry의 바운더리에 있는 각 점은 제공된 Geometry로부터 지정된 거리만큼 떨어져 있습니다. 결과 Geometry는 제공된 Geometry의 공간 참조 시스템에 표시됩니다.

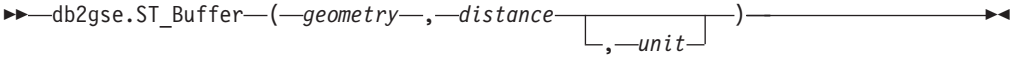
측지 데이터의 경우 음수 거리를 지정하면 ST_Buffer는 입력 Geometry의 모든 점으로부터 지정된 거리보다 먼 영역을 리턴합니다. 즉, 음수 거리는 보완 영역을 리턴합니다.

결과 Geometry의 바운더리에 있는 임의의 순환 곡선은 선형 스트링에 가깝습니다. 예를 들어, 순환 영역을 생성하는 점 주위의 버퍼는 바운더리가 선스트링인 다각형에 가깝습니다.

제공된 Geometry가 널(NULL)이거나 비어 있는 경우, 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

유형 ST_Geometry 값 또는 버퍼 주위를 작성하기 위해 Geometry를 표시하는 해당 부속 유형 중 하나의 값 측지 데이터의 경우 ST_Buffer는 ST_Point 및 ST_MultiPoint 데이터 유형만을 지원합니다.

거리 *geometry* 주위의 버퍼에 사용될 거리를 지정하는 DOUBLE PRECISION 값 측지 데이터의 경우 거리는 지구의 적도 반경보다 길지 않아야 합니다. WGS-84 타원체의 경우, 이 길이는 6378137.0미터입니다.

unit *unit*를 측정하는 단위를 식별하는 VARCHAR(128) 값. 지원되는 측정 단위는 DB2GSE.ST_UNITS_OF_MEASURE 카탈로그 뷰에 나열되어 있습니다.

단위 매개변수를 생략하면 다음 규칙을 사용하여 거리에 사용되는 측정 단위를 판별합니다.

- *Geometry*가 투영 또는 지구 중심 좌표 시스템에 있는 경우, 이 좌표 시스템과 연관된 선형 단위가 디폴트값입니다.
- *Geometry*가 지리 좌표 시스템에 있지만 측지 공간 참조 시스템(SRS)에 없는 경우, 이 좌표 시스템과 연관된 각도 단위가 디폴트값입니다.
- *Geometry*가 측지 SRS에 있는 경우, 디폴트 측정 단위는 미터입니다.

단위 변환에 대한 제한사항: 다음과 같은 조건이 발생하는 경우 오류(SQLSTATE 38SU4)가 리턴됩니다.

- Geometry가 지정되지 않은 좌표 시스템에 있고 *unit* 매개변수가 지정됩니다.
- Geometry가 투영 좌표 시스템에 있고 각도 단위가 지정됩니다.
- Geometry는 지리 좌표 시스템에 있지만 측지 SRS에는 없으며 선형 단위가 지정됩니다.
- Geometry가 지리 좌표 시스템에 있고 측지 SRS에도 있으며 각도 단위가 지정됩니다.

리턴 유형

db2gse.ST_Geometry

예

다음 예에서는 읽기 쉽도록 결과를 다시 형식화했습니다. 결과의 스페이스는 사용자의 표시 장치에 따라 다릅니다.

예 1

다음 코드는 공간 참조 시스템을 작성하고 SAMPLE_GEOMETRIES 테이블을 작성한 후 데이터를 채웁니다.

```
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))
```

예 2

다음 SELECT문은 ST_Buffer 함수를 사용하여 버퍼 10을 적용합니다.

```
SELECT id, spatial_type,
       cast(geometry..ST_Buffer(10)..ST_AsText AS varchar(470)) AS buffer_10
FROM sample_geometries
```

결과:

```
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))
```

예 3

다음 SELECT문은 ST_Buffer 함수를 사용하여 음수 버퍼 5를 적용합니다.

```
SELECT id, spatial_type,
       cast(ST_AsText(ST_Buffer(geometry, -5)) AS varchar(150))
       AS buffer_negative_5
FROM   sample_geometries
WHERE  id = 3
```

결과:

```
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))
```

예 4

다음 SELECT문에서는 단위 매개변수가 지정된 버퍼를 적용한 결과를 보여줍니다.

```
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
```

결과:

```
ID          SPATIAL_TYPE       BUFFER_10_METER
----------- ------------------ -------------------------------------
3           st_polygon         POLYGON (( 163.00000000 120.00000000,
   163.00000000 140.00000000, 162.00000000 149.00000000, 159.00000000
   157.00000000, 152.00000000 165.00000000, 143.00000000 170.00000000,
   130.00000000 173.00000000, 110.00000000 173.00000000, 101.00000000
   172.00000000, 92.00000000 167.00000000, 84.00000000 160.00000000,
   79.00000000 151.00000000, 77.00000000 140.00000000, 77.00000000
   120.00000000, 78.00000000 111.00000000, 83.00000000 102.00000000,
   90.00000000 94.00000000, 99.00000000 89.00000000, 110.00000000
   87.00000000, 130.00000000 87.00000000, 139.00000000 88.00000000,
   147.00000000 91.00000000, 155.00000000 98.00000000, 160.00000000
   107.00000000, 163.00000000 120.00000000))
```

ST_Centroid

ST_Centroid는 입력 매개변수로 Geometry를 가져오며 점으로 제공된 Geometry의 최소 바운딩 직사각형의 중심인 Geometry 중심을 리턴합니다. 결과 점은 제공된 Geometry의 공간 참조 시스템에 표시됩니다.

제공된 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 값 또는 Geometry 중심을 판별하기 위해 Geometry를 표시하는 해당 부속 유형 중 하나의 값

리턴 유형

db2gse.ST_Point

예

이 예는 두 개의 Geometry를 작성하고 둘 사이의 중심을 찾습니다.

```
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
```

결과:

```
ID CENTROID
---------- ----------------------------------------
         1 POINT ( 65.00000000 135.00000000)
         2 POINT ( 30.00000000 20.00000000)
```

ST_ChangePoint

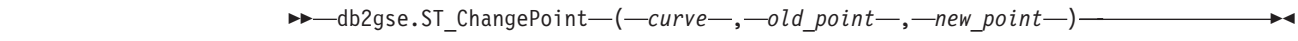
ST_ChangePoint는 입력 매개변수로 곡선과 두 개의 점을 가져옵니다. 제공된 곡선의 발생한 모든 첫 번째 점을 두 번째 점으로 바꾸고 결과 곡선을 리턴합니다. 결과 Geometry는 제공된 Geometry의 공간 참조 시스템에 표시됩니다.

두 개의 점이 곡선과 동일한 공간 참조 시스템에 표시되지 않으면, 곡선에 사용된 공간 참조 시스템으로 변환됩니다.

제공된 곡선이 비어 있으면, 빈 값이 리턴됩니다. 제공된 곡선이 널(NULL)이거나 제공된 점 중 하나가 널(NULL)이거나 비어 있으면, 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

curve 유형 ST_Curve의 값이나 *old_point*로 식별되는 점이 *new_point*로 변경되는 곡선을 표시하는 해당 부속 유형 중 하나의 값

old_point

*new_point*로 변경되는 곡선의 점을 식별하는 유형 ST_Point의 값

new_point

*old_point*로 식별되는 곡선에서 점의 새 위치를 표시하는 유형 ST_Point의 값

리턴 유형

db2gse.ST_Curve

제한사항

곡선에서 변경할 점은 곡선을 정의하는 데 사용된 점 중 하나여야 합니다.

곡선에 Z 또는 M 좌표가 있는 경우, 제공된 점에도 Z 또는 M 좌표가 있어야 합니다.

예

예 1

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

다음 코드는 SAMPLE_LINES 테이블을 작성한 후 데이터를 채웁니다.

```
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) )
```

예 2

이 예는 선스트링에 있는 점 (5, 5)의 모든 어커런스를 점 (6, 6)으로 변경합니다.

```
SELECT cast(ST_AsText(ST_ChangePoint(line, ST_Point(5, 5),
                                     ST_Point(6, 6))) as VARCHAR(160))
FROM   sample_lines
WHERE  id=1
```

예 3

이 예는 선스트링에 있는 점 (5, 5, 5)의 모든 어커런스를 점 (6, 6, 6)으로 변경합니다.

```
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
```

결과:

```
NEW
--------------------------------------------------------------------------------
LINESTRING Z ( 0.00000000 0.00000000 4.00000000, 6.00000000 6.00000000
6.00000000, 10.00000000 10.00000000 6.00000000, 5.00000000 5.00000000
7.00000000)
```

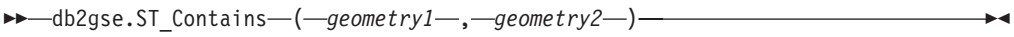
ST_Contains

ST_Contains는 두 개의 Geometry를 입력 매개변수로 가져오며 첫 번째 Geometry가 두 번째 Geometry를 완전히 포함하면 1을 리턴하고, 그렇지 않으면 0을 리턴하여 첫 번째 Geometry가 두 번째 Geometry를 완전히 포함하지 않음을 나타냅니다.

제공된 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

비측지 데이터의 경우 두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템으로 표현되지 않으면 다른 공간 참조 시스템으로 변환됩니다. 측지 데이터의 경우, 두 Geometry는 모두 동일한 측지 공간 참조 시스템(SRS) 내에 있어야 합니다.

구문



매개변수

geometry1

*geometry2*를 완전히 포함하는지를 테스트할 Geometry를 나타내는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

geometry2

완전히 *geometry1* 내에 들어 있는지를 테스트할 Geometry를 나타내는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

제한사항: 측지 데이터의 경우, 두 Geometry는 모두 측지 Geometry로서 동일한 측지 SRS 내에 있어야 합니다.

리턴 유형

INTEGER

예

예 1

다음 코드는 SAMPLE_LINES 테이블을 작성한 후 데이터를 채웁니다.

```
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) )
```

예 2

다음 코드 조각은 ST_Contains 함수를 사용하여 특정 다각형에 포함되는 점을 판별합니다.

```
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
```

결과:

```
POLYGON_ID CONTAINS         POINT_ID
---------- ---------------- --------
       100 does contain            1
       100 does not contain        2
```

예 3

다음 코드 조각은 ST_Contains 함수를 사용하여 특정 다각형에 포함되는 선을 판별합니다.

```
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
```

결과:

```
POLYGON_ID CONTAINS         LINE_ID
---------- ---------------- -------
       100 does contain          10
       100 does not contain      20
```

ST_ConvexHull

ST_ConvexHull은 입력 매개변수로 Geometry를 가져오며 해당 convex hull을 리턴합니다.

결과 Geometry는 제공된 Geometry의 공간 참조 시스템에 표시됩니다.

가능한 경우, 리턴된 Geometry의 특정 유형은 ST_Point, ST_LineString 또는 ST_Polygon입니다. 예를 들어, 홀(hole)이 없는 다각형의 바운더리는 ST_LineString으로 표시되는 단일 선스트링입니다. 하나 이상의 홀이 있는 다각형의 바운더리는 ST_MultiLineString으로 표시되는 다중 선스트링으로 구성됩니다.

제공된 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 값 또는 convex hull을 계산할 Geometry를 표시하는 해당 부속 유형 중 하나의 값

리턴 유형

db2gse.ST_Geometry

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

다음 코드는 SAMPLE_GEOMETRIES 테이블을 작성하고 상주시킵니다.

```
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))
```

다음 SELECT문은 위에서 구성한 모든 Geometry에 대한 볼록한 선체(convex hull)를 계산하고 결과를 표시합니다.

```
SELECT id, spatial_type, cast(geometry..ST_ConvexHull..ST_AsText
   AS varchar(300)) AS convexhull
FROM   sample_geometries
```

결과:

```
ID     SPATIAL_TYPE       CONVEXHULL
------ ------------------ --------------------------------------------
    1 ST_LineString      POLYGON (( 20.00000000 40.00000000,
                           20.00000000 20.00000000, 30.00000000
                           30.00000000, 30.00000000 50.00000000,
                           20.00000000 40.00000000))

    2 ST_Polygon         POLYGON (( 110.00000000 140.00000000,
                           110.00000000 120.00000000, 120.00000000
                           130.00000000,  110.00000000 140.00000000))

    3 ST_Polygon         POLYGON (( 15.00000000 50.00000000,
                           25.00000000 35.00000000, 30.00000000
                           30.00000000, 60.00000000 30.00000000,
                           75.00000000 40.00000000, 80.00000000
                           90.00000000, 40.00000000 85.00000000,
                           35.00000000 80.00000000, 15.00000000
                           50.00000000))

    4 ST_MultiPoint      POLYGON (( 20.00000000 40.00000000,
                           20.00000000 20.00000000, 30.00000000
                           30.00000000, 30.00000000 50.00000000,
                           20.00000000 40.00000000))
```

ST_CoordDim

ST_CoordDim은 입력 매개변수로 Geometry를 가져오며 해당 좌표의 차원성을 리턴합니다.

제공된 Geometry에 Z 및 M 좌표가 없으면 차원은 2입니다. Z 좌표는 있고 M 좌표는 없거나 M 좌표는 있고 Z 좌표는 없으면 차원은 3입니다. Z 및 M 좌표가 있으면 차원은 4입니다. Geometry가 널(NULL)이면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 값 또는 차원성을 검색할 Geometry를 표시하는 해당 부속 유형 중 하나의 값

리턴 유형

INTEGER

예

이 예는 여러 개의 Geometry를 작성한 후 해당 좌표의 차원성을 판별합니다.

```
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
```

결과:

```
ID              COORDDIM
--------------- -----------
Empty Point               2
Linestring                2
Polygon                   2
Multipoint M              3
Multipoint Z              3
Point ZM                  4
```

ST_Crosses

ST_Crosses는 두 개의 Geometry를 입력 매개변수로 가져오며 첫 번째 Geometry가 두 번째 Geometry와 교차할 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다.

두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템에 표시되지 않으면, 다른 공간 참조 시스템으로 변환됩니다.

첫 번째 Geometry가 다각형 또는 다중 다각형이거나 두 번째 Geometry가 점 또는 다중점이거나 Geometry 중 하나가 널(NULL) 값 또는 비어 있는 경우, 널(NULL) 값이 리턴됩니다. 두 Geometry가 교차되며 제공된 두 Geometry의 최대 차원보다 작은 차원을 갖는 Geometry를 생성하거나 결과 Geometry가 제공된 두 Geometry 중 어느 것과도 동일하지 않을 경우에 1이 리턴됩니다. 그렇지 않으면, 결과는 0(영)입니다.

구문

매개변수

geometry1

*geometry2*와 교차 여부를 테스트할 Geometry를 나타내는 ST_Geometry 또는 해당 부속 유형 중 하나의 값

geometry2

*geometry1*과 교차하는지 여부를 판별하기 위해 테스트할 Geometry를 나타내는 ST_Geometry 또는 해당 부속 유형 중 하나의 값

리턴 유형

INTEGER

예

다음 코드는 구성된 Geometry가 서로 교차하는지를 판별합니다.

```
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
```

결과:

```
ID          ID          CROSSES
----------- ----------- -----------
          1           1           -
          2           1           0
          3           1           1
          1           2           -
          2           2           0
          3           2           1
          1           3           -
          2           3           1
          3           3           0
```

ST_Difference

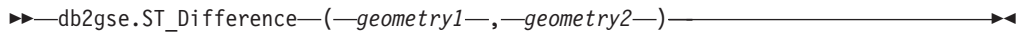
ST_Difference에서는 두 개의 Geometry를 입력 매개변수로 사용하며 두 번째 Geometry와 교차하지 않는 첫 번째 Geometry의 파트를 리턴합니다.

두 Geometry의 차원은 동일해야 합니다. Geometry가 널(NULL)이면 널(NULL)이 리턴됩니다. 첫 번째 Geometry가 비어 있으면, 유형 ST_Point의 빈 Geometry가 리턴됩니다. 두 번째 Geometry가 비어 있으면, 첫 번째 Geometry가 변경되지 않은 상태로 리턴됩니다.

비측지 데이터의 경우 두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템으로 표현되지 않으면 다른 공간 참조 시스템으로 변환됩니다. 측지 데이터의 경우, 두 Geometry는 모두 동일한 측지 공간 참조 시스템(SRS) 내에 있어야 합니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry1

*geometry2*와의 차이를 계산하는 데 사용할 첫 번째 Geometry를 표시하는 유형 ST_Geometry의 값

geometry2

*geometry1*과의 차이를 계산하는 데 사용할 두 번째 Geometry를 표시하는 유형 ST_Geometry의 값

측지 데이터의 제한사항:

- 두 Geometry는 모두 측지 Geometry로서 동일한 측지 SRS 내에 있어야 합니다.
- ST_Difference는 ST_Point, ST_Polygon, ST_MultiPoint 및 ST_MultiPolygon 데이터 유형만 지원합니다.

리턴 유형

db2gse.ST_Geometry

리턴된 Geometry의 차원은 입력 Geometry의 차원과 동일합니다.

예

다음 예에서 결과는 읽기 쉽도록 다시 형식화된 것입니다. 결과의 스페이스는 사용자의 표시 장치에 따라 다릅니다.

다음 코드는 SAMPLE_GEOMETRIES 테이블을 작성하고 상주시킵니다.

```
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))
```

예 1

이 예는 두 개의 분리된 다각형 간의 차이점을 찾습니다.

```
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
```

결과:

```
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))
```

예 2

이 예는 두 개의 교차하는 다각형 간의 차이점을 찾습니다.

```
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
```

결과:

```
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))
```

예 3

이 예는 두 개의 겹치는 선스트링 간의 차이점을 찾습니다.

```
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
```

결과:

```
ID       ID       DIFFERENCE
-------- -------- ----------------------------------------------------
       4        5 LINESTRING ( 70.00000000 70.00000000, 75.00000000
                     75.00000000)
```

ST_Dimension

ST_Dimension은 입력 매개변수로 Geometry를 가져오며 차원을 리턴합니다.

제공된 Geometry가 비어 있으면, -1이 리턴됩니다. 점 및 다중점의 경우 차원은 0이고, 곡선 및 다중 곡선의 경우 차원은 1이며, 다각형 및 다중 다각형의 경우 차원은 2입니다. 제공된 Geometry가 널(NULL)이면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

차원이 리턴되는 Geometry를 표시하는 유형 ST_Geometry의 값

리턴 유형

INTEGER

예

이 예는 여러 개의 다른 Geometry를 작성하고 Geometry의 차원을 찾습니다.

```
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
```

결과:

```
ID              DIMENSION
--------------- -----------
Empty Point              -1
Point ZM                  0
MultiPoint M              0
LineString                1
Polygon                   2
```

ST_Disjoint

ST_Disjoint는 입력 매개변수로 두 개의 Geometry를 가져오며 제공된 Geometry가 교차하지 않는 경우 1을 리턴합니다. Geometry가 교차하면, 0이 리턴됩니다.

두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템에 표시되지 않으면, 다른 공간 참조 시스템으로 변환됩니다.

두 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry1

*geometry2*와 분리되어 있는지를 테스트할 Geometry를 표시하는 ST_Geometry 유형의 값

geometry2

*geometry1*과 분리되어 있는지를 테스트할 Geometry를 표시하는 ST_Geometry 유형의 값

리턴 유형

INTEGER

예

예 1

다음 코드는 SAMPLE_GEOMETRIES 테이블에 여러 개의 Geometry를 작성합니다.

```
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))
```

예 2

이 예는 첫 번째 다각형이 다른 Geometry와 분리되어 있는지를 판별합니다.

```
SELECT a.id, b.id, ST_Disjoint(a.geometry, b.geometry) DisJoint
FROM sample_geoms a, sample_geoms b
WHERE a.id = 1
```

결과:

```
ID          ID          DISJOINT
----------- ----------- -----------
          1           1           0
          1           2           0
          1           3           1
          1           4           1
          1           5           0
```

예 3

이 예는 세 번째 다각형이 다른 Geometry와 분리되어 있는지를 판별합니다.

```
SELECT a.id, b.id, ST_Disjoint(a.geometry, b.geometry) DisJoint
FROM sample_geoms a, sample_geoms b
WHERE a.id = 3
```

결과:

```
ID          ID          DISJOINT
----------- ----------- -----------
          3           1           1
          3           2           0
          3           3           0
          3           4           0
          3           5           0
```

예 4

이 예는 두 번째 선스트링이 다른 Geometry와 분리되어 있는지를 판별합니다.

```
SELECT a.id, b.id, ST_Disjoint(a.geometry, b.geometry) DisJoint
FROM sample_geoms a, sample_geoms b
WHERE a.id = 5
```

결과:

```
ID          ID          DISJOINT
----------- ----------- -----------
          5           1           0
          5           2           0
          5           3           0
          5           4           1
          5           5           0
```

ST_Distance

ST_Distance에서는 두 개의 Geometry와 단위(선택적)를 입력 매개변수로 사용하며 첫 번째 Geometry 점과 두 번째 Geometry 점 사이의 최단 거리를 디폴트 단위 또는 지정된 단위로 측정하여 리턴합니다.

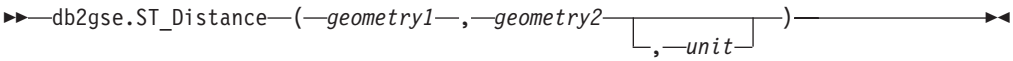
측지 데이터의 경우, ST_Distance는 두 Geometry 사이의 *측지 거리*를 리턴합니다. 측지 거리는 ellipsoid 표면의 최단 거리입니다.

두 Geometry 중 하나가 널(NULL)이거나 비어 있는 경우, 널(NULL)이 리턴됩니다.

비측지 데이터의 경우 두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템으로 표현되지 않으면 다른 공간 참조 시스템으로 변환됩니다. 측지 데이터의 경우, 두 Geometry는 모두 동일한 측지 공간 참조 시스템(SRS) 내에 있어야 합니다.

수치 단위를 제공할 때 이 함수를 메소드로 호출할 수도 있습니다.

구문



매개변수

geometry1

*geometry2*에 대한 거리를 계산하는 데 사용되는 Geometry를 표시하는 ST_Geometry 유형의 값

geometry2

*geometry1*과의 차이를 계산하는 데 사용할 Geometry를 표시하는 유형 ST_Geometry의 값

unit 결과를 측정하는 단위를 식별하는 VARCHAR(128) 값. 지원되는 측정 단위는 DB2GSE.ST_UNITS_OF_MEASURE 카탈로그 뷰에 나열되어 있습니다.

측지 데이터의 경우, 두 Geometry는 모두 측지 Geometry로서 동일한 측지 SRS 내에 있어야 합니다.

unit 매개변수를 생략하면 결과에 사용되는 측정 단위를 판별하기 위해 다음과 같은 규칙이 사용됩니다.

- *geometry1*이 투영 또는 지구 중심 좌표 시스템에 있는 경우 이 좌표 시스템과 연관된 선형 단위가 디폴트입니다.
- *geometry1*이 지구 중심 좌표 시스템에는 있지만 측지 SRS에는 없는 경우, 이 좌표 시스템과 연관된 각도 단위는 디폴트입니다.
- *geometry1*이 측지 SRS에 있는 경우, 디폴트 측정 단위는 미터입니다.

단위 변환에 대한 제한사항: 다음과 같은 조건이 발생하는 경우 오류(SQLSTATE 38SU4)가 리턴됩니다.

- Geometry가 지정되지 않은 좌표 시스템에 있고 *unit* 매개변수가 지정됩니다.
- Geometry가 투영 좌표 시스템에 있고 각도 단위가 지정됩니다.
- Geometry가 지리 좌표 시스템에 있고 측지 SRS에도 있으며 각도 단위가 지정됩니다.

리턴 유형

DOUBLE

예

예 1

다음 SQL문은 SAMPLE_GEOMETRIES1 및 SAMPLE_GEOMETRIES2 테이블을 작성하고 데이터를 채웁니다.

```
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) )
```

예 2

다음 SELECT문은 SAMPLE_GEOMTRIES1 및 SAMPLE_GEOMTRIES2 테이블의 여러 Geometry 간의 거리를 계산합니다.

```
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
```

결과:

```
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
```

예 3

다음 SELECT문은 거리가 100 이하인 모든 Geometry를 찾는 방법을 보여줍니다.

```
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
```

결과:

```
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
```

예 4

다음 SELECT문은 여러 Geometry 간의 거리를 킬로미터로 계산합니다.

```
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
```

결과:

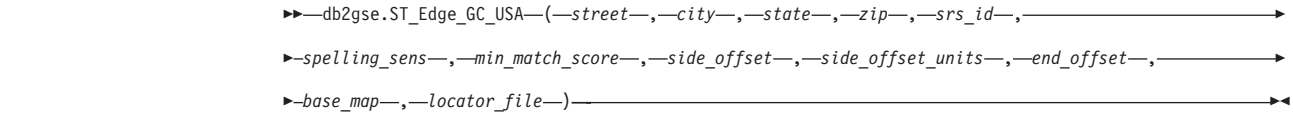
```
SG1_ID SG1_TYPE      SG1_ID SG2_TYPE      DISTANCE
------ ------------- ------ ------------- ------------
     1 ST_Point         101 ST_Point        12373.2168
     1 ST_Point         102 ST_Point        16311.3816
     1 ST_Point         103 ST_Point         9809.4713
     1 ST_Point         110 ST_LineString    1707.4463
     1 ST_Point         120 ST_Polygon      12373.2168
    10 ST_LineString    101 ST_Point         8648.2333
    10 ST_LineString    102 ST_Point        11317.3934
    10 ST_LineString    103 ST_Point        10959.7313
    10 ST_LineString    110 ST_LineString    3753.5862
    10 ST_LineString    120 ST_Polygon      10891.1254
    20 ST_Polygon       101 ST_Point         7700.5333
    20 ST_Polygon       102 ST_Point        15039.8109
    20 ST_Polygon       103 ST_Point         7284.8552
    20 ST_Polygon       110 ST_LineString    6001.8407
    20 ST_Polygon       120 ST_Polygon      14515.8872
```

ST_Edge_GC_USA

ST_Edge_GC_USA는 미국에 있는 주소를 점으로 지오코드하는 DB2SE_USA_GEOCODER를 구현하는 함수입니다. 주소는 지오코더(Geocoder) 데이터 CD에 제공된 EDGE 파일과 비교(일치)됩니다.

함수는 입력 매개변수로서 거리 번호와 이름, 도시 이름, 주, 우편번호 및 공간 참조 시스템 ID를 가져와 ST_Point 값을 리턴합니다. 또한 지오코딩(Geocoding) 프로세스에 영향을 미치는 여러 구성 매개변수를 지정할 수 있습니다.

구문



매개변수

street 지오코드할 주소의 거리 번호 및 이름을 포함하는 VARCHAR(128) 유형의 값.

이 값은 널(NULL)이어서는 안됩니다.

city 지오코드할 주소의 시 이름을 포함하는 VARCHAR(128) 유형의 값.

zip 매개변수가 지정된 경우 이 값은 널(NULL)일 수 있습니다.

state 지오코드할 주소의 주 이름을 포함하는 VARCHAR(128) 유형의 값. 주는 약어 또는 완전한 철자를 사용할 수 있습니다.

zip 매개변수가 지정된 경우 이 값은 널(NULL)일 수 있습니다.

zip 지오코드할 주소의 우편번호를 포함하는 VARCHAR(10) 유형의 값. 우편번호는 5자리 또는 5+4 표기법으로 제공할 수 있습니다.

이 값은 *city* 및 *state* 매개변수가 지정된 경우에는 널(NULL)일 수 있습니다.

srs_id 결과 점에 대한 공간 참조 시스템의 숫자 ID를 포함하는 INTEGER 유형의 값. 값은 결과는 지리적 좌표 시스템인 GCS_NORTH_AMERICAN_1983을 기본으로 한 프로젝트화 좌표 시스템을 사용하는 기존의 공간 참조 시스템 또는 지리적 좌표 시스템인 GCS_NORTH_AMERICAN_1983 자체를 사용하는 기존의 공간 참조 시스템을 식별해야 합니다.

*srs_id*가 카탈로그 뷰 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS에 나열된 공간 참조 시스템을 식별하지 못하면 오류(SQLSTATE 38SU1)가 리턴됩니다.

spelling_sens

제공된 주소에 적용할 철자의 중요도를 지정하는 INTEGER 유형의 값. 값의 범위는 0 - 100입니다. 값이 클수록 제공된 주소에 적힌 철자의 차이를 보다 중요하게 생각합니다. 편차가 있으면 최종 일치도에 적용할 높은 벌점이 부과됩니다.

철자 중요도를 너무 높게 설정하면 지오코드되는 주소의 수가 줄고 대신 널(NULL)이 리턴됩니다. 철자 중요도를 너무 낮게 설정하면, 주소에 적힌 철자의 승인 레벨이 차이가 나기 때문에 일치하지 않는 주소도 일치하는 주소로 간주됩니다. **권장사항:** 이 값을 60으로 설정하십시오.

이 값이 널(NULL)이면 철자 중요도는 로케이터 파일에서 파생됩니다. 로케이터 파일에 값이 지정되지 있지 않으면 철자 중요도 60이 사용됩니다.

min_match_score

제공된 주소에 대해 점이 일치하는 것으로 간주해야 할 최소 범위 값을 포함하는 INTEGER 유형의 값. 최소 값의 범위는 0 - 100이어야 합니다. 점의 범위가 *min_match_score* 값보다 작으면 점 대신 널(NULL)이 리턴되며 주소는 지오코드되지 않습니다.

주소가 점의 범위에 영향을 미칠 경우, 기본 맵의 수량, 철자 중요도 또는 정확도와 같은 다른 인수. **권장사항:** 이 값을 80으로 설정하십시오.

이 값이 널(NULL)이면 최소 일치 범위는 로케이터 파일에서 파생됩니다. 로케이터 파일에 값이 지정되지 있지 않으면 최소 범위 값 80이 사용됩니다.

side_offset

결과 점이 거리의 중심에서 얼마나 멀리 떨어져 있는지를 지정하는 DOUBLE 유형의 값. 값은 0보다 크거나 같아야 합니다. *side_offset_unit* 매개변수는 측면 오프셋을 측정하는 데 사용되는 단위를 지정합니다.

이 값이 널(NULL)이면 측면 오프셋은 로케이터 파일에서 파생됩니다. 로케이터 파일에 값이 지정되지 있지 않으면 측면 오프셋 0.0이 사용됩니다.

side_offset_units

side_offset 매개변수를 측정하는 단위를 포함하는 VARCHAR(128) 유형의 값. 값은 다음 값 중 하나여야 합니다.

- 인치
- 점
- 피트
- 야드
- 마일
- 해리
- 밀리미터
- 센티미터
- 미터
- 킬로미터
- 10진수(도)
- 프로젝트화 미터
- 참조 데이터 단위

이 값이 널(NULL)이면 측면 오프셋은 로케이터 파일에서 파생됩니다. 로케이터 파일에 값이 지정되지 있지 않으면 측면 오프셋은 피트 단위로 특정됩니다.

end_offset

정확히 거리 세그먼트의 끝에 있어야 하는 점이 세그먼트 내 어디쯤에 위치하는지를 나타냅니다. 값은 0보다 크거나 같아야 합니다. 이 매개변수는 결과 점이 교차로 중간에 놓이지 않도록 하는 데 사용됩니다. 끝 오프셋은 기본 맵에서 점(가능한 최소 해상도)으로 측정됩니다.

이 값이 널(NULL)이면 끝 오프세트는 로케이터 파일에서 파생됩니다. 로케이터 파일에 값이 지정되지 있지 않으면 끝 오프셋 3이 사용됩니다.

base_map

기본 이름을 비롯하여 기본 맵(.edg) 파일에 대한 완전한 경로를 포함하는 VARCHAR(256) 유형의 값. 기본 맵 파일은 지오코더(Geocoder)가 제공된 주소를 일치시킬 때 사용합니다. DB2 Spatial Extender에서 제공하는 기본 맵이 사용됩니다. 다른 디렉토리에 기본 맵을 배치한 경우에는 이 매개변수를 사용할 수 있습니다.

이 값이 널(NULL)이면 기본 맵에 대한 경로는 로케이터 파일에서 파생됩니다. 로케이터 파일에서 지정되지 않으면, 현재 인스턴스의 sqllib 디렉토리인 gse/refdata 서브디렉토리에서 기본 맵을 검색합니다. 검색할 파일의 기본 이름은 usa.edg입니다.

locator_file

기본 이름을 비롯하여 지오코더(Geocoder)에 대한 추가 구성 매개변수를 포함하는 로케이터 파일의 완전한 경로를 포함하는 VARCHAR(256) 유형의 값. DB2 Spatial Extender에서 제공하는 로케이터 파일이 사용됩니다.

이 값이 널이면, 현재 인스턴스의 sqllib 디렉토리인 gse/cfg/geocoder 서브디렉토리에서 로케이터 파일을 검색합니다. 검색할 파일의 기본 이름은 EDGELocator.loc입니다.

리턴 유형

db2gse.ST_Point

예

예 1

다음 코드는 SAMPLE_GEOCODING 테이블을 작성하고 계속해서 지오코드되는 두 개의 주소를 삽입합니다. 제공된 주소에 대한 최소 일치도는 50으로 설정되며 결과 점에 대한 공간 참조 시스템은 1입니다.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geocoding (
 street VARCHAR(128),
  city   VARCHAR(128),
 state  VARCHAR(128),
 zip    VARCHAR(5) )

INSERT INTO geocoding(street, city, state, zip)
 VALUES ('1212 New York Ave NW', 'Washington', 'DC', '20005'),
 ('100 First North Street', 'San Jose', 'CA', NULL)

SELECT VARCHAR(ST_AsText(ST_Edge_GC_USA(street, city, state, zip, 1,
    CAST(NULL AS INTEGER), 50, CAST(NULL AS DOUBLE),
    CAST(NULL AS VARCHAR(128)), CAST(NULL AS INTEGER),
     CAST(NULL AS VARCHAR(256)), CAST(NULL AS VARCHAR(256)))), 50)
FROM sample_geocoding
```

결과:

```
1
--------------------------------------------------
POINT ( -77.02829300 38.90049000)
POINT ( -121.94507200 37.28766700)
```

예 2

이 예에서는 프로젝트화 좌표 시스템을 사용하는 공간 참조 시스템을 작성합니다. 지오코딩(Geocoding) 함수에 대한 인터페이스를 단순화하기 위해 ST_Edge_GC_USA 함수를 랩하는 사용자 정의 함수를 작성합니다.

```
db2se create_srs <db_name> -srsName CALIFORNIA -srsId 101 -xScale 1
  -coordsysName NAD_1983_STATEPLANE_CALIFORNIA_I_FIPS_0401

SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE FUNCTION California_GC (
  street VARCHAR(128), city VARCHAR(128), zip VARCHAR(10))
```

```
 RETURNS db2gse.ST_Point
   LANGUAGE SQL
   RETURN db2gse.ST_Edge_GC_USA(street, city, 'CA', zip, 101,
     CAST(NULL AS INTEGER), CAST(NULL AS INTEGER),
     CAST(NULL AS DOUBLE), CAST(NULL AS VARCHAR(128)),
       CAST(NULL AS INTEGER), CAST(NULL AS VARCHAR(256)))

CREATE TABLE sample_geocoding (
  street VARCHAR(128),
  city  VARCHAR(128),
  state  VARCHAR(128),
    zip    VARCHAR(5) )

INSERT INTO geocoding(street, city, state, zip)
VALUES ('100 First North Street', 'San Jose', 'CA', NULL)

SELECT VARCHAR(ST_AsText(California_GC(street, city, zip)), 50)
FROM sample_geocoding
```

결과:

```
1
--------------------------------------------------
POINT ( 2004879.00000000 272723.00000000)
```

다른 공간 참조 시스템을 사용하므로 점의 X 및 Y 좌표 값은 첫 번째 예의 값과는 다릅니다.

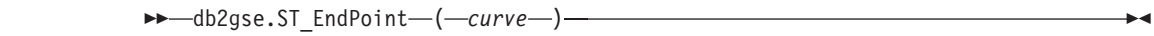
ST_Endpoint

ST_Endpoint는 입력 매개변수로 곡선을 가져오며 곡선의 마지막 점을 리턴합니다. 결과 점은 제공된 곡선의 공간 참조 시스템에 표시됩니다.

제공된 곡선이 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

curve 마지막 점이 리턴되는 Geometry를 표시하는 유형 ST_Curve의 값

리턴 유형

db2gse.ST_Point

예

SELECT문은 SAMPLE_LINES 테이블에서 각 Geometry의 끝점을 찾습니다.

```
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
```

결과:

```
ID          ENDPOINT
----------- --------------------------------------------------
          1 POINT ( 0.00000000 10.00000000)
          2 POINT Z ( 5.00000000 5.00000000 7.00000000)
```

ST_Envelope

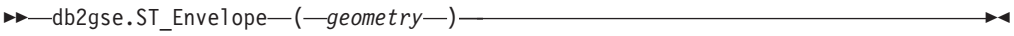
ST_Envelope는 입력 매개변수로 Geometry를 가져오며 Geometry 주위에 인벨로프를 리턴합니다. 인벨로프는 다각형으로 표시되는 직사각형입니다.

제공된 Geometry가 점, 수평 선스트링 또는 수직 선스트링이면, 제공된 Geometry보다 조금 큰 직사각형이 리턴됩니다. 그렇지 않으면, Geometry의 최소 바운딩 직사각형이 인벨로프로 리턴됩니다. 제공된 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다. 모든 Geometry에 대해 정확한 최소 바운딩 직사각형을 리턴하려면 ST_MBR 함수를 사용하십시오.

측지 데이터의 경우, 인벨로프는 Geometry의 최소 바운드 원을 감싸는 다각형입니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

인벨로프(envelope)를 리턴하기 위해 Geometry를 표시하는 유형 ST_Geometry의 값

리턴 유형

db2gse.ST_Polygon

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 여러 개의 Geometry를 작성한 다음 해당 인벨로프를 판별합니다. 비어 있지 않은 점 및 선스트링(수평)의 경우, 인벨로프는 Geometry보다 약간 큰 직사각형입니다.

```
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
```

결과:

```
ID          ENVELOPE
----------- ----------------------------------------------------------------
          1 -

          2 POLYGON (( 9.00000000 9.00000000, 11.00000000 9.00000000,
   11.00000000 11.00000000, 9.00000000 11.00000000, 9.00000000 9.00000000))

          3 POLYGON (( 10.00000000 10.00000000, 50.00000000 10.00000000,
   50.00000000 30.00000000, 10.00000000 30.00000000, 10.00000000
   10.00000000))

          4 POLYGON (( 10.00000000 9.00000000, 20.00000000 9.00000000,
   20.00000000  11.00000000, 10.00000000 11.00000000, 10.00000000
   9.00000000))

          5 POLYGON (( 40.00000000 120.00000000, 90.00000000 120.00000000,
   90.00000000 150.00000000, 40.00000000 150.00000000, 40.00000000
   120.00000000))
```

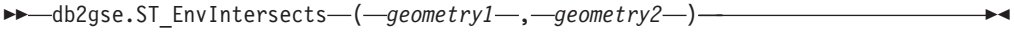
ST_EnvIntersects

ST_EnvIntersect는 입력 매개변수로 Geometry를 가져오며 두 Geometry의 인벨로프가 교차하는 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다.

두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템에 표시되지 않으면, 다른 공간 참조 시스템으로 변환됩니다.

제공된 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

구문



매개변수

geometry1

인벨로프가 *geometry2*의 인벨로프와 교차되는지를 테스트할 Geometry를 표시하는 유형 ST_Geometry 또는 해당 부속 유형 중 하나의 값

geometry2

유형 ST_Geometry 값 또는 인벨로프가 *geometry1*의 인벨로프와의 교차에 대해 테스트할 Geometry를 표시하는 해당 부속 유형 중 하나의 값

리턴 유형

INTEGER

예

이 예는 두 개의 병렬 선스트링을 작성하고 교차 여부를 점검합니다. 선스트링 자체는 교차하지 않으나 선스트링의 인벨로프는 교차합니다.

```
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
```

결과:

```
ID          ID          INTERSECTS  ENVELOPE_INTERSECTS
----------- ----------- ----------- -------------------
          1           2           0                   1
```

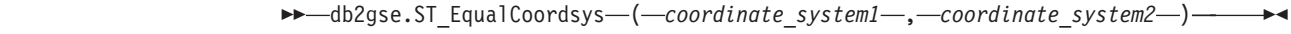
ST_EqualCoordsys

ST_EqualCoordsys는 입력 매개변수로 두 개의 좌표 시스템 정의를 가져오며 제공된 정의가 동일할 경우 정수 값 1을 리턴합니다. 그렇지 않으면 정수 값 0이 리턴됩니다.

좌표 시스템 정의는 스페이스, 괄호, 대문자 및 소문자, 부동 소수점 숫자의 표현의 차이점에 관계없이 비교됩니다.

제공된 좌표 시스템 정의 중 하나가 널(NULL)이면 널(NULL)이 리턴됩니다.

구문



매개변수

coordinate_system1

*coordinate_system2*와 비교할 첫 번째 좌표 시스템을 정의하는 VARCHAR(2048) 유형의 값

coordinate_system2

*coordinate_system1*과 비교할 두 번째 좌표 시스템을 정의하는 VARCHAR(2048) 유형

리턴 유형

INTEGER

예

이 예는 두 개의 오스트리아 좌표 시스템을 비교하여 동일한지를 알아봅니다.

```
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')
      )
```

결과:

```
1
-----------
          0
```

ST_Equals

ST_Equals는 입력 매개변수로 두 개의 Geometry를 가져오며 Geometry가 동일할 경우 1을 리턴합니다. 그렇지 않은 경우에는 0(영)이 리턴됩니다. Geometry를 정의하는 데 사용되는 점의 순서는 동등성에 대한 테스트에 관련되지 않습니다.

두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템에 표시되지 않으면, 다른 공간 참조 시스템으로 변환됩니다.

두 Geometry 중 하나가 널(NULL)이면 널(NULL)이 리턴됩니다.

구문

매개변수

geometry1

*geometry2*와 비교할 Geometry를 표시하는 유형 ST_Geometry의 값

geometry2

*geometry1*과 비교할 Geometry를 표시하는 유형 ST_Geometry의 값

리턴 유형

INTEGER

예

예 1

이 예는 좌표의 순서가 서로 다른 두 개의 다각형을 작성합니다. ST_Equal은 이 다각형이 동일한 것으로 간주되는지를 알아보는 데 사용됩니다.

```
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
```

결과:

```
ID          ID          EQUALS
----------- ----------- -----------
          1           2           1
```

예 2

이 예에서는 X 및 Y 좌표는 동일하나 M 좌표(측정 단위)는 다른 두 개의 Geometry가 작성됩니다. Geometry를 ST_Equal 함수와 비교하면 Geometry가 다르다는 것을 나타내는 0이 리턴됩니다.

```
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
```

결과:

```
ID          ID          EQUALS
----------- ----------- -----------
          3           4           0
```

예 3

이 예에서는 다른 좌표 세트를 사용하여 두 개의 Geometry를 작성하지만 둘 다 동일한 Geometry를 표시합니다. ST_Equal은 Geometry를 비교한 후 두 Geometry가 동일함을 표시합니다.

```
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
```

결과:

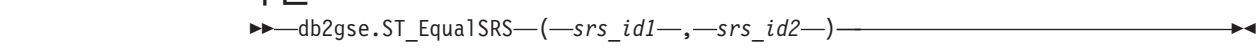
```
ID          ID          EQUALS
----------- ----------- -----------
          5           6           1
```

ST_EqualSRS

ST_EqualSRS는 입력 매개변수로 두 개의 공간 참조 시스템 ID를 가져오며 제공된 공간 참조 시스템이 동일한 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다. 오프셋, 스케일 인수 및 좌표 시스템이 비교됩니다.

임의의 제공된 공간 참조 시스템 ID가 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

구문



매개변수

srs_id1

*srs_id2*로 식별되는 공간 참조 시스템과 비교할 첫 번째 공간 참조 시스템을 식별하는 유형 INTEGER의 값

srs_id2

*srs_id1*로 식별되는 공간 참조 시스템과 비교할 두 번째 공간 참조 시스템을 식별하는 유형 INTEGER의 값

리턴 유형

INTEGER

예

다음과 같은 db2se에 대한 호출을 사용하여 두 개의 유사한 공간 참조 시스템을 작성합니다.

```
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
```

이러한 SRS는 동일한 오프셋 및 스케일 값을 가지며 동일한 좌표 시스템을 참조합니다. 정의된 이름 및 SRS ID만이 다릅니다. 그러므로, 비교 결과 동일하다는 것을 나타내는 1이 리턴됩니다.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

VALUES ST_EqualSRS(12, 22)
```

결과:

```
1
-----------
          1
```

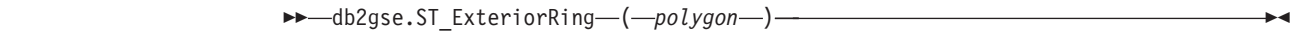
ST_ExteriorRing

ST_ExteriorRing은 입력 매개변수로 다각형을 가져오며 해당 외부 링을 곡선으로 리턴합니다. 결과 곡선은 제공된 다각형의 공간 참조 시스템에 표시됩니다.

제공된 다각형이 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다. 다각형에 내부 링이 없으면, 리턴된 외부 링은 다각형 바운더리와 동일합니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

polygon
외부 링이 리턴될 다각형을 표시하는 유형 ST_Polygon의 값

리턴 유형

db2gse.ST_Curve

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 두 개의 다각형(하나는 두 개의 내부 링을 갖고 있고 다른 하나는 내부 링이 없음)을 작성한 다음 외부 링을 판별합니다.

```
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
```

결과:

```
ID          EXTERIOR_RING
----------- ------------------------------------------------------------
          1 LINESTRING ( 40.00000000 120.00000000, 90.00000000
   120.00000000,  90.00000000 150.00000000, 40.00000000 150.00000000,
   40.00000000 120.00000000)

          2 LINESTRING ( 10.00000000 10.00000000, 50.00000000
   10.00000000, 10.00000000 30.00000000, 10.00000000 10.00000000)
```

ST_FindMeasure 또는 ST_LocateAlong

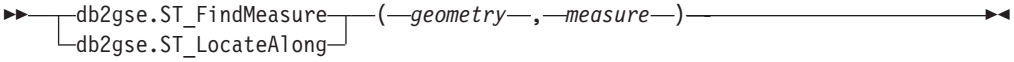
ST_FindMeasure 또는 ST_LocateAlong은 입력 매개변수로 Geometry 및 측정 단위를 가져오며 지정된 측정 단위를 포함하는 제공된 Geometry에 지정된 것과 동일한 측정 단위를 갖는 제공된 Geometry의 파트인 다중점 또는 다중 곡선을 리턴합니다.

점 및 다중점의 경우, 지정된 측정 단위를 갖는 모든 점이 리턴됩니다. 곡선, 다중 곡선, 표면 및 다중 표면의 경우, 결과를 계산하기 위해 보간법(interpolation)이 수행됩니다. Geometry의 바운더리에서 표면 또는 다중 표면에 대한 계산이 수행됩니다.

점 및 다중점의 경우, 제공된 점을 찾을 수 없으면 비어 있는 Geometry가 리턴됩니다. 다른 모든 Geometry의 경우, 제공된 측정 단위가 Geometry의 최저 측정 단위보다 낮거나 Geometry의 최고 측정 단위보다 높으면 비어 있는 측정 단위가 리턴됩니다. 제공된 Geometry가 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

M 좌표(측정 단위)가 *measure*를 포함하는 파트를 검색할 Geometry를 나타내는 ST_Geometry 또는 해당 부속 유형 중 하나의 값

measure

결과에 *geometry*의 파트가 포함되어 있어야 하는 측정 단위인 DOUBLE 유형의 값

리턴 유형

db2gse.ST_Geometry

예

예 1

다음의 CREATE TABLE 명령문은 SAMPLE_GEOMETRIES 테이블을 작성합니다. SAMPLE_GEOMETRIES에는 각 행을 고유하게 식별하는 ID 컬럼과, 샘플 Geometry를 저장하는 GEOMETRY ST_Geometry 컬럼이 있습니다.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sample_geometries(id SMALLINT, geometry ST_GEOMETRY)
```

다음의 INSERT문은 두 행을 삽입합니다. 첫 번째는 선스트링이고 두 번째는 다중점입니다.

```
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))
```

예 2

다음 SELECT문과 해당 결과 세트에서 ST_FindMeasure 함수는 측정 단위가 7인 점을 찾도록 지정됩니다. 첫 번째 행은 점을 리턴합니다. 그러나 두 번째 행에서는 빈 점을 리턴합니다. 선형 기능(0보다 큰 차원의 Geometry)의 경우 ST_FindMeasure가 점을 삽입하지만, 다중점의 경우에는 목표 측정 단위가 정확하게 일치해야 합니다.

```
SELECT id, cast(ST_AsText(ST_FindMeasure(geometry, 7))
   AS varchar(45)) AS measure_7
FROM   sample_geometries
```

결과:

```
ID     MEASURE_7
------ ---------------------------------------------
     1 POINT M ( 3.50000000 3.50000000 7.00000000)
     2 POINT EMPTY
```

예 3

다음 SELECT문과 해당 결과 세트에서 ST_FindMeasure 함수는 점과 다중점을 리턴합니다. 목표 측정 단위 6은 ST_FindMeasure 및 다중점 소스 데이터 모두에서 해당 측정 단위와 일치합니다.

```
SELECT id, cast(ST_AsText(ST_FindMeasure(geometry, 6))
   AS varchar(120)) AS measure_6
FROM   sample_geometries
```

결과:

```
ID     MEASURE_6
------ ------------------------------------------------------------
1      POINT M ( 3.00000000 3.00000000 6.00000000)

2      MULTIPOINT M ( 3.00000000 3.00000000 6.00000000, 4.00000000
        4.00000000 6.00000000, 5.00000000 5.00000000 6.00000000)
```

ST_Generalize

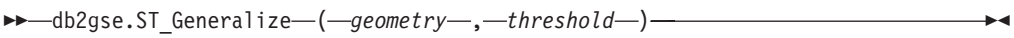
Geometry의 일반 특성을 보존하는 동안 ST_Generalize는 입력 매개변수로 Geometry와 임계값을 가져오며 점 수가 줄어든 제공된 Geometry를 표시합니다.

Douglas-Peucker line-simplification 알고리즘이 사용되며, 이 알고리즘에서는 Geometry를 정의하는 점 시퀀스가 연속된 점이 직선 세그먼트로 대체될 때까지 순환적으로 세분화됩니다. 이 선 세그먼트에서, 정의한 점은 제공된 임계값을 초과하여 직선 세그먼트에서 벗어나지 않습니다. Z 및 M 좌표는 단순화를 위해 고려되지 않습니다. 결과 Geometry는 제공된 Geometry의 공간 참조 시스템에 있습니다.

제공된 Geometry가 비어 있는 경우 ST_Point 유형의 빈 Geometry가 리턴됩니다. 제공된 Geometry 또는 임계값이 널(NULL)인 경우, 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

유형 ST_Geometry 값 또는 line-simplification이 적용되는 Geometry를 표시하는 해당 부속 유형 중 하나의 값

threshold

line-simplification 알고리즘에 사용될 임계값을 식별하는 유형 DOUBLE의 값. 임계값은 0보다 크거나 같아야 합니다. 임계값이 커질수록, 일반화된 Geometry를 표시하는 데 사용되는 점의 수는 더 적어집니다. 측지 데이터의 경우, 임계값 단위는 미터입니다.

리턴 유형

db2gse.ST_Geometry

예

다음 예에서는 읽기 쉽도록 결과를 다시 형식화했습니다. 결과의 스페이스는 사용자의 표시 장치에 따라 다릅니다.

예 1

(10, 10)에서 (80, 80)까지 8개의 점을 가진 선스트링을 작성합니다. 경로는 거의 직선이나 점은 선에서 약간 벗어나 있습니다. ST_Generalize 함수를 사용하여 선 내 점의 수를 줄일 수 있습니다.

```
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))
```

예 2

일반화 인수 3을 사용할 경우, 선스트링이 4개의 좌표로 줄어들고 원래의 선스트링 표현에 아주 근접합니다.

```
SELECT CAST(ST_AsText(ST_Generalize(geometry, 3)) as VARCHAR(115))
  Generalize_3
FROM sample_lines
```

결과:

```
GENERALIZE 3
------------------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 34.00000000 26.00000000,
  59.00000000  63.00000000, 80.00000000 80.00000000)
```

예 3

일반화 인수 6을 사용할 경우, 선스트링은 단 두 개의 좌표로 줄어듭니다. 이는 이전 예제에서보다 단순한 선스트링을 생성하나 원래 표현에서 약간 벗어납니다.

```
SELECT CAST(ST_AsText(ST_Generalize(geometry, 6)) as VARCHAR(65))
  Generalize_6
FROM sample_lines
```

결과:

```
GENERALIZE 6
-----------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 80.00000000 80.00000000)
```

ST_GeomCollection

ST_GeomCollection을 사용하여 Geometry 콜렉션을 생성하십시오.

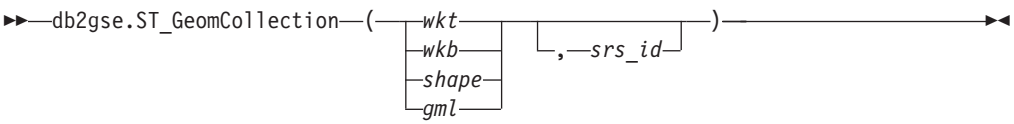
ST_GeomCollection은 다음 입력 중 하나로부터 Geometry 콜렉션을 구성합니다.

- Well-Known Text 표현
- Well-Known Binary 표현
- ESRI 모양 표현
- GML(Geography Markup Language) 표현

선택적 공간 참조 시스템 ID를 지정하여 결과 Geometry 콜렉션이 있는 공간 참조 시스템을 식별할 수 있습니다.

Well-Known Text 표현, Well-Known Binary 표현, ESRI 모양 표현 또는 GML 표현이 널(NULL)일 경우, 널(NULL)이 리턴됩니다.

구문



매개변수

wkt 결과 Geometry 콜렉션에서 Well-Known Text 표현을 포함하는 CLOB(2G) 유형의 값

wkb 결과 Geometry 콜렉션에서 Well-Known Binary 표현을 포함하는 BLOB(2G) 유형의 값

shape 결과 Geometry 콜렉션에서 ESRI 모양 표현을 포함하는 BLOB(2G) 유형의 값

gml GML(Geography Markup Language)을 사용하는 결과 Geometry 콜렉션을 나타내는 CLOB(2G) 유형의 값

srs_id 결과 Geometry 콜렉션에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 내재적으로 사용됩니다.

*srs_id*가 카탈로그 뷰 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS에 나열된 공간 참조 시스템을 식별하지 못하면 오류(SQLSTATE 38SU1)가 리턴됩니다.

리턴 유형

db2gse.ST_GeomCollection

참고

srs_id 매개변수를 생략하면 *wkt* 및 *gml*을 CLOB 데이터 유형으로 캐스트해야 합니다. 그렇지 않으면, DB2는 값을 참조 유형 REF(ST_GeomCollection)에서 ST_GeomCollection 유형으로 캐스트하는 데 사용된 함수를 해결할 수 없습니다. 다음 예는 DB2가 올바른 함수를 해결할 수 있게 합니다.

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

다음 코드는 ST_GeomCollection 함수를 사용하여 Well-Known Text 표현으로부터 다중점, 다중 행, 다중 다각형을 작성하고 GML(Geographic Markup Language)로부터 다중점을 작성하여 GeomCollection 컬럼에 삽입하는 방법을 보여줍니다.

```
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
```

결과:

```
ID          GEOMCOLLECTION
----------- ------------------------------------------------------------
4001         MULTIPOINT ( 1.00000000 2.00000000, 4.00000000 3.00000000,
    5.00000000 6.00000000)

4002         MULTILINESTRING (( 33.00000000 2.00000000, 34.00000000
   3.00000000, 35.00000000 6.00000000),( 28.00000000 4.00000000,
   29.00000000 5.00000000, 31.00000000 8.00000000, 43.00000000
   12.00000000),(39.00000000 3.00000000, 37.00000000 4.00000000,
   36.00000000 7.00000000))

4003         MULTIPOLYGON ((( 13.00000000 33.00000000, 10.00000000
    43.00000000, 1.00000000 40.00000000, 7.00000000 36.00000000,
    13.00000000 33.00000000)),(( 8.00000000 24.00000000, 9.00000000
    25.00000000, 1.00000000 28.00000000, 8.00000000 24.00000000)),
    (( 3.00000000 3.00000000,5.00000000 3.00000000, 4.00000000
    6.00000000,3.00000000 3.00000000)))

4004         MULTIPOINT ( 10.00000000 20.00000000, 30.00000000
    40.00000000)
```

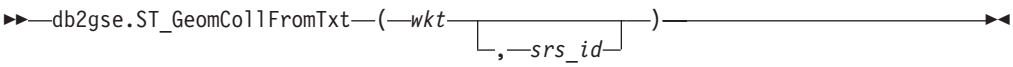
ST_GeomCollFromTxt

ST_GeomCollFromTxt는 Geometry 콜렉션 중에서 Well-Known Text 표현과 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 Geometry 콜렉션을 리턴합니다.

제공된 Well-Known Text 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

동일한 결과를 얻기 위해서는 ST_GeomCollection 함수를 사용하도록 권장합니다. 이 함수는 유연성이 뛰어나기 때문에 권장됩니다. ST_GeomCollection은 Well-Known Binary 표현뿐 아니라 다른 입력 양식도 가져옵니다.

구문



매개변수

wkt 결과 Geometry 콜렉션에서 Well-Known Text 표현을 포함하는 CLOB(2G) 유형의 값

srs_id 결과 Geometry 콜렉션에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 내재적으로 사용됩니다.

*srs_id*가 카탈로그 뷰 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS에 나열된 공간 참조 시스템을 식별하지 못하면 오류(SQLSTATE 38SU1)가 리턴됩니다.

리턴 유형

db2gse.ST_GeomCollection

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

다음 코드는 ST_GeomCollFromTxt 함수를 사용하여 Well-Known Text 표현으로부터 다중점, 다중 행, 다중 다각형을 작성하여 GeomCollection 컬럼에 삽입하는 방법을 보여줍니다.

```
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
```

결과:

```
ID          GEOMCOLLECTION
----------- ------------------------------------------------------------------
4011        MULTIPOINT ( 1.00000000 2.00000000, 4.00000000 3.00000000,
   5.00000000 6.00000000)

4012        MULTILINESTRING (( 33.00000000 2.00000000, 34.00000000
   3.00000000, 35.00000000 6.00000000),( 28.00000000 4.00000000, 29.00000000
   5.00000000, 31.00000000 8.00000000, 43.00000000 12.00000000),( 39.00000000
   3.00000000, 37.00000000 4.00000000, 36.00000000 7.00000000))

4013        MULTIPOLYGON ((( 13.00000000 33.00000000, 10.00000000 43.00000000,
   1.00000000 40.00000000, 7.00000000 36.00000000, 13.00000000 33.00000000)),
   (( 8.00000000 24.00000000, 9.00000000 25.00000000, 1.00000000 28.00000000,
   8.00000000 24.00000000)),(( 3.00000000 3.00000000, 5.00000000 3.00000000,
   4.00000000 6.00000000, 3.00000000 3.00000000)))
```

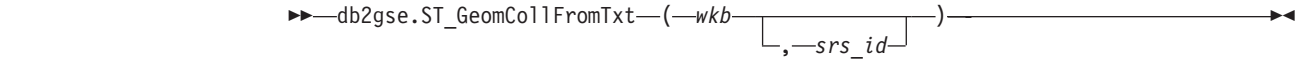
ST_GeomCollFromWKB

ST_GeomCollFromWKB는 Geometry 콜렉션의 Well-Known Binary 표현 및 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 Geometry 콜렉션을 리턴합니다.

제공된 Well-Known Binary 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

이 기능에 우선하는 버전은 ST_GeomCollection입니다.

구문



매개변수

wkb 결과 Geometry 콜렉션에서 Well-Known Binary 표현을 포함하는 BLOB(2G) 유형의 값

srs_id 결과 Geometry 콜렉션에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 내재적으로 사용됩니다.

*srs_id*가 카탈로그 뷰 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS에 나열된 공간 참조 시스템을 식별하지 못하면 오류(SQLSTATE 38SU1)가 리턴됩니다.

리턴 유형

db2gse.ST_GeomCollection

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

다음 코드는 ST_GeomCollFromWKB 함수를 사용하여 Well-Known Binary 표현으로 Geometry 콜렉션의 좌표를 작성 및 쿼리하는 방법을 보여줍니다. ID 4021 및 4022와 공간 참조 시스템 1의 Geometry 콜렉션을 가진 행을 SAMPLE_GEOMCOLLECTION 테이블에 삽입합니다.

```
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
```

결과:

```
ID          GEOMCOLLECTION
----------- ----------------------------------------------------------
       4021 MULTIPOINT ( 1.00000000 2.00000000, 4.00000000
  3.00000000, 5.00000000 6.00000000)

       4022 MULTILINESTRING (( 33.00000000 2.00000000,
 34.00000000 3.00000000, 35.00000000 6.00000000),( 28.00000000
 4.00000000, 29.00000000 5.00000000, 31.00000000 8.00000000,
 43.00000000 12.00000000))
```

ST_Geometry

ST_Geometry는 다음 입력 중 하나로부터 Geometry를 구성합니다.

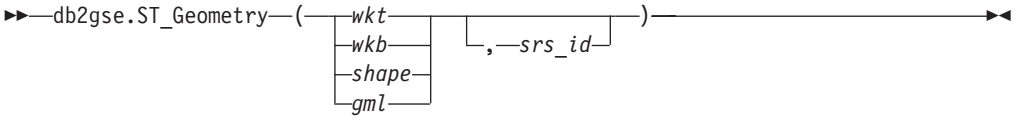
- Well-Known Text 표현
- Well-Known Binary 표현
- ESRI 모양 표현
- GML(Geography Markup Language) 표현

선택적 공간 참조 시스템 ID를 지정하여 결과 Geometry가 있는 공간 참조 시스템을 식별할 수 있습니다.

결과 Geometry의 동적 유형은 ST_Geometry의 인스턴스화할 수 있는 부속 유형 중 하나입니다.

Well-Known Text 표현, Well-Known Binary 표현, ESRI 모양 표현 또는 GML 표현이 널(NULL)일 경우, 널(NULL)이 리턴됩니다.

구문



매개변수

wkt 결과 Geometry의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

wkb 결과 Geometry의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

shape 결과 Geometry의 ESRI 모양 표현을 나타내는 BLOB(2G) 유형의 값

gml GML(Geography Markup Language)을 사용하는 결과 Geometry를 나타내는 CLOB(2G) 유형의 값

srs_id 결과 Geometry에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 내재적으로 사용됩니다.

*srs_id*가 카탈로그 뷰 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS에 나열된 공간 참조 시스템을 식별하지 못하면 오류(SQLSTATE 38SU1)가 리턴됩니다.

리턴 유형

db2gse.ST_Geometry

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

다음 코드는 ST_Geometry 함수를 사용하여 Well-Known Text 점 표현으로부터 점을, GML(Geographic Markup Language) 행 표현으로부터 행을 작성하여 삽입하는 방법을 보여줍니다.

ST_Geometry 함수는 여러 가지 Geometry 표현으로부터 모든 공간 유형을 작성할 수 있기 때문에 공간 유형 컨스트럭터 함수 중에서 가장 유연성이 뛰어납니다. ST_LineFromText는 WKT 행 표현을 통해서만 행을 작성할 수 있습니다. ST_WKTToSql은 모든 유형을 작성할 수 있으나 WKT 표현을 통해서만 가능합니다.

```
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
```

결과:

```
ID          GEOMETRY
----------- ------------------------------------------------------------
7001        POINT ( 1.00000000 2.00000000)

7002        LINESTRING ( 33.00000000 2.00000000, 34.00000000 3.00000000,
  35.00000000 6.00000000)

7003        POLYGON (( 3.00000000 3.00000000, 5.00000000 3.00000000,
  4.00000000 6.00000000, 3.00000000 3.00000000))

7004        POINT ( 50.00000000 60.00000000)
```

ST_GeometryN

ST_GeometryN은 입력 매개변수로 Geometry 콜렉션과 인덱스를 가져오며 인덱스로 식별되는 콜렉션의 Geometry를 리턴합니다. 결과 Geometry는 제공된 Geometry 콜렉션의 공간 참조 시스템에 표시됩니다.

제공된 Geometry 콜렉션이 널(NULL) 값 또는 비어 있거나 인덱스가 1보다 작거나 콜렉션의 Geometry 수보다 크면, 널(NULL) 값이 리턴되고 경고 조건(O1HSO)이 발생합니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

collection

*n*번째 Geometry를 찾을 Geometry 콜렉션을 나타내는 ST_GeomCollection 유형 또는 해당 부속 유형 중 하나의 값

index *collection*에서 리턴될 *n*번째 Geometry를 식별하는 유형 INTEGER의 값

*index*가 1보다 작거나 콜렉션의 Geometry 수보다 크면, 널(NULL) 값이 리턴되거나 경고가 리턴됩니다(SQLSTATE 01HS0).

리턴 유형

db2gse.ST_Geometry

예

다음 코드는 Geometry 콜렉션 내에서 두 번째 Geometry를 선택하는 방법을 보여줍니다.

```
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
```

결과:

```
ID          SECOND_GEOMETRY
----------- ------------------------------------------------------------
       4001 POINT ( 4.00000000 3.00000000)

       4002 LINESTRING ( 28.00000000 4.00000000, 29.00000000 5.00000000,
   31.00000000 8.00000000, 43.00000000 12.00000000)

       4003 POLYGON (( 8.00000000 24.00000000, 9.00000000 25.00000000,
   1.00000000 28.00000000, 8.00000000 24.00000000))
```

ST_GeometryType

ST_GeometryType은 입력 매개변수로 Geometry를 가져오며 해당 Geometry 동적 유형의 완전한 유형 이름을 리턴합니다.

DB2 함수 TYPE_SCHEMA 및 TYPE_NAME의 효과는 동일합니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

Geometry 유형이 리턴된 유형 ST_Geometry의 값

리턴 유형

VARCHAR(128)

예

다음 코드는 Geometry의 유형을 판별하는 방법을 보여줍니다.

```
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
```

결과:

```
ID          GEOMETRY_TYPE
----------- ------------------------------
       7101 "DB2GSE  "."ST_POINT"
       7102 "DB2GSE  "."ST_LINESTRING"
       7103 "DB2GSE  "."ST_POLYGON"
       7104 "DB2GSE  "."ST_MULTIPOINT"
```

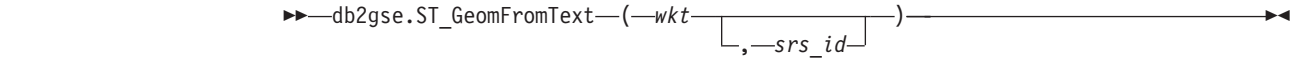
ST_GeomFromText

ST_GeomFromText는 Geometry의 Well-Known Text 표현과 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 Geometry를 리턴합니다.

제공된 Well-Known Text 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

이 기능에 우선하는 버전은 ST_Geometry입니다.

구문



매개변수

wkt 결과 Geometry의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

srs_id 결과 Geometry에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 내재적으로 사용됩니다.

*srs_id*가 카탈로그 뷰 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS에 나열된 공간 참조 시스템을 식별하지 못하면 오류(SQLSTATE 38SU1)가 리턴됩니다.

리턴 유형

db2gse.ST_Geometry

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예에서는 ST_GeomFromText 함수를 사용하여 Well Known Text 점 표현으로부터 점을 작성하여 삽입합니다.

다음 코드는 WKT 표현을 사용하여 ID 및 공간 참조 시스템 1의 Geometry를 가진 행을 SAMPLE_POINTS 테이블에 삽입합니다.

```
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))
```

다음 SELECT문은 SAMPLE_GEOMETRIES 테이블로부터 ID 및 GEOMETRIES를 리턴합니다.

```
SELECT id, cast(geometry..ST_AsText AS varchar(105))
   AS geometry
FROM   sample_geometries
```

결과:

```
ID          GEOMETRY
----------- ----------------------------------------------------------------
       1251 POINT ( 1.00000000 2.00000000)

       1252 LINESTRING ( 33.00000000 2.00000000, 34.00000000 3.00000000,
              35.00000000 6.00000000)

       1253 POLYGON (( 3.00000000 3.00000000, 5.00000000 3.00000000,
              4.00000000 6.00000000, 3.00000000 3.00000000))
```

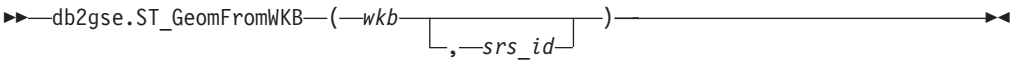
ST_GeomFromWKB

ST_GeomFromWKB는 Geometry의 Well-Known Binary 표현과 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 Geometry를 리턴합니다.

제공된 Well-Known Binary 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

이 기능에 우선하는 버전은 ST_Geometry입니다.

구문



매개변수

wkb 결과 Geometry의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

srs_id 결과 Geometry에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 내재적으로 사용됩니다.

지정된 srs_id 매개변수가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 오류(SQLSTATE 38SU1)가 리턴됩니다.

리턴 유형

db2gse.ST_Geometry

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

다음 코드는 ST_GeomFromWKB 함수를 사용하여 Well-Known Binary 행 표현으로부터 행을 작성하여 삽입하는 방법을 보여줍니다.

다음 예는 WKB 표현을 사용하여 ID 및 공간 참조 시스템 1의 Geometry를 가진 레코드를 SAMPLE_GEOMETRIES 테이블에 삽입합니다.

```
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
```

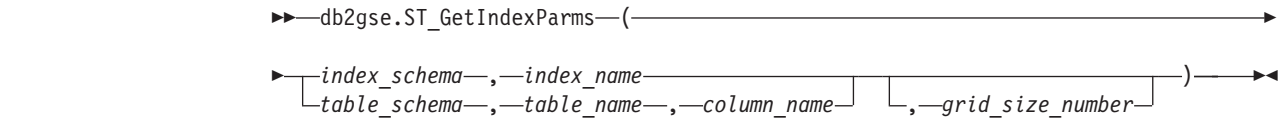
결과:

```
 ID          GEOMETRY
 ----------- ------------------------------------------------------------
1901         POINT ( 1.00000000 2.00000000)

1902         LINESTRING ( 33.00000000 2.00000000, 34.00000000
   3.00000000, 35.00000000 6.00000000)

1903         POLYGON (( 3.00000000 3.00000000, 5.00000000 3.00000000,
   4.00000000 6.00000000, 3.00000000 3.00000000))
```

ST_GetIndexParms

ST_GetIndexParms는 공간 인덱스 또는 공간 컬럼에 대한 ID를 입력 매개변수로 가져오며 인덱스 또는 공간 컬럼에 대한 인덱스를 정의하는 데 사용되는 매개변수를 리턴합니다. 추가 매개변수 번호를 지정할 경우 숫자로 식별되는 격자 *크기*만이 리턴됩니다.

구문



매개변수

index_schema

규정되지 않은 *index_name* 이름을 가진 공간 인덱스가 있는 스키마를 식별하는 VARCHAR(128) 유형의 값. 스키마 이름은 대소문자를 구분하며 SYSCAT.SCHEMATA 카탈로그 뷰에 나열되어 있어야 합니다.

이 매개변수가 널(NULL)이면 CURRENT SCHEMA 특수 레지스터의 값이 공간 인덱스의 스키마 이름으로 사용됩니다.

index_name

인덱스 매개변수를 리턴하는 공간 인덱스의 규정되지 않은 이름을 포함하는 VARCHAR(128) 유형의 값. 인덱스 이름은 대소문자를 구분하며 *index_schema* 스키마에 대한 SYSCAT.INDEXES 카탈로그 뷰에 나열되어 있어야 합니다.

table_schema

규정되지 않은 *table_name* 이름을 가진 테이블이 있는 스키마를 식별하는 VARCHAR(128) 유형의 값. 스키마 이름은 대소문자를 구분하며 SYSCAT.SCHEMATA 카탈로그 뷰에 나열되어 있어야 합니다.

이 매개변수가 널(NULL)이면 CURRENT SCHEMA 특수 레지스터의 값이 공간 인덱스에 대한 스키마 이름으로 사용됩니다.

table_name

column_name 공간 컬럼이 있는 테이블의 규정되지 않은 이름을 포함하는 VARCHAR(128) 유형의 값. 테이블 이름은 대소문자를 구분하며 *table_schema* 스키마에 대한 SYSCAT.TABLES 카탈로그 뷰에 나열되어 있어야 합니다.

column_name

해당 컬럼에 대한 공간 인덱스의 인덱스 매개변수가 리턴되는 *table_schema.table_name* 테이블의 컬럼을 식별하는 VARCHAR(128) 유형의 값. 컬럼 이름은 대소문자를 구분하며 *table_schema.table_name* 테이블에 대한 SYSCAT.COLUMNS 카탈로그 뷰에 나열되어 있습니다.

컬럼에 정의된 공간 인덱스가 없으면 오류(SQLSTATE 38SQ0)가 발생합니다.

grid_size_number

값을 리턴하는 매개변수를 식별하는 DOUBLE 값

이 값이 1보다 작거나 3보다 크면 오류(SQLSTATE 38SQ1)가 발생합니다.

리턴 유형

DOUBLE(*grid_size_number*를 지정할 경우)

*grid_size_number*를 지정하지 않으면 ORDINAL 및 VALUE의 두 컬럼이 있는 테이블이 리턴됩니다. ORDINAL 컬럼의 유형은 INTEGER이며 VALUE 컬럼의 유형은 DOUBLE입니다.

격자 인덱스에 대한 매개변수가 리턴될 경우, 첫 번째, 두 번째 및 세 번째 *크기*의 값으로 ORDINAL 컬럼에 값 1, 2 및 3이 포함됩니다. VALUE 컬럼에는 격자 *크기*가 포함되어 있습니다.

VALUE 컬럼에는 각 매개변수에 해당하는 값이 포함되어 있습니다.

예

예 1

다음 코드는 공간 컬럼 및 공간 인덱스가 있는 테이블을 작성합니다.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse

CREATE TABLE sch.offices (name VARCHAR(30), location ST_Point )

 CREATE INDEX sch.idx ON sch.offices(location)
      EXTEND USING db2gse.spatial_index(1e0, 10e0, 1000e0)
```

ST_GetIndexParms 함수를 사용하여 공간 인덱스를 작성할 때 사용한 매개변수의 값을 검색할 수 있습니다.

예 2

이 예는 리턴할 매개변수(번호로 식별됨)를 명시적으로 지정하여 공간 격자 인덱스에 대한 세 개의 격자 *크기*를 별도로 검색하는 방법을 보여줍니다.

```
VALUES ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION', 1)
```

결과:

```
1
------------------------
  +1.00000000000000E+000

VALUES ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION', 2)
```

결과:

```
1
------------------------
  +1.00000000000000E+001
```

```
VALUES ST_GetIndexParms('SCH', 'IDX', 3)
```

결과:

```
1
------------------------
  +1.00000000000000E+003
```

예 3

이 예는 공간 격자 인덱스의 모든 매개변수를 검색하는 방법을 보여줍니다. ST_GetIndexParms 함수는 매개변수 번호 및 해당 격자 크기를 표시하는 테이블을 리턴합니다.

```
SELECT * FROM TABLE ( ST_GetIndexParms('SCH', 'OFFICES', 'LOCATION') ) AS t
```

결과:

```
ORDINAL     VALUE
----------- ------------------------
          1   +1.00000000000000E+000
          2   +1.00000000000000E+001
          3   +1.00000000000000E+003
```

```
SELECT * FROM TABLE ( ST_GetIndexParms('SCH', 'IDX') ) AS t
```

결과:

```
ORDINAL     VALUE
----------- ------------------------
          1   +1.00000000000000E+000
          2   +1.00000000000000E+001
          3   +1.00000000000000E+003
```

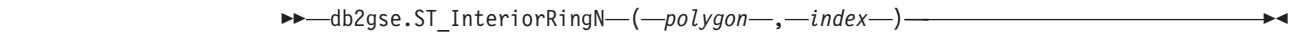
ST_InteriorRingN

ST_InteriorRingN은 입력 매개변수로 다각형과 인덱스를 가져오며 제공된 인덱스로 식별되는 내부 링을 선스트링으로 리턴합니다. 내부 링은 내부 Geometry 검증 루틴에서 정의한 규칙에 따라 구성됩니다.

제공된 다각형이 널(NULL) 또는 비어 있거나 내부 링이 없으면 널(NULL)이 리턴됩니다. 인덱스가 1보다 작거나 다각형의 내부 링 수보다 크면 널(NULL)이 리턴되고 경고 조건(1HS1)이 발생합니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

polygon

*index*에 의해 식별되는 내부 링이 리턴되는 Geometry를 표시하는 유형 ST_Polygon의 값

index 리턴된 *n*번째 내부 링을 식별하는 유형 INTEGER의 값. *index*로 식별되는 내부 링이 없으면 경고 조건(01HS1)이 발생합니다.

리턴 유형

db2gse.ST_Curve

예

이 예에서는 두 개의 내부 링을 가진 다각형이 작성됩니다. 두 번째 내부 링을 검색하는 데는 ST_InteriorRingN 호출이 사용됩니다.

```
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
```

결과:

```
ID          INTERIOR_RING
----------- ----------------------------------------------------------------
      1 LINESTRING ( 70.00000000 130.00000000, 70.00000000 140.00000000,
80.00000000 140.00000000, 80.00000000 130.00000000, 70.00000000 130.00000000)
```

ST_Intersection

ST_Intersection은 입력 매개변수로 두 개의 Geometry를 가져오며 두 개의 제공된 Geometry의 교차인 Geometry를 리턴합니다. 교차는 첫 번째 Geometry와 두 번째 Geometry의 공통 파트입니다. 결과 Geometry는 첫 번째 Geometry의 공간 참조 시스템에 표시됩니다.

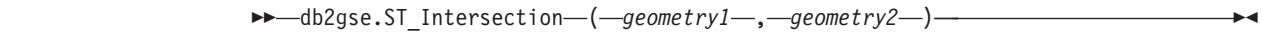
가능한 경우, 리턴된 Geometry의 특정 유형은 ST_Point, ST_LineString 또는 ST_Polygon입니다. 예를 들어, 점과 다각형의 교차는 비어 있거나 ST_MultiPoint로 표시되는 단일 점입니다.

두 Geometry 중 하나가 널(NULL)인 경우, 널(NULL)이 리턴됩니다.

비측지 데이터의 경우 두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템으로 표현되지 않으면 다른 공간 참조 시스템으로 변환됩니다. 측지 데이터의 경우, 두 Geometry는 모두 동일한 측지 공간 참조 시스템(SRS) 내에 있어야 합니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry1

ST_Geometry의 값 또는 *geometry2*와의 교차를 계산하기 위해 첫 번째 Geometry를 표시하는 해당 부속 유형 중 하나의 값

geometry2

ST_Geometry의 값 또는 *geometry1*과의 교차를 계산하기 위해 두 번째 Geometry를 표시하는 해당 부속 유형 중 하나의 값

측지 데이터의 경우, 두 Geometry는 모두 측지 Geometry로서 동일한 측지 SRS 내에 있어야 합니다.

리턴 유형

db2gse.ST_Geometry

리턴된 Geometry의 차원은 차원이 낮은 입력의 차원입니다(측지 데이터의 선스트링 제외). 측지 데이터의 경우, 두 선스트링의 교차 차원은 0입니다. 즉, 교차는 점 또는 다중점입니다.

예

다음 예에서는 읽기 쉽도록 결과를 다시 형식화했습니다. 결과의 스페이스는 사용자의 표시 장치에 따라 다릅니다.

이 예에서는 여러 개의 다른 Geometry를 작성한 후 첫 번째 Geometry와의 교차 여부를 판별합니다(있을 경우).

```
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
```

결과:

```
ID          ID          INTERSECTION
----------- ----------- ----------------------------------------------
          1           1 POLYGON (( 30.00000000 30.00000000, 50.00000000
30.00000000, 50.00000000 50.00000000, 30.00000000 50.00000000, 30.00000000
30.00000000))

          1           2 LINESTRING ( 30.00000000 40.00000000, 30.00000000
30.00000000)

          1           3 POLYGON (( 40.00000000 40.00000000, 50.00000000
40.00000000, 50.00000000 50.00000000, 40.00000000 50.00000000, 40.00000000
40.00000000))

          1           4 POINT EMPTY

          1           5 LINESTRING ( 30.00000000 30.00000000, 50.00000000
50.00000000)

  5 record(s) selected.
```

ST_Intersects

ST_Intersects는 입력 매개변수로 두 개의 Geometry를 가져오며 제공된 Geometry가 교차하는 경우 1을 리턴합니다. Geometry가 교차하지 않으면 0이 리턴됩니다.

두 Geometry 중 하나가 널(NULL)이거나 비어 있는 경우, 널(NULL)이 리턴됩니다.

비측지 데이터의 경우 두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템으로 표현되지 않으면 다른 공간 참조 시스템으로 변환됩니다. 측지 데이터의 경우, 두 Geometry는 모두 동일한 측지 공간 참조 시스템(SRS) 내에 있어야 합니다.

구문

매개변수

geometry1

*geometry2*와의 교차를 테스트하기 위한 Geometry를 나타내는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

geometry2

*geometry1*과의 교차를 테스트하기 위한 Geometry를 나타내는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

제한사항: 측지 데이터의 경우, 두 Geometry는 모두 측지 Geometry로서 동일한 측지 SRS 내에 있어야 합니다.

리턴 유형

INTEGER

예

다음 명령문은 SAMPLE_GEOMETRIES1 및 SAMPLE_GEOMETRIES2 테이블을 작성하고 채웁니다.

```
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) )
```

다음 SELECT문은 SAMPLE_GEOMTRIES1 및 SAMPLE_GEOMTRIES2 테이블의 여러 Geometry가 교차하는지를 판별합니다.

```
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
```

결과:

```
SG1_ID SG1_TYPE      SG2_ID SG2_TYPE      INTERSECTS
------ ------------- ------ ------------- --------------------------
     1 ST_Point         101 ST_Point      Geometries intersect
     1 ST_Point         102 ST_Point      Geometries do not intersect
     1 ST_Point         103 ST_Point      Geometries do not intersect
     1 ST_Point         110 ST_LineString Geometries do not intersect
     1 ST_Point         120 ST_Polygon    Geometries do not intersect
     1 ST_Point         121 ST_Polygon    Geometries do not intersect
    10 ST_LineString    101 ST_Point      Geometries do not intersect
    10 ST_LineString    102 ST_Point      Geometries do not intersect
    10 ST_LineString    103 ST_Point      Geometries intersect
    10 ST_LineString    110 ST_LineString Geometries intersect
    10 ST_LineString    120 ST_Polygon    Geometries do not intersect
    10 ST_LineString    121 ST_Polygon    Geometries do not intersect
    20 ST_Polygon       101 ST_Point      Geometries intersect
    20 ST_Polygon       102 ST_Point      Geometries intersect
    20 ST_Polygon       103 ST_Point      Geometries do not intersect
    20 ST_Polygon       110 ST_LineString Geometries do not intersect
    20 ST_Polygon       120 ST_Polygon    Geometries intersect
    20 ST_Polygon       121 ST_Polygon    Geometries do not intersect
```

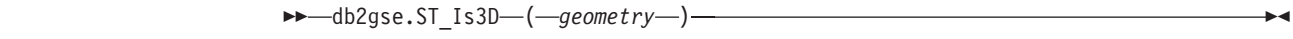
ST_Is3d

ST_Is3d는 입력 매개변수로 Geometry를 가져오며 제공된 Geometry에 Z 좌표가 있는 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다.

제공된 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

유형 ST_Geometry의 값 또는 Z 좌표의 존재에 대해 테스트할 Geometry를 표시하는 해당 부속 유형 중 하나의 값

리턴 유형

INTEGER

예

이 예에서는 여러 개의 Geometry(Z 좌표 및 M 좌표(측정 단위)가 있는 것과 없는 것)를 작성합니다. 그런 다음 ST_Is3d를 사용하여 Z 좌표를 포함하는 Geometry를 판별합니다.

```
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
```

결과:

```
ID          IS_3D
----------- -----------
          1           0
          2           0
          3           0
          4           1
          5           1
```

ST_IsClosed

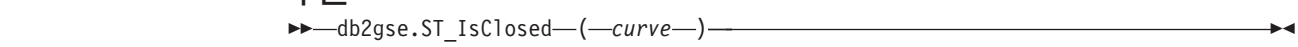
ST_IsClosed는 입력 매개변수로 곡선 또는 다중 곡선을 가져오며 제공된 곡선이나 다중 곡선이 닫혀 있는 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다.

시작점과 끝점이 같은 경우 곡선은 닫혀 있습니다. 곡선에 Z 좌표가 있으면, 시작점 및 끝점의 Z 좌표는 동일해야 합니다. 그렇지 않으면, 점은 동일한 것으로 간주되지 않으며 곡선은 닫혀 있지 않습니다. 해당 곡선 각각이 닫혀 있으면 다중 곡선은 닫혀 있습니다.

제공된 곡선 또는 다중 곡선이 비어 있으면, 0(영)이 리턴됩니다. 곡선이 널(NULL)이면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

curve 유형 ST_Curve 또는 ST_MultiCurve의 값 또는 테스트할 곡선 또는 다중 곡선을 표시하는 해당 부속 유형 중 하나의 값

리턴 유형

INTEGER

예

예 1

이 예는 여러 개의 선스트링을 작성합니다. 마지막 두 개의 선스트링은 동일한 X 및 Y 좌표를 갖고 있으나 한 선스트링은 선스트링이 닫히지 않도록 하는 변하는 Z 좌표를 포함하고 다른 선스트링은 선스트링의 닫힘 여부에 영향을 미치지 않는 변하는 M 좌표를 포함합니다.

```
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
```

결과:

```
ID          IS_CLOSED
----------- -----------
          1           0
          2           0
          3           1
          4           1
          5           0
```

예 2

이 예에서 두 개의 다중 선스트링이 작성됩니다. ST_IsClosed를 사용하여 다중 선스트링의 닫힘 여부를 판별합니다. 모든 곡선이 함께 완전히 닫힌 루프를 구성한다 할지라도 첫 번째 선스트링은 닫혀 있지 않습니다. 이는 각 곡선 자체가 닫혀 있지 않기 때문입니다.

두 번째 다중 선스트링은 각 곡선 자체가 닫혀 있기 때문에 닫혀 있습니다.

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_mlines (id INTEGER, geometry ST_MultiLinestring)
INSERT INTO sample_mlines VALUES
       (6, ST_MultiLinestring('multilinestring((10 10, 20 10, 20 20),
                                               (20 20, 30 20, 30 30),
                                               (30 30, 10 30, 10 10))',0))

INSERT INTO sample_mlines VALUES
       (7, ST_MultiLinestring('multilinestring((10 10, 20 10, 20 20, 10 10 ),
                                                (30 30, 50 30, 50 50,
                                                 30 30 ))',0))

SELECT id, ST_IsClosed(geometry)  Is_Closed
FROM sample_mlines
```

결과:

```
ID          IS_CLOSED
----------- -----------
          6           0
          7           1
```

ST_IsEmpty

ST_IsEmpty는 입력 매개변수로 Geometry를 가져오며 제공된 Geometry가 비어 있는 경우 1을 리턴합니다. 그렇지 않은 경우에는 0(영)이 리턴됩니다.

제공된 Geometry가 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 값 또는 테스트할 Geometry를 표시하는 해당 부속 유형 중 하나의 값

리턴 유형

INTEGER

예

다음 코드는 세 개의 Geometry를 작성한 다음 비어 있는지를 판별합니다.

```
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
```

결과:

```
ID          IS_EMPTY
----------- -----------
          1           1
          2           0
          3           0
          4           0
          5           0
```

ST_IsMeasured

ST_IsMeasured는 입력 매개변수로 Geometry를 가져오며 제공된 Geometry에 M 좌표(측정 단위)가 있는 경우 1을 리턴합니다. 그렇지 않은 경우에는 0(영)이 리턴됩니다.

제공된 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry의 값 또는 M 좌표(측정 단위)의 존재에 대해 테스트할 Geometry를 표시하는 해당 부속 유형 중 하나의 값

리턴 유형

INTEGER

예

이 예에서는 여러 개의 Geometry(Z 좌표 및 M 좌표(측정 단위)가 있는 것과 없는 것)를 작성합니다. 그런 다음 ST_IsMeasured를 사용하여 측정 단위가 포함된 Geometry를 판별합니다.

```
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
```

결과:

```
ID          IS_MEASURED
----------- -----------
          1           0
          2           0
          3           1
          4           0
          5           1
```

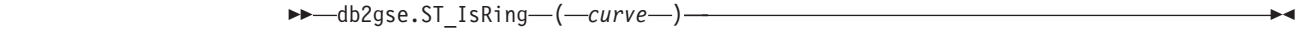
ST_IsRing

ST_IsRing은 입력 매개변수로 곡선을 가져오며 링인 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다. 곡선이 단순하고 닫혀 있는 경우 곡선은 링입니다.

제공된 곡선이 비어 있으면, 0이 리턴됩니다. 곡선이 널(NULL)이면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

curve 유형 ST_Curve의 값 또는 테스트할 곡선을 표시하는 해당 부속 유형 중 하나의 값

리턴 유형

INTEGER

예

이 예에서는 네 개의 선스트링이 작성됩니다. ST_IsRing은 선스트링이 링인지를 점검합니다. 경로가 선스트링 자체를 교차하기 때문에 마지막 선스트링은 닫혀 있더라도 링으로 간주되지 않습니다.

```
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
```

결과:

```
ID          IS_CLOSED   IS_RING
----------- ----------- -----------
          1           1           0
          2           0           0
          3           1           1
          4           1           0
```

ST_IsSimple

ST_IsSimple은 입력 매개변수로 Geometry를 가져오며 제공된 Geometry가 단순한 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다.

점, 표면 및 다중 표면은 항상 단순합니다. 곡선은 동일한 점을 두 번 통과하지 않는 경우 단순합니다. 다중점은 두 개의 동일한 점을 포함하지 않는 경우 단순합니다. 모든 곡선이 단순하고 다중 곡선의 곡선 바운더리에 있는 점에서 유일한 교차가 발생하는 경우 다중 곡선은 단순합니다.

제공된 Geometry가 비어 있으면, 1이 리턴됩니다. Geometry가 널(NULL)이면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry의 값 또는 테스트할 Geometry를 표시하는 해당 부속 유형 중 하나의 값

리턴 유형

INTEGER

예

이 예에서는 여러 개의 Geometry를 작성하여 단순 여부를 점검합니다. ID가 4인 Geometry는 동일한 하나 이상의 점을 포함하기 때문에 단순 Geometry로 간주되지 않습니다. ID가 6인 Geometry는 선스트링이 자신과 교차하기 때문에 단순 Geometry로 간주되지 않습니다.

```
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
```

결과:

```
ID          IS_SIMPLE
----------- -----------
          1           1
          2           1
          3           1
          4           0
          5           1
          6           0
          7           1
```

ST_IsValid

ST_IsValid는 Geometry를 입력 매개변수로 가져오며 유효한 경우 1을 리턴합니다. 그렇지 않은 경우에는 0(영)이 리턴됩니다.

Geometry는 구조화된 유형의 모든 속성이 Geometry 데이터의 내부 표현과 일치할 경우와 내부 표현이 손상되지 않은 경우에만 유효합니다.

제공된 Geometry가 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 또는 해당 부속 유형 중 하나의 값

리턴 유형

INTEGER

예

이 예는 여러 개의 Geometry를 작성한 다음 ST_IsValid를 사용하여 Geometry가 유효한지를 판별합니다. ST_Geometry와 같은 컨스트럭터는 유효하지 않은 Geometry의 구성을 허용하지 않기 때문에 모든 Geometry가 유효합니다.

```
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
```

결과:

```
ID          IS_VALID
----------- -----------
          1           1
          2           1
          3           1
          4           1
          5           1
```

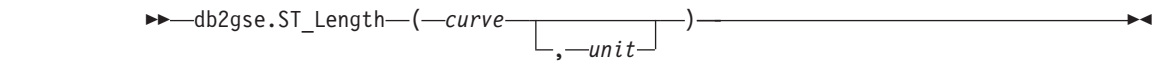
ST_Length

ST_Length는 곡선 또는 다중 곡선과 단위(선택적)를 입력 매개변수로 사용하고 디폴트값 또는 지정된 측정 단위로 지정된 곡선 또는 다중 곡선의 길이를 리턴합니다.

제공된 곡선 또는 다중 곡선이 널(NULL)이거나 비어 있는 경우, 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

curve 길이를 리턴하는 곡선을 표시하는 ST_Curve 또는 ST_MultiCurve 유형의 값

unit 곡선의 길이를 측정하는 단위를 식별하는 VARCHAR(128) 값. 지원되는 측정 단위는 DB2GSE.ST_UNITS_OF_MEASURE 카탈로그 뷰에 나열되어 있습니다.

unit 매개변수를 생략하면 다음 규칙을 사용하여 길이 측정 단위를 판별합니다.

- *curve*가 투영 또는 지구 중심 좌표 시스템에 있는 경우, 이 좌표 시스템과 연관된 선형 단위가 디폴트값입니다.
- *curve*가 지리 좌표 시스템에 있지만 측지 공간 참조 시스템(SRS)에 없는 경우, 이 좌표 시스템과 연관된 각도 단위가 디폴트입니다.
- *curve*가 측지 SRS에 있는 경우, 디폴트 측정 단위는 미터입니다.

단위 변환에 대한 제한사항: 다음과 같은 조건이 발생하는 경우 오류(SQLSTATE 38SU4)가 리턴됩니다.

- *curve*가 지정되지 않은 좌표 시스템에 있고 *unit* 매개변수가 지정됩니다.
- *curve*가 투영 좌표 시스템에 있고 각도 단위가 지정됩니다.
- *curve*가 지리 좌표 시스템에 있지만 측지 SRS에는 없고 선형 단위가 지정되었습니다.
- *curve*가 측지 SRS에 있고 각도 단위가 지정되었습니다.

리턴 유형

DOUBLE

예

예 1

다음 SQL문은 SAMPLE_GEOMETRIES 테이블을 작성하여 이 테이블에 행 및 다중 행을 삽입합니다.

```
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) )
```

예 2

다음 SELECT문은 SAMPLE_GEOMTRIES 테이블에 있는 행의 길이를 계산합니다.

```
SELECT id, spatial_type, cast(ST_Length(geometry..ST_ToLineString)
   AS DECIMAL(7, 2)) AS "Line Length"
FROM   sample_geometries
WHERE  id = 1110
```

결과:

```
ID     SPATIAL_TYPE         Line Length
------ -------------------- -----------
  1110 ST_LineString              10.00
```

예 3

다음 SELECT문은 SAMPLE_GEOMTRIES 테이블에 있는 다중 행의 길이를 계산합니다.

```
SELECT id, spatial_type, ST_Length(ST_ToMultiLine(geometry))
   AS multiline_length
FROM   sample_geometries
WHERE  id = 1111
```

결과:

```
ID     SPATIAL_TYPE         MULTILINE_LENGTH
 ------ -------------------- ------------------------
   1111 ST_MultiLineString     +2.76437123387202E+001
```

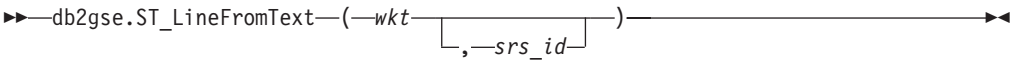
ST_LineFromText

ST_LineFromText는 선스트링에 대한 Well-Known Text 표현과 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 선스트링을 리턴합니다.

제공된 Well-Known Text 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

이 기능에 우선하는 버전은 ST_LineString입니다.

구문



매개변수

wkt 결과 선스트링의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

srs_id 결과 선스트링에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

*srs_id*가 카탈로그 뷰 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS에 나열된 공간 참조 시스템을 식별하지 못하면 오류(SQLSTATE 38SU1)가 리턴됩니다.

리턴 유형

db2gse.ST_LineString

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

다음 코드는 ST_LineFromText 함수를 사용하여 Well-Known Text 행 표현을 작성하여 삽입합니다. WKT 표현을 사용하여 ID 및 공간 참조 시스템 1의 행 값을 가진 행을 SAMPLE_LINES 테이블에 삽입합니다.

```
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
```

결과:

```
ID     LINESTRING
------ ---------------------------------------------------------------------
  1110 LINESTRING ( 850.00000000 250.00000000, 850.00000000 850.00000000)
  1111 LINESTRING EMPTY
```

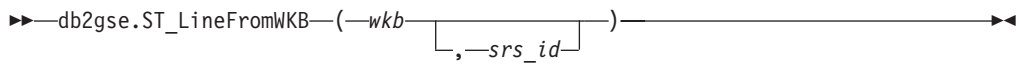
ST_LineFromWKB

ST_LineFromWKB는 선스트링에 대한 Well-Known Binary 표현 및 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 선스트링을 리턴합니다.

제공된 Well-Known Binary 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

이 기능에 우선하는 버전은 ST_LineString입니다.

구문



매개변수

wkb 결과 선스트링의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

srs_id 결과 선스트링에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

*srs_id*가 카탈로그 뷰 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS에 나열된 공간 참조 시스템을 식별하지 못하면 오류(SQLSTATE 38SU1)가 리턴됩니다.

리턴 유형

db2gse.ST_LineString

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

다음 코드는 ST_LineFromWKB 함수를 사용하여 Well-Known Binary 표현으로부터 행을 작성하여 삽입합니다. WKB 표현을 사용하여 ID 및 공간 참조 시스템 1의 행을 가진 행을 SAMPLE_LINES 테이블에 삽입합니다.

```
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
```

결과:

```
ID     LINE
------ ------------------------------------------------------------------------
  1901 LINESTRING ( 850.00000000 250.00000000, 850.00000000 850.00000000)

  1902 LINESTRING ( 33.00000000 2.00000000, 34.00000000 3.00000000,
35.00000000 6.00000000)
```

ST_LineString

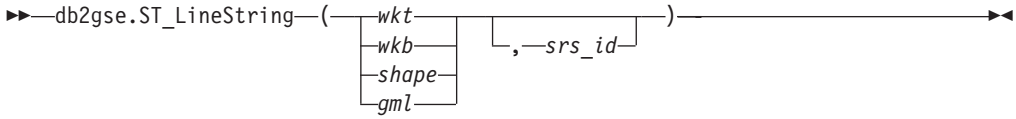
ST_LineString은 다음 입력 중 하나로부터 선스트링을 구성합니다.

- Well-Known Text 표현
- Well-Known Binary 표현
- ESRI 모양 표현
- GML(Geography Markup Language) 표현

결과 선스트링이 있는 공간 참조 시스템을 식별하기 위해 선택적으로 공간 참조 시스템 ID를 제공할 수 있습니다.

Well-Known Text 표현, Well-Known Binary 표현, ESRI 모양 표현 또는 GML 표현이 널(NULL)일 경우, 널(NULL)이 리턴됩니다.

구문



매개변수

wkt 결과 다각형의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

wkb 결과 다각형의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

shape 결과 다각형의 ESRI 모양 표현을 나타내는 BLOB(2G) 유형의 값

gml GML(Geography Markup Language)을 사용하는 결과 다각형을 나타내는 CLOB(2G) 유형의 값

srs_id 결과 다각형에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 식별자가 0인 공간 참조 시스템이 사용됩니다.

*srs_id*가 카탈로그 뷰 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS에 나열된 공간 참조 시스템을 식별하지 못하면 오류(SQLSTATE 38SU1)가 리턴됩니다.

리턴 유형

db2gse.ST_LineString

예

다음 코드는 ST_LineString 함수를 사용하여 Well-Known Text 행 표현 또는 Well-Known Binary 표현으로부터 행을 작성하여 삽입합니다.

다음 예는 WKT 및 GML 표현을 사용하여 ID 및 공간 참조 시스템 1의 행을 가진 행을 SAMPLE_LINES 테이블에 삽입합니다.

```
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
```

결과:

```
ID     LINESTRING
------ ---------------------------------------------------------------------
  1110 LINESTRING ( 850.00000000 250.00000000, 850.00000000 850.00000000)
  1111 LINESTRING ( 90.00000000 90.00000000, 100.00000000 100.00000000)
```

ST_LineStringN

ST_LineStringN은 입력 매개변수로 다중 선스트링 및 인덱스를 가져오며 인덱스에 의해 식별되는 선스트링을 리턴합니다. 결과 선스트링은 제공된 다중 선스트링의 공간 참조 시스템에 표시됩니다.

제공된 다중 선스트링이 널(NULL) 또는 비어 있거나 인덱스가 1보다 작거나 선스트링 수보다 큰 경우 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

multi_linestring

*index*에 의해 식별되는 선스트링이 리턴되는 다중 선스트링을 표시하는 유형 ST_MultiLineString의 값

index *multi_linestring*에서 리턴될 *n*번째 선스트링을 식별하는 유형 INTEGER의 값

*인덱스*가 1보다 작거나 *multi_linestring*의 선스트링 수보다 크면 널(NULL)이 리턴되고 경고 조건(SQLSTATE 01HS0)이 리턴됩니다.

리턴 유형

db2gse.ST_LineString

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

SELECT문은 SAMPLE_MLINES 테이블의 다중 선스트링 내에서 두 번째 Geometry를 선택하는 방법을 보여줍니다.

```
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
```

결과:

```
ID          SECOND_LINESTRING
----------- ----------------------------------------------------------------
1110         LINESTRING ( 28.00000000 4.00000000, 29.00000000
   5.00000000, 31.00000000 8.00000000, 43.00000000 12.00000000)

1111         LINESTRING ( 58.00000000 4.00000000, 59.00000000
   5.00000000, 61.00000000 8.00000000)
```

ST_M

ST_M은 다음 중 하나를 수행할 수 있습니다.

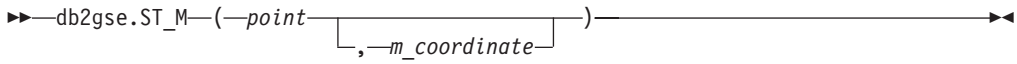
- 점을 입력 매개변수로 가져오며 해당 M(측정 단위) 좌표를 리턴합니다.
- 지정된 점에 기존의 M 좌표가 없더라도 점과 M 좌표를 가져오며 M 좌표가 제공된 측정 단위로 설정된 점 자체를 리턴합니다.

지정된 M 좌표가 널(NULL)이면 점의 M 좌표가 제거됩니다.

지정된 점이 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

point M 좌표를 리턴하거나 수정하는 ST_Point 유형의 값

m_coordinate

*point*에 대한 새 M 좌표를 표시하는 DOUBLE 유형의 값

*m_coordinate*가 널(NULL)이면 *point*에서 M 좌표가 제거됩니다.

리턴 유형

- DOUBLE, *m_coordinate*면 지정되지 않습니다.
- db2gse.ST_Point, *m_coordinate*면 지정됩니다.

예

예 1

이 예에서는 ST_M 함수의 사용법을 보여줍니다. 세 개의 점을 작성하여 SAMPLE_POINTS 테이블에 삽입합니다. 이 점은 모두 ID가 1인 공간 참조 시스템에 있습니다.

```
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))
```

예 2

이 예에서는 SAMPLE_POINTS 테이블에 있는 점의 M 좌표를 찾습니다.

```
SELECT id, ST_M (geometry) M_COORD
  FROM sample_points
```

결과:

```
ID          M_COORD
----------- ------------------------
          1 +5.00000000000000E+000
          2 +4.00000000000000E+000
          3 +7.00000000000000E+000
```

예 3

이 예는 M 좌표가 40으로 설정된 점 중 하나를 리턴합니다.

```
SELECT id, CAST (ST_AsText (ST_M (geometry, 40) )
  AS VARCHAR(60) ) M_COORD_40
  FROM sample_points
  WHERE id=3
```

결과:

```
ID         M_COORD_40
---------- ------------------------------------------------------------
         3 POINT ZM (3.00000000 8.00000000 23.00000000 40.00000000)
```

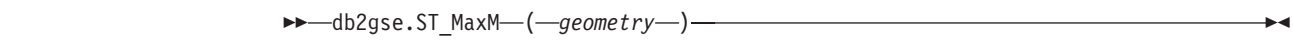
ST_MaxM

ST_MaxM은 입력 매개변수로 Geometry를 가져오며 해당 최대 M 좌표를 리턴합니다.

제공된 Geometry가 널(NULL) 값 또는 비어 있거나 M 좌표를 갖지 않으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

유형 ST_Geometry 값 또는 최대 M 좌표가 리턴되는 해당 부속 유형 중 하나의 값

리턴 유형

DOUBLE

예

예 1

이 예는 ST_MaxM 함수의 사용법을 보여줍니다. 세 개의 다각형을 작성하여 SAMPLE_POLYS 테이블에 삽입합니다.

```
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) )
```

예 2

이 예는 SAMPLE_POLYS에 있는 각 다각형의 최대 M 좌표를 찾습니다.

```
SELECT id, CAST ( ST_MaxM(geometry) AS INTEGER) MAX_M
  FROM sample_polys
```

결과:

```
ID          MAX_M
----------- ------------
          1            4
          2           12
          3           16
```

예 3

이 예는 GEOMETRY 컬럼의 모든 다각형에 대해 존재하는 최대 M 좌표를 찾습니다.

```
SELECT CAST ( MAX ( ST_MaxM(geometry) ) AS INTEGER) OVERALL_MAX_M
  FROM sample_polys
```

결과:

```
OVERALL_MAX_M
--------------
           16
```

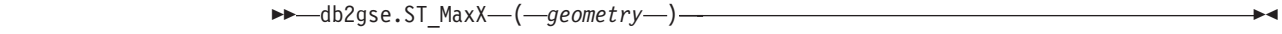
ST_MaxX

ST_MaxX는 입력 매개변수로 Geometry를 가져오며 해당 최대 X 좌표를 리턴합니다.

제공된 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

유형 ST_Geometry 값 또는 최대 X 좌표가 리턴되는 해당 부속 유형 중 하나의 값

리턴 유형

DOUBLE

예

예 1

이 예에서는 ST_MaxX 함수 사용법에 대해 설명합니다. 세 개의 다각형을 작성하여 SAMPLE_POLYS 테이블에 삽입합니다. 세 번째 예는 최대 및 최소 좌표값을 리턴하는 모든 함수를 사용하여 특정 공간 컬럼에 저장된 Geometry의 공간 범위를 평가할 수 있는 방법을 보여줍니다.

```
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) )
```

예 2

이 예에서는 SAMPLE_POLYS에 있는 각 다각형의 최대 X 좌표를 찾습니다.

```
SELECT id, CAST ( ST_MaxX(geometry) AS INTEGER) MAX_X_COORD
  FROM sample_polys
```

결과:

```
ID          MAX_X_COORD
----------- ------------
          1          120
          2            5
          3           12
```

예 3

이 예에서는 GEOMETRY 컬럼의 모든 다각형에 대해 존재하는 최대 X 좌표를 찾습니다.

```
SELECT CAST ( MAX ( ST_MaxX(geometry) ) AS INTEGER) OVERALL_MAX_X
  FROM sample_polys
```

결과:

```
OVERALL_MAX_X
-------------
          120
```

예 4

이 예에서는 SAMPLE_POLYS 테이블의 모든 다각형에 대한 공간 extent(전체 최소값에서 전체 최대값까지)를 찾습니다. 이 계산은 일반적으로 Geometry의 실제 공간 extent를 데이터에 연관된 공간 참조 시스템의 공간 extent와 비교하여 데이터에 증가할 공간이 있는지를 판별합니다.

```
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
```

결과:

```
MIN_X    MIN_Y    MIN_Z    MIN_M    MAX_X    MAX_Y    MAX_Z    MAX_M
------   ------   ------   ------   ------   ------   ------   ------
     0        0       10        3      120      140       40       16
```

ST_MaxY

ST_MaxY는 입력 매개변수로 Geometry를 가져오며 해당 최대 Y 좌표를 리턴합니다.

제공된 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 값 또는 최대 Y 좌표가 리턴되는 해당 부속 유형 중 하나의 값

리턴 유형

DOUBLE

예

예 1

이 예는 ST_MaxY 함수의 사용법을 보여줍니다. 세 개의 다각형을 작성하여 SAMPLE_POLYS 테이블에 삽입합니다.

```
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) )
```

예 2

이 예에서는 SAMPLE_POLYS에 있는 각 다각형에 대한 최대 Y 좌표를 찾습니다.

```
SELECT id, CAST ( ST_MaxY(geometry) AS INTEGER) MAX_Y
  FROM sample_polys
```

결과:

```
ID          MAX_Y
----------- ------------
          1          140
          2            4
          3           13
```

예 3

이 예에서는 GEOMETRY 컬럼의 모든 다각형에 대해 존재하는 최대 Y 좌표를 찾습니다.

```
SELECT CAST ( MAX ( ST_MaxY(geometry) ) AS INTEGER) OVERALL_MAX_Y
  FROM sample_polys
```

결과:

```
OVERALL_MAX_Y
--------------
           140
```

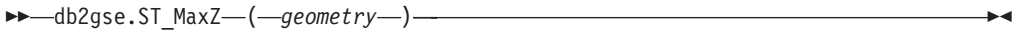
ST_MaxZ

ST_MaxZ는 입력 매개변수로 Geometry를 가져오며 해당 최대 Z 좌표를 리턴합니다.

제공된 Geometry가 널(NULL) 값 또는 비어 있거나 Z 좌표를 갖지 않으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

유형 ST_Geometry 값 또는 최대 Z 좌표가 리턴되는 해당 부속 유형 중 하나의 값

리턴 유형

DOUBLE

예

예 1

이 예는 ST_MaxZ 함수의 사용법을 보여줍니다. 세 개의 다각형을 작성하여 SAMPLE_POLYS 테이블에 삽입합니다.

```
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) )
```

예 2

이 예에서는 SAMPLE_POLYS에 있는 각 다각형에 대한 최대 Z 좌표를 찾습니다.

```
SELECT id, CAST ( ST_MaxZ(geometry) AS INTEGER) MAX_Z
  FROM sample_polys
```

결과:

```
ID          MAX_Z
----------- ------------
          1           26
          2           40
          3           12
```

예 3

이 예에서는 GEOMETRY 컬럼의 모든 다각형에 대해 존재하는 최대 Z 좌표를 찾습니다.

```
SELECT CAST ( MAX ( ST_MaxZ(geometry) ) AS INTEGER) OVERALL_MAX_Z
  FROM sample_polys
```

결과:

```
OVERALL_MAX_Z
--------------
           40
```

ST_MBR

ST_MBR은 입력 매개변수로 Geometry를 가져오며 해당 최소 바운딩 직사각형을 리턴합니다.

제공된 Geometry가 점이면, 점 자체가 리턴됩니다. Geometry가 수평 선스트링 또는 수직 선스트링이고 공간 참조 시스템이 측지 시스템이 아닌 경우, 수평 또는 수직 선스트링 자체가 리턴됩니다. 그렇지 않으면, Geometry의 최소 바운딩 직사각형이 다각형으로 리턴됩니다. 제공된 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

최소 바운딩 직사각형이 리턴되는 Geometry를 표시하는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

리턴 유형

db2gse.ST_Geometry

예

이 예는 ST_MBR 함수를 사용하여 다각형의 최소 바운딩 직사각형을 리턴하는 방법을 보여줍니다. 지정된 Geometry가 다각형이기 때문에 최소 바운딩 직사각형이 다각형으로 리턴됩니다.

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

```
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
```

결과:

```
ID        MBR
--------- ------------------------------------------------------------
        1 POLYGON (( 5.00000000 5.00000000, 15.00000000 5.00000000,
             15.00000000 11.00000000, 5.00000000 11.00000000,
              5.00000000 5.00000000))
```

```
2 POLYGON (( 20.00000000 30.00000000, 30.00000000 30.00000000,
     30.00000000 35.00000000, 20.00000000 35.00000000,
     20.00000000 30.00000000 ))
```

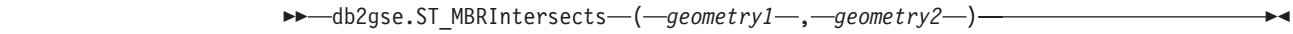
ST_MBRIntersects

ST_MBRIntersects는 두 개의 Geometry를 입력 매개변수로 가져오며 두 Geometry의 최소 바운딩 직사각형이 교차할 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다. 점의 최소 바운딩 직사각형 및 수평 및 수직 선스트링은 Geometry 자체입니다.

두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템에 표시되지 않으면, 다른 공간 참조 시스템으로 변환됩니다.

제공된 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

구문



매개변수

geometry1

최소 바운딩 직사각형이 *geometry2*의 최소 바운딩 직사각형과 교차하는지를 테스트할 Geometry를 나타내는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

geometry2

최소 바운딩 직사각형이 *geometry1*의 최소 바운딩 직사각형과 교차하는지를 테스트할 Geometry를 나타내는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

리턴 유형

INTEGER

예

예 1

이 예에서는 ST_MBRIntersects를 사용하여 MBR(Minimum Bounding Rectangle)의 교차 여부를 조사하여 두 개의 교차하지 않는 다각형이 서로 얼마나 가까이 있는지를 알아보는 방법을 보여줍니다. 첫 번째 예에서는 SQL CASE 표현식을 사용합니다. 두 번째 예에서는 하나의 SELECT문을 사용하여 ID가 2인 다각형의 최소 바운딩 직사각형이 교차하는 다각형을 찾습니다.

```
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) )
```

예 2

다음 SELECT문은 CASE 표현식을 사용하여 교차하는 최소 바운딩 직사각형을 갖는 다각형의 ID를 찾습니다.

```
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
```

결과:

```
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
```

예 3

다음 SELECT문은 Geometry에 대한 최소 바운딩 직사각형이 ID가 2인 다각형과 교차하는지를 판별합니다.

```
SELECT a.id, b.id, ST_MBRIntersects (a.geometry, b.geometry) MBR_INTERSECTS
  FROM sample_polys a, sample_polys b
  WHERE a.id = 2
```

결과

```
ID         ID          MBR_INTERSECTS
---------- ----------- --------------
         2           1              1
         2           2              1
         2           3              0
```

ST_MeasureBetween, ST_LocateBetween

ST_MeasureBetween 또는 ST_LocateBetween은 하나의 Geometry 및 두 개의 M 좌표(측정 단위)를 입력 매개변수로 가져오며 두 개의 M 좌표 간에 연결이 끊어진 경로 또는 점 세트를 나타내는 제공된 Geometry의 일부를 리턴합니다.

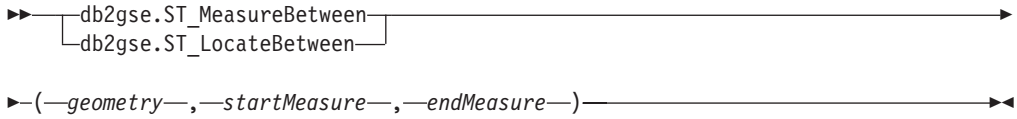
곡선, 다중 곡선, 표면 및 다중 표면의 경우, 결과를 계산하기 위해 보간법(interpolation)이 수행됩니다. 결과 Geometry는 제공된 Geometry의 공간 참조 시스템에 표시됩니다.

제공된 Geometry가 표면 또는 다중 표면인 경우, ST_MeasureBetween 또는 ST_LocateBetween이 Geometry의 외부 및 내부에 적용됩니다. 제공된 Geometry의 어느 한 부분도 M 좌표에 의해 정의된 구간에 들어 있지 않은 경우 빈 Geometry가 리턴됩니다. 제공된 Geometry가 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

결과 Geometry가 비어 있지 않은 경우 다중점 또는 다중 선스트링 유형이 리턴됩니다.

두 함수를 메소드로 호출할 수도 있습니다.

구문



매개변수

geometry

ST_Geometry 유형의 값 또는 부분의 측정값이 *startMeasure*와 *endMeasure* 사이에 있는 Geometry를 나타내는 해당 부속 유형 중 하나의 값

startMeasure

측정 구간의 하한을 나타내는 DOUBLE 유형의 값. 이 값이 널(NULL)이면 하한은 적용되지 않습니다.

endMeasure

측정 구간의 상한을 나타내는 DOUBLE 유형의 값. 이 값이 널(NULL)이면 상한이 적용되지 않습니다.

리턴 유형

db2gse.ST_Geometry

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

Geometry의 M 좌표(측정 단위)는 사용자가 정의합니다. 측정하고자 하는 모든 단위(고속도로의 거리, 온도, 압력 또는 pH 측정)를 나타낼 수 있기 때문에 M 좌표는 매우 다양합니다.

이 예는 M 좌표를 사용하여 수집된 pH 측정 데이터를 기록하는 것을 보여줍니다. 조사자는 고속도로를 따라 특정 지점에서 토양의 pH를 수집합니다. 표준 작업 프로시저에 따라 조사자는 토양 샘플을 채취한 모든 지역에서 필요한 값(해당 지역의 X 및 Y 좌표와 측정한 pH)을 기록합니다.

```
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 ) )
```

토양의 산성도가 4 - 6에 있는 경로를 찾기 위해 조사자는 다음 SELECT문을 사용합니다.

```
SELECT id, CAST( ST_AsText( ST_MeasureBetween( 4, 6) )
  AS VARCHAR(150) ) MEAS_BETWEEN_4_AND_6
  FROM sample_lines
```

결과:

```
ID          MEAS_BETWEEN_4_AND_6
----------- ------------------------------------------------------------
          1 LINESTRING M (3.00000000 4.33333300 4.00000000,
                          3.00000000 3.00000000 6.00000000,
                          4.00000000 4.00000000 6.00000000,
                          5.00000000 5.00000000 6.00000000)
```

ST_MidPoint

ST_MidPoint는 입력 매개변수로 곡선을 가져오며 곡선과 함께 측정된 곡선의 양쪽 끝점에서 같은 거리에 있는 곡선의 점을 리턴합니다. 결과 점은 제공된 곡선의 공간 참조 시스템에 표시됩니다.

제공된 곡선이 비어 있으면, 빈 점이 리턴됩니다. 제공된 곡선이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

곡선에 Z 좌표 또는 M 좌표(측정 단위)가 포함되어 있으면 중간점은 곡선의 X 및 Y 좌표값에 의해서만 결정됩니다. 리턴된 점의 Z 좌표 및 측정 단위가 보간(삽입)됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

curve 유형 ST_Curve의 값 또는 중간의 점이 리턴되는 곡선을 표시하는 해당 부속 유형 중 하나의 값

리턴 유형

db2gse.ST_Point

예

이 예는 ST_MidPoint를 사용하여 곡선의 중간점을 리턴하는 방법을 보여줍니다.

```
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
```

결과:

```
ID          MID_POINT
----------- ------------------------------------
          1 POINT ( 0.00000000 20.00000000)
          2 POINT ( 3.00000000 3.45981800)
          3 POINT ( 5.00000000 0.00000000)
          4 POINT ( 7.50000000 20.00000000)
```

ST_MinM

ST_MinM은 입력 매개변수로 Geometry를 가져오며 해당 최소 M 좌표를 리턴합니다.

제공된 Geometry가 널(NULL) 값 또는 비어 있거나 M 좌표를 갖지 않으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 값 또는 최소 M 좌표가 리턴되는 해당 부속 유형 중 하나의 값

리턴 유형

DOUBLE

예

예 1

이 예는 ST_MinM 함수 사용법을 보여줍니다. 세 개의 다각형을 작성하여 SAMPLE_POLYS 테이블에 삽입합니다.

```
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) )
```

예 2

이 예에서는 SAMPLE_POLYS의 각 다각형에 대한 최소 M 좌표를 찾습니다.

```
SELECT id, CAST ( ST_MinM(geometry) AS INTEGER) MIN_M
  FROM sample_polys
```

결과:

```
ID          MIN_M
----------- -----------
          1           3
          2           5
          3          11
```

예 3

이 예에서는 GEOMETRY 열의 모든 다각형에 대해 존재하는 최소 M 좌표를 찾습니다.

```
SELECT CAST ( MIN ( ST_MinM(geometry) ) AS INTEGER) OVERALL_MIN_M
  FROM sample_polys
```

결과:

```
OVERALL_MIN_M
-------------
            3
```

ST_MinX

ST_MinX는 입력 매개변수로 Geometry를 가져오며 해당 최소 X 좌표를 리턴합니다.

제공된 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 값 또는 최소 X 좌표가 리턴되는 해당 부속 유형 중 하나의 값

리턴 유형

DOUBLE

예

예 1

이 예는 ST_MinX 함수의 사용법을 보여줍니다. 세 개의 다각형을 작성하여 SAMPLE_POLYS 테이블에 삽입합니다.

```
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) )
```

예 2

이 예에서는 SAMPLE_POLYS의 각 다각형에 대한 최소 X 좌표를 찾습니다.

```
SELECT id, CAST ( ST_MinX(geometry) AS INTEGER) MIN_X
  FROM sample_polys
```

결과:

```
ID          MIN_X
----------- -----------
          1         110
          2           0
          3           8
```

예 3

이 예에서는 GEOMETRY 컬럼의 모든 다각형에 대해 존재하는 최소 X 좌표를 찾습니다.

```
SELECT CAST ( MIN ( ST_MinX(geometry) ) AS INTEGER) OVERALL_MIN_X
  FROM sample_polys
```

결과:

```
OVERALL_MIN_X
-------------
            0
```

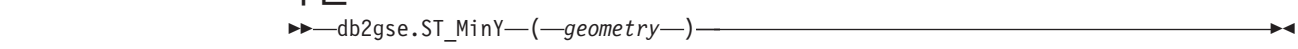
ST_MinY

ST_MinY는 입력 매개변수로 Geometry를 가져오며 해당 최소 Y 좌표를 리턴합니다.

제공된 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

유형 ST_Geometry 값 또는 최소 Y 좌표가 리턴되는 해당 부속 유형 중 하나의 값

리턴 유형

DOUBLE

예

예 1

이 예는 ST_MinY 함수의 사용법을 보여줍니다. 세 개의 다각형을 작성하여 SAMPLE_POLYS 테이블에 삽입합니다.

```
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) )
```

예 2

이 예에서는 SAMPLE_POLYS의 각 다각형에 대한 최소 Y 좌표를 찾습니다.

```
SELECT id, CAST ( ST_MinY(geometry) AS INTEGER) MIN_Y
  FROM sample_polys
```

결과:

```
ID          MIN_Y
----------- -----------
          1         120
          2           0
          3           4
```

예 3

이 예에서는 GEOMETRY 컬럼의 모든 다각형에 대해 존재하는 최소 Y 좌표를 찾습니다.

```
SELECT CAST ( MIN ( ST_MinY(geometry) ) AS INTEGER) OVERALL_MIN_Y
  FROM sample_polys
```

결과:

```
OVERALL_MIN_Y
-------------
            0
```

ST_MinZ

ST_MinZ는 입력 매개변수로 Geometry를 가져오며 해당 최소 Z 좌표를 리턴합니다.

제공된 Geometry가 널(NULL) 값 또는 비어 있거나 Z 좌표를 갖지 않으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 값 또는 최소 Z 좌표가 리턴되는 해당 부속 유형 중 하나의 값

리턴 유형

DOUBLE

예

예 1

이 예는 ST_MinZ 함수의 사용법을 보여줍니다. 세 개의 다각형을 작성하여 SAMPLE_POLYS 테이블에 삽입합니다.

```
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) )
```

예 2

이 예에서는 SAMPLE_POLYS의 각 다각형에 대한 최소 Z 좌표를 찾습니다.

```
SELECT id, CAST ( ST_MinZ(geometry) AS INTEGER) MIN_Z
  FROM sample_polys
```

결과:

```
ID          MIN_Z
----------- ------------
          1           20
          2           31
          3           10
```

예 3

이 예에서는 GEOMETRY 컬럼의 모든 다각형에 대해 존재하는 최소 Z 좌표를 찾습니다.

```
SELECT CAST ( MIN ( ST_MinZ(geometry) ) AS INTEGER) OVERALL_MIN_Z
  FROM sample_polys
```

결과:

```
OVERALL_MIN_Z
--------------
            10
```

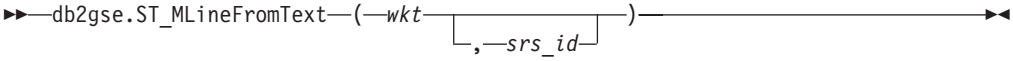
ST_MLineFromText

ST_MLineFromText는 다중 선스트링에 대한 Well-Known Text 표현과 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 다중 선스트링을 리턴합니다.

제공된 Well-Known Text 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

동일한 결과를 얻기 위해서는 ST_MultiLineString 함수를 사용하도록 권장합니다. 이 함수는 유연성이 뛰어나기 때문에 권장됩니다. ST_MultiLineString은 Well-Known Text 표현뿐 아니라 다른 입력 양식도 가져옵니다.

구문



매개변수

wkt 결과 다중 선스트링의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

srs_id 결과 다중 선스트링에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

지정된 *srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_MultiLineString

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_MLineFromText를 사용하여 Well-Known Text 표현으로부터 다중 선스트링을 작성하고 삽입하는 방법을 보여줍니다. 삽입된 레코드의 ID는 1110이고 Geometry는 공간 참조 시스템 1의 다중 선스트링입니다. 다중 선스트링은 다중 선스트링에 대한 Well-Known Text 표현으로 표시됩니다. 이 Geometry에 대한 X 및 Y 좌표는 다음과 같습니다.

- 행 1: (33, 2) (34, 3) (35, 6)
- 행 2: (28, 4) (29, 5) (31, 8) (43, 12)
- 행 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) )
```

다음 SELECT문은 테이블에 기록된 다중 선스트링을 리턴합니다.

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(280) ) MULTI_LINE_STRING
  FROM sample_mlines
  WHERE id = 1110
```

결과:

```
ID         MULTI_LINE_STRING
---------- ------------------------------------------------------------------
      1110 MULTILINESTRING (( 33.00000000 2.00000000, 34.00000000 3.00000000,
             35.00000000 6.00000000),
                          ( 28.00000000 4.00000000, 29.00000000 5.00000000,
             31.00000000 8.0000000, 43.00000000 12.00000000),
                          ( 39.00000000 3.00000000, 37.00000000 4.00000000,
             36.00000000 7.00000000 ))
```

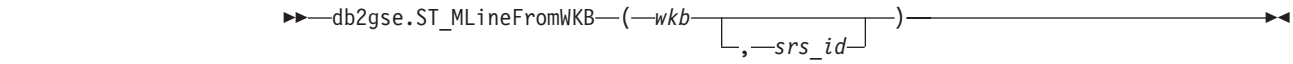
ST_MLineFromWKB

ST_MLineFromWKB는 다중 선스트링에 대한 Well-Known Binary 표현과 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 다중 선스트링을 리턴합니다.

제공된 Well-Known Binary 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

동일한 결과를 얻기 위해서는 ST_MultiLineString 함수를 사용하도록 권장합니다. 이 함수는 유연성이 뛰어나기 때문에 권장됩니다. ST_MultiLineString은 Well-Known Binary 표현뿐 아니라 다른 입력 양식도 가져옵니다.

구문



매개변수

wkb 결과 다중 선스트링의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

srs_id 결과 다중 선스트링에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

지정된 *srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_MultiLineString

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_MLineFromWKB를 사용하여 해당 Well-Known Binary 표현으로부터 다중 선스트링을 작성하는 방법을 보여줍니다. Geometry는 공간 참조 시스템 1에 있는 다중 선스트링입니다. 이 예에서 SAMPLE_MLINES 테이블의 GEOMETRY 컬럼에 ID가 10인 다중 선스트링을 저장한 다음 ST_AsBinary 함수를 사용하여 Well-Known Binary 표현으로 WKB 컬럼을 갱신합니다. 마지막으로 ST_MLineFromWKB 함수를 사용하여 WKB 컬럼에 있는 다중 선스트링을 리턴합니다. 이 Geometry에 대한 X 및 Y 좌표는 다음과 같습니다.

- 행 1: (61, 2) (64, 3) (65, 6)
- 행 2: (58, 4) (59, 5) (61, 8)
- 행 3: (69, 3) (67, 4) (66, 7) (68, 9)

SAMPLE_MLINES 테이블에는 다중 선스트링을 저장하는 GEOMETRY 컬럼과 다중 선스트링의 Well-Known Binary 표현을 저장하는 WKB 컬럼이 있습니다.

```
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
```

다음 SELECT문에서는 ST_MLineFromWKB 함수를 사용하여 WKB 컬럼에서 다중 선스트링을 검색합니다.

```
SELECT id, CAST( ST_AsText( ST_MLineFromWKB (wkb) )
       AS VARCHAR(280) ) MULTI_LINE_STRING
  FROM sample_mlines
  WHERE id = 10
```

결과:

```
ID         MULTI_LINE_STRING
---------- ------------------------------------------------------------------
        10 MULTILINESTRING (( 61.00000000 2.00000000, 64.00000000 3.00000000,
              65.00000000 6.00000000),
```

```
                ( 58.00000000 4.00000000, 59.00000000 5.00000000,
61.00000000 8.0000000),
                ( 69.00000000 3.00000000, 67.00000000 4.00000000,
66.00000000 7.00000000, 68.00000000 9.00000000 ))
```

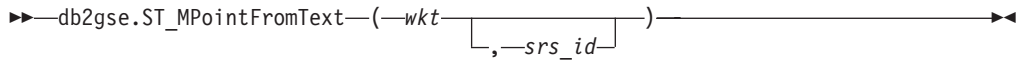
ST_MPointFromText

ST_MPointFromText는 다중점에 대한 Well-Known Text 표현과 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 다중점을 리턴합니다.

제공된 Well-Known Text 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

동일한 결과를 얻기 위해서는 ST_MultiPoint 함수를 사용하도록 권장합니다. 이 함수는 유연성이 뛰어나기 때문에 권장됩니다. ST_MultiPoint는 Well-Known Text 표현뿐 아니라 다른 입력 양식도 가져옵니다.

구문



매개변수

wkt 결과 다중점의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

srs_id 결과 다중점에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

지정된 *srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_MultiPoint

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_MPointFromText를 사용하여 Well-Known Text 표현으로부터 다중점을 작성하여 삽입하는 방법을 보여줍니다. 삽입된 레코드의 ID는 1110이고 Geometry는 공간 참조 시스템 1의 다중점입니다. 다중점은 다중점에 대한 Well-Known Text 표현으로 표시됩니다. 이 Geometry의 X 및 Y 좌표는 (1, 2) (4, 3) (5, 6)입니다.

```
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) )
```

다음 SELECT문은 테이블에 기록된 다중점을 리턴합니다.

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(280) ) MULTIPOINT
  FROM sample_mpoints
  WHERE id = 1110
```

결과:

```
ID         MULTIPOINT
---------- ------------------------------------------------------------------
      1110 MULTIPOINT (1.00000000 2.00000000, 4.00000000 3.00000000,
                       5.00000000 6.00000000)
```

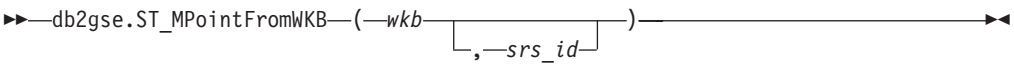
ST_MPointFromWKB

ST_MPointFromWKB는 다중점에 대한 Well-Known Binary 표현과 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 다중점을 리턴합니다.

제공된 Well-Known Binary 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

동일한 결과를 얻기 위해서는 ST_MultiPoint 함수를 사용하도록 권장합니다. 이 함수는 유연성이 뛰어나기 때문에 권장됩니다. ST_MultiPoint는 Well-Known Binary 표현뿐 아니라 다른 입력 양식도 가져옵니다.

구문



매개변수

wkb 결과 다중점의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

srs_id 결과 다중점에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

지정된 *srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_MultiPoint

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_MPointFromWKB를 사용하여 해당 Well-Known Binary 표현으로부터 다중점을 작성하는 방법을 보여줍니다. Geometry는 공간 참조 시스템 1에 있는 다중점입니다. 이 예에서 SAMPLE_MPOINTS 테이블의 GEOMETRY 컬럼에 ID가 10인 다중점을 저장한 다음 ST_AsBinary 함수를 사용하여 Well-Known Binary 표현으로 WKB 컬럼을 갱신합니다. 마지막으로 ST_MPointFromWKB 함수를 사용하여 WKB 컬럼으로부터 다중점을 리턴합니다. 이 Geometry에 대한 X 및 Y 좌표는 (44, 14) (35, 16) (24, 13)입니다.

SAMPLE_MPOINTS 테이블에는 다중점을 저장하는 GEOMETRY 컬럼과 다중점의 Well-Known Binary 표현을 저장하는 WKB 컬럼이 있습니다.

```
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
```

다음 SELECT문에서는 ST_MPointFromWKB 함수를 사용하여 WKB 컬럼으로부터 다중점을 검색합니다.

```
SELECT id, CAST( ST_AsText( ST_MLineFromWKB (wkb)) AS VARCHAR(100)) MULTIPOINT
  FROM sample_mpoints
  WHERE id = 10
```

결과:

```
ID         MULTIPOINT
---------- ------------------------------------------------------------------
        10 MULTIPOINT (44.00000000 14.00000000, 35.00000000
              16.00000000 24.00000000 13.00000000)
```

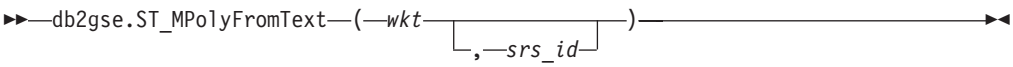
ST_MPolyFromText

ST_MPolyFromText는 다중 다각형에 대한 Well-Known Text 표현과 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 다중 다각형을 리턴합니다.

제공된 Well-Known Text 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

동일한 결과를 얻기 위해서는 ST_MultiPolygon 함수를 사용하도록 권장합니다. 이 함수는 유연성이 뛰어나기 때문에 권장됩니다. ST_MultiPolygon은 Well-Known Text 표현뿐 아니라 다른 입력 양식도 가져옵니다.

구문



매개변수

wkt 결과 다중 다각형의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

srs_id 결과 다중 다각형에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

지정된 *srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_MultiPolygon

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_MPolyFromText를 사용하여 Well-Known Text 표현으로부터 다중 다각형을 작성하여 삽입하는 방법을 보여줍니다. 삽입된 레코드의 ID는 1110이고 Geometry는 공간 참조 시스템 1의 다중 다각형입니다. 다중 다각형은 다중 다각형에 대한 Well-Known Text 표현으로 표시됩니다. 이 Geometry에 대한 X 및 Y 좌표는 다음과 같습니다.

- 다각형 1: (3, 3) (4, 6) (5, 3) (3, 3)
- 다각형 2: (8, 24) (9, 25) (1, 28) (8, 24)
- 다각형 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) )
```

다음 SELECT문은 테이블에 기록된 다중 다각형을 리턴합니다.

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(350) ) MULTI_POLYGON
  FROM sample_mpolys
  WHERE id = 1110
```

결과:

```
ID         MULTI_POLYGON
------- ------------------------------------------------------------------
   1110 MULTIPOLYGON ((( 13.00000000 33.00000000, 10.00000000 43.00000000,
           1.00000000 40.00000000, 7.00000000 36.00000000,
           13.00000000 33.00000000)),
                      (( 8.00000000 24.00000000, 9.00000000 25.00000000,
           1.00000000 28.0000000, 8.00000000 24.00000000)),
                       ( 3.00000000 3.00000000, 5.00000000 3.00000000,
           4.00000000 6.00000000, 3.00000000 3.00000000)))
```

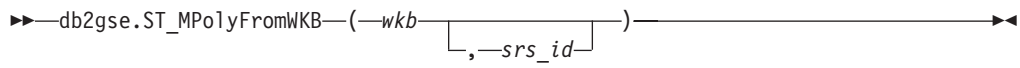
ST_MPolyFromWKB

ST_MPolyFromWKB는 다중 다각형에 대한 Well-Known Binary 표현과 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 다중 다각형을 리턴합니다.

제공된 Well-Known Binary 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

동일한 결과를 얻기 위해서는 ST_MultiPolygon 함수를 사용하도록 권장합니다. 이 함수는 유연성이 뛰어나기 때문에 권장됩니다. ST_MultiPolygon은 Well-Known Binary 표현뿐 아니라 다른 입력 양식도 가져옵니다.

구문



매개변수

wkb 결과 다중 다각형의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

srs_id 결과 다중 다각형에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

지정된 *srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_MultiPolygon

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_MPolyFromWKB를 사용하여 해당 Well-Known Binary 표현으로부터 다중 다각형을 작성하는 방법을 보여줍니다. Geometry는 공간 참조 시스템 1에 있는 다중 다각형입니다. 이 예에서 SAMPLE_MPOLYS 테이블의 GEOMETRY 컬럼에 ID가 10인 다중 다각형을 저장한 다음 ST_AsBinary 함수를 사용하여 Well-Known Binary 표현으로 WKB 컬럼을 갱신합니다. 마지막으로 ST_MPolyFromWKB 함수를 사용하여 WKB 컬럼으로부터 다중 다각형을 리턴합니다. 이 Geometry에 대한 X 및 Y 좌표는 다음과 같습니다.

- 다각형 1: (1, 72) (4, 79) (5, 76) (1, 72)
- 다각형 2: (10, 20) (10, 40) (30, 41) (10, 20)
- 다각형 3: (9, 43) (7, 44) (6, 47) (9, 43)

SAMPLE_MPOLYS 테이블에는 다중 다각형을 저장하는 GEOMETRY 컬럼과 다중 다각형의 Well-Known Binary 표현을 저장하는 WKB 컬럼이 있습니다.

```
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
```

다음 SELECT문에서는 ST_MPolyFromWKB 함수를 사용하여 WKB 컬럼에서 다중 다각형을 검색합니다.

```
SELECT id, CAST( ST_AsText( ST_MPolyFromWKB (wkb) )
    AS VARCHAR(320) ) MULTIPOLYGON
  FROM sample_mpolys
  WHERE id = 10
```

결과:

```
ID         MULTIPOLYGON
---------- ------------------------------------------------------------------
        10 MULTIPOLYGON ((( 10.00000000 20.00000000, 30.00000000
              41.00000000, 10.00000000 40.00000000, 10.00000000
              20.00000000)),
                          ( 1.00000000 72.00000000, 5.00000000
              76.00000000, 4.00000000 79.0000000, 1.00000000
```

```
72,00000000)),
              ( 9.00000000 43.00000000, 6.00000000
 47.00000000, 7.00000000 44.00000000, 9.00000000
 43.00000000 )))
```

ST_MultiLineString

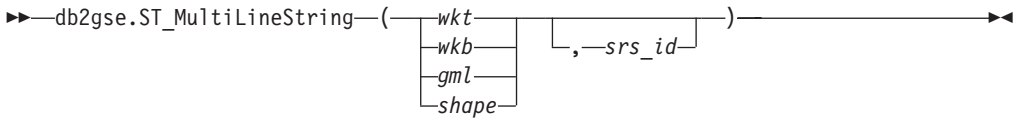
ST_MultiLineString은 다음을 입력하여 다중 선스트링을 구성합니다.

- Well-Known Text 표현
- Well-Known Binary 표현
- 모양 표현
- GML(Geography Markup Language) 표현

선택적 공간 참조 시스템 ID를 지정하여 결과 다중 선스트링이 있는 공간 참조 시스템을 식별할 수 있습니다.

Well-Known Text 표현, Well-Known Binary 표현, 모양 표현 또는 GML 표현이 널(NULL)일 경우, 널(NULL)이 리턴됩니다.

구문



매개변수

wkt 결과 다중 선스트링의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

wkb 결과 다중 선스트링의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

gml GML(Geography Markup Language)을 사용하는 결과 다중 선스트링을 나타내는 CLOB(2G) 유형의 값

shape 결과 다중 선스트링의 모양 표현을 나타내는 BLOB(2G) 유형의 값

srs_id 결과 다중 선스트링에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 식별자가 0인 공간 참조 시스템이 사용됩니다.

*srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_MultiLineString

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_MultiLineString을 사용하여 해당 Well-Known Text 표현으로부터 다중 선스트링을 작성하여 삽입하는 방법을 보여줍니다. 삽입된 레코드의 ID는 1110이고 Geometry는 공간 참조 시스템 1의 다중 선스트링입니다. 다중 선스트링은 다중 선스트링에 대한 Well-Known Text 표현으로 표시됩니다. 이 Geometry에 대한 X 및 Y 좌표는 다음과 같습니다.

- 행 1: (33, 2) (34, 3) (35, 6)
- 행 2: (28, 4) (29, 5) (31, 8) (43, 12)
- 행 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) )
```

다음 SELECT문은 테이블에 기록된 다중 선스트링을 리턴합니다.

```
SELECT id,
       CAST( ST_AsText( geometry ) AS VARCHAR(280) )
  MULTI_LINE_STRING
  FROM sample_mlines
  WHERE id = 1110
```

결과:

```
ID         MULTI_LINE_STRING
------- ------------------------------------------------------------------
   1110 MULTILINESTRING (( 33.00000000 2.00000000, 34.00000000 3.00000000,
              35.00000000 6.00000000),
                        ( 28.00000000 4.00000000, 29.00000000 5.00000000,
              31.00000000 8.0000000, 43.00000000 12.00000000),
                        ( 39.00000000 3.00000000, 37.00000000 4.00000000,
              36.00000000 7.00000000 ))
```

ST_MultiPoint

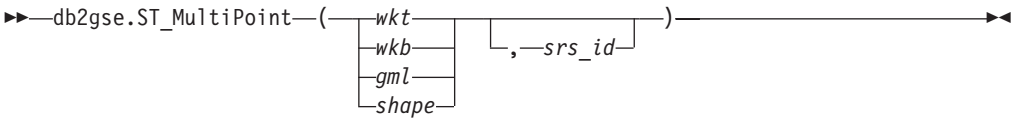
ST_MultiPoint는 다음과 같은 입력 중 하나로부터 다중점을 구성합니다.

- Well-Known Text 표현
- Well-Known Binary 표현
- 모양 표현
- GML(Geography Markup Language) 표현

선택적 공간 참조 시스템 ID를 지정하여 결과 다중점이 있는 공간 참조 시스템을 표시할 수 있습니다.

Well-Known Text 표현, Well-Known Binary 표현, 모양 표현 또는 GML 표현이 널(NULL)일 경우, 널(NULL)이 리턴됩니다.

구문



매개변수

wkt 결과 다중점의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

wkb 결과 다중점의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

gml GML(Geography Markup Language)을 사용하는 결과 다중점을 나타내는 CLOB(2G) 유형의 값

shape 결과 다중점의 모양 표현을 나타내는 BLOB(2G) 유형의 값

srs_id 결과 다중점에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

*srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_Point

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_MultiPoint를 사용하여 Well-Known Text 표현으로부터 다중점을 작성하여 삽입하는 방법을 보여줍니다. 삽입된 레코드의 ID는 1110이고 Geometry는 공간 참조 시스템 1의 다중점입니다. 다중점은 다중점에 대한 Well-Known Text 표현으로 표시됩니다. 이 Geometry의 X 및 Y 좌표는 (1, 2) (4, 3) (5, 6)입니다.

```
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))
```

다음 SELECT문은 테이블에 기록된 다중점을 리턴합니다.

```
SELECT id, CAST( ST_AsText(geometry) AS VARCHAR(90)) MULTIPOINT
  FROM sample_mpoints
  WHERE id = 1110
```

결과:

```
ID         MULTIPOINT
---------- ----------------------------------------------------
      1110 MULTIPOINT (1.00000000 2.00000000, 4.00000000
           3.00000000, 5.00000000 6.00000000)
```

ST_MultiPolygon

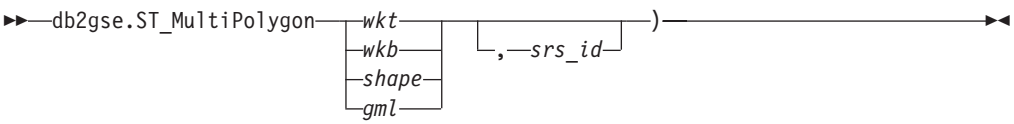
ST_MultiPolygon은 다음 입력 중 하나로부터 다중 다각형을 구성합니다.

- Well-Known Text 표현
- Well-Known Binary 표현
- 모양 표현
- GML(Geography Markup Language) 표현

선택적 공간 참조 시스템 ID를 지정하여 결과 다중 다각형이 있는 공간 참조 시스템을 식별할 수 있습니다.

Well-Known Text 표현, Well-Known Binary 표현, 모양 표현 또는 GML 표현이 널(NULL)일 경우, 널(NULL)이 리턴됩니다.

구문



매개변수

wkt 결과 다중 다각형의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

wkb 결과 다중 다각형의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

gml GML(Geography Markup Language)을 사용하는 결과 다중 다각형을 나타내는 CLOB(2G) 유형의 값

shape 결과 다각형의 모양 표현을 나타내는 BLOB(2G) 유형의 값

srs_id 결과 다중 다각형에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

*srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_MultiPolygon

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_MultiPolygon을 사용하여 Well-Known Text 표현으로부터 다중 다각형을 작성하여 삽입하는 방법을 보여줍니다. 삽입된 레코드의 ID는 1110이고 Geometry는 공간 참조 시스템 1의 다중 다각형입니다. 다중 다각형은 다중 다각형에 대한 Well-Known Text 표현으로 표시됩니다. 이 Geometry에 대한 X 및 Y 좌표는 다음과 같습니다.

- 다각형 1: (3, 3) (4, 6) (5, 3) (3, 3)
- 다각형 2: (8, 24) (9, 25) (1, 28) (8, 24)
- 다각형 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) )
```

다음 SELECT문은 테이블에 기록된 다중 다각형을 리턴합니다.

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(350) ) MULTI_POLYGON
  FROM sample_mpolys
  WHERE id = 1110
```

결과:

```
ID          MULTI_POLYGON
------- ------------------------------------------------------------------
   1110 MULTIPOLYGON ((( 13.00000000 33.00000000, 10.00000000 43.00000000,
           1.00000000 40.00000000, 7.00000000 36.00000000,
          13.00000000 33.00000000)),
                      (( 8.00000000 24.00000000, 9.00000000 25.00000000,
           1.00000000 28.0000000, 8.00000000 24.00000000)),
                      (( 3.00000000 3.00000000, 5.00000000 3.00000000,
           4.00000000 6.00000000, 3.00000000 3.00000000)))
```

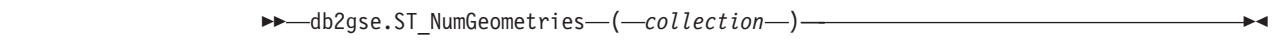
ST_NumGeometries

ST_NumGeometries는 Geometry 콜렉션을 입력 매개변수로 가져오며 콜렉션 내의 Geometry 수를 리턴합니다.

제공된 Geometry 콜렉션이 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

collection

Geometry 수를 리턴하는 Geometry 콜렉션을 나타내는 ST_GeomCollection 유형 또는 해당 부속 유형 중 하나의 값

리턴 유형

INTEGER

예

두 Geometry 콜렉션은 SAMPLE_GEOMCOLL 테이블에 저장됩니다. 하나는 다중 다각형이고 다른 하나는 다중점입니다. ST_NumGeometries 함수는 각 Geometry 콜렉션 내에 있는 개별 Geometry 수를 판별합니다.

```
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
```

결과:

```
ID          NUM_GEOMS_IN_COLL
----------- -----------------
          1                 3
          2                 5
```

ST_NumInteriorRing

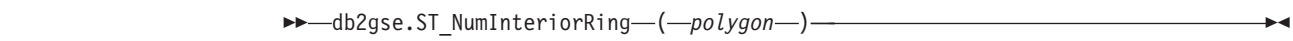
ST_NumInteriorRing은 입력 매개변수로 다각형을 가져오며 내부 링의 수를 리턴합니다.

제공된 다각형이 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

다각형에 내부 링이 없으면 0이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

polygon
내부 링의 수가 리턴되는 다각형을 표시하는 ST_Polygon 유형의 값

리턴 유형

INTEGER

예

다음 예에서는 두 개의 다각형을 작성합니다.

- 내부 링이 있는 다각형
- 내부 링이 없는 다각형

```
SET CURRENT FUNCTION PATH = CURRENT FUNCTION PATH, db2gse
CREATE TABLE sample_polys (id INTEGER, geometry ST_Polygon)

INSERT INTO sample_polys
  VALUES (1, ST_Polygon('polygon
        ((40 120, 90 120, 90 150, 40 150, 40 120),
         (50 130, 60 130, 60 140, 50 140, 50 130),
         (70 130, 80 130, 80 140, 70 140, 70 130))' , 0) )
```

```
INSERT INTO sample_polys
  VALUES (2, ST_Polygon('polygon ((5 15, 50 15, 50 105, 5 15))' , 0) )
```

ST_NumInteriorRing 함수는 Geometry의 링 수를 테이블로 리턴하는 데 사용됩니다.

```
SELECT id, ST_NumInteriorRing(geometry) NUM_RINGS
FROM sample_polys
```

결과:

```
ID         NUM_RINGS
---------- --------------
         1              2
         2              0
```

ST_NumLineStrings

ST_NumLineStrings는 다중 선스트링을 입력 매개변수로 가져오며 포함하는 선스트링 수를 리턴합니다.

제공된 다중 선스트링이 널(NULL)이거나 비어 있으면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

multilinestring
선스트링 수가 리턴되는 다중 선 스트링을 표시하는 유형 ST_MultilLineString의 값

리턴 유형

INTEGER

예

다중 선스트링은 SAMPLE_MLINES 테이블에서 저장됩니다. ST_NumLineStrings 함수는 각 다중 선스트링에 있는 개별 Geometry의 수를 판별합니다.

```
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
```

결과:

```
ID          NUM_WITHIN
----------- ----------
        110          3
        111          2
```

ST_NumPoints

ST_NumPoints는 Geometry를 입력 매개변수로 가져오며 해당 Geometry를 정의하는 데 사용된 점의 수를 리턴합니다. 예를 들어, Geometry가 다각형이고 이 다각형을 정의하는 데 다섯 개의 점이 사용된 경우 숫자 5가 리턴됩니다.

제공된 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

점의 수가 리턴되는 Geometry를 표시하는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

리턴 유형

INTEGER

예

다양한 Geometry가 테이블에 저장됩니다. ST_NumPoints 함수는 SAMPLE_GEOMETRIES 테이블의 각 Geometry에 있는 점의 수를 판별합니다.

```
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
```

결과:

```
SPATIAL_TYPE    NUM_POINTS
--------------- ----------
st_point                 1
st_linestring            3
st_polygon               4
```

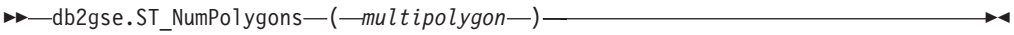
ST_NumPolygons

ST_NumPolygons는 다중 다각형을 입력 매개변수로 가져오며 포함하는 다각형의 수를 리턴합니다.

제공된 다중 다각형이 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

multipolygon
다각형의 수가 리턴되는 다중 다각형을 표시하는 ST_MultiPolygon의 값

리턴 유형

INTEGER

예

다중 다각형은 SAMPLE_MPOLYS 테이블에 저장됩니다. ST_NumPolygons 함수는 각 다중 다각형에 있는 개별적 Geometry의 수를 판별합니다.

```
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
```

결과:

```
ID          NUM_WITHIN
----------- ----------
          1          3
          2          0
          3          2
```

ST_Overlaps

ST_Overlaps는 두 개의 Geometry를 입력 매개변수로 가져오며 교차된 부분이 차원은 동일하나 제공된 Geometry와 다른 다각형을 생성할 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다.

두 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템에 표시되지 않으면, 다른 공간 참조 시스템으로 변환됩니다.

구문

매개변수

geometry1

*geometry2*와 겹치는지를 테스트할 Geometry를 나타내는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

geometry2

*geometry1*과 겹치는지를 테스트할 Geometry를 나타내는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

리턴 유형

INTEGER

예

예 1

이 예는 ST_Overlaps의 사용법을 보여줍니다. 여러 Geometry를 작성하여 SAMPLE_GEOMETRIES 테이블에 삽입합니다.

```
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) )
```

예 2

이 예는 겹치는 점의 ID를 찾습니다.

```
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
```

결과:

```
ID          ID         OVERLAP
----------- ---------- ------------------------
          1          1 Points_do_not_overlap
          2          1 Points_do_not_overlap
          2          2 Points_do_not_overlap
```

예 3

이 예는 겹치는 선의 ID를 찾습니다.

```
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
```

결과:

```
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
```

예 4

이 예는 겹치는 다각형의 ID를 찾습니다.

```
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
```

결과:

```
ID          ID         OVERLAP
----------- ---------- ------------------------
        100        100 Polygons_do_not_overlap
        110        100 Polygons_overlap
        120        100 Polygons_do_not_overlap
        110        110 Polygons_do_not_overlap
        120        110 Polygons_do_not_overlap
        120        120 Polygons_do_not_overlap
```

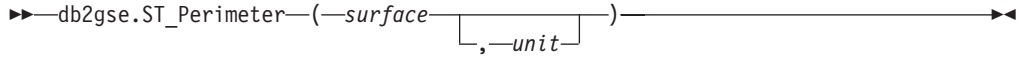
ST_Perimeter

ST_Perimeter는 표면 또는 다중 표면과 단위(선택적)를 입력 매개변수로 사용하고 디폴트 또는 제공된 단위로 측정한 바운더리의 길이인 표면 또는 다중 표면의 둘레를 리턴합니다.

제공된 표면 또는 다층이 널(NULL)이거나 비어 있는 경우, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

surface

둘레가 리턴되는 ST_Surface, ST_MultiSurface 유형 또는 해당 부속 유형 중 하나의 값

unit 둘레를 측정하는 단위를 식별하는 VARCHAR(128) 값. 지원되는 측정 단위는 DB2GSE.ST_UNITS_OF_MEASURE 카탈로그 뷰에 나열되어 있습니다.

unit 매개변수를 생략하면 다음 규칙을 사용하여 둘레를 측정한 단위를 판별합니다.

- *surface*가 투영 또는 지구 중심 좌표 시스템에 있는 경우 이 좌표 시스템과 연관된 선형 단위가 디폴트값입니다.
- *surface*가 지리 좌표 시스템에 있지만 측지 공간 참조 시스템(SRS)에는 없는 경우 이 좌표 시스템과 연관된 각도 단위가 디폴트값입니다.
- *surface*가 측지 SRS에 있는 경우, 디폴트 측정 단위는 미터입니다.

단위 변환에 대한 제한사항: 다음과 같은 조건이 발생하는 경우 오류(SQLSTATE 38SU4)가 리턴됩니다.

- Geometry가 지정되지 않은 좌표 시스템에 있고 *unit* 매개변수가 지정됩니다.
- Geometry가 투영 좌표 시스템에 있고 각도 단위가 지정됩니다.
- Geometry는 지리 좌표 시스템에 있지만 측지 SRS에는 없으며 선형 단위가 지정됩니다.
- Geometry가 지리 좌표 시스템에 있고 측지 SRS에도 있으며 각도 단위가 지정됩니다.

리턴 유형

DOUBLE

예

예 1

이 예는 ST_Perimeter 함수의 사용법을 보여줍니다. db2se에 대한 호출을 사용하여 ID가 4000인 공간 참조 시스템을 작성하고 해당 공간 참조 시스템에 다각형을 작성합니다.

```
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
```

둘레가 18인 Geometry를 보유할 SAMPLE_POLYS 테이블이 작성됩니다.

```
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))
```

예 2

이 예는 ID와 다각형의 둘레를 나열합니다.

```
SELECT id, ST_Perimeter (geometry) AS PERIMETER
  FROM sample_polys
```

결과:

```
ID          PERIMETER
----------- ------------------------
          1 +1.80000000000000E+001
```

예 3

이 예는 ID와 미터 단위로 측정된 다각형의 둘레를 나열합니다.

```
SELECT id, ST_Perimeter (geometry, 'METER') AS PERIMETER_METER
  FROM sample_polys
```

결과:

```
ID          PERIMETER_METER
----------- ------------------------
          1 +5.48641097282195E+000
```

ST_PerpPoints

ST_PerpPoints는 곡선 또는 다중 곡선 및 점을 입력 매개변수로 가져오며 곡선 또는 다중 곡선의 제공된 점에 대해 수직인 프로젝션을 리턴합니다. 제공된 점과 수직 점 간의 거리가 가장 짧은 점이 리턴됩니다. 제공된 점에서 둘 이상의 수직 프로젝트화된 점까지의 거리가 동일하면 그 점이 모두 리턴됩니다. 수직인 점을 구성할 수 없으면 비어 있는 점이 리턴됩니다.

제공된 곡선 또는 다중 곡선에 Z 또는 M 좌표가 있으면 제공된 곡선 또는 다중 곡선에 대해 보간법(interpolation)을 사용하여 점의 Z 또는 M 좌표를 계산합니다.

제공된 곡선 또는 점이 비어 있으면, 빈 점이 리턴됩니다. 제공된 곡선 또는 점이 널(NULL)인 경우 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

curve *point*에 대한 수직 프로젝션이 리턴되는 곡선 또는 다중 곡선을 표시하는 ST_Curve, ST_MultiCurve 유형 또는 해당 부속 유형 중 하나의 값

point *curve*로 수직으로 프로젝트화된 점을 표시하는 ST_Point 유형의 값

리턴 유형

db2gse.ST_MultiPoint

예

예 1

이 예는 ST_PerpPoints 함수를 사용하여 다음 테이블에 저장된 선스트링에 수직인 점을 찾는 방법을 보여줍니다. INSERT문에 ST_LineString 함수를 사용하여 선스트링을 작성합니다.

```
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) )
```

예 2

이 예에서는 좌표가 (5, 0)인 점의 선스트링에 대한 수직 프로젝션을 찾습니다. ST_AsText 함수는 리턴된 값(다중점)을 Well-Known Text 표현으로 변환하는 데 사용됩니다.

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(5, 0) ) )
  AS VARCHAR(50) ) PERP
  FROM sample_lines
```

결과:

```
PERP
----------------------------------------
MULTIPOINT ( 5.00000000 0.00000000)
```

예 3

이 예에서는 좌표가 (5, 0)인 점의 선스트링에 대한 수직 프로젝션을 찾습니다. 이 경우, 선스트링에 제공된 위치까지의 거리가 동일한 점이 세 개가 있습니다. 따라서, 세 개의 모든 점으로 구성된 다중점이 리턴됩니다.

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(5, 5) ) )
  AS VARCHAR160) ) PERP
  FROM sample_lines
```

결과:

```
PERP
--------------------------------------------------------------------------------
MULTIPOINT ( 0.00000000 5.00000000, 5.00000000 0.00000000, 10.00000000 5.00000000)
```

예 4

이 예에서는 좌표가 (5, 10)인 점의 선스트링에 대한 수직 프로젝션을 찾습니다. 이 경우, 세 개의 다른 수직 점을 찾을 수 있습니다. 그러나, ST_PerpPoints 함수는 제공된 점에 근접한 점만 리턴합니다. 따라서, 두 개의 근접한 점으로만 구성된 다중점만이 리턴됩니다. 세 번째 점은 포함되지 않습니다.

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(5, 10) ) )
  AS VARCHAR(80) ) PERP
  FROM sample_lines
```

결과:

```
PERP
------------------------------------------------------------
MULTIPOINT ( 0.00000000 10.00000000, 10.00000000 10.00000000 )
```

예 5

이 예에서는 좌표가 (5, 15)인 점의 선스트링에 대한 수직 프로젝션을 찾습니다.

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point('point(5 15)', 0 ) ) )
  AS VARCHAR(80) ) PERP
  FROM sample_lines
```

결과:

```
PERP
-----------------------------------
MULTIPOINT ( 5.00000000 0.00000000 )
```

예 6

이 예에서 좌표가 (15 15)인 지정된 점은 선스트링에 대해 수직인 프로젝션이 없습니다. 그러므로 빈 Geometry가 리턴됩니다.

```
SELECT CAST ( ST_AsText( ST_PerpPoints( line, ST_Point(15, 15) ) )
  AS VARCHAR(80) ) PERP
  FROM sample_lines
```

결과:

```
PERP
--------------------------------------------
MULTIPOINT EMPTY
```

ST_Point

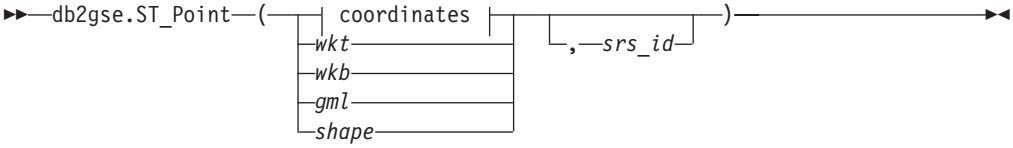
ST_Point는 다음 입력 세트 중 하나로부터 점을 구성합니다.

- X 및 Y 좌표만
- X, Y 및 Z 좌표
- X, Y, Z 및 M 좌표
- Well-Known Text 표현
- Well-Known Binary 표현
- 모양 표현
- GML(Geography Markup Language) 표현

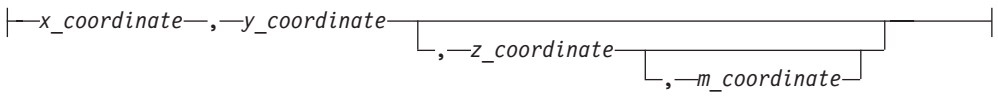
선택적 공간 참조 시스템 ID를 지정하여 결과 다중점이 있는 공간 참조 시스템을 표시할 수 있습니다.

좌표로부터 점을 구성할 때 X 또는 Y 좌표가 널(NULL)이면 예외 조건(SQLSTATE 38SUP)이 발생합니다. Z 또는 M 좌표가 널(NULL)이면 결과 점에 Z 또는 M 좌표가 없습니다. Well-Known Text 표현, Well-Known Binary 표현, 모양 표현 또는 GML 표현으로부터 점을 구성할 경우 널(NULL)이 리턴됩니다.

구문



좌표:



매개변수

wkt 결과 점의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

wkb 결과 점의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

gml GML(Geography Markup Language)을 사용하는 결과 다중점을 나타내는 CLOB(2G) 유형의 값

shape 결과 점의 모양 표현을 나타내는 BLOB(2G) 유형의 값

srs_id 결과 점에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

*srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

x_coordinate

결과 점에 대한 X 좌표를 지정하는 DOUBLE 유형의 값

y_coordinate

결과 점에 대한 Y 좌표를 지정하는 DOUBLE 유형의 값

z_coordinate

결과 점에 대한 Z 좌표를 지정하는 DOUBLE 유형의 값

z_coordinate 매개변수를 생략하면 결과 점에 Z 좌표가 없습니다. 이러한 점에 대한 ST_Is3D 결과는 0입니다.

m_coordinate

결과 점에 대한 M 좌표를 지정하는 DOUBLE 유형의 값

m_coordinate 매개변수를 생략하면 결과 점에는 측정 단위가 없습니다. 이러한 점에 대한 ST_IsMeasured 결과는 0입니다.

리턴 유형

db2gse.ST_Point

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

예 1

이 예는 ST_Point를 사용하여 점을 작성하여 삽입하는 방법을 보여줍니다. 첫 번째 점은 X 및 Y 좌표 세트를 사용하여 작성됩니다. 두 번째 점은 Well-Known Text 표현을 사용하여 작성됩니다. 두 점은 공간 참조 시스템 1에 있는 Geometry입니다.

```
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) )
```

다음 SELECT문은 테이블에 기록된 점을 리턴합니다.

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(90)) POINTS
  FROM sample_points
```

결과:

```
ID         POINTS
---------- ----------------------------------
      1110 POINT ( 10.00000000 20.00000000)
      1101 POINT ( 30.00000000 40.00000000)
```

예 2

이 예는 ID가 1103이고 X 좌표는 120, Y 좌표는 358, M 좌표는 34이며 Z 좌표는 없는 점 값을 가진 레코드를 SAMPLE_POINTS 테이블에 삽입합니다.

```
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
```

결과:

```
ID         POINTS
 ---------- -------------------------------------------------
       1103 POINT M ( 120.0000000 358.0000000 34.00000000)
```

예 3

이 예는 해당 표현에 대한 GML(Geography Markup Language)을 사용하여 ID가 1104이고 X 좌표는 1003, Y 좌표는 9876, Z 좌표는 20인 행을 공간 참조 시스템 0의 SAMPLE_POINTS 테이블에 삽입합니다.

```
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
```

결과:

```
ID         POINTS
 ---------- ----------------------------------------------------
       1104 POINT Z ( 1003.000000 9876.000000 20.00000000)
```

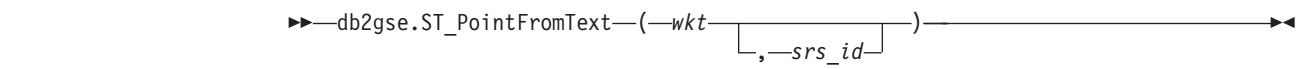
ST_PointFromText

ST_PointFromText는 점에 대한 Well-Known Text 표현과 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 점을 리턴합니다.

제공된 Well-Known Text 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

동일한 결과를 얻기 위해서는 ST_Point 함수를 사용하도록 권장합니다. 이 함수는 유연성이 뛰어나기 때문에 권장됩니다. ST_Point는 Well-Known Text 표현뿐 아니라 다른 입력 양식도 가져옵니다.

구문



매개변수

wkt 결과 점의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

srs_id 결과 점에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

*srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_Point

예

이 예는 ST_PointFromText를 사용하여 Well-Known Text 표현으로부터 점을 작성하여 삽입하는 방법을 보여줍니다. 삽입된 레코드의 ID는 1110이고 Geometry는 공간 참조 시스템 1의 다중점입니다. 점은 점에 대한 Well-Known Text 표현으로 표시됩니다. 이 Geometry의 X 및 Y 좌표는 (10, 20)입니다.

```
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) )
```

다음 SELECT문은 테이블에 기록된 다각형을 리턴합니다.

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(35) ) POINTS
  FROM sample_points
  WHERE id = 1110
```

결과:

```
ID          POINTS
----------- -----------------------------------
       1110 POINTS ( 30.00000000 40.00000000)
```

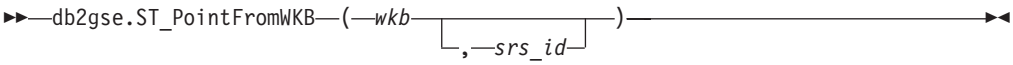
ST_PointFromWKB

ST_PointFromWKB는 점에 대한 Well-Known Binary 표현 및 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 점을 리턴합니다.

제공된 Well-Known Binary 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

동일한 결과를 얻기 위해서는 ST_Point 함수를 사용하도록 권장합니다. 이 함수는 유연성이 뛰어나기 때문에 권장됩니다. ST_Point는 Well-Known Binary 표현뿐 아니라 다른 입력 양식도 가져옵니다.

구문



매개변수

wkb 결과 점의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

srs_id 결과 점에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 내재적으로 사용됩니다.

*srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_Point

예

이 예는 ST_PointFromWKB를 사용하여 해당 Well-Known Binary 표현으로부터 점을 작성하는 방법을 보여줍니다. Geometry는 공간 참조 시스템 1에 있는 점입니다. 이 예에서는 점을 SAMPLE_POLYS 테이블의 GEOMETRY 컬럼에 저장한 다음 ST_AsBinary 함수를 사용한 Well-Known Binary 표현을 사용하여 WKB 컬럼을 갱신합니다. 마지막으로 ST_PointFromWKB 함수를 사용하여 WKB 컬럼으로부터 점을 리턴합니다.

SAMPLE_POINTS 테이블에는 점을 저장하는 GEOMETRY 컬럼과 점의 Well-Known Binary 표현을 저장하는 WKB 컬럼이 있습니다.

```
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
```

다음 SELECT문에서는 ST_PointFromWKB 함수를 사용하여 WKB 컬럼으로부터 점을 검색합니다.

```
SELECT id, CAST( ST_AsText( ST_PolyFromWKB (wkb) ) AS VARCHAR(35) ) POINTS
  FROM sample_points
```

결과:

```
ID         POINTS
---------- -----------------------------------
        10 POINT ( 44.00000000 14.00000000)
        11 POINT ( 24.00000000 13.00000000)
```

ST_PointN

ST_PointN은 입력 매개변수로 선스트링이나 다중점 및 인덱스를 가져오며 인덱스에 의해 식별되는 선스트링이나 다중점의 점을 리턴합니다. 결과 점은 제공된 선스트링이나 다중점의 공간 참조 시스템에 표시됩니다.

제공된 선스트링 또는 다중점이 널(NULL)이거나 비어 있는 경우 널(NULL)이 리턴됩니다. 인덱스가 1보다 작거나 선스트링 또는 다중점의 점 수보다 크면, 널(NULL) 값이 리턴되거나 경고 조건이 리턴됩니다(SQLSTATE 01HS2).

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

*index*에 의해 식별되는 점이 리턴되는 Geometry를 표시하는 유형 ST_LineString 또는 ST_MultiPoint의 값

index *Geometry*에서 리턴될 *n*번째 점을 식별하는 유형 INTEGER의 값

리턴 유형

db2gse.ST_Point

예

다음 예는 ST_PointN의 사용법을 보여줍니다.

```
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
```

결과:

```
ID        SECOND_INDEX
--------- ------------------------------
        1 POINT (5.00000000 5.00000000)
```

ST_PointOnSurface

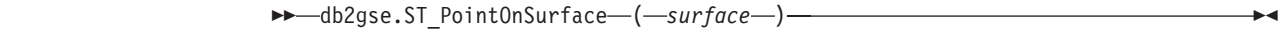
ST_PointOnSurface는 표면 또는 다중 표면을 입력 매개변수로 가져오며 표면 또는 다중 표면의 내부에 있는 점을 리턴합니다. 이 점은 표면의 paracentroid입니다.

결과 점은 제공된 표면 또는 다층의 공간 참조 시스템에 표시됩니다.

제공된 표면 또는 다층이 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

surface

표면의 점이 리턴되는 Geometry를 표시하는 ST_Surface, ST_MultiSurface 유형 또는 해당 부속유형 중 하나의 값

리턴 유형

db2gse.ST_Point

예

다음 예에서는 두 개의 다각형을 작성한 다음 ST_PointOnSurface를 사용합니다. 다각형 중 하나는 가운데에 홀(hole)이 있습니다. 리턴된 점은 다각형의 표면에 있습니다. 점이 다각형의 정 중앙에 있을 필요는 없습니다.

```
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
```

결과:

```
ID          POINT_ON_SURFACE
----------- -----------------------------------
          1 POINT ( 65.00000000 125.00000000)
          2 POINT ( 30.00000000 15.00000000)
```

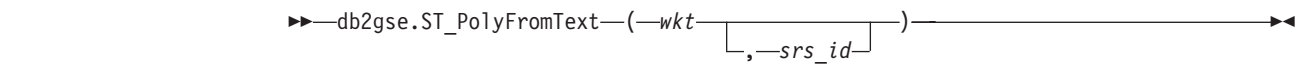
ST_PolyFromText

ST_PolyFromText는 다각형의 Well-Known Text 표현 및 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 다각형을 리턴합니다.

제공된 Well-Known Text 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

동일한 결과를 얻기 위해서는 ST_Polygon 함수를 사용하도록 권장합니다. 이 함수는 유연성이 뛰어나기 때문에 권장됩니다. ST_Polygon은 Well-Known Text 표현뿐 아니라 다른 입력 양식도 가져옵니다.

구문



매개변수

wkt 결과 다각형의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

srs_id 결과 다각형에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

*srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_Polygon

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_PolyFromText를 사용하여 Well-Known Text 표현으로부터 다각형을 작성하여 삽입하는 방법을 보여줍니다. 삽입된 레코드의 ID는 1110이고 Geometry는 공간 참조 시스템 1의 다각형입니다. 다각형은 다각형에 대한 Well-Known Text 표현으로 표시됩니다. 이 Geometry에 대한 X 및 Y 좌표는 (50, 20) (50, 40) (70, 30)입니다.

```
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) )
```

다음 SELECT문은 테이블에 기록된 다각형을 리턴합니다.

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(120) ) POLYGON
  FROM sample_polys
  WHERE id = 1110
```

결과:

```
ID         POLYGON
---------- ------------------------------------------------------------
      1110 POLYGON (( 50.00000000 20.00000000, 70.00000000 30.00000000,
                      50.00000000 40.00000000, 50.00000000 20.00000000))
```

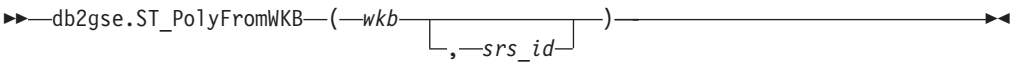
ST_PolyFromWKB

ST_PolyFromWKB는 다각형에 대한 Well-Known Binary 표현과 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져오며 해당 다각형을 리턴합니다.

제공된 Well-Known Binary 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

동일한 결과를 얻기 위해서는 ST_Polygon 함수를 사용하도록 권장합니다. 이 함수는 유연성이 뛰어나기 때문에 권장됩니다. ST_Polygon은 Well-Known Binary 표현뿐 아니라 다른 입력 양식도 가져옵니다.

구문



매개변수

wkb 결과 다각형의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

srs_id 결과 다각형에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

srs_id 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

*srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_Polygon

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_PolyFromWKB을 사용하여 해당 Well-Known Binary 표현으로부터 다각형을 작성하는 방법을 보여줍니다. Geometry는 공간 참조 시스템 1에 있는 다각형입니다. 이 예에서 SAMPLE_POLYS 테이블의 GEOMETRY 컬럼에 ID가 1115인 다각형을 저장한 다음 ST_AsBinary 함수를 사용하여 해당 Well-Known Binary 표현으로 WKB 컬럼을 갱신합니다. 마지막으로 ST_PolyFromWKB 함수를 사용하여 WKB 컬럼으로부터 다중 다각형을 리턴합니다. 이 Geometry에 대한 X 및 Y 좌표는 (50, 20) (50, 40) (70, 30)입니다.

SAMPLE_POLYS 테이블에는 다각형을 저장하는 GEOMETRY 컬럼과 다각형의 Well-Known Binary 표현을 저장하는 WKB 컬럼이 있습니다.

```
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
```

다음 SELECT문에서는 ST_PolyFromWKB 함수를 사용하여 WKB 컬럼으로부터 다각형을 검색합니다.

```
SELECT id, CAST( ST_AsText( ST_PolyFromWKB (wkb) )
  AS VARCHAR(120) ) POLYGON
  FROM sample_polys
  WHERE id = 1115
```

결과:

```
ID          POLYGON
----------- ------------------------------------------------------------
       1115 POLYGON (( 50.00000000 20.00000000, 70.00000000
               30.00000000,50.00000000 40.00000000, 50.00000000
               20.00000000))
```

ST_Polygon

ST_Polygon은 다음 입력 중 하나로부터 다각형을 구성합니다.

- 결과 다각형의 외부 링을 정의하는 닫힌 선스트링
- Well-Known Text 표현
- Well-Known Binary 표현

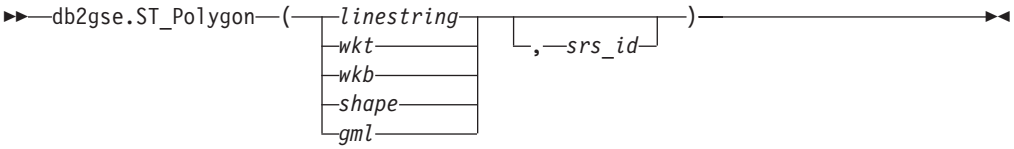
- 모양 표현
- GML(Geography Markup Language) 표현

선택적 공간 참조 시스템 ID를 지정하여 결과 다각형이 있는 공간 참조 시스템을 식별할 수 있습니다.

선스트링으로부터 다각형을 구성할 경우 제공된 선스트링이 널(NULL)이면 널(NULL)이 리턴됩니다. 제공된 선스트링이 비어 있으면, 빈 선스트링이 리턴됩니다. Well-Known Text 표현, Well-Known Binary 표현, 모양 표현 또는 GML 표현으로부터 다각형을 구성할 경우 널(NULL)이 리턴됩니다.

이 함수는 ST_Polygon(*linestring*) 및 ST_Polygon(*linestring*, *srs_id*)과 같은 경우에만 메소드로 호출할 수 있습니다.

구문



매개변수

linestring

외부 바운더리에 대한 외부 링을 정의하는 선스트링을 표시하는 ST_LineString 유형의 값. *linestring*이 닫혀 있지 않고 단순하면 예외 조건(SQLSTATE 38SSL)이 발생합니다.

wkt 결과 다각형의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

wkb 결과 다각형의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

shape 결과 다각형의 모양 표현을 표시하는 BLOB(2G) 유형의 값

gml GML(Geography Markup Language)을 사용하여 결과 다각형을 표시하는 CLOB(2G) 유형의 값

srs_id 결과 다각형에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

linestring 매개변수로부터 다각형을 구성할 때 *srs_id* 매개변수를 생략할 경우, 내재적으로 *linestring*의 공간 참조 시스템이 사용됩니다. 그렇지 않을 경우에 *srs_id* 매개변수를 생략하면 숫자 ID가 0인 공간 참조 시스템이 사용됩니다.

*srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_Polygon

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_Polygon을 사용하여 다각형을 작성하여 삽입하는 방법을 보여줍니다. 세 개의 다각형을 작성하여 삽입합니다. 모두 공간 참조 시스템 1에 있는 Geometry입니다.

- 첫 번째 다각형은 링(닫힌 단순한 선스트링)으로부터 작성됩니다. 이 다각형에 대한 X 및 Y 좌표는 (10, 20) (10, 40) (20, 30)입니다.
- 두 번째 다각형은 Well-Known Text 표현을 사용하여 작성됩니다. 이 다각형에 대한 X 및 Y 좌표는 (110, 120) (110, 140) (120, 130)입니다.
- 세 번째 다각형은 고리형 다각형입니다. 고리형 다각형은 내부 및 외부 다각형으로 구성됩니다. 이 고리형 다각형은 해당 Well-Known Text 표현으로 작성됩니다. 외부 다각형에 대한 X 및 Y 좌표는 (110, 120) (110, 140) (130, 140) (130, 120) (110, 120)입니다. 내부 다각형에 대한 X 및 Y 좌표는 (115, 125) (115, 135) (125, 135) (125, 135) (115, 125)입니다.

```
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))
```

다음 SELECT문은 테이블에 기록된 다각형을 리턴합니다.

```
SELECT id, CAST( ST_AsText( geometry ) AS VARCHAR(120) ) POLYGONS
  FROM sample_polys
```

결과:

```
ID          POLYGONS
------- ------------------------------------------------------------------
   1110 POLYGON (( 10.00000000 20.00000000, 20.00000000 30.00000000
```

```
                  10.00000000 40.00000000, 10.00000000 20.00000000))

   1101 POLYGON (( 110.00000000 120.00000000, 120.00000000 130.00000000
                   110.00000000 140.00000000, 110.00000000 120.00000000))

   1102 POLYGON (( 110.00000000 120.00000000, 130.00000000 120.00000000
                   130.00000000 140.00000000, 110.00000000 140.00000000
                   110.00000000 120.00000000),
                 ( 115.00000000 125.00000000, 115.00000000 135.00000000
                   125.00000000 135.00000000, 125.00000000 135.00000000
                   115.00000000 125.00000000))
```

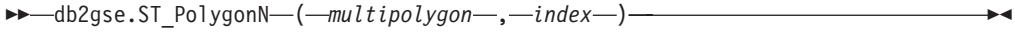
ST_PolygonN

ST_PolygonN은 입력 매개변수로 다중 다각형 및 인덱스를 가져오며 인덱스에 의해 식별되는 다각형을 리턴합니다. 결과 다각형은 제공된 다중 다각형의 공간 참조 시스템에 표시됩니다.

제공된 다중 다각형이 널(NULL) 또는 비어 있거나 인덱스가 1보다 작거나 다각형 수보다 큰 경우, 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

multipolygon

*index*에 의해 식별되는 다각형이 리턴되는 다중 다각형을 표시하는 유형 ST_MultiPolygon의 값

index *multipolygon*에서 리턴될 *n*번째 다각형을 식별하는 유형 INTEGER의 값

리턴 유형

db2gse.ST_Polygon

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_PolygonN의 사용법을 보여줍니다.

```
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
```

결과:

```
ID        SECOND_INDEX
--------- ------------------------------------------------------------
        1 POLYGON (( 8.00000000 24.00000000, 9.00000000 25.00000000,
                     1.00000000 28.00000000, 8.00000000 24.00000000))
```

ST_Relate

ST_Relate는 입력 매개변수로 두 개의 Geometry와 DE-9IM 매트릭스를 가져오며 제공된 Geometry가 매트릭스에 의해 지정된 조건을 만족할 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다.

제공된 Geometry가 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템에 표시되지 않으면, 다른 공간 참조 시스템으로 변환됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry1

*geometry2*에 대해 테스트하는 Geometry를 나타내는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

geometry2

유형 ST_Geometry 값 또는 *geometry1*에 대해 테스트된 Geometry를 표시하는 해당 부속 유형 중 하나의 값

matrix

geometry1 및 *geometry2*의 테스트에 사용될 DE-9IM(Dimensionally Extended 9 Intersection Model) 매트릭스를 표시하는 CHAR(9)의 값

리턴 유형

INTEGER

예

다음 코드는 두 개의 별도의 다각형을 작성합니다. 그런 다음, ST_Relate 함수를 사용하여 두 다각형 간의 여러 관계를 판별합니다. 예를 들어, 두 다각형이 겹치는지를 판별합니다.

```
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
```

결과:

```
Overlaps    Contains    Within      Intersects  Equals
----------- ----------- ----------- ----------- -----------
          1           0           0           1           0
```

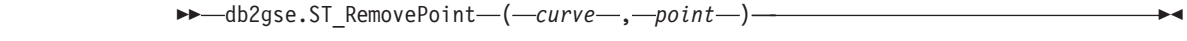
ST_RemovePoint

ST_RemovePoint는 입력 매개변수로 곡선과 점을 가져오며 여기에서 제거된 지정된 점과 동일한 모든 점이 있는 제공된 곡선을 리턴합니다. 제공된 곡선에 Z 또는 M 좌표가 있는 경우, 점에도 Z 또는 M 좌표가 있어야 합니다. 결과 Geometry는 제공된 Geometry의 공간 참조 시스템에 표시됩니다.

제공된 곡선이 비어 있으면, 빈 곡선이 리턴됩니다. 제공된 곡선이 널(NULL)이거나 제공된 점이 널(NULL)이면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

curve 유형 ST_Curve의 값 또는 *point*가 제거된 곡선을 표시하는 해당 부속 유형 중 하나의 값

point *curve*에서 제거된 점을 식별하는 유형 ST_Point의 값

리턴 유형

db2gse.ST_Curve

예

예 1

이 예에서는 두 개의 선스트링을 SAMPLE_LINES 테이블에 추가합니다. 이 선스트링은 아래의 예에 사용됩니다.

```
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))
```

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

예 2

이 예에서는 ID가 1인 선스트링에서 점 (5, 5)를 제거합니다. 이 점은 선 스트링에서 두 번 발생합니다. 따라서 두 어커런스 모두 제거합니다.

```
SELECT CAST(ST_AsText (ST_RemovePoint (line, ST_Point(5, 5) ) )
  AS VARCHAR(120) ) RESULT
  FROM sample_lines
  WHERE id = 1
```

결과:

```
RESULT
------------------------------------------------------------
LINESTRING ( 10.00000000 10.00000000, 0.00000000 0.00000000,
             10.00000000 0.00000000, 0.00000000 10.00000000)
```

예 3

이 예에서는 ID가 2인 선스트링에서 점 (5, 5, 5)를 제거합니다. 이 점은 한 번만 발생하므로 하나의 어커런스만 제거합니다.

```
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
```

결과:

```
RESULT
------------------------------------------------------------------------
LINESTRING Z ( 0.00000000 0.00000000 4.00000000, 10.00000000 10.00000000
               6.00000000,  5.00000000 5.00000000 7.00000000, 0.00000000
               0.00000000 8.00000000)
```

ST_SrsId, ST_SRID

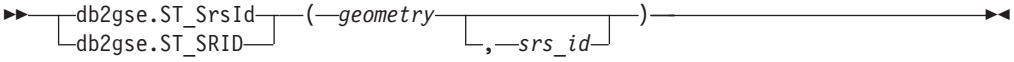
ST_SrsId(또는 ST_SRID)는 Geometry 및 공간 참조 시스템 ID(선택적)를 입력 매개변수로 가져옵니다. 리턴되는 정보는 지정된 입력 매개변수에 따라 다릅니다.

- 공간 참조 시스템 ID를 지정하면 이 함수는 지정된 공간 참조 시스템으로 변경된 공간 참조 시스템과 함께 Geometry를 리턴합니다. Geometry는 변환되지 않습니다.
- 입력 매개변수로 공간 참조 시스템 ID를 제공하지 않은 경우 제공된 Geometry의 현재 공간 참조 시스템 ID가 리턴됩니다.

제공된 Geometry가 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수도 있습니다.

구문



매개변수

geometry
공간 참조 시스템 ID를 설정하거나 리턴할 Geometry를 나타내는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

srs_id 결과 Geometry에 사용되는 공간 참조 시스템을 식별하는 INTEGER 유형의 값

주의: 이 매개변수를 지정하면 Geometry는 변환되지 않으나 지정된 공간 참조 시스템으로 변경된 해당 공간참조 시스템과 함께 리턴됩니다. 새 공간 참조 시스템으로 변경함으로써 데이터가 손상될 수 있습니다. 변환하려면 대신 ST_Transform을 사용하십시오.

*srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

- INTEGER, srs_id면 지정되지 않습니다.
- db2gse.ST_Geometry, srs_id면 지정됩니다.

예

두 점이 두 개의 다른 공간 참조 시스템에 작성됩니다. ST_SrsId 함수를 사용하여 각 점에 연관된 공간 참조 시스템의 ID를 찾을 수 있습니다.

```
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
```

결과:

```
ID          SRSID
----------- ------------
          1            0
          2            1
```

ST_SrsName

ST_SrsName은 입력 매개변수로 Geometry를 가져오며 제공된 Geometry가 표시되는 공간 참조 시스템의 이름을 리턴합니다.

제공된 Geometry가 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 값 또는 공간 참조 시스템 이름이 리턴되는 Geometry를 표시하는 해당 부속 유형 중 하나의 값

리턴 유형

VARCHAR(128)

예

두 점이 두 개의 다른 공간 참조 시스템에 작성됩니다. ST_SrsName 함수를 사용하여 각 점에 연관된 공간 참조 시스템의 이름을 찾을 수 있습니다.

```
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
```

결과:

```
ID          SRSNAME
----------- ------------------------------
          1 DEFAULT_SRS
          2 NAD83_SRS_1
```

ST_StartPoint

ST_StartPoint는 입력 매개변수로 곡선을 가져오며 곡선의 시작 점을 리턴합니다. 결과 점은 제공된 곡선의 공간 참조 시스템에 표시됩니다. 이 결과는 ST_PointN(*curve*, 1) 함수 호출에 대한 결과와 동등합니다.

제공된 곡선이 널(NULL) 값이거나 비어 있으면, 널(NULL) 값이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

curve 첫 번째 점이 리턴되는 Geometry를 표시하는 ST_Curve 유형 또는 해당 부속 유형 중 하나의 값

리턴 유형

db2gse.ST_Point

예

이 예에서는 두 개의 선스트링을 SAMPLE_LINES 테이블에 추가합니다. 첫 번째 것은 X 및 Y 좌표를 갖고 있는 선스트링입니다. 두 번째 것은 X, Y 및 Z 좌표가 있는 선스트링입니다. ST_StartPoint 함수를 사용하여 각 선스트링의 첫 번째 점을 리턴합니다.

```
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
```

결과:

```
ID          START_POINT
----------- ----------------------------------------
          1 POINT ( 10.00000000 10.00000000)
          2 POINT Z ( 0.00000000 0.00000000 4.00000000)
```

ST_SymDifference

ST_SymDifference는 두 개의 Geometry를 입력 매개변수로 사용하고 두 개 Geometry의 대칭적 차이인 Geometry를 리턴합니다. 대칭적 차이는 제공된 두 Geometry의 교차하지 않는 부분입니다. 결과 Geometry는 첫 번째 Geometry의 공간 참조 시스템에 표시됩니다. 리턴된 Geometry의 차원은 입력 Geometry의 차원과 동일합니다. 두 Geometry의 차원은 동일해야 합니다.

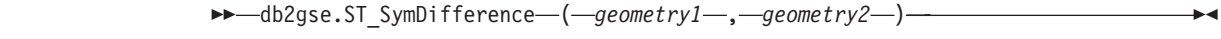
비측지 데이터의 경우 두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템으로 표현되지 않으면 다른 공간 참조 시스템으로 변환됩니다. 측지 데이터의 경우, 두 Geometry는 모두 동일한 측지 공간 참조 시스템(SRS) 내에 있어야 합니다.

Geometry가 동일한 경우, ST_Point 유형의 빈 Geometry가 리턴됩니다. Geometry중 하나가 널(NULL)이면 널(NULL)이 리턴됩니다.

결과 Geometry는 가장 적절한 공간 유형으로 표시됩니다. 점, 선스트링 또는 다각형으로 표시될 수 있는 경우, 이 유형 중 하나가 사용됩니다. 그렇지 않으면 다중점, 다중 선스트링 또는 다중 다각형 유형이 사용됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry1

*geometry2*와의 대칭적 차이를 계산할 첫 번째 Geometry를 표시하는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

geometry2

*geometry1*과의 대칭적 차이를 계산할 두 번째 Geometry를 표시하는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

측지 데이터의 제한사항:

- 두 Geometry는 모두 측지 Geometry로서 동일한 측지 SRS 내에 있어야 합니다.
- ST_SymDifference는 ST_Point, ST_Polygon, ST_MultiPoint 및 ST_MultiPolygon 데이터 유형만 지원합니다.

리턴 유형

db2gse.ST_Geometry

예

예 1

이 예는 ST_SymDifference 함수의 사용법을 보여줍니다. Geometry는 SAMPLE_GEOMS 테이블에서 저장됩니다.

```
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));
```

다음 예에서는 읽기 쉽도록 결과를 다시 형식화했습니다. 결과는 표시 장치에 따라 다릅니다.

예 2

이 예에서는 ST_SymDifference를 사용하여 SAMPLE_GEOMS 테이블에 있는 두 개의 다른 다각형의 대칭적 차이를 리턴합니다.

```
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
```

결과:

```
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)))
```

예 3

이 예에서는 ST_SymDifference를 사용하여 SAMPLE_GEOMS 테이블의 두 개의 교차하는 다각형에 대한 대칭적 차이를 리턴합니다.

```
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
```

결과:

```
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)))
```

예 4

이 예에서는 ST_SymDifference를 사용하여 SAMPLE_GEOMS 테이블의 두 개의 교차하는 선스트링에 대한 대칭적 차이를 리턴합니다.

```
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
```

결과:

```
ID   ID  SYM_DIFF
---- --- ------------------------------------------------------------------
   4   5 MULTILINESTRING (( 70.00000000 70.00000000, 75.00000000 75.00000000),
                ( 80.00000000 80.00000000, 90.00000000 90.00000000))
```

ST_ToGeomColl

ST_ToGeomColl은 입력 매개변수로 Geometry를 가져오며 이를 Geometry 콜렉션으로 변환합니다. 결과 Geometry 콜렉션은 제공된 Geometry의 공간 참조 시스템에 표시됩니다.

지정된 Geometry가 비어 있는 경우 어떤 유형이든지 사용할 수 있습니다. 그러나 ST_Multipoint, ST_MultiLineString 또는 ST_MultiPolygon 유형으로 적절하게 변환됩니다.

지정된 Geometry가 비어 있지 않는 경우 ST_Point, ST_LineString 또는 ST_Polygon 유형 중 하나이어야 합니다. 이는 각각 ST_Multipoint, ST_MultiLineString 또는 ST_MultiPolygon으로 변환됩니다.

제공된 Geometry가 널(NULL)이면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 값 또는 Geometry 콜렉션으로 변환된 Geometry를 표시하는 해당 부속 유형 중 하나의 값

리턴 유형

db2gse.ST_GeomCollection

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_ToGeomColl 함수의 사용법을 보여줍니다.

```
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))
```

다음 SELECT문에서는 ST_ToGeomColl 함수를 사용하여 Geometry를 해당 Geometry 콜렉션 부속 유형으로 리턴합니다.

```
SELECT id, CAST( ST_AsText( ST_ToGeomColl(geometry) )
    AS VARCHAR(120) ) GEOM_COLL
FROM sample_geometries
```

결과:

```
ID          GEOM_COLL
----------- ----------------------------------------------------
          1 MULTIPOLYGON ((( 3.00000000 3.00000000, 5.00000000
                             3.00000000, 4.00000000 6.00000000,
                             3.00000000 3.00000000)))
          2 MULTIPOINT ( 1.00000000 2.00000000)
```

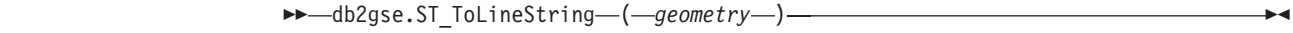
ST_ToLineString

ST_ToLineString은 입력 매개변수로 Geometry를 가져오며 이를 선스트링으로 변환합니다. 결과 선스트링은 제공된 Geometry의 공간 참조 시스템에 표시됩니다.

제공된 Geometry는 비어 있거나 선스트링이어야 합니다. 제공된 Geometry가 널(NULL)이면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

유형 ST_Geometry 값 또는 선스트링으로 변환된 Geometry를 표시하는 해당 부속 유형 중 하나의 값

Geometry는 비어 있거나 선스트링인 경우 선스트링으로 변환될 수 있습니다. 변환을 수행할 수 없으면 예외 조건(SQLSTATE 38SUD)이 발생합니다.

리턴 유형

db2gse.ST_LineString

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_ToLineString 함수의 사용법을 보여줍니다.

```
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) )
```

다음 SELECT문에서는 ST_ToLineString 함수를 사용하여 정적 유형의 ST_Geometry에서 ST_LineString으로 변환된 선스트링을 리턴합니다.

```
SELECT CAST( ST_AsText( ST_ToLineString(geometry) )
     AS VARCHAR(130) ) LINES
FROM sample_geometries
```

결과:

```
LINES
------------------------------------------------------------------
LINESTRING Z ( 0.00000000 10.00000000 1.00000000, 0.00000000
               0.00000000 3.00000000, 10.00000000 0.00000000
               5.00000000)
LINESTRING EMPTY
LINESTRING EMPTY
```

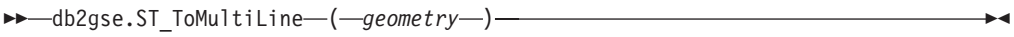
ST_ToMultiLine

ST_ToMultiLine은 입력 매개변수로 Geometry를 가져오며 이를 다중 선스트링으로 변환합니다. 결과 다중 선스트링은 제공된 Geometry의 공간 참조 시스템에 표시됩니다.

제공된 Geometry는 비어 있거나 다중 선스트링 또는 선스트링이어야 합니다. 제공된 Geometry가 널(NULL)이면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

유형 ST_Geometry 값 또는 다중 선스트링으로 변환된 Geometry를 표시하는 해당 부속 유형 중 하나의 값

Geometry는 비어 있거나 선스트링이거나 다중 선스트링인 경우 다중 선스트링으로 변환될 수 있습니다. 변환을 수행할 수 없으면 예외 조건(SQLSTATE 38SUD)이 발생합니다.

리턴 유형

db2gse.ST_MultiLineString

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_ToMultiLine 함수의 사용법을 보여줍니다.

```
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) )
```

다음 SELECT문에서는 ST_ToMultiLine 함수를 사용하여 정적 유형의 ST_Geometry에서 ST_MultiLineString으로 변환된 다중 선스트링을 리턴합니다.

```
SELECT CAST( ST_AsText( ST_ToMultiLine(geometry) )
  AS VARCHAR(130) ) LINES
  FROM sample_geometries
```

결과:

```
LINES
-----------------------------------------------------
MULTILINESTRING Z ( 0.00000000 10.00000000 1.00000000,
                    0.00000000 0.00000000 3.00000000,
                   10.00000000 0.00000000 5.00000000)
MULTILINESTRING EMPTY
MULTILINESTRING EMPTY
```

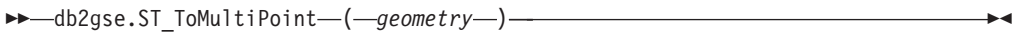
ST_ToMultiPoint

ST_ToMultiPoint는 입력 매개변수로 Geometry를 가져오며 이를 다중점으로 변환합니다. 결과 다중점은 제공된 Geometry의 공간 참조 시스템에 표시됩니다.

제공된 Geometry는 비어 있거나 점 또는 다중점이어야 합니다. 제공된 Geometry가 널(NULL)이면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

유형 ST_Geometry 값 또는 다중점으로 변환된 Geometry를 표시하는 해당 부속 유형 중 하나의 값

Geometry는 비어 있거나 다중점인 경우 다중점으로 변환될 수 있습니다. 변환을 수행할 수 없으면 예외 조건(SQLSTATE 38SUD)이 발생합니다.

리턴 유형

db2gse.ST_MultiPoint

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_ToMultiPoint 함수의 사용법을 보여줍니다.

```
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) )
```

다음 SELECT문에서는 ST_ToMultiPoint 함수를 사용하여 정적 유형의 ST_Geometry에서 ST_MultiPoint로 변환된 다중점을 리턴합니다.

```
SELECT CAST( ST_AsText( ST_ToMultiPoint(geometry))
  AS VARCHAR(62) ) MULTIPOINTS
FROM sample_geometries
```

결과:

```
MULTIPOINTS
------------------------------------------------------------
MULTIPOINT ( 0.00000000 0.00000000, 0.00000000 4.00000000)
MULTIPOINT ( 30.00000000 40.00000000)
MULTIPOINT EMPTY
```

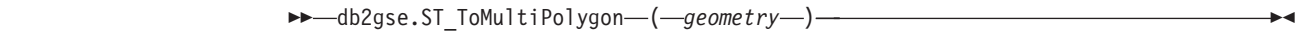
ST_ToMultiPolygon

ST_ToMultiPolygon은 입력 매개변수로 Geometry를 가져오며 이를 다중 다각형으로 변환합니다. 결과 다중 다각형은 제공된 Geometry의 공간 참조 시스템에 표시됩니다.

제공된 Geometry는 비어 있거나 다각형 또는 다중 다각형이어야 합니다. 제공된 Geometry가 널(NULL)이면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

유형 ST_Geometry 값 또는 다중 다각형으로 변환된 Geometry를 표시하는 해당 부속 유형 중 하나의 값

Geometry는 비어 있거나 다중 다각형인 경우 다중 다각형으로 변환될 수 있습니다. 변환을 수행할 수 없으면 예외 조건(SQLSTATE 38SUD)이 발생합니다.

리턴 유형

db2gse.ST_MultiPolygon

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 여러 개의 Geometry를 작성한 다음 ST_ToMultiPolygon을 사용하여 다중 다각형을 리턴합니다.

```
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))
```

다음 SELECT문에서는 ST_ToMultiPolygon 함수를 사용하여 정적 유형의 ST_Geometry에서 ST_MultiPolygon으로 변환된 다중 다각형을 리턴합니다.

```
SELECT CAST( ST_AsText( ST_ToMultiPolygon(geometry) )
  AS VARCHAR(130) ) POLYGONS
  FROM sample_geometries
```

결과:

```
POLYGONS
------------------------------------------------------------------------
MULTIPOLYGON (( 0.00000000 0.00000000, 5.00000000 0.00000000,
                5.00000000 4.00000000, 0.00000000 4.00000000,
                 0.00000000 0.00000000))
```

```
MULTIPOLYGON EMPTY

MULTIPOLYGON EMPTY
```

ST_ToPoint

ST_ToPoint는 입력 매개변수로 Geometry를 가져오며 이를 점으로 변환합니다. 결과 점은 제공된 Geometry의 공간 참조 시스템에 표시됩니다.

제공된 Geometry는 비어 있거나 점이어야 합니다. 제공된 Geometry가 널(NULL)이면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

유형 ST_Geometry 값 또는 점으로 변환된 Geometry를 표시하는 해당 부속 유형 중 하나의 값

Geometry는 비어 있거나 점인 경우 점으로 변환될 수 있습니다. 변환을 수행할 수 없으면 예외 조건(SQLSTATE 38SUD)이 발생합니다.

리턴 유형

db2gse.ST_Point

예

이 예는 SAMPLE_GEOMETRIES에 세 개의 Geometry를 작성한 후 각각을 점으로 변환합니다.

```
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) )
```

다음 SELECT문에서는 ST_ToPoint 함수를 사용하여 정적 유형의 ST_Geometry에서 ST_Point로 변환된 점을 리턴합니다.

```
SELECT CAST( ST_AsText( ST_ToPoint(geometry) ) AS VARCHAR(35) ) POINTS
  FROM sample_geometries
```

결과:

```
POINTS
--------------------------------
POINT ( 30.00000000 40.00000000)
POINT EMPTY
POINT EMPTY
```

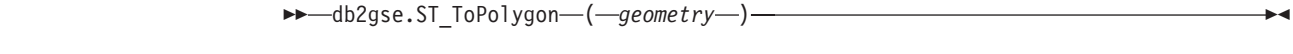
ST_ToPolygon

ST_ToPolygon은 입력 매개변수로 Geometry를 가져오며 이를 다각형으로 변환합니다. 결과 다각형은 제공된 Geometry의 공간 참조 시스템에 표시됩니다.

제공된 Geometry는 비어 있거나 다각형이어야 합니다. 제공된 Geometry가 널(NULL)이면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry

유형 ST_Geometry 값 또는 다각형으로 변환된 Geometry를 표시하는 해당 부속 유형 중 하나의 값

Geometry는 비어 있거나 다각형인 경우 다각형으로 변환될 수 있습니다. 변환을 수행할 수 없으면 예외 조건(SQLSTATE 38SUD)이 발생합니다.

리턴 유형

db2gse.ST_Polygon

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예에서는 SAMPLE_GEOMETRIES에 세 개의 Geometry를 작성하여 각각을 다각형으로 변환합니다.

```
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) )
```

다음 SELECT문에서는 ST_ToPolygon 함수를 사용하여 정적 유형의 ST_Geometry로부터 ST_Polygon으로 변환된 다각형을 리턴합니다.

```
SELECT CAST( ST_AsText( ST_ToPolygon(geometry) ) AS VARCHAR(130) ) POLYGONS
  FROM sample_geometries
```

결과:

```
POLYGONS
--------------------------------------------------------------------------------
POLYGON (( 0.00000000 0.00000000, 5.00000000 0.00000000,
           5.00000000 4.00000000,0.00000000 4.00000000,
            0.00000000 0.00000000))

POLYGON EMPTY

POLYGON EMPTY
```

ST_Touches

ST_Touches는 두 개의 Geometry를 입력 매개변수로 가져오며 제공된 Geometry가 공간적으로 인접할 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다.

두 Geometry의 내부가 교차하지는 않으나 Geometry 중 하나의 바운더리가 다른 Geometry의 바운더리 또는 내부와 교차할 경우 두 Geometry는 인접합니다.

두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템에 표시되지 않으면, 다른 공간 참조 시스템으로 변환됩니다.

제공된 두 Geometry가 모두 점 또는 다중점이거나 제공된 Geometry 중 하나가 널(NULL)이거나 비어 있으면 널(NULL)이 리턴됩니다.

구문

매개변수

geometry1

*geometry2*와 인접하는지를 테스트할 Geometry를 나타내는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

geometry2

*geometry1*과 인접하는지를 테스트할 Geometry를 나타내는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

리턴 유형

INTEGER

예

여러 Geometry를 SAMPLE_GEOMS 테이블에 추가합니다. 그런 다음 ST_Touches 함수를 사용하여 서로 인접하는 Geometry를 판별합니다.

```
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
```

결과:

```
ID          ID          TOUCHES
----------- ----------- -----------
          1           1           0
          1           2           1
          1           3           0
          1           4           0
          1           5           1
          2           2           0
          2           3           0
          2           4           0
          2           5           1
          3           3           0
          3           4           1
          3           5           1
          4           4           0
          4           5           1
          5           5           0
```

ST_Transform

ST_Transform은 Geometry 및 공간 참조 시스템 ID를 입력 매개변수로 가져오며 제공된 공간 참조 시스템에 표시할 수 있도록 Geometry를 변환합니다. 다른 좌표 시스템 간의 프로젝션 및 변환이 수행되고 Geometry의 좌표가 그에 맞게 조정됩니다.

Geometry의 현재 공간 참조 시스템이 지정된 공간 참조 시스템과 동일한 지리적 좌표 시스템을 기본으로 할 경우에만 지정된 공간 참조 시스템으로 변환할 수 있습니다.

Geometry의 현재 공간 참조 시스템 또는 지정된 공간 참조 시스템이 프로젝트화 좌표 시스템을 기본으로 할 경우, 역 프로젝션을 수행하여 프로젝트화 좌표 시스템을 기본으로 하는 지리적 좌표 시스템을 판별합니다.

제공된 Geometry가 널(NULL)이면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문

매개변수

geometry

*srs_id*로 식별되는 공간 참조 시스템으로 변환할 Geometry를 표시하는 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

srs_id 결과 Geometry에 대한 공간 참조 시스템을 식별하는 INTEGER 유형의 값

*geometry*의 현재 공간 참조 시스템이 *srs_id*로 식별되는 공간 참조 시스템과 호환 가능하지 않아 지정된 공간 참조 시스템으로 변환을 수행할 수 없는 경우 예외 조건(SQLSTATE 38SUC)이 발생합니다.

*srs_id*가 DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS 카탈로그 뷰에 나열된 공간 참조 시스템을 식별하지 못하면 예외 조건(SQLSTATE 38SU1)이 발생합니다.

리턴 유형

db2gse.ST_Geometry

예

예 1

다음 예는 ST_Transform을 사용하여 한 공간 참조 시스템에서 다른 공간 참조 시스템으로 Geometry를 변환하는 방법을 보여줍니다.

먼저, db2se에 대한 호출을 사용하여 ID가 3인 state plane 공간 참조 시스템을 작성합니다.

```
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
```

점을 추가합니다.

- 해당 공간 참조 시스템을 사용하는 state plane 좌표의 SAMPLE_POINTS_SP 테이블

• 경도 및 위도에 지정된 좌표를 사용하는 SAMPLE_POINTS_LL 테이블

```
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) )
```

ST_Transform 함수를 사용하여 Geometry를 변환합니다.

예 2

이 예에서는 경도 및 위도에 있는 점을 state plane 좌표로 변환합니다.

```
SELECT id, CAST( ST_AsText( ST_Transform( geometry, 3) )
    AS VARCHAR(100) ) STATE_PLANE
  FROM sample_points_ll
```

결과:

```
ID          STATE_PLANE
----------- ----------------------------------------
      12457 POINT ( 567176.00000000 1166411.00000000)
      12477 POINT ( 637948.00000000 1177640.00000000)
```

예 3

이 예에서는 state plane 좌표에 있는 점을 경도 및 위도 좌표로 변환합니다.

```
SELECT id, CAST( ST_AsText( ST_Transform( geometry, 1) )
    AS VARCHAR(100) ) LAT_LONG
  FROM sample_points_sp
```

결과:

```
ID          LAT_LONG
----------- ------------------------------------
      12457 POINT ( -74.22371500 42.03498800)
      12477 POINT ( -73.96293100 42.06488000)
```

ST_Union

ST_Union는 입력 매개변수로 두 개의 Geometry를 가져오며 제공된 Geometry의 합쳐진 Geometry를 리턴합니다. 결과 Geometry는 첫 번째 Geometry의 공간 참조 시스템에 표시됩니다.

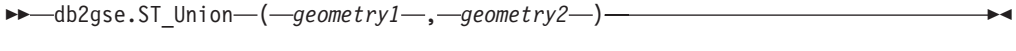
두 Geometry의 차원은 동일해야 합니다. 제공된 두 Geometry 중 하나가 널(NULL)인 경우 널(NULL)이 리턴됩니다.

비측지 데이터의 경우 두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템으로 표현되지 않으면 다른 공간 참조 시스템으로 변환됩니다. 측지 데이터의 경우, 두 Geometry는 모두 동일한 측지 공간 참조 시스템(SRS) 내에 있어야 합니다.

결과 Geometry는 가장 적절한 공간 유형으로 표시됩니다. 점, 선스트링 또는 다각형으로 표시될 수 있는 경우, 이 유형 중 하나가 사용됩니다. 그렇지 않으면 다중점, 다중 선스트링 또는 다중 다각형 유형이 사용됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

geometry1

*geometry2*와 조합된 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

geometry2

*geometry1*과 조합된 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

측지 데이터의 제한사항: 두 Geometry는 모두 측지 Geometry로서 동일한 측지 SRS 내에 있어야 합니다.

리턴 유형

db2gse.ST_Geometry

예

예 1

다음 SQL문은 SAMPLE_GEOMS 테이블을 작성하여 데이터를 채웁니다.

```
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))
```

다음 예에서는 읽기 쉽도록 결과를 다시 형식화했습니다. 결과는 표시 장치에 따라 다릅니다.

예 2

이 예는 두 개의 분리된 다각형을 통합한 다각형을 찾습니다.

```
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
```

결과:

```
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)))
```

예 3

이 예는 두 개의 교차하는 다각형을 통합한 다각형을 찾습니다.

```
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
```

결과:

```
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))
```

예 4

두 선스트링을 통합한 선스트링을 찾습니다.

```
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
```

결과:

```
ID    ID    UNION
----- ----- ------------------------------------------------------------------
    4     5 MULTILINESTRING (( 70.00000000 70.00000000, 80.00000000 80.00000000),
              ( 80.00000000 80.00000000, 100.00000000 70.00000000))
```

ST_Within

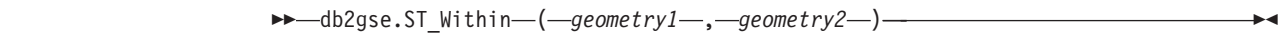
ST_Within은 두 개의 Geometry를 입력 매개변수로 가져오며 첫 번째 Geometry가 완전히 두 번째 Geometry에 포함될 경우 1을 리턴합니다. 그렇지 않으면 0이 리턴됩니다.

제공된 Geometry가 널(NULL)이거나 비어 있는 경우, 널(NULL)이 리턴됩니다.

비측지 데이터의 경우 두 번째 Geometry가 첫 번째 Geometry와 동일한 공간 참조 시스템으로 표현되지 않으면 다른 공간 참조 시스템으로 변환됩니다. 측지 데이터의 경우, 두 Geometry는 모두 동일한 측지 공간 참조 시스템(SRS) 내에 있어야 합니다.

ST_Within은 매개변수를 반대로 사용하여 ST_Contains가 수행하는 것과 동일한 조작을 수행합니다.

구문



매개변수

geometry1

geometry2 내에 완전히 들어 있는지를 테스트할 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

geometry2

geometry1 내에 완전히 들어 있는지를 테스트할 ST_Geometry 유형 또는 해당 부속 유형 중 하나의 값

측지 데이터의 제한사항: 두 Geometry는 모두 측지 Geometry로서 동일한 측지 SRS 내에 있어야 합니다.

리턴 유형

INTEGER

예

예 1

이 예에서는 ST_Within 함수의 사용법을 보여줍니다. Geometry를 작성하여 세 개의 테이블(SAMPLE_POINTS, SAMPLE_LINES 및 SAMPLE_POLYGONS)에 삽입합니다.

```
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) )
```

예 2

이 예에서는 SAMPLE_POLYGONS 테이블의 다각형에 있는 SAMPLE_POINTS 테이블에서 점을 찾습니다.

```
SELECT a.id POINT_ID_WITHIN_POLYGONS
  FROM sample_points a, sample_polygons b
  WHERE ST_Within( b.geometry, a.geometry) = 0
```

결과:

```
POINT_ID_WITHIN_POLYGONS
----------------------
                     2
```

예 3

이 예에서는 SAMPLE_POLYGONS 테이블의 다각형에 있는 SAMPLE_LINES 테이블에서 선스트링을 찾습니다.

```
SELECT a.id LINE_ID_WITHIN_POLYGONS
  FROM sample_lines a, sample_polygons b
  WHERE ST_Within( b.geometry, a.geometry) = 0
```

결과:

```
LINE_ID_WITHIN_POLYGONS
----------------------
                     1
```

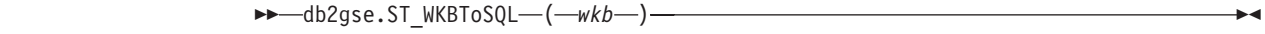
ST_WKBToSQL

ST_WKBToSQL은 Geometry의 well-known binary 표현을 사용하며 해당 Geometry를 리턴합니다. ID가 0인 공간 참조 시스템은 결과 Geometry에 사용됩니다.

제공된 Well-Known Binary 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

ST_WKBToSQL(*wkb*)은 ST_Geometry(*wkb*,0)와 동일한 결과를 제공합니다. 유연성 때문에 ST_WKBToSQL보다는 ST_Geometry 함수를 사용하도록 권장합니다. ST_Geometry는 Well-Known Binary 표현뿐 아니라 다른 입력 양식도 가져옵니다.

구문



매개변수

wkb 결과 Geometry의 Well-Known Binary 표현이 들어 있는 유형 BLOB(2G) 값

리턴 유형

db2gse.ST_Geometry

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_WKBToSQL 함수의 사용법을 보여줍니다. 먼저 Geometry를 SAMPLE_GEOMETRIES 테이블의 GEOMETRY 컬럼에 저장합니다. 그런 다음, UPDATE 문의 ST_AsBinary 함수를 사용하여 Well-Known Binary 표현을 WKB 컬럼에 저장합니다. 마지막으로, ST_WKBToSQL 함수를 사용하여 WKB 컬럼의 Geometry 좌표를 리턴합니다.

```
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
```

이 SELECT문을 사용하여 WKB 컬럼의 Geometry를 보십시오.

```
SELECT id, CAST( ST_AsText( ST_WKBToSQL(wkb) ) AS VARCHAR(120) ) GEOMETRIES
  FROM sample_geometries
```

결과:

```
ID          GEOMETRIES
----------- ------------------------------------------------------------
         10 POINT ( 44.00000000 14.00000000)
         11 POINT ( 24.00000000 13.00000000)
         12 POLYGON (( 50.00000000 20.00000000, 70.00000000 30.00000000,
                       50.00000000 40.00000000, 50.00000000 20.00000000))
```

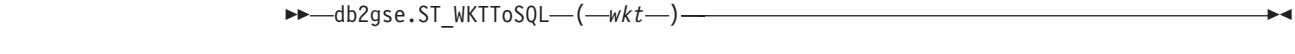
ST_WKTToSQL

ST_WKTToSQL은 Geometry의 Well-Known Text 표현을 사용하며 해당 Geometry를 리턴합니다. ID가 0인 공간 참조 시스템은 결과 Geometry에 사용됩니다.

제공된 Well-Known Text 표현이 널(NULL) 값이면, 널(NULL) 값이 리턴됩니다.

ST_WKTToSQL(*wkt*)은 ST_Geometry(*wkt*,0)와 동일한 결과를 제공합니다. 유연성 때문에 ST_WKTToSQL보다는 ST_Geometry 함수를 사용하도록 권장합니다. ST_Geometry는 Well-Known Text 표현뿐 아니라 다른 입력 양식도 가져옵니다.

구문



매개변수

wkt 결과 Geometry의 Well-Known Text 표현이 들어 있는 유형 CLOB(2G) 값

리턴 유형

db2gse.ST_Geometry

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 ST_WKTToSQL이 Well-Known Text 표현을 사용하여 Geometry를 작성한 다음 삽입하는 방법을 보여줍니다.

```
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))' ) )
```

이 SELECT문은 삽입한 Geometry를 리턴합니다.

```
SELECT id, CAST( ST_AsText(geometry) AS VARCHAR(120) ) GEOMETRIES
  FROM sample_geometries
```

결과:

```
ID          GEOMETRIES
----------- ------------------------------------------------------------
         10 POINT ( 44.00000000 14.00000000)
         11 POINT ( 24.00000000 13.00000000)
         12 POLYGON (( 50.00000000 20.00000000, 70.00000000 30.00000000,
                       50.00000000 40.00000000, 50.00000000 20.00000000))
```

ST_X

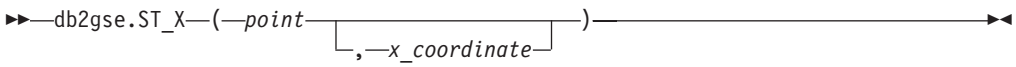
ST_X는 다음을 가져옵니다.

- 점을 입력 매개변수로 가져오며 해당 X 좌표를 리턴합니다.
- 점 및 X 좌표를 가져오며 해당 X 좌표가 제공된 값으로 설정된 점 자체를 리턴합니다.

점이 널(NULL)이거나 비어 있으면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

point X 좌표를 리턴하거나 수정할 ST_Point 유형의 값

x_coordinate

*point*에 대한 새로운 X 좌표를 표시하는 DOUBLE 유형의 값

리턴 유형

- DOUBLE, *x_coordinate*면 지정되지 않습니다.
- db2gse.ST_Point, *x_coordinate*면 지정됩니다.

예

예 1

이 예에서는 ST_X 함수의 사용법을 보여줍니다. Geometry를 작성하여 SAMPLE_POINTS 테이블에 삽입합니다.

```
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) )
```

예 2

이 예는 테이블에 있는 점의 X 좌표를 찾습니다.

```
SELECT id, ST_X (geometry) X_COORD
  FROM sample_points
```

결과:

```
ID          X_COORD
---------- ----------------------
         1 +2.00000000000000E+000
         2 +4.00000000000000E+000
         3 +3.00000000000000E+000
```

예 3

이 예는 해당 X 좌표가 40으로 설정된 점을 리턴합니다.

```
SELECT id, CAST( ST_AsText( ST_X (geometry, 40)) AS VARCHAR(60) )
  X_40
  FROM sample_points
  WHERE id=3
```

결과:

```
ID          X_40
------ ------------------------------------------------------------
     3 POINT ZM ( 40.00000000 8.00000000 23.00000000 7.00000000)
```

ST_Y

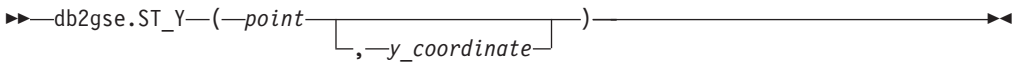
ST_Y는 다음을 가져옵니다.

- 점을 입력 매개변수로 가져오며 해당 Y 좌표를 리턴합니다.
- 점 및 Y 좌표를 가져오며 해당 Y 좌표가 제공된 값으로 설정된 점 자체를 리턴합니다.

점이 널(NULL)이거나 비어 있으면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

point Y 좌표를 리턴하거나 수정할 ST_Point 유형의 값

y_coordinate

*point*에 대한 새로운 Y 좌표를 표시하는 DOUBLE 유형의 값

리턴 유형

- DOUBLE, *y_coordinate*면 지정되지 않습니다.
- db2gse.ST_Point, *y_coordinate*면 지정됩니다.

예

예 1

이 예는 ST_Y 함수의 사용법을 보여줍니다. Geometry를 작성하여 SAMPLE_POINTS 테이블에 삽입합니다.

```
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) )
```

예 2

이 예는 테이블에 있는 점의 Y 좌표를 찾습니다.

```
SELECT id, ST_Y (geometry) Y_COORD
  FROM sample_points
```

결과:

```
ID Y_COORD
---------- ----------------------
         1 +3.00000000000000E+000
         2 +5.00000000000000E+000
         3 +8.00000000000000E+000
```

예 3

이 예는 해당 Y 좌표가 40으로 설정된 점을 리턴합니다.

```
SELECT id, CAST( ST_AsText( ST_Y (geometry, 40)) AS VARCHAR(60) )
  Y_40
  FROM sample_points
  WHERE id=3
```

결과:

```
ID          Y_40
--------- ----------------------------------------------------------
     3 POINT ZM ( 3.00000000 40.00000000 23.00000000 7.00000000)
```

ST_Z

ST_Y는 다음을 가져옵니다.

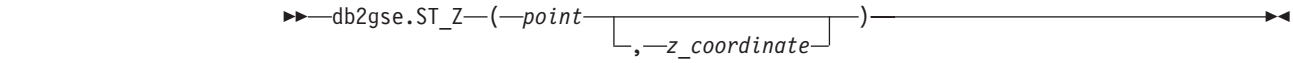
- 점을 입력 매개변수로 가져오며 해당 Z 좌표를 리턴합니다.
- 지정된 점에 기존의 Z 좌표가 없더라도 점과 Z 좌표를 가져오며 Z 좌표가 제공된 값으로 설정된 점 자체를 리턴합니다.

지정된 Z 좌표가 널(NULL)이면 점으로부터 Z 좌표를 제거합니다.

지정된 점이 널(NULL)이거나 비어 있으면 널(NULL)이 리턴됩니다.

이 함수를 메소드로 호출할 수 있습니다.

구문



매개변수

point Z 좌표를 리턴하거나 수정할 ST_Point 유형의 값

z_coordinate

*point*에 대한 새로운 Z 좌표를 표시하는 DOUBLE 유형의 값

*z_coordinate*가 널(NULL)이면 *point*에서 Z 좌표를 제거합니다.

리턴 유형

- DOUBLE, *z_coordinate*면 지정되지 않습니다.
- db2gse.ST_Point, *z_coordinate*면 지정됩니다.

예

예 1

이 예는 ST_Z 함수의 사용법을 보여줍니다. Geometry를 작성하여 SAMPLE_POINTS 테이블에 삽입합니다.

```
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) )
```

예 2

이 예는 테이블에 있는 점의 Z 좌표를 찾습니다.

```
SELECT id, ST_Z (geometry) Z_COORD
  FROM sample_points
```

결과:

```
ID          Z_COORD
----------- ----------------------
          1 +3.20000000000000E+001
          2 +2.00000000000000E+001
          3 +2.30000000000000E+001
```

예 3

이 예는 해당 Z 좌표가 40으로 설정된 점을 리턴합니다.

```
SELECT id, CAST( ST_AsText( ST_Z (geometry, 40)) AS VARCHAR(60) )
  Z_40
  FROM sample_points
  WHERE id=3
```

결과:

```
ID          Z_40
------ ------------------------------------------------------
     3 POINT ZM ( 3.00000000 8.00000000 40.00000000 7.00000000)
```

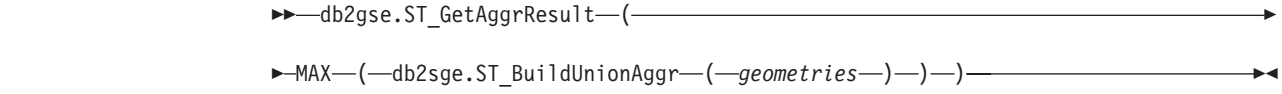
총계

통합 집계는 ST_BuildUnionAggr 및 ST_GetAggrResult 함수를 조합한 것입니다. 이러한 조합은 통합 세트를 구성함으로써 테이블에 있는 Geometry 컬럼을 단일 Geometry로 집계합니다.

통합 Geometry에 조합할 모든 Geometry가 널(NULL)이면 널(NULL)이 리턴됩니다. 통합 Geometry에 조합할 각 Geometry가 널(NULL)이거나 비어 있으면 ST_Point 유형의 비어 있는 Geometry가 리턴됩니다.

ST_BuildUnionAggr 함수를 메소드로 호출할 수도 있습니다.

구문



매개변수

Geometry

ST_Geometry 또는 부속 유형 중 하나를 가지며 통합에 조합될 모든 Geometry를 나타내는 테이블의 컬럼

리턴 유형

db2gse.ST_Geometry

제한사항

다음과 같은 상황에서는 테이블에 있는 공간 컬럼의 통합 집계를 구성할 수 없습니다.

- 대량 병렬 처리(MPP) 환경
- SELECT문에 GROUP BY절을 사용한 경우
- DB2 집계 함수 MAX가 아닌 다른 함수를 사용할 경우

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변합니다.

이 예는 통합 집계를 사용하여 점 세트를 다중점으로 조합하는 방법을 보여줍니다. 여러 개의 점이 SAMPLE_POINTS 테이블에 추가됩니다. ST_GetAggrResult 및 ST_BuildUnionAggr 함수를 사용하여 통합된 점 세트를 구성합니다.

```
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
```

결과:

```
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)
```

제 24 장 그룹 변환

그룹 변환

Spatial Extender는 DB2 서버와 클라이언트 응용프로그램 간에 Geometry를 전송하는 데 사용되는 네 가지 변환 그룹을 제공합니다. 이러한 변환 그룹은 다음 데이터 교환 형식을 수용합니다.

- Well-Known Text 표현
- Well-Known Binary 표현
- ESRI 모양 표현
- GML(Geography Markup Language)

공간 컬럼을 포함하는 테이블에서 데이터를 검색할 때 공간 컬럼의 데이터는 변환된 데이터가 2진 또는 텍스트 표현으로 나타나는 여부를 표시하는지에 따라 CLOB(2G) 또는 BLOB(2G)으로 변환됩니다. 변환 그룹을 사용하여 공간 데이터를 데이터베이스로 변환할 수도 있습니다.

데이터를 전송할 때 사용할 변환 그룹을 선택하려면 SET CURRENT DEFAULT TRANSFORM GROUP문을 사용하여 DB2 특수 레지스터 CURRENT DEFAULT TRANSFORM GROUP을 수정하십시오. DB2는 이러한 특수 레지스터의 값을 사용하여 필수 변환을 수행하기 위해 호출해야 할 변환 함수를 판별합니다.

변환 그룹은 응용프로그램 프로그래밍을 단순화할 수 있습니다. SQL문에 명시적으로 변환 함수를 사용하는 대신 변환 그룹을 지정하여 DB2가 해당 태스크를 처리하게 할 수 있습니다.

ST_WellKnownText 변환 그룹

ST_WellKnownText 변환 그룹을 사용하여 WKT(Well-Known Text) 표현을 사용하는 데이터를 DB2®로 전송하거나 이로부터 수신할 수 있습니다.

값을 데이터베이스 서버에서 클라이언트로 바인드할 때 ST_AsText()에서 제공하는 것과 동일한 기능을 사용하여 Geometry를 WKT로 변환합니다. Well-known Text 표현으로 된 Geometry를 데이터베이스 서버로 전송할 경우, 내재적으로 ST_Geometry (CLOB) 함수를 사용하여 ST_Geometry 값으로 변환을 수행합니다. 값을 DB2에 바인드할 때 변환 그룹을 사용하면 숫자 ID가 0인 공간 참조 시스템에 Geometry가 표현됩니다.

© Copyright IBM Corp. 1998, 2006

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변할 수 있습니다.

예 1

다음 SQL 스크립트는 명시적 ST_AsText 함수를 사용하지 않고 ST_WellKnownText 변환 그룹을 사용하여 해당 Well-Known Text 표현으로 된 Geometry를 검색하는 방법을 보여줍니다.

```
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
```

결과:

```
ID   GEOM
---  ------------------------------------------------------------------
  1  LINESTRING ( 100.00000000 100.00000000, 200.00000000 100.00000000)
```

예 2

다음 C 코드는 ST_WellKnownText 변환 그룹을 사용하여 호스트 변수 wkt_buffer(CLOB 유형을 갖고 있으며 WKT 표현으로 표시된 삽입할 점 (10 10) 포함)에 대해 명시적 ST_Geometry 함수를 사용하는 Geometry를 삽입하는 방법을 보여줍니다.

```
EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS db2gse.ST_Geometry AS CLOB(1000) wkt_buffer;
EXEC SQL END DECLARE SECTION;

// set the transform group for all subsequent SQL statements
EXEC SQL
   SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownText;

id = 100;
strcpy(wkt_buffer.data, "point ( 10 10 )");
wkt_buffer.length = strlen(wkt_buffer.data);

// insert point using WKT into column of type ST_Geometry
```

```
EXEC SQL
   INSERT
    INTO   transforms_sample(id, geom)
    VALUES (:id, :wkt_buffer);
```

ST_WellKnownBinary 변환 그룹

ST_WellKnownBinary 변환 그룹을 사용하여 WKB(Well-Known Binary) 표현을 사용하는 데이터를 DB2®로 전송하거나 이로부터 수신하십시오.

값을 데이터베이스 서버에서 클라이언트로 바인드할 때 ST_AsBinary()에서 제공하는 것과 동일한 기능을 사용하여 Geometry를 WKB로 변환합니다. Well-Known Binary 표현으로 된 Geometry를 데이터베이스 서버로 전송할 경우, 내재적으로 ST_Geometry(BLOB) 함수를 사용하여 ST_Geometry 값으로 변환을 수행합니다. 값을 DB2에 바인드할 때 변환 그룹을 사용하면 숫자 ID가 0인 공간 참조 시스템에 Geometry가 표현됩니다.

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변할 수 있습니다.

예 1

다음 SQL 스크립트는 명시적 ST_AsBinary 함수를 사용하지 않고 ST_WellKnownBinary 변환 그룹을 사용하여 해당 Well-Known Binary 표현으로 된 Geometry를 검색하는 방법을 보여줍니다.

```
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
```

결과:

```
ID    GEOM
----   ----------------------------------------------
   1   x'010300000001000000050000000000000000000244000
```

```
00000000002440000000000002440000000000003440
00000000000034400000000000034400000000000034
40000000000002440000000000002440000000000000
2440'
```

예 2

다음 C 코드는 ST_WellKnownBinary 변환 그룹을 사용하여 호스트 변수 wkb_buffer(BLOB 유형을 가지며 WKB 표현으로 표시된 삽입할 Geometry 포함)에 대해 명시적 ST_Geometry 함수를 사용하는 Geomatry를 삽입하는 방법을 보여줍니다.

```
EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS db2gse.ST_Geometry AS BLOB(1000) wkb_buffer;
EXEC SQL END DECLARE SECTION;

// set the transform group for all subsequent SQL statements
EXEC SQL
   SET CURRENT DEFAULT TRANSFORM GROUP = ST_WellKnownBinary;

// initialize host variables
...
// insert geometry using WKB into column of type ST_Geometry

EXEC SQL
   INSERT
    INTO transforms_sample(id, geom)
    VALUES ( :id, :wkb_buffer );
```

ST_Shape 변환 그룹

ST_Shape 변환 그룹을 사용하여 ESRI 모양 표현을 사용하는 데이터를 DB2®로 전송하거나 이로부터 수신하십시오.

값을 데이터베이스 서버에서 클라이언트로 바인드할 때 ST_AsShape()에서 제공하는 것과 동일한 기능을 사용하여 Geometry를 해당 모양 표현으로 변환합니다. 모양 표현으로 된 Geometry를 데이터 서버로 전송할 때 내재적으로 ST_Geometry(BLOB) 함수를 사용하여 ST_Geometry 값으로 변환을 수행합니다. 값을 DB2에 바인드할 때 변환 그룹을 사용하면 숫자 ID가 0인 공간 참조 시스템에 Geometry가 표현됩니다.

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변할 수 있습니다.

예 1

다음 SQL 스크립트는 명시적 ST_AsShape 함수를 사용하지 않고 ST_Shape 변환 그룹을 사용하여 해당 모양 표현으로 된 Geometry를 검색할 수 있는 방법을 보여줍니다.

```
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
```

결과:

```
ID    GEOM
----   ------------------------------------------
   1    x'0100000000000000000034400000000000003E40'
```

예 2

다음 C 코드는 ST_Shape 변환 그룹을 사용하여 호스트 변수 shape_buffer(BLOB 유형을 가지며 모양 표현으로 표시된 삽입할 Geometry 포함)에 대해 명시적 ST_Geometry 함수를 사용하는 Geometry를 삽입하는 방법을 보여줍니다.

```
EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS db2gse.ST_Geometry AS BLOB(1000) shape_buffer;
EXEC SQL END DECLARE SECTION;

// set the transform group for all subsequent SQL statements
EXEC SQL
   SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape;

// initialize host variables
...

SET CURRENT DEFAULT TRANSFORM GROUP = ST_Shape;

// insert geometry using shape representation into column of type ST_Geometry
EXEC SQL
     INSERT
    INTO transforms_sample(id, geom)
    VALUES ( :id, :shape_buffer );
```

ST_GML 변환 그룹

ST_GML 변환 그룹을 사용하여 GML(Geography Markup Language)을 사용하는 데이터를 DB2®로 전송하거나 이로부터 수신하십시오.

값을 데이터베이스 서버에서 클라이언트로 바인드할 때 ST_AsGML()에서 제공하는 것과 동일한 기능을 사용하여 Geometry를 해당 GML 표현으로 변환합니다. GML 표현으로 된 Geometry를 데이터 서버로 전송할 경우, 내재적으로 ST_Geometry(CLOB)

함수를 사용하여 ST_Geometry 값으로 변환을 수행합니다. 값을 DB2에 바인드할 때 변환 그룹을 사용하면 숫자 ID가 0인 공간 참조 시스템에 Geometry가 표현됩니다.

예

다음 예에서 결과 행은 읽기 쉽도록 다시 포맷된 것입니다. 결과의 스페이스는 온라인 표시장치에 따라 변할 수 있습니다.

예 1

다음 SQL 스크립트는 명시적 ST_AsGML 함수를 사용하지 않고 ST_GML 변환 그룹을 사용하여 해당 GML 표현으로 된 Geometry를 검색할 수 있는 방법을 보여줍니다.

```
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
```

결과:

```
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>
```

예 2

다음 C 코드는 ST_GML 변환 그룹을 사용하여 호스트 변수 gml_buffer(CLOB 유형을 갖고 있으며 GML 표현으로 표시된 삽입할 점 (20 20) 포함)에 대해명시적으로 ST_Geometry 함수를 사용하지 않고 Geometry를 삽입하는 방법을 보여줍니다.

```
EXEC SQL BEGIN DECLARE SECTION;
   sqlint32 id = 0;
   SQL TYPE IS db2gse.ST_Geometry AS CLOB(1000) gml_buffer;
EXEC SQL END DECLARE SECTION;
```

```
// set the transform group for all subsequent SQL statements
EXEC SQL
   SET CURRENT DEFAULT TRANSFORM GROUP = ST_GML;
 id = 100;
strcpy(gml_buffer.data, "<gml:point><gml:coord>"
   "<gml:X>20</gml:X> <gml:Y>20</gml:Y></gml:coord></gml:point>");

//initialize host variables
wkt_buffer.length = strlen(gml_buffer.data);

// insert point using WKT into column of type ST_Geometry
EXEC SQL
   INSERT
   INTO transforms_sample(id, geom)
   VALUES ( :id, :gml_buffer );
```

제 25 장 데이터 형식 지원

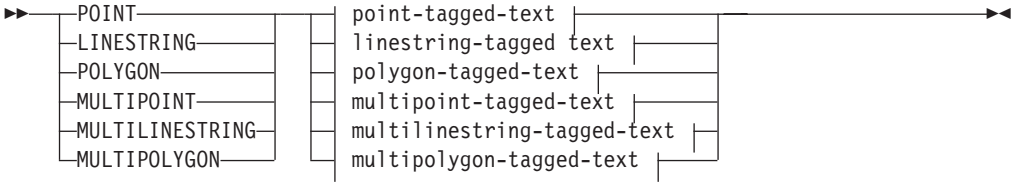
이 장에서는 DB2 Spatial Extender에서 사용할 수 있는 산업 표준 공간 데이터 형식에 대해 설명합니다. 다음과 같은 네 가지 공간 데이터 형식에 대해 설명합니다.

- Well-Known Text 표현
- Well-Known Binary 표현
- 모양 표현
- GML(Geography Markup Language) 표현

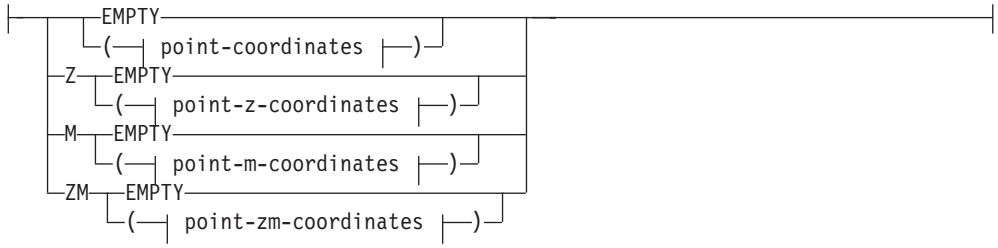
Well-Known Text 표현

OpenGIS 컨소시엄 "SQL용 단순 기능" 스펙에서는 Geometry 데이터를 ASCII 형식으로 교환할 수 있도록 Well-Known Text 표현을 정의합니다. 이 표현은 ISO "SQL/MM 파트: 3 공간" 표준이라고도 합니다. WKT 데이터를 승인하고 작성하는 함수에 대한 정보는 Geometry를 데이터 교환 형식으로(또는 그 반대로) 변환하는 공간 함수를 참조하십시오.

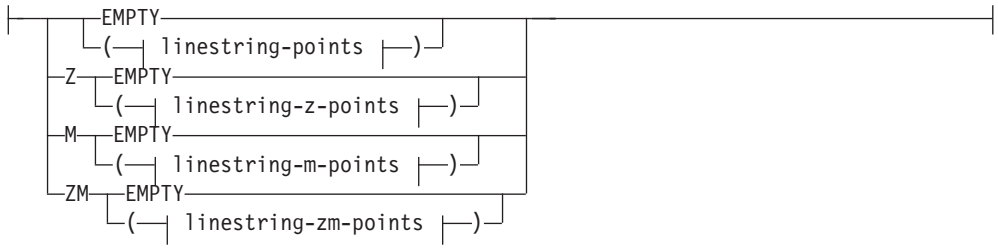
Geometry의 Well-Known Text 표현은 다음과 같이 정의됩니다.



point-tagged-text:

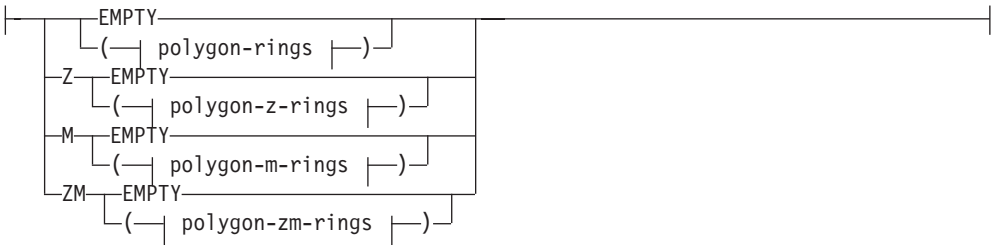


linestring-tagged-text:

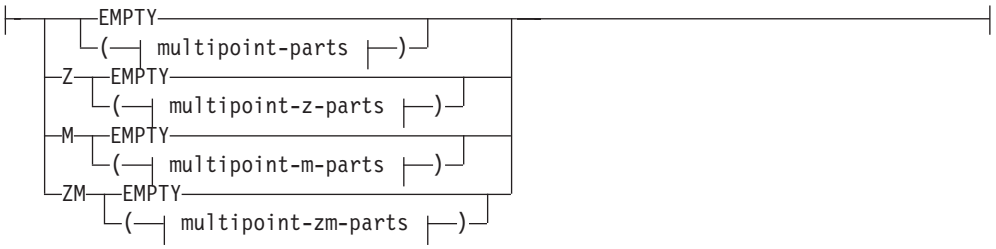


© Copyright IBM Corp. 1998, 2006

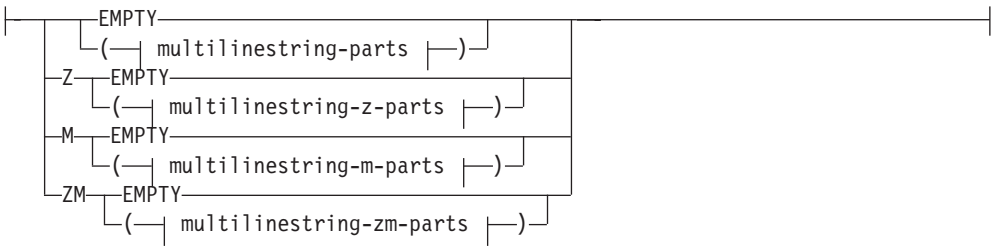
polygon-tagged-text:



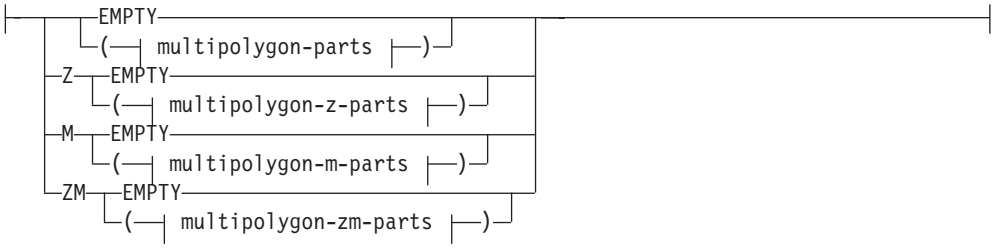
multipoint-tagged-text:



multilinestring-tagged-text:

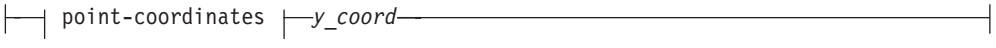


multipolygon-tagged-text:



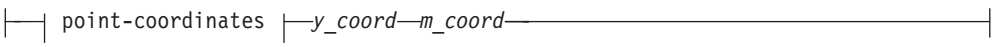
point-coordinates:

point-z-coordinates:

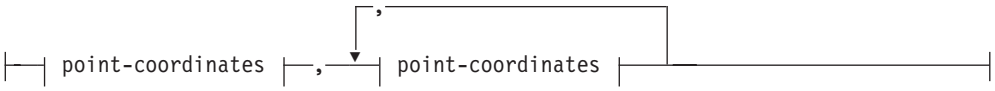


point-m-coordinates:

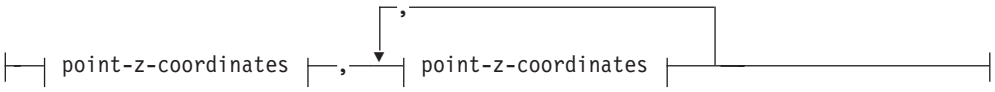
point-zm-coordinates:



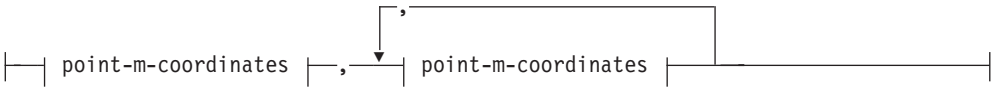
linestring-points:



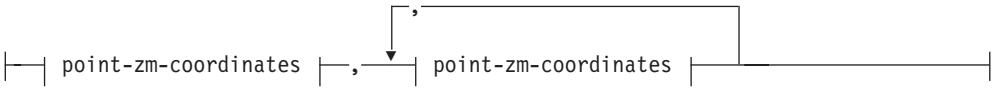
linestring-z-points:



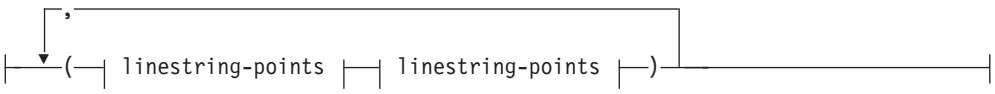
linestring-m-points:



linestring-zm-points:



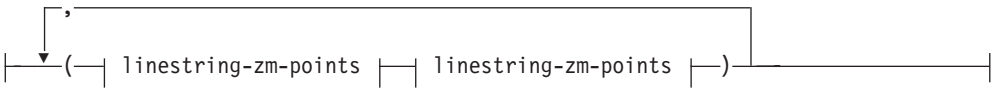
polygon-rings:



polygon-z-rings:

polygon-m-rings:

polygon-zm-rings:



multipoint-parts:

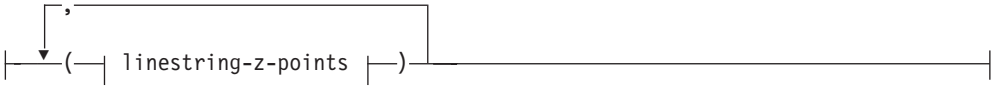
multipoint-z-parts:

multipoint-m-parts:

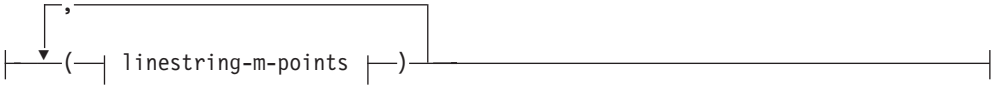
multipoint-zm-parts:

multilinestring-parts:

multilinestring-z-parts:



multilinestring-m-parts:



multilinestring-zm-parts:

multipolygon-parts:

multipolygon-z-parts:

multipolygon-m-parts:

multipolygon-zm-parts:

매개변수

x_coord

점의 X 좌표를 나타내는 숫자 값(고정, 정수 또는 부동 소수점)

y_coord

점의 Y 좌표를 나타내는 숫자 값(고정, 정수 또는 부동 소수점)

z_coord

점의 Z 좌표를 나타내는 숫자 값(고정, 정수 또는 부동 소수점)

m_coord

점의 M 좌표(측정 단위)를 나타내는 숫자 값(고정, 정수 또는 부동 소수점)

Geometry가 비어 있으면 좌표 목록 대신 키워드 EMPTY가 지정됩니다. 좌표 목록에서 EMPTY 키워드가 임베드되면 안됩니다.

다음 테이블은 가능한 텍스트 표현에 대한 몇 가지 예를 제공합니다.

표 59. Geometry 유형 및 텍스트 표현

Geometry 유형	**WKT 표현**	**주석**
point	POINT EMPTY	빈 점
point	POINT (10.05 10.28)	point
point	POINT Z(10.05 10.28 2.51)	Z 좌표가 있는 점
point	POINT M(10.05 10.28 4.72)	M 좌표가 있는 점
point	POINT ZM(10.05 10.28 2.51 4.72)	Z 좌표 및 M 좌표가 있는 점
linestring	LINESTRING EMPTY	빈 선스트링
polygon	POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))	polygon

표 59. *Geometry 유형 및 텍스트 표현 (계속)*

Geometry 유형	WKT 표현	주석
다중점	MULTIPOINT Z(10 10 2, 20 20 3)	Z 좌표가 있는 다중점
multilinestring	MULTILINESTRING M((310 30 1, 40 30 20, 50 20 10)(10 10 0, 20 20 1))	M 좌표가 있는 다중 선스트링
multipolygon	MULTIPOLYGON ZM(((1 1 1 1, 1 2 3 4, 2 2 5 6, 2 1 7 8, 1 1 1 1)))	Z 좌표 및 M 좌표가 있는 다중 다각형

Well-Known Binary 표현

이 절에서는 Geometry에 대한 Well-Known Binary 표현에 대해 설명합니다.

OpenGIS 컨소시엄 ″SQL용 단순 기능″ 스펙에서는 Well-Known Binary 표현을 정의합니다. 이 표현은 국제 표준화 기구(ISO) ″SQL/MM 파트: 3 공간″ 표준에서도 정의됩니다. WKB를 승인 및 작성하는 함수에 대한 자세한 정보는 이 주제의 끝에 있는 관련 참조 절을 참조하십시오.

Well-Known Binary 표현에 대한 기본 빌딩 블록은 점에 대한 바이트 스트림이며, 이는 두 개의 DOUBLE 값으로 구성되어 있습니다. 다른 Geometry에 대한 바이트 스트림은 이미 정의된 Geometry에 대한 바이트 스트림을 사용하여 빌드됩니다.

다음 예는 Well-Known Binary 표현에 대한 기본 빌딩 블록을 보여줍니다.

```
// Basic Type definitions
// byte : 1 byte
// uint32 : 32 bit unsigned integer (4 bytes)
// double : double precision number (8 bytes)

// Building Blocks : Point, LinearRing

Point {
  double x;
  double y;
};
LinearRing {
  uint32 numPoints;
  Point points[numPoints];
};
enum wkbGeometryType {
  wkbPoint = 1,
  wkbLineString = 2,
  wkbPolygon = 3,
  wkbMultiPoint = 4,
  wkbMultiLineString = 5,
  wkbMultiPolygon = 6
```

```
};
enum wkbByteOrder {
  wkbXDR = 0, // Big Endian
  wkbNDR = 1 // Little Endian
};
WKBPoint {
  byte byteOrder;
  uint32 wkbType; // 1=wkbPoint
  Point point;
};
WKBLineString {
  byte byteOrder;
  uint32 wkbType; // 2=wkbLineString
  uint32 numPoints;
  Point points[numPoints];
};

WKBPolygon {
  byte byteOrder;
  uint32 wkbType; // 3=wkbPolygon
  uint32 numRings;
  LinearRing rings[numRings];
};
WKBMultiPoint {
  byte byteOrder;
  uint32 wkbType; // 4=wkbMultipoint
  uint32 num_wkbPoints;
  WKBPoint WKBPoints[num_wkbPoints];
};
WKBMultiLineString {
  byte byteOrder;
  uint32 wkbType; // 5=wkbMultiLineString
  uint32 num_wkbLineStrings;
  WKBLineString WKBLineStrings[num_wkbLineStrings];
};

wkbMultiPolygon {
  byte byteOrder;
  uint32 wkbType; // 6=wkbMultiPolygon
  uint32 num_wkbPolygons;
  WKBPolygon wkbPolygons[num_wkbPolygons];
};

WKBGeometry {
  union {
    WKBPoint point;
    WKBLineString linestring;
    WKBPolygon polygon;
    WKBMultiPoint mpoint;
    WKBMultiLineString mlinestring;
    WKBMultiPolygon mpolygon;
  }
};
```

다음 그림은 NDR 코딩을 사용한 Well-Known Binary 표현으로 된 Geometry 예를 표시합니다.

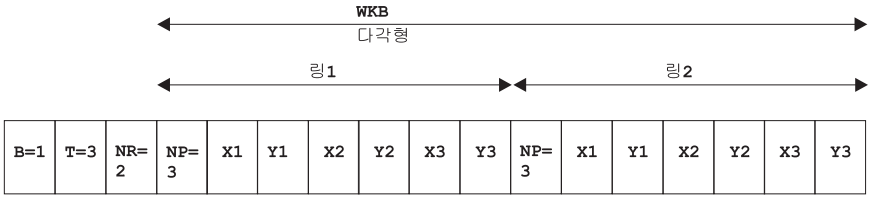


그림 58. NDR 형식의 Geometry 표현. 2개의 선형(NR=2)이 있는 유형 다각형(T=3)의 (B=1), 여기서 각 링에는 3개의 점(NP=3)이 있습니다.

모양 표현

모양 표현은 ESRI에서 정의한 광범위하게 사용되는 산업 표준입니다. 모양 표현에 대한 자세한 설명은 http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf에 있는 ESRI 웹 사이트를 참조하십시오.

GML(Geography Markup Language) 표현

DB2 Spatial Extender에는 GML(Geography Markup Language) 표현의 표현으로부터 Geometry를 생성하는 여러 가지 함수가 있습니다.

GML(Geography Markup Language)은 OpenGIS 컨소시엄 ″Geography Markup Language V2″ 스펙에 의해 정의된 지리적 정보에 대한 XML 인코딩입니다. 이 OpenGIS 컨소시엄 스펙은 http://www.opengis.org/techno/implementation.htm에 있습니다.

제 26 장 지원되는 좌표 시스템

이 주제에서는 좌표 시스템 구문에 대한 설명을 제공하며 DB2 Spatial Extender에 의해 지원되는 좌표 시스템 값을 나열합니다.

좌표 시스템 구문

공간 참조 시스템의 Well-Known Text 표현에서는 좌표 시스템 정보에 대한 표준 텍스트 표현을 제공합니다. Well-Known Text 표현의 정의는 OGC SQL용 단순 기능 스펙 및 ISO SQL/MM 파트 3: 공간 표준으로 정의됩니다.

좌표 시스템이란 지리적(위도-경도), 프로젝트화된(X,Y) 또는 지심(X,Y,Z) 좌표 시스템입니다. 좌표 시스템은 여러 개의 오브젝트로 구성됩니다. 각 오브젝트는 DATUM 또는 UNIT와 같은 대문자 키워드를 가지며, 괄호안에 쉼표로 구분하여 정의한 오브젝트 매개변수들이 따라 나옵니다. 일부 오브젝트는 다른 오브젝트들로 구성되므로 결과가 중첩된 구조를 가질 수 있습니다.

주: 구현 시 대괄호[] 대신 표준 괄호()를 사용할 수 있지만 두 괄호 양식 모두를 읽을 수 있어야 합니다.

대괄호를 사용하는 좌표 시스템의 문자열 표현에 대한 EBNF(Extended Backus Naur Form) 정의는 다음과 같습니다(괄호 사용에 관해서는 위의 주 참조).

```
<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>
```

좌표 시스템의 유형은 사용되는 키워드에 의해 식별됩니다.

PROJCS

데이터 세트의 좌표 시스템은 데이터가 프로젝트화된 좌표인 경우에 PROJCS 키워드로 식별됩니다.

GEOGCS

데이터 세트의 좌표 시스템은 데이터가 지리적 좌표인 경우에 GEOGCS 키워드로 식별됩니다.

© Copyright IBM Corp. 1998, 2006

GEOCCS

데이터 세트의 좌표 시스템은 데이터가 지리적 좌표인 경우에 GEOCCS 키워드로 식별됩니다.

PROJCS 키워드 뒤에는 투영 좌표 시스템을 정의하는 모든 "조각들"이 수반됩니다. 모든 오브젝트의 첫 번째 조각은 항상 이름입니다. 투영 좌표 시스템 이름 뒤에는 지리 좌표 시스템, 맵 투영, 하나 이상의 매개변수 및 선형 도량 단위와 같은 여러 오브젝트가 따라 나옵니다. 투영 좌표 시스템은 모두 지리적 좌표 시스템에 근거하므로, 이 절에서는 투영 좌표 시스템에 특정한 조각을 먼저 설명합니다. 예를 들어, NAD83 기준상의 UTM 존 10N은 다음과 같이 정의됩니다.

```
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]]
```

이름 및 여러 오브젝트(기준, 자오선 및 각도 수치)가 지리적 좌표 시스템 오브젝트를 차례로 정의합니다.

```
<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>
```

긴 반지름(semi major axis)은 미터로 측정되며 0보다 커야 합니다.

NAD83상의 UTM 존 10에 대한 지리적 좌표 시스템 문자열

```
GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",
SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],
UNIT["Degree",0.0174532925199433]]
```

UNIT 오브젝트는 각도 또는 선형 도량 단위를 나타낼 수 있습니다.

```
<angular unit> = <unit>
<linear unit> = <unit>
<unit> = UNIT["<name>", <conversion factor>]
<conversion factor> = <number>
```

변환 인수는 단위당 미터 수(선형 단위의 경우) 또는 라디안 수(각도 단위의 경우)를 지정하며 이는 0보다 커야 합니다.

그러므로 UTM 존 10N의 전체 문자열 표현은 다음과 같습니다.

```
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]]
```

지심 좌표 시스템은 지리적 좌표 시스템과 유사합니다.

```
<geocentric cs> = GEOCCS["<name>", <datum>, <prime meridian>, <linear unit>]
```

지원되는 선형 단위

표 60. 지원되는 선형 단위

단위	전환 계수
Meter	1.0
Foot(International)	0.3048
U.S. Foot	12/39.37
Modified American Foot	12.0004584/39.37
Clarke's Foot	12/39.370432
Indian Foot	12/39.370141
Link	7.92/39.370432
Link (Benoit)	7.92/39.370113
Link (Sears)	7.92/39.370147
Chain (Benoit)	792/39.370113
Chain (Sears)	792/39.370147
Yard (Indian)	36/39.370141
Yard (Sears)	36/39.370147
Fathom	1.8288
Nautical Mile	1852.0

지원되는 각도 단위

표 61. 지원되는 각도 단위

단위	유효한 위도 범위	유효한 경도 범위	전환 계수
Radian	–pi/2 및 pi/2 라디안(포함)	–pi 및 pi 라디안(포함)	1.0
Decimal Degree	–90 및 90도(포함)	–180 및 180도(포함)	pi/180

표 61. 지원되는 각도 단위 (계속)

단위	유효한 위도 범위	유효한 경도 범위	전환 계수
Decimal Minute	–5400 및 5400분(포함)	–10800 및 10800분(포함)	(pi/180)/60
Decimal Second	–324000 및 324000초(포함)	–648000 및 648000초(포함)	(pi/180)*3600
Gon	–100 및 100 그라디언(포함)	–200 및 200 그라디언(포함)	pi/200
Grad	–100 및 100 그라디언(포함)	–200 및 200 그라디언(포함)	pi/200

표준 회전 타원체

표 62. 표준 회전 타원체

이름	긴 반지름	역 변환
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 정의)	6377299.36	300.8017
Everest 1830 Modified	6377304.063	300.8017
Everest Adjustment 1937	6377276.345	300.8017
Everest 1830 (1962 정의)	6377301.243	300.8017255

표 62. 표준 회전 타원체 (계속)

이름	긴 반지름	역 변환
Everest 1830 (1967 정의)	6377298.556	300.8017
Everest 1830 (1975 정의)	6377299.151	300.8017255
Everest 1969 수정	6377295.664	300.8017
Fischer 1960	6378166.0	298.3
Fischer 1968	6378150 .0	298.3
수정 Fischer	6378155 .0	298.3
GEM 10C	6378137.0	298.257222101
GRS 1967	6378160.0	298.247167427
GRS 1967 절단	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
구체	6371000.0	0.0
구체(ArcInfo)	6370997.0	0.0
Struve 1860	6378298.3	294.73
Walbeck	6376896.0	302.78
War Office	6378300.0	296.0
WGS 1966	6378145.0	298.25
WGS 1972	6378135.0	298.26
WGS 1984	6378137.0	298.257223563

지원되는 측지학 기준면

표 63. 지원되는 측지학 기준면

이름	측지 기준면
Adindan	Lisbon
Afgooye	Loma Quintana
Agadez	Lome
Australian Geodetic Datum 1966	Luzon 1911
Australian Geodetic Datum 1984	Mahe 1971
Ain el Abd 1970	Makassar
Amersfoort	Malongo 1987
Aratu	Manoca
Arc 1950	Massawa
Arc 1960	Merchich
Ancienne Triangulation Francaise	Militar-Geographische Institute
Barbados	Mhast
Batavia	Minna
Beduaram	Monte Mario
Beijing 1954	M'poraloko
Reseau National Belge 1950	NAD Michigan
Reseau National Belge 1972	North American Datum 1927
Bermuda 1957	North American Datum 1983
Bern 1898	Nahrwan 1967
Bern 1938	Naparima 1972
Ancienne Triangulation Francaise	Militar-Geographische Institute
Barbados	Mhast
Batavia	Minna
Beduaram	Monte Mario
Beijing 1954	M'poraloko
Reseau National Belge 1950	NAD Michigan
Reseau National Belge 1972	North American Datum 1927
Bermuda 1957	North American Datum 1983

표 63. 지원되는 측지학 기준면 (계속)

이름	측지 기준면
Bern 1898	Nahrwan 1967
Bern 1938	Naparima 1972
Ancienne Triangulation Francaise	Militar-Geographische Institute
Barbados	Mhast
Batavia	Minna
Beduaram	Monte Mario
Beijing 1954	M'poraloko
Reseau National Belge 1950	NAD Michigan
Reseau National Belge 1972	North American Datum 1927
Bermuda 1957	North American Datum 1983
Bern 1898	Nahrwan 1967
Bern 1938	Naparima 1972
Bogota	Nord de Guerre
Bukit Rimpah	NGO 1948
Camacupa	Nord Sahara 1959
Campo Inchauspe	NSWC 9Z-2
Cape	Nouvelle Triangulation Francaise
Carthage	New Zealand Geodetic Datum 1949
Chua	OS (SN) 1980
Conakry 1905	OSGB 1936
Corrego Alegre	OSGB 1970 (SN)
Cote d'Ivoire	Padang 1884
Datum 73	Palestine 1923
Deir ez Zor	Pointe Noire
Deutsche Hauptdreiecksnetz	Provisional South American Datum 1956
Douala	Pulkovo 1942
European Datum 1950	Qatar
European Datum 1987	Qatar 1948
Egypt 1907	Qornoq
European Reference System 1989	RT38

표 63. 지원되는 측지학 기준면 (계속)

이름	측지 기준면
Fahud	South American Datum 1969
Gandajika 1970	Sapper Hill 1943
Garoua	Schwarzeck
Geocentric Datum of Australia 1994	Segora
Guyane Francaise	Serindung
Herat North	Stockholm 1938
Hito XVIII 1963	Sudan
Hu Tzu	Shan Tananarive 1925
Hungarian Datum 1972	Timbalai 1948
Indian 1954	TM65
Indian 1975	TM75
Indonesian Datum 1974	Tokyo
Jamaica 1875	Trinidad 1903
Jamaica 1969	Trucial Coast 1948
Kalianpur	Voirol 1875
Kandawala	Voirol Unifie 1960
Kertau	WGS 1972
Kuwait Oil Company	WGS 1972 Transit Broadcast Ephemeris
La Canoa	WGS 1984
Lake	Yacare
Leigon	Yoff
Liberia 1964	Zanderij

지원되는 자오선

표 64. 지원되는 자오선

위치	좌표
Greenwich	0° 0’ 0″
Bern	7° 26’ 22.5″ E
Bogota	74° 4’ 51.3″ W
Brussels	4° 22’ 4.71″ E

표 64. 지원되는 자오선 (계속)

위치	좌표
Ferro	17° 40’ 0″ W
Jakarta	106° 48’ 27.79″ E
Lisbon	9° 7’ 54.862″ W
Madrid	3° 41’ 16.58″ W
Paris	2° 20’ 14.025″ E
Rome	12° 27’ 8.4″ E
Stockholm	18° 3’ 29″ E

지원되는 맵 투영

표 65. 원통형 공간 방사

원통형 공간 방사	의사 원통형 공간 방사
Behrmann	Craster parabolic
Cassini	Eckert I
Cylindrical equal area	Eckert II
Equirectangular	Eckert III
Gall’s stereographic	Eckert IV
Gauss-Kruger	Eckert V
Mercator	Eckert VI
Miller cylindrical	McBryde-Thomas flat polar quartic
Oblique	Mercator (Hotine) Mollweide
Plate-Carée	Robinson
Times	Sinusoidal (Sansom-Flamsteed)
Transverse Mercator	Winkel I

표 66. 원추형 공간 방사

이름	원추형 프로젝션
Albers conic equal-area	Chamberlin trimetric
Bipolar oblique conformal conic	Two-point equidistant
Bonne	Hammer-Aitoff equal-area
Equidistant conic	Van der Grinten I

표 66. 원추형 공간 방사 (계속)

이름	원추형 프로젝션
Lambert conformal conic	기타
Polyconic	Alaska series E
Simple conic	Alaska Grid (Modified-Stereographic by Snyder)

표 67. 맵 투영 매개변수

매개변수	설명
central_meridian	x 좌표의 원점으로 선택된 경도선
scale_factor	Scale_factor는 일반적으로 맵 투영에서 변형된 부분의 양을 줄이는 데 사용됩니다.
standard_parallel_1	변형 부분이 없는 위도선. ″실제 스케일의 위도″에도 사용됩니다.
standard_parallel_2	변형 부분이 없는 경도선
longitude_of_center	맵 투영의 중앙점을 정의하는 경도
latitude_of_center	맵 투영의 중앙점을 정의하는 위도
longitude_of_origin	x 좌표의 원점으로 선택된 경도
latitude_of_origin	y 좌표의 원점으로 선택된 위도
false_easting	모든 X 좌표가 양수가 되도록 X축에 추가된 값
false_northing	모든 Y 좌표가 양수가 되도록 Y 좌표에 추가된 값
azimuth	사각 공간 방사의 중앙점을 정의하는 동북쪽 각도
longitude_of_point_1	맵 투영에 필요한 첫 번째 점의 경도
latitude_of_point_1	맵 투영에 필요한 첫 번째 점의 위도
longitude_of_point_2	맵 투영에 필요한 두 번째 점의 경도
latitude_of_point_2	맵 투영에 필요한 두 번째 점의 위도
longitude_of_point_3	맵 투영에 필요한 세 번째 점의 경도
latitude_of_point_3	맵 투영에 필요한 세 번째 점의 위도
landsat_number	Landsat 위성의 수
path_number	특정 위성에 대한 궤도 경로 번호
perspective_point_height	맵 투영의 원근 화법 점에 대한 지표면으로부터의 높이
fipszone	State Plane 좌표 시스템 번호
zone	UTM 존 번호

제 27 장 DB2 제어 센터의 공간 태스크

Spatial Extender 사용자 인터페이스에서 사용 가능한 태스크는 태스크를 단순화하는 데 도움이 될 수 있습니다.

명령 프롬프트에서 명령으로, 응용프로그램에서 스토어드 프로시저로 또는 DB2 제어 센터에서 많은 태스크를 실행할 수 있습니다. 이 섹션에서는 DB2 제어 센터에서 실행할 수 있는 태스크를 설명합니다.

좌표 시스템 변경

좌표 시스템을 변경하면 Geometry의 실제 위치가 크게 변경될 수 있으며 불필요한 렌더링이 발생할 수 있습니다. 좌표 시스템 을변경하기 전에 변경 결과를 이해해야 합니다.

이름을 제외한 모든 필드를 변경할 수 있습니다.

- **소속** 및 **소속 ID:** 선택적인 값입니다. 이 매개변수는 모두 널이거나 또는 모두 널(NULL)이 아니어야 합니다. 매개변수 조합으로 좌표 시스템을 고유하게 식별합니다.
- **정의:** 정의는 최대 2048자까지 가능합니다. 좌표 시스템을 제공하는 벤더는 보통 이 값을 포함시킵니다.

좌표 시스템 작성

일반적으로 기존 좌표 시스템을 사용합니다.

고유의 좌표 시스템을 작성해야 하는 경우 제공하는 좌표 정보 모두가 정확하고 사용중인 geobrowser와 호환 가능한지 확인하십시오.

- **이름:** 좌표 시스템을 식별하는 데 도움이 됩니다. 소속 및 소속 ID로 좌표 시스템을 식별합니다.
- **소속** 및 **소속 ID:** 선택적인 값입니다. 이 매개변수는 모두 널이거나 또는 모두 널이 아니어야 합니다. 매개변수 조합으로 좌표 시스템을 고유하게 식별합니다.
- **정의:** 정의는 최대 2048자까지 가능합니다. 좌표 시스템을 제공하는 벤더는 보통 이 값을 포함시킵니다.

공간 컬럼 작성

공간 데이터를 기존 테이블에 추가하는 경우 공간 컬럼을 작성하고 공간 참조 시스템에서 컬럼을 등록하십시오.

© Copyright IBM Corp. 1998, 2006

공간 컬럼 작성 창을 사용하여 공간 컬럼을 작성하려면 테이블에서 공간 컬럼 창을 열어야 합니다.

- **컬럼:** 컬럼의 이름을 입력하십시오.
- **유형 스키마** 및 **유형 이름:** 목록에서 값을 선택하십시오.
- **공간 참조 시스템:** 선택사항: 목록에서 공간 참조 시스템을 선택하십시오. 측지 공간 참조 시스템에서 컬럼을 등록하려면 2000000000 - 2000001000 범위 내의 ID를 사용하여 측지 참조 시스템을 지정하십시오.

공간 인덱스 작성

공간 컬럼에서 공간 격자 인덱스 또는 측지 Voronoi 인덱스를 작성할 수 있습니다. 인덱스 작성 시 찾아 검색하기 쉽도록 DB2 데이터를 작성함으로써 성능을 향상시킬 수 있습니다.

권장사항: 인덱스에서 올바른 결과를 리턴하도록 하려면 인덱스를 작성하려는 공간 컬럼의 모든 데이터에 동일한 좌표 시스템을 사용하십시오. 동일한 공간 참조 시스템과 이에 따른 동일 좌표 시스템을 사용하도록 모든 데이터를 제한하는 공간 컬럼을 등록할 수 있습니다.

- **공간 컬럼:** 인덱스를 작성할 컬럼을 선택하십시오.
- **고밀도 격자:** Voronoi 인덱스를 작성하려면 -1을 입력하십시오. 그렇지 않으면 0 이외의 숫자를 입력하십시오.
- **보통 격자:** Voronoi 인덱스를 작성하려면 -1을 입력하십시오. 보통 격자를 사용하지 않으려면 0을 입력하거나 고밀도 격자보다 큰 숫자를 입력할 수 있습니다.
- **저밀도 격자:** Voronoi 인덱스를 작성하려면 사용할 셀 구조의 ID(1 - 14)를 입력하십시오. 저밀도 격자를 사용하려면 보통 격자를 사용해야 합니다. 이 값은 보통 격자보다 커야 합니다. 저밀도 격자를 사용하지 않으려면 0을 입력할 수 있습니다.

지오코딩(Geocoding) 실행

DB2 제어 센터에서 지오코딩 노트북 실행을 사용하여 공간 데이터를 일괄처리 모드로 지오코딩하십시오.

기본 페이지

지오코더(Geocoder)를 지정한 다음 결과 컬럼을 선택하십시오. **결과 유형** 필드는 지정하는 지오코더(Geocoder)를 기초로 자동으로 완료됩니다. 커미트 범위와 Where절을 지정하여 일괄처리 작업을 사용자 정의할 수도 있습니다.

지오코더(Geocoder) 매개변수 페이지

고유한 컬럼 이름 또는 매개변수 값을 지정할 수 있습니다.

지오코딩(Geocoding) 설정

지오코딩(Geocoding)을 설정할 경우 데이터를 지오코딩하는 데 사용할 지오코더(Geocoder)와 컬럼을 연관시키고 이후 지오코더(Geocoder)에서 사용할 해당 지오코딩 매개변수와 기타 관련 정보를 설정합니다.

기본 페이지

설정할 지오코더(Geocoder)를 선택하십시오. **결과 유형** 필드는 지정하는 지오코더(Geocoder)를 기초로 자동으로 완료됩니다. **커미트 범위** 필드를 사용하여 커미트하기 전에 지오코딩할 행 수를 선택할 수 있습니다.

추가 정보: 목록에서 지오코더(Geocoder)를 선택하고 **자동 지역코드**를 선택하면 갱신할 때 공간 컬럼을 자동으로 지오코딩할 수 있습니다.

지오코더(Geocoder) 매개변수 페이지

고유한 컬럼 이름 또는 매개변수 값을 지정할 수 있습니다.

공간 참조 시스템 변경

주의: 설명 이외의 속성을 변경하면 해당 공간 참조 시스템과 연관된 모든 공간 데이터 값이 변경되어 더 이상 유효할 수 없습니다.

- **ID** 필드에서 변경하는 공간 참조 시스템의 ID를 입력하십시오. 측지 공간 참조 시스템의 경우 2000000318 - 2000001000 범위 내에서만 ID 값을 변경할 수 있습니다.
- **오프셋:** 오프셋 변환의 경우, X, Y, Z 또는 M 좌표의 스케일 및 오프셋 값을 입력하십시오. 스케일 값은 수치의 배수 값입니다. 디폴트값은 1입니다. 좌표를 10진수 값에서 정수로 변환할 때 정확도를 유지하기 위해 값을 변경할 수 있습니다. 오프셋 값은 좌표 및 수치 값에 대한 양수를 구하기 위해 각 값에서 빼는 숫자입니다. 스케일 및 오프셋 값은 공간 참조 시스템을 작성하는 데 필요합니다.
- **Extent:** Extent 변환의 경우 X, Y, Z 또는 M 좌표의 최소 및 최대값을 입력하십시오. 스케일 값은 수치의 배수 값입니다. 디폴트값은 1입니다. 좌표를 10진수 값에서 정수로 변환할 때 정확도를 유지하기 위해 값을 변경할 수 있습니다. 스케일 및 Extent 값은 **Extent** 단일 선택 단추를 선택할 경우에만 필요합니다.

공간 데이터 임포트

모양 파일이나 SDE 파일을 임포트할 수 있습니다.

기본 페이지

이 페이지를 사용하여 소스 및 목표 정보를 지정하십시오. 또한 임포트에 실패했을 경우 문제점 해결에 도움이 될 수 있는 예외 및 메시지 파일을 지정할 수도 있습니다.

고급 페이지

이 페이지를 사용하여 테이블 스페이스 세부사항 및 임포트 처리 방법에 대한 세부사항을 지정하십시오.

DB2 기술 라이브러리(PDF 형식)

다음 테이블에서는 IBM® Publications Center(www.ibm.com/shop/publications/order)에 있는 DB2® 라이브러리를 설명합니다.

테이블에서 인쇄 사용 가능한 책으로 식별된 경우라도, 해당 국가 또는 지역에서 책이 사용 가능하지 않을 수 있습니다.

이 책들에 있는 정보는 모든 DB2 사용자에게 기본이 되고, DB2 Connect™ 또는 기타 DB2 제품을 사용하는 프로그래머, 데이터베이스 관리자나 누구에게라도 유용합니다.

표 68. DB2 기술 정보

이름	문서 번호	인쇄 사용 가능
관리 안내서: 구현	SA30-2857	예
관리 안내서: 계획	SA30-2859	예
Administrative API Reference	SC10-4231	예
관리 SQL 루틴 및 뷰	SA30-2895	아니오
Call Level Interface Guide and Reference, Volume 1	SC10-4224	예
Call Level Interface Guide and Reference, Volume 2	SC10-4225	예
Command Reference	SC10-4226	아니오
데이터 이동 유틸리티 안내 및 참조서	SA30-2863	예
데이터 복구 및 고가용성 안내 및 참조서	SA30-2860	예
ADO.NET 및 OLE DB 응용프로그램 개발	SA30-2866	예
Embedded SQL 응용프로그램 개발	SA30-2870	예
Java™ 응용프로그램 개발	SA30-2869	예
Perl and PHP 응용프로그램 개발	SA30-2867	아니오
데이터베이스 응용프로그램 개발 시작하기	SA30-2868	예
Linux® 및 Windows®에서 DB2 개발 시작하기	GA30-2875	예
메시지 참조서 볼륨 1	SA30-2880	아니오
메시지 참조서 볼륨 2	SA30-2881	아니오
이주 안내서	GA30-2876	예
Net Search Extender 관리 및 사용자 안내서 **주:** 이 문서에 대한 HTML은 HTML 문서 CD에서 설치되지 않습니다.	SA30-3022	예
성능 안내서	SA30-2858	예
Query Patroller 관리 및 사용자 안내서	GA30-2865	예
DB2 Client 빠른 시작	GA30-2877	아니오
DB2 Server 빠른 시작	GA30-2874	예

© Copyright IBM Corp. 1998, 2006

표 68. DB2 기술 정보 (계속)

이름	문서 번호	인쇄 사용 가능
Spatial Extender 및 Geodetic Data Management Feature 사용자 안내서 및 참조서	SA30-3019	예
SQL 안내서	SA30-2871	예
SQL 참조서, 볼륨 1	SA30-2872	예
SQL 참조서, 볼륨 2	SA30-2873	예
시스템 모니터 안내 및 참조서	SA30-2862	예
문제점 해결 안내서	GA30-2864	아니오
Visual Explain 자습서	SA30-3234	아니오
새로운 기능	SA30-2882	예
XML Extender 관리 및 프로그래밍	SA30-3020	예
XML 안내서	SA30-2861	예
XQuery 참조서	SA30-3021	예

표 69. DB2 Connect 특정 기술 정보

이름	문서 번호	인쇄 사용 가능
DB2 Connect 사용자 안내서	SA30-2879	예
DB2 Connect Personal Edition 빠른 시작	GA30-3199	예
DB2 Connect Server 빠른 시작	GA30-2878	예

표 70. WebSphere Information Integration 기술 정보

이름	문서 번호	인쇄 사용 가능
WebSphere® Information Integration: Administration Guide for Federated Systems	SC19-1001	예
WebSphere Information Integration: ASNCLP Program Referance for Replication and Event Publishing	SC19-1000	예
WebSphere Information Integration: Configuration Guide for Federated Data Sources	문서 번호 없음	아니오
WebSphere Information Integration: SQL Replication Guide and Reference	SC19-1002	예

주: DB2 릴리스 정보는 제품 릴리스 및 Fixpack 레벨별 추가 정보를 제공합니다.

인쇄된 DB2 책 주문

인쇄된 DB2 책이 필요할 경우, 온라인으로 구매할 수 있습니다. (모든 국가 또는 지역에 해당되지 않습니다.)

언제든지 해당 지역의 IBM 담당자로부터 인쇄된 DB2 책을 구매할 수 있습니다. DB2 PDF 문서 CD에서 일부 소프트카피 책은 사용할 수 없습니다(예: DB2 메시지 참조서).

DB2 PDF 문서 CD에 있는 여러 DB2 책의 인쇄된 버전은 IBM에 비용을 지불하고 구매할 수 있습니다. 주문하는 지역에 따라 IBM Publications Center에서 온라인으로 책을 주문할 수도 있습니다. 해당 국가 또는 지역에서 온라인 주문이 사용 불가능하면 언제든지 해당 지역의 IBM 담당자로부터 인쇄된 DB2 책을 구매할 수 있습니다. DB2 PDF 문서 CD의 모든 책이 인쇄가 가능한 것은 아닙니다.

주: 최신 및 완료된 DB2 문서는 DB2 정보 센터(http://publib.boulder.ibm.com/infocenter/db2help/)에서 유지 관리됩니다.

인쇄된 DB2 책 주문:

1. 해당 국가 또는 지역에서 인쇄된 DB2 책을 온라인으로 구입할 수 있는지 여부를 확인하려면 IBM Publications Center 웹 사이트(http://www.ibm.com/shop/publications/order)를 확인하십시오. 서적 주문 정보를 액세스하려면 국가/지역/언어를 선택한 다음 해당 위치에서 주문 지시사항을 따르십시오.
2. 로컬 IBM 담당자로부터 인쇄된 DB2 책을 주문하려면 다음 웹 사이트 중 하나에서 해당 로컬 담당자 정보를 찾으십시오.
   a. IBM Worldwide Directory of Contacts(http://www.ibm.com/planetwide)
   b. IBM Publications Center 웹 사이트(http://www.ibm.com/shop/publications/order). 해당 지역의 알맞은 책에 대한 홈 페이지에 액세스하려면 국가/지역/언어를 선택해야 합니다. 이 페이지에서 "이 사이트 정보" 링크를 수행하십시오.
3. 전화로 주문할 경우, 주문하려는 DB2 서적을 지정하십시오.
4. 담당자에게 주문하려는 책의 제목 및 문서 번호를 제공하십시오.

© Copyright IBM Corp. 1998, 2006

DB2 문제점 해결 정보

광범위한 문제점 해결 및 문제점 판별 정보는 DB2 제품을 사용하여 지원합니다.

다음 자원이 사용 가능합니다.

DB2 문서

문제점 해결 정보는 DB2 문제점 해결 안내서 또는 DB2 정보 센터의 지원 및 문제점 해결 섹션에서 찾을 수 있습니다. 여기에서 가장 일반적인 몇몇 문제점에 대해 DB2 진단 도구 및 유틸리티, 솔루션을 사용하여 문제점을 분리하고 식별하는 방법에 대한 정보 및 DB2 제품을 사용하면서 발생할 수 있는 문제점을 해결하는 방법에 대한 기타 어드바이스를 찾을 수 있습니다.

DB2 기술 지원 웹 사이트

문제가 있는 경우, 가능한 원인 및 솔루션을 찾을 수 있는 도움말을 보려면 DB2 기술 지원 웹 사이트를 참조하십시오. 기술 지원 사이트에는 최신 DB2 서적, 기술 노트, APAR(Authorized Program Analysis Report) 또는 버그 수정, Fixpack과 나머지 자원에 대한 링크가 있습니다. 이러한 기술 자료를 검색하고 문제에 대해 가능한 솔루션을 찾을 수 있습니다.

DB2 기술 지원 웹 사이트(http://www.ibm.com/software/data/db2/udb/support.html)를 참조하십시오.

© Copyright IBM Corp. 1998, 2006

주의사항

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운용에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이센스까지 부여하는 것은 아닙니다. 라이센스에 대한 의문사항은 다음으로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2 바이트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 (단, 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증없이 이 책을 『현상태대로』 제공합니다. 일부 국가에서는 특정 거래에 있어서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 변경된 사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지없이 언제든지 개선 및/또는 변경할 수 있습니다.

© Copyright IBM Corp. 1998, 2006

이 정보에서 언급되는 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

(i) 독자적으로 작성된 프로그램과 다른 프로그램(본 프로그램 포함) 간의 정보 교환 및 (ii) 교환된 정보의 상호 이용을 목적으로 본 프로그램에 관한 정보를 얻고자 하는 라이센스 사용자는 다음 주소로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터

이러한 정보는 해당 조건(예를 들어, 사용료 지불 등)에 따라 사용할 수 있습니다.

이 정보에 기술된 라이센스가 있는 프로그램 및 프로그램에 대해 사용 가능한 모든 라이센스가 부여된 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

본 문서에 포함된 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 측정치는 개발 레벨 시스템에서 작성되었을 수 있으며, 따라서 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보장할 수 없습니다. 또한, 일부 성능은 추정을 통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 책의 사용자는 해당 데이터를 사용자의 특정 환경에서 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 비IBM 제품을 테스트하지 않았으므로, 이들 제품과 관련된 성능의 정확성, 호환성 또는 기타 주장에 대해서는 확인할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM이 제시하는 방향 또는 의도에 관한 어떠한 언급도 특별한 통지없이 변경될 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이 예제에는 가능한 완벽하게 개념을 설명하기 위해 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며, 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권:

이 정보에는 여러 가지 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원시 언어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 된 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포를 목적으로 추가 비용없이 이를 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 따라서 IBM은 이들 샘플 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증하거나 진술하지 않습니다.

이러한 샘플 프로그램 또는 파생 제품의 각 사본이나 일부에는 반드시 다음과 같은 저작권 표시가 포함되어야 합니다.

© (귀하의 회사명) (연도). 이 코드의 일부는 IBM Corp.의 샘플 프로그램에서 파생됩니다. © Copyright IBM Corp. _ 연도_. All rights reserved.

상표

DB2 버전 9 문서 라이브러리의 문서에서 식별되는 회사, 제품 및 서비스 이름은 IBM(International Business Machines) Corporation 또는 해당 회사의 상표 또는 서비스표입니다. 미국 또는 기타 국가에서 사용되는 IBM Corporation의 상표에 대한 정보는 http://www.ibm.com/legal/copytrade.shtml에 있습니다.

다음 용어는 기타 회사의 상표 또는 등록상표이며, DB2 문서 라이브러리에 있는 최소한 하나의 문서에서 사용됩니다.

Microsoft®, Windows, Windows NT® 및 Windows 로고는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 상표입니다.

Intel®, Itanium®, Pentium® 및 Xeon™은 미국 또는 기타 국가에서 사용되는 Intel Corporation의 상표입니다.

Java 및 모든 Java 기반 상표는 미국 또는 기타 국가에서 사용되는 Sun Microsystems, Inc.의 상표입니다.

UNIX®는 미국 또는 기타 국가에서 사용되는 Open Group의 등록상표입니다.

Linux는 미국 또는 기타 국가에서 사용되는 Linus Torvalds의 상표입니다.

기타 회사, 제품 및 서비스 이름은 타사의 상표 또는 서비스표입니다.

IBM에 문의

해당 국가 또는 지역의 IBM에 문의하려면 IBM Directory of Worldwide Contacts(http://www.ibm.com/planetwide)를 확인하십시오.

DB2 제품에 대한 자세한 내용을 보려면 http://www.ibm.com/software/data/db2/를 방문하십시오.

© Copyright IBM Corp. 1998, 2006

색인

[숫자]

180도 자오선
  교차 Gometry 188
  교차하는 최소 바운드 원 197
180도 자오선, 교차선 187

[가]

경도, 측지
  정의 149
공간 Extent
  정의 57
공간 참조 시스템 ID(SRID)
  측지용 147, 148

[다]

다중 다각형, Spatial Extender 동종 콜렉션 9
다중 선스트링, Spatial Extender 동종 콜렉션 9
다중점, Spatial Extender 동종 콜렉션 9
데이터 유형 정보, 획득 313

[마]

명령행 프로시저(CLP)
  Spatial Extender 명령 117
  메시지 141
모양 표현, 데이터 형식 534

[바]

반구, 다각형 표현 188

[아]

위도, 측지
  정의 149
인수, 변환
  좌표 62, 66
일괄처리 지오코딩(Geocoding) 82

[자]

지오코딩(Geocoding)에 사용되는 공식 62, 66

[차]

측지 Voronoi 인덱스
  공간 격자 인덱스와 비교 91
  대체 Voronoi 구조 선택 167
  사용하는 함수 165
  이용 115
  CREATE INDEX문 169
측지 공간 참조 시스템 ID
  ST_create_srs 225
측지 공간 참조 시스템(SRS)
  설명 57
치수 정보, 획득 313

A

ArcExplorer
  인터페이스로서 사용 113

C

CREATE INDEX문
  공간 격자 인덱스 100
  측지 Voronoi 인덱스 169
Control Center
  메시지 143

D

DB2 Geodetic Data Management Feature
  공간 함수 지원 197
DEFAULT _SRS
  공간 참조 시스템 60
DE_HDN _SRS_1004
  공간 참조 시스템 60
db2se 명령 117

G

GCSW _DEUTSCHE _HAUPTDRE IECKSNETZ
  좌표 시스템 60
GCS_NORTH _AMERICAN _1927
  좌표 시스템 60
GCS_NORTH _AMERICAN _1983
  좌표 시스템 60
GCS_WGS _1984
  좌표 시스템 60
GET GEOMETRY 명령
  구문 108
GML(Geographic Markup Language), 데이터 형식 534
GSE_export_sde 스토어드 프로시저 212
GSE_import_sde 스토어드 프로시저 214
Geodetic Data Management Feature
  공간 카탈로그 뷰 지원 201
  사용 시기 147
  설명 147
  타원체 209
Geodetic Extender
  공간 스토어드 프로시저 지원 201
  차이점 188
  ST_Geometry 속성 187
Geometry
  개요 9
  공간 데이터 6
  등록 정보
    개요 11
    "공간 함수, Geometry 등록 정보" 또한 참조 313
  새로 생성
    개요 320
    다른 대상으로 변환 321
    새 스페이스 구성 322
    많은 대상 중 하나 325
    기존 도량 단위 기반 326
    수정된 양식 327
  클라이언트-서버 데이터 전송 519
Geometry 거리 정보 329
Geometry 등록 정보
  개요 11

© Copyright IBM Corp. 1998, 2006

Geometry 등록 정보 *(계속)*
공간 함수 대상
데이터 유형 정보 313
좌표 및 치수 정보 313
Geometry 내 Geometry 316
바운더리 정보 317
구성 정보 319
차원 정보 319
공간 참조 시스템 320
Geometry 인덱스 정보 330
gse_disable_autogc 스토어드 프로시저 232
gse_disable_db 스토어드 프로시저 234
gse_disable_sref 스토어드 프로시저 236
gse_enable_autogc 스토어드 프로시저 238
gse_enable_db 스토어드 프로시저 240
gse_enable_sref 스토어드 프로시저 225
gse_export_shape 242
gse_import_sde 스토어드 프로시저 214
gse_import_shape 스토어드 프로시저 246
gse_register_gc 스토어드 프로시저 254
gse_register_layer 스토어드 프로시저 259
gse_run_gc 스토어드 프로시저 263
gse_unregist_gc 스토어드 프로시저 270
gse_unregist_layer 스토어드 프로시저 272

M

MBC(Minimum Bounding Circle)
공간 격자 인덱스 92
정의 165
ST_Geometry 속성 187
공간 함수 결과 197
정의 11
migrate 명령
설명 38

N

NAD27_ SRS_1002 (공간 참조 시스템) 60
NAD83_ SRS_1 (공간 참조 시스템) 60

S

SQL문
측지 Voronoi 인덱스 사용 169
ST_COORDINATE_ SYSTEMS 277
ST_Distance 370
ST_GEOCODERS 280
ST_GEOCODER_ PARAMETERS 279
ST_GEOCODING 281
ST_GEOCODING_ PARAMETERS 282
ST_GEOMETRY_ COLUMNS 275, 278
ST_Geometry 속성
측지 차이점 187
ST_SIZINGS 283
ST_SPATIAL_
REFERENCE_SYSTEMS 284
ST_UNITS_OF_ MEASURE 286
ST_UNITS_OF_ MEASURE 카탈로그 뷰 286
ST_alter_coordsys 스토어드 프로시저 216
ST_alter_srs 218
ST_create_coordsys 스토어드 프로시저 222
ST_create_srs 225
ST_disable_autogeocoding 232
ST_disable_db 스토어드 프로시저 234
ST_drop_coordsys 스토어드 프로시저 235
ST_drop_srs 236
ST_enable_autogeocoding 스토어드 프로시저 238
ST_enable_db 스토어드 프로시저 240
ST_export_shape 스토어드 프로시저 242
ST_import_shape 스토어드 프로시저 246
ST_register_geocoder 스토어드 프로시저 254
ST_register_spatial_column 스토어드 프로시저 259
ST_remove_geocoding_setup 스토어드 프로시저 261
ST_run_geocoding 스토어드 프로시저 263
ST_setup_geocoding 스토어드 프로시저 267
ST_unregister_geocoder 스토어드 프로시저 270
ST_unregister_spatial_column 스토어드 프로시저 272
Spatial Extender
공간 참조 시스템 제공 프로그램 60
사용 시기 147
참조 자료 46

V

Voronoi 모자이크 세공 166
Voronoi 셀 구조
설명 166
인덱스에 대한 대체 선택 167

W

WGS84_ SRS_1003
공간 참조 시스템 60
Well-Known Binary 표현, 데이터 형식 532
Well-Known Text 표현, 데이터 형식 527
각도 단위
좌표 시스템 535
거리
ST_Distance 함수 370
측지선상 150
격자 인덱스
개요 92
조정 101
공간 격자 인덱스
사용하는 SQL문 100
측지 Voronoi 인덱스와 비교 91
CREATE INDEX문 100
격자 레벨 및 크기 92, 94
사용하는 공간 함수 100
이용 115
인덱스 어드바이저 명령 108
공간 격자 인덱스 조정
인덱스 어드바이저 사용 101
공간 데이터
ST_GEOMETRY_ COLUMNS 275, 278
검색 및 분석
인덱스 이용 115
인터페이스 113
함수 113
데이터 유형 71
사용 6
설명 1
익스포트 77
임포트 77
지오코딩 82
컬럼 71
클라이언트에서 서버로 전송 519
공간 스토어드 프로시저
Geodetic Data Management Feature에서 지원하는 201
공간 인덱스
측지 Voronoi 165
유형 91
공간 참조 시스템
DB2 Spatial Extender에서 제공 60
설명 57

공간 참조 시스템 *(계속)*
작성 225
공간 카탈로그 뷰
Geodetic Data Management Feature에서 지원하는 201
공간 컬럼
지오코딩 82
공간 함수
데이터-교환 형식 변환
개요 289
ESRI 모양 표현 296
GML(Geography Markup Language) 표현 297
Well-Known Binary 표현 295
Well-Known Text 표현 294
새로운 Geometry 생성
개요 320
기존 도량 단위 기반 326
다른 대상으로 변환 321
많은 대상 중 하나 325
새 스페이스 구성 322
수정된 양식 327
측지 Voronoi 인덱스 사용 165, 169
EnvelopesIntersect 336
Geometries 변형 중 289
Geometry 등록 정보 313
공간 참조 시스템 320
구성 정보 319
데이터 유형 정보 313
바운더리 정보 317
좌표 및 치수 정보 313
차원 정보 319
Geometry 내 Geometry 316
Geometry 비교
개요 298
교차 304, 312
동일한 Geometry 310
컨테이너 관계 301
DE-9IM 패턴 행렬 스트링 313
Geometry 인벨로프 309
MBR aggregate 339
ST_AppendPoint 340
ST_Area 342
ST_AsBinary 345
ST_AsGML 347
ST_AsShape 348
ST_AsText 349
ST_Boundary 350

공간 함수 *(계속)*
ST_Buffer 352
ST_Centroid 355
ST_ChangePoint 356
ST_Contains 358
ST_ConvexHull 360
ST_CoordDim 362
ST_Crosses 363
ST_Difference 364
ST_Dimension 366
ST_Disjoint 368
ST_Distance 370
ST_Edge_GC_USA 373
ST_Endpoint 377
ST_EnvIntersects 380
ST_Envelope 378
ST_EqualCoordsys 381
ST_EqualSRS 383
ST_Equals 382
ST_ExteriorRing 385
ST_FindMeasure
ST_LocateAlong 386
ST_Generalize 388
ST_GeomCollFromTxt 391
ST_GeomCollFromWKB 393
ST_GeomCollection 389
ST_GeomFromText 399
ST_GeomFromWKB 400
ST_Geometry 395
ST_GeometryN 397
ST_GeometryType 398
ST_GetIndexParms 402
ST_InteriorRingN 404
ST_Intersection 405
ST_Intersects 407
ST_Is3d 409
ST_IsClosed 410
ST_IsEmpty 412
ST_IsMeasured 413
ST_IsRing 415
ST_IsSimple 416
ST_IsValid 417
ST_Length 418
ST_LineFromText 420
ST_LineFromWKB 421
ST_LineString 422
ST_LineStringN 424

공간 함수 *(계속)*
ST_LocateAlong
ST_FindMeasure 386
ST_LocateBetween
ST_MeasureBetween 437
ST_M 425
ST_MBR 434
ST_MBRIntersects 435
ST_MLineFromText 446
ST_MLineFromWKB 447
ST_MPointFromText 449
ST_MPointFromWKB 450
ST_MPolyFromText 451
ST_MPolyFromWKB 453
ST_MaxM 427
ST_MaxX 428
ST_MaxY 430
ST_MaxZ 432
ST_MeasureBetween
ST_LocateBetween 437
ST_MidPoint 438
ST_MinM 439
ST_MinX 441
ST_MinY 442
ST_MinZ 444
ST_MultiLineString 455
ST_MultiPoint 457
ST_MultiPolygon 458
ST_NumGeometries 460
ST_NumInteriorRing 461
ST_NumLineStrings 462
ST_NumPoints 463
ST_NumPolygons 464
ST_Overlaps 465
ST_Perimeter 467
ST_PerpPoints 469
ST_Point 472
ST_PointFromText 474
ST_PointFromWKB 475
ST_PointN 477
ST_PointOnSurface 478
ST_PolyFromText 479
ST_PolyFromWKB 480
ST_Polygon 481
ST_PolygonN 484
ST_Relate 485
ST_RemovePoint 486

공간 함수 *(계속)*
ST_SRID
ST_SrsId 488
ST_SrsID
ST_SRID 488
ST_SrsName 489
ST_StartPoint 490
ST_SymDifference 491
ST_ToGeomColl 494
ST_ToLineString 495
ST_ToMultiLine 496
ST_ToMultiPoint 497
ST_ToMultiPolygon 498
ST_ToPoint 500
ST_ToPolygon 501
ST_Touches 502
ST_Transform 503
ST_Union 506
ST_WKBToSQL 510
ST_WKTToSQL 511
ST_Within 508
ST_X 512
ST_Y 513
ST_Z 515
개요 289
거리 정보 329
고려사항 331
공간 인덱스를 이용하는 데 사용 115
관련된 데이터 유형 331
예 113
인덱스 정보 330
좌표 시스템 사이의 데이터 변환 330
총계 516
측지 동작 차이점 197
관리 통지 로그 145
그룹 변환
개요 519
극
포함 다각형 188
기술 라이브러리 549
다각형
Geometry 유형 9
측지 영역 정의 151
데이터 형식
GML(Geography Markup Language) 534
Well-Known Binary 표현 532
Well-Known Text 표현 527
데이터 형식 *(계속)*
모양 표현 534
데이터베이스
공간 데이터 조작에 사용
개요 45
데이텀
좌표 시스템 정의 내 209
측지 147, 148
도
경도 및 위도 149
동일 영역 투영 55
등각 투영 55
등거리 투영 55
로그
진단 145
링
설명 11
측지 영역 정의 151
맵 투영
좌표 시스템 535
메시지
Control Center 143
Spatial Extender
스토어드 프로시저 137
파트 135
CLP 141
모양 정보 141
이주 정보 141
함수 139
메시지 함수 139
명령
db2se 117
문제점 판별 553
문제점 해결
Spatial Extender
메시지 135
스토어드 프로시저 137
관리 통지 로그 145
모양 정보 메시지 141
이주 메시지 141
정보 553
함수 139
방위각 투영 55
법적 주의사항 555
변환
좌표 시스템 간의 공간 데이터 330
좌표 처리 향상 62, 66
본초 자오선 149
비교 함수
동일한 Geometry 310
DE-9IM 패턴 행렬 스트링 313
Geometry 간 교차 304, 312
Geometry 인벨로프 309
개요 298
컨테이너 관계 301
사용 가능
공간 데이터 관련 작업 45
샘플
Spatial Extender 127
선스트링 9
선형 단위
좌표 시스템 535
설치
DB2 Spatial Extender
하드웨어 및 소프트웨어 요구사항 22
성능
좌표-데이터 변환 62, 66
성능을 향상시키기 위한 증가
좌표 처리 62, 66
세계 인구 밀도
Voronoi 셀 구조 166
소프트웨어 요구사항
Spatial Extender 22
스케일 인수
개요 62, 66
스토어드 프로시저
GSE_export_sde 212
GSE_import_sde 214
ST_alter_coordsys 216
ST_alter_srs 218
ST_create_coordsys 222
ST_create_srs 225
ST_disable_autogeocoding 232
ST_disable_db 234
ST_drop_coordsys 235
ST_drop_srs 236
ST_enable_autogeocoding 238
ST_enable_db 240
ST_export_shape 242
ST_import_shape 246
ST_register_geocoder 254
ST_register_spatial_column 259
ST_remove_geocoding_ setup 261
ST_run_geocoding 263
ST_setup_geocoding 267
ST_unregister_geocoder 270

스토어드 프로시저 *(계속)*
ST_unregister_spatial_ column 272
문제점 137
시나리오
Spatial Extender 설치 15
오프셋 값
개요 62, 66
오프셋 값 및 스케일 인수의 단위 62, 66
응용프로그램
샘플 프로그램 127
이주
Spatial Extender 38
인덱스
측지 Voronoi
셀 구조 167
CREATE INDEX문 169
공간 격자 인덱스
CREATE INDEX문 100
설명 92
인덱스 어드바이저 명령 108
인덱스 어드바이저
호출할 GET GEOMETRY 명령 108
목적 92, 101
사용 시기 94
인터페이스
DB2 Spatial Extender 15
자동 지오코딩 82
자오선 149
좌표 시스템 535
적도 149
다각형 표현 내용 188
점 9
좌표
공간 참조 시스템 57
공간 참조 시스템의 변환 57
성능을 향상시키기 위한 변환 62, 66
획득 313
좌표 시스템
ST_COORDINATE_ SYSTEMS 카탈로그 뷰 277
ST_SPATIAL_ REFERENCE_SYSTEMS 카탈로그 뷰 284
개요 49
지원됨 535
좌표 참조 시스템
경도 및 위도 147
주문
책 551
지구 전체
표현 188
지리 대상물
설명 1
지리 좌표 시스템 49
지오코더
ST_GEOCODERS 카탈로그 뷰 280
ST_GEOCODER_ PARAMETERS 카탈로그 뷰 279
ST_GEOCODING 카탈로그 뷰 281
ST_GEOCODING_ PARAMETERS 카탈로그 뷰 282
ST_SIZINGS 카탈로그 뷰 283
개요 82
지오코딩
개요 82
진방위 투영 55
집계 함수
공간 컬럼 339, 516
참조 자료
DB2 Spatial Extender
설명 46
측지
예 188
정의 150
측지 경도 149
측지 공간 참조 시스템 147
측지 다각형 151
측지 데이텀 148
측지 동작
ST_Area 342
ST_Buffer 352
ST_Contains 358
ST_Difference 364
ST_Distance 370
ST_Generalize 388
ST_Intersection 405
ST_Intersects 407
ST_Length 418
ST_Perimeter 467
ST_SymDifference 491
ST_Union 506
ST_Within 508
측지 영역
설명 151
측지 위도 149
측지학 147
측지학 데이텀
ST_SPATIAL_
REFERENCE_SYSTEMS 284
설명 147
좌표 시스템 535

[카]

카탈로그 뷰
ST_COORDINATE_ SYSTEMS 277
ST_GEOCODERS 280
ST_GEOCODER_ PARAMETERS 279
ST_GEOCODING 281
ST_GEOCODING_
PARAMETERS 282
ST_GEOMETRY_ COLUMNS 275, 278
ST_SIZINGS 283
ST_SPATIAL_
REFERENCE_SYSTEMS 284
ST_UNITS_OF_ MEASURE 286
컨스트럭터 함수
ESRI 모양 표현 296
GML(Geography Markup Language) 표현 297
Well-Known Binary 표현 295
Well-Known Text 표현 294
개요 289
쿼리
공간 인덱스, 이용 115
공간, 제출할 인터페이스 113
수행할 공간 함수 113

[타]

타원체
Geodetic Extender 209
태스크
Spatial Extender 설치 15
통합 집계 함수 516
투영 좌표 시스템 49, 55

[하]

하드웨어 요구사항
Spatial Extender 22

함수
공간
데이터-교환 형식 변환 289
개요 289
회전 타원체
정의 148
좌표 시스템 535
좌표 시스템 정의 내 209

SA30-3019-00