INTERNATIONAL STANDARD

ISO/IEC 13249-3

Fourth edition
2011-04-15

Information technology — Database languages — SQL multimedia and application packages —

Part 3: **Spatial**

Technologies de l'information — Langages de bases de données — Multimédia SQL et paquetages d'application —

Partie 3: Spatial

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

Reference number
ISO/IEC FDIS 13249-3:2011(E)

© ISO/IEC 2011

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2011

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

 © ISO/IEC 2011 – All rights reserved

Contents

Page

Foreword xii

Introduction xiii

1 Scope 1

2 Normative references 1

3 Terms and definitions, notations, and conventions 1
3.1 Terms and definitions 1
3.1.1 Terms and definitions provided in ISO/IEC 13249-1 1
3.1.2 Terms and definitions provided in this part of ISO/IEC 13249 1
3.1.3 Terms and definitions taken from ISO 19107 7
3.1.4 Terms and definitions taken from ISO 19111 8
3.2 Notations 8
3.2.1 Notations provided in ISO/IEC 13249-1 8
3.2.2 Notations provided in this part of ISO/IEC 13249 8
3.3 Conventions 9

4 Concepts 10
4.1 Concepts provided in Part 1 10
4.2 Geometry Types 10
4.2.1 ST_Geometry 10
4.2.2 Spatial Relationships using ST_Geometry 17
4.2.3 ST_Point 22
4.2.4 ST_Curve 22
4.2.5 ST_LineString 23
4.2.6 ST_CircularString 23
4.2.7 ST_CompoundCurve 24
4.2.8 ST_Surface 25
4.2.9 ST_CurvePolygon 25
4.2.10 ST_Polygon 26
4.2.11 ST_Triangle 27
4.2.12 ST_PolyhdrlSurface 27
4.2.13 ST_TIN 28
4.2.14 ST_GeomCollection 28
4.2.15 ST_MultiPoint 29
4.2.16 ST_MultiCurve 29
4.2.17 ST_MultiLineString 30
4.2.18 ST_MultiSurface 30
4.2.19 ST_MultiPolygon 31
4.3 Topology-Geometry 32
4.3.1 <topology-name>.ST_NODE 32
4.3.2 <topology-name>.ST_EDGE 33
4.3.3 <topology-name>.ST_FACE 35
4.4 Topology-Network 38
4.4.1 <network-name>.ST_NODE 38
4.4.2 <network-name>.ST_LINK 38
4.5 General Routines 41
4.5.1 ST_ShortestUndPath Function 41
4.5.2 ST_ShortestDirPath Function 41
4.6 Spatial Reference System Type 42
4.6.1 ST_SpatialRefSys 42
4.7 Angle and Direction Types 42
4.7.1 ST_Angle 42

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

4.7.2 ST_Direction 43
4.8 Support Types 44
4.8.1 ST_TINElement 44
4.9 Support Routines 45
4.9.1 ST_Geometry ARRAY Support Routines 45
4.10 Tables with columns using geometry types 46
4.11 The Spatial Information Schema 46

5 Geometry Types 47
5.1 ST_Geometry Type and Routines 47
5.1.1 ST_Geometry Type 47
5.1.2 ST_Dimension Method 59
5.1.3 ST_CoordDim Method 60
5.1.4 ST_GeometryType Method 61
5.1.5 ST_SRID Methods 63
5.1.6 ST_Transform Method 64
5.1.7 ST_IsEmpty Method 65
5.1.8 ST_IsSimple Method 66
5.1.9 ST_3DIsSimple Method 67
5.1.10 ST_IsValid Method 68
5.1.11 ST_Is3D Method 69
5.1.12 ST_IsMeasured Method 70
5.1.13 ST_LocateAlong Method 71
5.1.14 ST_3DLocateAlong Method 72
5.1.15 ST_LocateBetween Method 73
5.1.16 ST_3DLocateBetween Method 75
5.1.17 ST_Boundary Method 77
5.1.18 ST_3DBoundary Method 78
5.1.19 ST_Envelope Method 79
5.1.20 ST_ConvexHull Method 80
5.1.21 ST_Buffer Methods 81
5.1.22 ST_Intersection Method 83
5.1.23 ST_3DIntersection Method 84
5.1.24 ST_Union Method 85
5.1.25 ST_3DUnion Method 86
5.1.26 ST_Difference Method 87
5.1.27 ST_3DDifference Method 88
5.1.28 ST_SymDifference Method 89
5.1.29 ST_3DSymDifference Method 90
5.1.30 Return Types from ST_Intersection, ST_Union, ST_Difference, and ST_SymDifference 91
5.1.31 Return Types from ST_3DIntersection, ST_3DUnion, ST_3DDifference, and ST_3DSymDifference 94
5.1.32 ST_Distance Methods 95
5.1.33 ST_3DDistance Methods 97
5.1.34 ST_Equals Method 99
5.1.35 ST_3DEquals Method 100
5.1.36 ST_Relate Method 101
5.1.37 ST_Disjoint Method 105
5.1.38 ST_3DDisjoint Method 106
5.1.39 ST_Intersects Method 107
5.1.40 ST_3DIntersects Method 108
5.1.41 ST_Touches Method 109
5.1.42 ST_Crosses Method 110
5.1.43 ST_Within Method 111
5.1.44 ST_Contains Method 112
5.1.45 ST_Overlaps Method 113
5.1.46 Cast 114
5.1.47 ST_WKTToSQL Method 126
5.1.48 ST_AsText Method 127
5.1.49 ST_WKBToSQL Method 128

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

 © ISO/IEC 2011 – All rights reserved

5.1.50 ST_AsBinary Method 129
5.1.51 ST_GMLToSQL Method 130
5.1.52 ST_AsGML Method 132
5.1.53 ST_GeomFromText Functions 133
5.1.54 ST_GeomFromWKB Functions 134
5.1.55 ST_GeomFromGML Functions 135
5.1.56 ST_Geometry Ordering Definition 137
5.1.57 SQL Transform Functions 138
5.1.58 <well-known text representation> 139
5.1.59 <well-known binary representation> 151

6 Point Types 182
6.1 ST_Point Type and Routines 182
6.1.1 ST_Point Type 182
6.1.2 ST_Point Methods 187
6.1.3 ST_X Methods 194
6.1.4 ST_Y Methods 195
6.1.5 ST_Z Methods 196
6.1.6 ST_M Methods 197
6.1.7 ST_ExplicitPoint Method 198
6.1.8 ST_PointFromText Functions 199
6.1.9 ST_PointFromWKB Functions 200
6.1.10 ST_PointFromGML Functions 201

7 Curve Types 202
7.1 ST_Curve Type and Routines 202
7.1.1 ST_Curve Type 202
7.1.2 ST_Length Methods 205
7.1.3 ST_3DLength Methods 207
7.1.4 ST_StartPoint Method 209
7.1.5 ST_EndPoint Method 210
7.1.6 ST_IsClosed Method 211
7.1.7 ST_3DIsClosed Method 212
7.1.8 ST_IsRing Method 213
7.1.9 ST_3DIsRing Method 214
7.1.10 ST_CurveToLine Method 215
7.2 ST_LineString Type and Routines 216
7.2.1 ST_LineString Type 216
7.2.2 ST_LineString Methods 219
7.2.3 ST_Points Methods 221
7.2.4 ST_NumPoints Method 223
7.2.5 ST_PointN Method 224
7.2.6 ST_StartPoint Method 225
7.2.7 ST_EndPoint Method 226
7.2.8 ST_LineFromText Functions 227
7.2.9 ST_LineFromWKB Functions 228
7.2.10 ST_LineFromGML Functions 229
7.3 ST_CircularString Type and Routines 230
7.3.1 ST_CircularString Type 230
7.3.2 ST_CircularString Methods 234
7.3.3 ST_Points Methods 236
7.3.4 ST_NumPoints Method 238
7.3.5 ST_PointN Method 239
7.3.6 ST_MidPointRep Method 240
7.3.7 ST_StartPoint Method 241
7.3.8 ST_EndPoint Method 242
7.3.9 ST_CircularFromTxt Functions 243
7.3.10 ST_CircularFromWKB Functions 244
7.3.11 ST_CircularFromGML Functions 245
7.4 ST_CompoundCurve Type and Routines 246
7.4.1 ST_CompoundCurve Type 246

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.4.2 ST_CompoundCurve Methods 250
7.4.3 ST_Curves Methods 253
7.4.4 ST_NumCurves Method 255
7.4.5 ST_CurveN Method 256
7.4.6 ST_StartPoint Method 257
7.4.7 ST_EndPoint Method 258
7.4.8 ST_CompoundFromTxt Functions 259
7.4.9 ST_CompoundFromWKB Functions 260
7.4.10 ST_CompoundFromGML Functions 261

8 Surface Types 262
8.1 ST_Surface Type and Routines 262
8.1.1 ST_Surface Type 262
8.1.2 ST_Area Methods 265
8.1.3 ST_3DArea Methods 267
8.1.4 ST_Perimeter Methods 269
8.1.5 ST_3DPerimeter Methods 271
8.1.6 ST_Centroid Method 273
8.1.7 ST_3DCentroid Method 274
8.1.8 ST_PointOnSurface Method 275
8.1.9 ST_3DPointOnSurf Method 276
8.1.10 ST_IsWorld Method 277
8.2 ST_CurvePolygon Type and Routines 278
8.2.1 ST_CurvePolygon Type 278
8.2.2 ST_CurvePolygon Methods 282
8.2.3 ST_ExteriorRing Methods 285
8.2.4 ST_InteriorRings Methods 287
8.2.5 ST_NumInteriorRing Method 290
8.2.6 ST_InteriorRingN Method 291
8.2.7 ST_CurvePolyToPoly Method 292
8.2.8 ST_CPolyFromText Functions 293
8.2.9 ST_CPolyFromWKB Functions 294
8.2.10 ST_CPolyFromGML Functions 295
8.3 ST_Polygon Type and Routines 296
8.3.1 ST_Polygon Type 296
8.3.2 ST_Polygon Methods 299
8.3.3 ST_ExteriorRing Methods 302
8.3.4 ST_InteriorRings Methods 303
8.3.5 ST_InteriorRingN Method 305
8.3.6 ST_PolyFromText Functions 306
8.3.7 ST_PolyFromWKB Functions 307
8.3.8 ST_PolyFromGML Functions 308
8.3.9 ST_BdPolyFromText Functions 309
8.3.10 ST_BdPolyFromWKB Functions 311
8.4 ST_TriangleType and Routines 313
8.4.1 ST_Triangle Type 313
8.4.2 ST_Triangle Methods 317
8.4.3 ST_Points Methods 321
8.4.4 ST_3DSlope Method 322
8.4.5 ST_ExteriorRing Methods 323
8.4.6 ST_InteriorRings Methods 324
8.4.7 ST_InteriorRingN Method 325
8.4.8 ST_TriFromText Functions 326
8.4.9 ST_TriFromWKB Functions 327
8.4.10 ST_TriFromGML Functions 328
8.5 ST_PolyhdrlSurface Type and Routines 329
8.5.1 ST_PolyhdrlSurface Type 329
8.5.2 ST_PolyhdrlSurface Methods 333
8.5.3 ST_Patches Methods 336
8.5.4 ST_NumPatches Method 339

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

 © ISO/IEC 2011 – All rights reserved

8.5.5 ST_PatchN Method 340
8.5.6 ST_PhSFromText Functions 341
8.5.7 ST_PhSFromWKB Functions 342
8.5.8 ST_PhSFromGML Functions 343
8.6 ST_TIN Type and Routines 344
8.6.1 ST_TIN Type 344
8.6.2 ST_TIN Methods 349
8.6.3 ST_TINElements Methods 353
8.6.4 ST_MaxSideLength Methods 355
8.6.5 ST_TINTable Methods 357
8.6.6 ST_Clip Method 373
8.6.7 ST_Patches Methods 374
8.6.8 ST_TINFromText Functions 375
8.6.9 ST_TINFromWKB Functions 376
8.6.10 ST_TINFromGML Functions 377

9 Geometry Collection Types 378
9.1 ST_GeomCollection Type and Routines 378
9.1.1 ST_GeomCollection Type 378
9.1.2 ST_GeomCollection Methods 382
9.1.3 ST_Geometries Methods 385
9.1.4 ST_NumGeometries Method 387
9.1.5 ST_GeometryN Method 388
9.1.6 ST_GeomCollFromTxt Functions 389
9.1.7 ST_GeomCollFromWKB Functions 390
9.1.8 ST_GeomCollFromGML Functions 391
9.2 ST_MultiPoint Type and Routines 392
9.2.1 ST_MultiPoint Type 392
9.2.2 ST_MultiPoint Methods 395
9.2.3 ST_Geometries Methods 397
9.2.4 ST_MPointFromText Functions 399
9.2.5 ST_MPointFromWKB Functions 400
9.2.6 ST_MPointFromGML Functions 401
9.3 ST_MultiCurve Type and Routines 402
9.3.1 ST_MultiCurve Type 402
9.3.2 ST_MultiCurve Methods 406
9.3.3 ST_IsClosed Method 408
9.3.4 ST_3DIsClosed Method 409
9.3.5 ST_Length Methods 410
9.3.6 ST_3DLength Methods 412
9.3.7 ST_Geometries Methods 414
9.3.8 ST_MCurveFromText Functions 416
9.3.9 ST_MCurveFromWKB Functions 417
9.3.10 ST_MCurveFromGML Functions 418
9.4 ST_MultiLineString Type and Routines 419
9.4.1 ST_MultiLineString Type 419
9.4.2 ST_MultiLineString Methods 422
9.4.3 ST_Geometries Methods 424
9.4.4 ST_MLineFromText Functions 426
9.4.5 ST_MLineFromWKB Functions 427
9.4.6 ST_MLineFromGML Functions 428
9.5 ST_MultiSurface Type and Routines 429
9.5.1 ST_MultiSurface Type 429
9.5.2 ST_MultiSurface Methods 433
9.5.3 ST_Area Methods 435
9.5.4 ST_3DArea Methods 437
9.5.5 ST_Perimeter Methods 439
9.5.6 ST_3DPerimeter Methods 441
9.5.7 ST_Centroid Method 443
9.5.8 ST_3DCentroid Method 444

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.5.9 ST_PointOnSurface Method 445
9.5.10 ST_3DPointOnSurf Method 446
9.5.11 ST_Geometries Methods 447
9.5.12 ST_MSurfaceFromTxt Functions 449
9.5.13 ST_MSurfaceFromWKB Functions 450
9.5.14 ST_MSurfaceFromGML Functions 451
9.6 ST_MultiPolygon Type and Routines 452
9.6.1 ST_MultiPolygon Type 452
9.6.2 ST_MultiPolygon Methods 455
9.6.3 ST_Geometries Methods 457
9.6.4 ST_MPolyFromText Functions 459
9.6.5 ST_MPolyFromWKB Functions 460
9.6.6 ST_MPolyFromGML Functions 461
9.6.7 ST_BdMPolyFromText Functions 462
9.6.8 ST_BdMPolyFromWKB Functions 464

10 Topology-Geometry 466
10.1 Topo-Geo Topology Schema 466
10.1.1 Introduction 466
10.1.2 ST_NODE view 467
10.1.3 ST_EDGE view 468
10.1.4 ST_FACE view 469
10.2 Topo-Geo Definition Schema 470
10.2.1 Introduction 470
10.2.2 ST_NODE base table 471
10.2.3 ST_EDGE base table 472
10.2.4 ST_FACE base table 474
10.3 Topo-Geo Routines 475
10.3.1 ST_AddIsoNode Function 475
10.3.2 ST_MoveIsoNode Procedure 477
10.3.3 ST_RemIsoNode Procedure 479
10.3.4 ST_AddIsoEdge Function 480
10.3.5 ST_GetFaceEdges Function 482
10.3.6 ST_ChangeEdgeGeom Procedure 483
10.3.7 ST_RemIsoEdge Procedure 485
10.3.8 ST_NewEdgesSplit Function 487
10.3.9 ST_ModEdgeSplit Function 489
10.3.10 ST_NewEdgeHeal Function 491
10.3.11 ST_ModEdgeHeal Procedure 494
10.3.12 ST_AddEdgeNewFaces Function 497
10.3.13 ST_AddEdgeModFace Function 500
10.3.14 ST_RemEdgeNewFace Function 503
10.3.15 ST_RemEdgeModFace Procedure 505
10.3.16 ST_GetFaceGeometry Function 507
10.3.17 ST_InitTopoGeo Procedure 509
10.3.18 ST_CreateTopoGeo Procedure 510
10.3.19 ST_ValidateTopoGeo Function 513

11 Topology-Network 517
11.1 Topo-Net Network Schema 517
11.1.1 Introduction 517
11.1.2 ST_NODE view 518
11.1.3 ST_LINK view 519
11.2 Topo-Net Definition Schema 520
11.2.1 Introduction 520
11.2.2 ST_NODE base table 521
11.2.3 ST_LINK base table 522
11.3 Topo-Net Routines 523
11.3.1 ST_AddIsoNetNode Function 523
11.3.2 ST_MoveIsoNetNode Procedure 524
11.3.3 ST_RemIsoNetNode Procedure 525

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.4 ST_AddLink Function 526
11.3.5 ST_ChangeLinkGeom Procedure 528
11.3.6 ST_RemoveLink Procedure 530
11.3.7 ST_InitTopoNet Procedure 531
11.3.8 ST_NewLogLinkSplit Function 532
11.3.9 ST_ModLogLinkSplit Function 534
11.3.10 ST_NewGeoLinkSplit Function 536
11.3.11 ST_ModGeoLinkSplit Function 538
11.3.12 ST_NewLinkHeal Function 540
11.3.13 ST_ModLinkHeal Procedure 543
11.3.14 ST_LogiNetFromTGeo Procedure 546
11.3.15 ST_SpatNetFromTGeo Procedure 548
11.3.16 ST_SpatNetFromGeom Procedure 550
11.3.17 ST_ValidLogicalNet Function 552
11.3.18 ST_ValidSpatialNet Function 554

12 General Routines 557
12.1 Shortest Path Routines 557
12.1.1 ST_ShortestUndPath Function 557
12.1.2 ST_ShortestDirPath Function 560

13 Spatial Reference System Type 563
13.1 ST_SpatialRefSys Type and Routines 563
13.1.1 ST_SpatialRefSys Type 563
13.1.2 ST_SpatialRefSys Methods 565
13.1.3 ST_AsWKTSRS Method 566
13.1.4 ST_WKTSRSToSQL Method 567
13.1.5 ST_SRID Method 568
13.1.6 ST_Equals Method 569
13.1.7 ST_OrderingEquals Function 570
13.1.8 ST_WellKnownText SQL Transform Group 571
13.1.9 <spatial reference system> 572

14 Angle and Direction Types 576
14.1 ST_Angle Type and Routines 576
14.1.1 ST_Angle Type 576
14.1.2 ST_Angle Methods 581
14.1.3 ST_Radians Methods 589
14.1.4 ST_Degrees Methods 590
14.1.5 ST_DegreeComponent Method 591
14.1.6 ST_MinuteComponent Method 592
14.1.7 ST_SecondComponent Method 593
14.1.8 ST_String Methods 594
14.1.9 ST_Gradians Methods 596
14.1.10 ST_Add Method 597
14.1.11 ST_Subtract Method 598
14.1.12 ST_Multiply Method 599
14.1.13 ST_Divide Method 600
14.1.14 ST_AsText Method 601
14.1.15 ST_Angle Ordering Definition 602
14.1.16 SQL Transform Functions 603
14.2 ST_Direction Type and Routines 604
14.2.1 ST_Direction Type 604
14.2.2 ST_Direction Methods 609
14.2.3 ST_Radians Method 614
14.2.4 ST_AngleNAzimuth Methods 615
14.2.5 ST_AsText Method 616
14.2.6 ST_RadianBearing Method 617
14.2.7 ST_DegreesBearing Method 619
14.2.8 ST_DMSBearing Method 621
14.2.9 ST_RadianNAzimuth Method 623

ECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

14.2.10 ST_DegreesNAzimuth Method 624
14.2.11 ST_DMSNAzimuth Method 625
14.2.12 ST_RadianSAzimuth Method 626
14.2.13 ST_DegreesSAzimuth Method 628
14.2.14 ST_DMSSAzimuth Method 630
14.2.15 ST_AddAngle Method 632
14.2.16 ST_SubtractAngle Method 633
14.2.17 ST_Direction Ordering Definition 634
14.2.18 SQL Transform Functions 635

15 Support Types 636
15.1 ST_TINElement Type and Routines 636
15.1.1 ST_TINElement Type 636
15.1.2 ST_TINElement Methods 640
15.1.3 ST_ElementType Methods 643
15.1.4 ST_ElementID Methods 644
15.1.5 ST_ElementTag Methods 645
15.1.6 ST_ElementGeometry Methods 646
15.1.7 ST_IsEmpty Method 649

16 Support Routines 650
16.1 ST_Geometry ARRAY Support Routines 650
16.1.1 ST_MaxDimension Function 650
16.1.2 ST_CheckSRID Function 652
16.1.3 ST_GetCoordDim Functions 653
16.1.4 ST_GetIs3D Function 655
16.1.5 ST_GetIsMeasured Function 656
16.1.6 ST_CheckNulls Procedure 657
16.1.7 ST_CheckConsecDups Procedure 658
16.1.8 ST_ToPointAry Cast Function 659
16.1.9 ST_ToCurveAry Cast Function 661
16.1.10 ST_ToLineStringAry Cast Function 663
16.1.11 ST_ToCircularAry Cast Function 665
16.1.12 ST_ToCompoundAry Cast Function 667
16.1.13 ST_ToSurfaceAry Cast Function 669
16.1.14 ST_ToCurvePolyAry Cast Function 671
16.1.15 ST_ToPolygonAry Cast Function 673
16.1.16 ST_ToTriangleAry Cast Function 675
16.1.17 ST_ToPolyhdrlAry Cast Function 677
16.1.18 ST_ToTINAry Cast Function 679

17 SQL/MM Spatial Information Schema 681
17.1 Introduction 681
17.2 ST_GEOMETRY_COLUMNS view 682
17.3 ST_SPATIAL_REFERENCE_SYSTEMS view 683
17.4 ST_UNITS_OF_MEASURE view 684
17.5 ST_SIZINGS view 685
17.6 Short name views 686

18 SQL/MM Spatial Definition Schema 687
18.1 Introduction 687
18.2 ST_GEOMETRY_COLUMNS base table 688
18.3 ST_SPATIAL_REFERENCE_SYSTEMS base table 689
18.4 ST_UNITS_OF_MEASURE base table 691
18.5 ST_SIZINGS base table 692

19 Status Codes 693

20 Conformance 696

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

 © ISO/IEC 2011 – All rights reserved

20.1 Requirements for conformance **696**
20.2 Features of ISO/IEC 9075 required for this part of ISO/IEC 13249 ..... **696**
20.3 Claims of conformance **696**

Annex A (informative) **Implementation-defined elements** **703**
A.1 General **703**
A.2 Implementation-defined Meta-variables **718**

Annex B (informative) **Implementation-dependent elements** **720**

Annex C (informative) **Deprecated features** **721**

Annex D (informative) **Incompatibilities with ISO/IEC 13249-3:2006** **722**

Annex E (informative) **Geometry Type Hierarchy** **723**

Bibliography **725**

Index 726

Figures

Page

Figure E.1 — Geometry Type Hierarchy Diagram **723**

Tables

Page

Table 1 — Symbols **9**
Table 2 — Data Type Codes **13**
Table 3 — Cast Codes **14**
Table 4 — Supported Casts **14**
Table 5 — DE-9IM **18**
Table 6 — Parameter Types **91**
Table 7 — Return Type Sets **91**
Table 8 — Return Type Matrix for the ST_Intersection Method **92**
Table 9 — Return Type Matrix for the ST_Union Method **93**
Table 10 — Return Type Matrix for the ST_Difference Method **93**
Table 11 — Return Type Matrix for the ST_SymDifference Method **93**
Table 12 — DE-9IM Mapping **103**
Table 13 — Cell Values **103**
Table 14 — Mapping between ST_Geometry values and GML representation **130**
Table 15 — <well-known binary representation> <uint32> Values **179**
Table 16 — SQLSTATE class and subclass values **693**

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 13249 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 13249-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

This fourth edition cancels and replaces the third edition (ISO/IEC 13249-3:2006), which has been technically revised.

ISO/IEC 13249 consists of the following parts, under the general title *Information technology — Database languages — SQL multimedia and application packages*:

— *Part 1: Framework*

— *Part 2: Full-Text*

— *Part 3: Spatial*

— *Part 5: Still image*

— *Part 6: Data mining*

— *Part 7: History*

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Introduction

The purpose of ISO/IEC 13249 is to define multimedia and application specific types and their associated routines using the user-defined features in ISO/IEC 9075.

ISO/IEC 13249 is based on the content of ISO/IEC International Standard Database Language (SQL).

The organization of this part of ISO/IEC 13249 is as follows:

Clause 1, "Scope", specifies the scope of this part of ISO/IEC 13249.

Clause 2, "Normative references", identifies additional standards that, through reference in this part of ISO/IEC 13249, constitute provisions of this part of ISO/IEC 13249.

Clause 3, "Terms and definitions, notations, and conventions", defines the notations and conventions used in this part of ISO/IEC 13249.

Clause 4, "Concepts", presents concepts used in the definition of this part of ISO/IEC 13249.

Clause 5, "Geometry Types", defines the geometry supertype.

Clause 6, "Point Types", defines primitive 0-dimensional geometry types.

Clause 7, "Curve Types", defines primitive 1-dimensional geometry types.

Clause 8, "Surface Types", defines primitive 2-dimensional geometry types.

Clause 9, "Geometry Collection Types", defines the geometry collection types.

Clause 10, "Topology-Geometry", defines node, edge, and face topology-geometry primitives.

Clause 11, "Topology-Network", defines node and link topology-network primitives.

Clause 12, "General Routines", defines the routines to determine shortest path in directed or undirected graphs.

Clause 13, "Spatial Reference System Type", defines the user-defined type to manage spatial reference systems.

Clause 14, "Angle and Direction Types", defines the angles and direction types.

Clause 15, "Support Types", defines supporting types and routines used by this part of ISO/IEC 13249.

Clause 16, "Support Routines", defines supporting functions and procedures used by this part of ISO/IEC 13249.

Clause 17, "SQL/MM Spatial Information Schema" defines the SQL/MM Spatial Information Schema.

Clause 18, "SQL/MM Spatial Definition Schema" defines the SQL/MM Spatial Definition Schema.

Clause 19, "Status Codes", defines the SQLSTATE codes used in this part of ISO/IEC 13249.

Clause 20, "Conformance", defines the criteria for conformance to this part of ISO/IEC 13249.

Annex A, "Implementation-defined elements", is an informative annex. It lists those features for which the body of this part of ISO/IEC 13249 states that the syntax or meaning or effect on the database is partly or wholly implementation-defined, and describes the defining information that an implementer needs to provide in each case.

Annex B, "Implementation-dependent elements", is an informative annex. It lists those features for which the body of this part of ISO/IEC 13249 states explicitly that the meaning or effect on the database is implementation-dependent.

Annex C, "Deprecated features", is an informative annex. It lists features that the responsible Technical Committee intend will not appear in a future revised version of this part of ISO/IEC 13249.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Annex D, "Incompatibilities with ISO/IEC 13249-3:2006", is an informative annex. It lists incompatibilities with the previous version of this part of ISO/IEC 13249-3.

Annex E, "Geometry Type Hierarchy", is an informative annex. It visually describes the inheritance relationship between user-defined types in this part of ISO/IEC 13249.

Bibliography is the last informative annex. It is a list of selective reading relating to this part of ISO/IEC 13249.

In the text of this part of ISO/IEC 13249, in Clause 5, "Geometry Types", through Clause 18, "SQL/MM Spatial Definition Schema", subclauses begin on a new page. Any resulting blank space is not significant.

The spatial user-defined types defined in this part of ISO/IEC 13249 adhere to the following:

— A spatial user-defined type is generic to spatial data handling. It addresses the need to store, manage and retrieve information based on aspects of spatial data such as geometry, location, and topology.

— A spatial user-defined type does not redefine the database language SQL directly or in combination with another spatial data type.

Implementations of this part of ISO/IEC 13249 can exist in environments that also support geographic information, decision support, data mining, and data warehousing systems.

Application areas addressed by implementations of this part of ISO/IEC 13249 include, but are not restricted to, automated mapping, desktop mapping, facilities management, geoengineering, graphics, location based services, multimedia, and resource management applications.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Information technology — Database languages — SQL multimedia and application packages —

Part 3: Spatial

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

1 Scope

This part of ISO/IEC 13249 defines

a) concepts specific to this part of ISO/IEC 13249, and

b) spatial user-defined types and their associated routines.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 13249-1, *Information technology — Database languages — SQL multimedia and application packages — Part 1: Framework*

ISO 19107, *Geographic information — Spatial schema*

ISO 19111, *Geographic information — Spatial referencing by coordinates*

ISO 19136, *Geographic information — Geography Markup Language (GML)*

IEC 559:1989, *Binary floating-point arithmetic for microprocessor systems*

3 Terms and definitions, notations, and conventions

3.1 Terms and definitions

3.1.1 Terms and definitions provided in ISO/IEC 13249-1

For the purposes of this document, the terms and definitions given in ISO/IEC 13249-1 apply.

3.1.2 Terms and definitions provided in this part of ISO/IEC 13249

For the purposes of this document, the following terms and definitions apply.

3.1.2.1
0-dimensional geometry
geometry with a geometric dimension of 0 (zero)

3.1.2.2
1-dimensional geometry
geometry with a geometric dimension of 1 (one)

3.1.2.3
2-dimensional geometry
geometry with a geometric dimension of 2

© ISO/IEC 2011 – All rights reserved

3.1.2.4
angle
inclination to each other of two intersecting lines

NOTE The angle can be measured by the arc of a circle intercepted between the two lines forming the angle, the center of the circle being the point of intersection. The value of an angle can be expressed in degrees, in degrees, minutes and seconds, in radians, or in gradians.

3.1.2.5
azimuth
representation of a geographic heading that is given by a rotation measured clockwise either from True North (North azimuth) or True South (South azimuth), having a value greater than or equal to 0 (zero) and less than 360 degrees (or 2pi radians, or 400 gradians)

3.1.2.6
bearing
representation of a geographic heading that is given by a rotation measured from True North or True South towards East or West

NOTE A bearing is specified with three parts: the prefix is either 'N' or 'S' for North or South; an angle in the range of 0 (zero) to 90 degrees, 0 to pi/2 radians, or 0 to 100 gradians; and a suffix of 'E' or 'W' for East or West. For example, a direction of Northeast is defined as the bearing N 45 E, where 45 is in degrees. A bearing of S 30 E is 30 degrees measured counterclockwise from due South and is equivalent to a North azimuth of 150 degrees.

3.1.2.7
boundary of a curve
empty set if the curve is closed, otherwise the set containing the start and end points of the curve

3.1.2.8
boundary of a point
empty set

3.1.2.9
boundary of a surface
set of curves that delineate the edge of the surface, including interior and exterior rings

3.1.2.10
break void
void whose boundary elevations take precedence over existing surface elevations

3.1.2.11
breakline
discontinuity in the slope of a Triangular Irregular Network surface caused by a natural feature (local ridge or depression) or a man-made feature (road, building)

NOTE When inserting a breakline into a surface, the elevations along the breakline take precedence over the original elevations on the surface.

3.1.2.12
closed curve
curve such that its start point is equal to its end point

3.1.2.13
closure
topological function to cause an open point-set to include its boundary, making the point-set topologically closed

3.1.2.14
control contour
breakline having a constant elevation throughout its length

3.1.2.15
degree
unit of measurement for angles such that there are 360 degrees in a circle

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3.1.2.16
degrees, minutes, and seconds representation
⟨of an angle⟩ way of documenting the value of an angle comprised of a whole number of degrees, a whole number of minutes (between 0 and 59), and a decimal number of seconds (greater than or equal to zero and less than 60), e.g., 180 00 00.0 for an angle of pi radians

NOTE For angles less than zero, only the degrees part is negative; however the positive minutes and seconds values are interpreted as increasing the negativeness of the angle. An angle of –180 30 00.0 is less than –pi radians.

3.1.2.17
dimension
geometric dimension

3.1.2.18
direction
position of one place relative to another without reference to the distance between them and usually indicated as an angular distance, measured in degrees of arc, usually from the direction of True North or South

3.1.2.19
distance between two geometries
minimum of the distances between all pairs of points composed of one point from each of the two geometries

NOTE Let g_1 and g_2 be two geometries, dist(g_1, g_2) = Min ({ distance between p and q | $p \in g_1 \wedge q \in g_2$ })

3.1.2.20
distance between two points
minimum of the lengths of all curves connecting two points

3.1.2.21
drape void
void whose boundary elevations are ignored

NOTE When inserting a drape void into a surface, the elevation of the surface takes precedence over any elevations which might have been included in the drape void boundary.

3.1.2.22
geometry
shape and geographic location of a feature

3.1.2.23
GML
Geography Markup Language as defined in ISO 19136

3.1.2.24
GML representation
XML element that is valid against an XML element definition in the Geometry Schemas as defined in ISO 19136

3.1.2.25
gradian
unit of measurement for angles such that there are 400 gradians in a circle

3.1.2.26
heading
geographic direction that is measured as a rotation from some reference direction

3.1.2.27
heal
topological operation which amalgamates two edges (or links) into one by deleting their common node or which amalgamates two faces into one by deleting their common edge

NOTE Heal is the reverse operation of split. When healing two successive edges (or links), the connecting node is deleted and a single edge (or link) replaces the original two. When healing two adjacent faces, their common edge is deleted and a single face replaces the original two.

3.1.2.28
intersection
⟨of two geometries⟩ set of points that are common to both geometries

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3.1.2.29
isolated edge
edge which is not part of the boundary of any face and is not connected to any other edge

3.1.2.30
isolated link
link which is not connected to any other link

3.1.2.31
linear ring
linestring that is closed and simple

3.1.2.32
linestring
curve with linear interpolation between control points

3.1.2.33
link
one-dimensional topological primitive that represents a relationship between two nodes

3.1.2.34
logical network
network which contains connectivity information but no geometric information

3.1.2.35
minute
unit of measurement for angles such that there are 60 minutes in a degree

3.1.2.36
mod 2 union rule
only elements that occur an odd number of times in a multiset are in the result set

NOTE For example, by applying the mod 2 union rule to the multiset: A = { 10, 20, 20, 30, 30, 30, 40, 40, 40, 40 }, the result is the set: R = { 10, 30 }. In this example, R contains element 10 and 30 because these elements occur an odd number of times in A. Elements 20 and 40 are not in R because they occur an even number of times in A.

3.1.2.37
network
set of nodes and links

3.1.2.38
non-closed curve
curve such that its start point is not equal to its end point

3.1.2.39
non-universal face
face other than the universal face

3.1.2.40
North azimuth
azimuthal heading whose rotation is measured clockwise from True North

3.1.2.41
patch
planar component of a polyhedral surface

3.1.2.42
pi
real-number mathematical constant that represents the circumference of a circle with unit diameter

NOTE This number is transcendental and cannot be represented exactly in any algebraic form, so the precision is implementation-defined.

3.1.2.43
planar graph
graph that can be drawn in a plane without graph edges crossing

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3.1.2.44
polygon
surface that is defined using linear rings to define its boundary

3.1.2.45
point set
representation of a geometry as a finite set or infinite set of points

NOTE Mathematical set intersection (∩), set union (∪) and set difference (—) operations work on point sets.

3.1.2.46
radian
unit of measurement for angles such that there are 2pi radians in a circle

3.1.2.47
random point
point of known elevation from which a Triangular Irregular Network surface can be generated

3.1.2.48
(regular) void
area within a Triangular Irrregular Network surface which signifies that the triangles within (or originally within) this area are voided

NOTE When inserting a (regular) void into a surface, the vertices of the void boundary are added to the other points in the surface and re-triangulation (perhaps only local to the boundary) occurs.

3.1.2.49
ring
curve that is closed and simple

3.1.2.50
rotation
angle with a specific sense, which can be either clockwise or counterclockwise

3.1.2.51
second
unit of measurement for angles such that there are 60 seconds in a minute

3.1.2.52
slope
⟨of a line segment⟩ ratio of (a) the absolute difference in z coordinate values of the two end points to (b) the distance between them, ignoring their z coordinates

3.1.2.53
slope
⟨of a triangle⟩ maximal slope of any line segment in a triangle

3.1.2.54
soft break
surface discontinuity which behaves as a break except that contour lines can be smoothed where they cross a soft break

3.1.2.55
South azimuth
azimuthal heading whose rotation is measured clockwise from True South

3.1.2.56
spatial network
network which contains both connectivity and geometric information

3.1.2.57
spatially 2D equals
test for equivalence between geometry values

NOTE Ignoring any z coordinate values, if the point sets of two geometry values are equal, then the geometries are spatially 2D equal.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3.1.2.58
spatially 3D equals
test for equivalence between geometry values

NOTE Considering their z coordinate values, if the point sets of two geometry values are equal, then the geometries are spatially 3D equal.

3.1.2.59
split
topological operation which creates two edges (or links) from one by inserting a node or which creates two faces from one by inserting an edge.

NOTE Split is the reverse operation of heal.

3.1.2.60
stop line
where the local continuity or regularity of a Triangular Irregular Network surface is questionable

3.1.2.61
TIN
Triangular Irregular Network as defined in ISO 19107

3.1.2.62
topologically closed
characteristic of a geometry type that every value includes its own boundary

3.1.2.63
topologically consistent
topology which exhibits the following characteristics:

1) all topological complexes are fully decomposed into their topological primitives
2) no two nodes exist at the same position in space
3) a node exists at the beginning and end of every edge
4) no edge has a geometry which crosses the geometry of a node
5) no edge has a geometry which crosses, overlaps, or is contained within the geometry of another edge
6) all edge geometries are simple
7) all edge geometries have a start point equal to the geometry of their start node
8) all edge geometries have a end point equal to the geometry of their end node
9) no face has a geometry which overlaps the geometry of another face
10) no face has a geometry within the geometry of another face
11) a universal face exists
12) a valid ST_Surface geometry can be constructed for all faces except the universal face
13) all geometries for the topology have the same spatial reference system

3.1.2.64
topology-geometry
model assuming full planar topology, comprised of nodes, edges, and faces

3.1.2.65
topology-network
model for linear applications including node and link topological primitives

3.1.2.66
True North
geographic reference direction towards the Earth's geographic North pole

3.1.2.67
True South
geographic reference direction towards the Earth's geographic South pole

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3.1.2.68
unit of measure
defined quantity in which dimensioned parameters are expressed

3.1.2.69
universal face
face containing everything else in a topology-geometry that is exterior to all other faces in that topology-geometry

3.1.2.70
voided area
⟨within a TIN surface⟩ a collection of contiguous triangles that remain in the surface but are designated as being void

NOTE A voided area is bounded by a break void, drape void, or void.

3.1.2.71
XML element
element as defined by ISO 19136

3.1.3 Terms and definitions taken from ISO 19107

For the purposes of this document, the following terms defined in ISO 19107 apply.

a) boundary
b) buffer
c) computational topology
d) connected
e) connected node
f) convex hull of a geometric object
g) coordinate
h) coordinate dimension
i) coordinate reference system
j) coordinate system
k) curve
l) directed edge
m) directed face
n) directed topological object
o) edge
p) end node
q) end point
r) exterior
s) face
t) geometric complex
u) geometric dimension
v) geometric object
w) geometric primitive
x) homomorphism
y) interior
z) isolated node

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

aa) isomorphism
ab) node
ac) point
ad) start node
ae) start point
af) surface
ag) topological complex
ah) topological object
ai) topological primitive

3.1.4 Terms and definitions taken from ISO 19111

For the purposes of this document, the following terms defined in ISO 19111 apply.

a) datum
b) ellipsoid
c) flattening
d) geodetic coordinate system, ellipsoidal coordinate system
e) meridian
f) prime meridian, zero meridian
g) projected coordinate system
h) semi-major axis
i) semi-minor axis

3.2 Notations

3.2.1 Notations provided in ISO/IEC 13249-1

For the purposes of this document, the notations given in ISO/IEC 13249-1 apply.

3.2.2 Notations provided in this part of ISO/IEC 13249

This part of ISO/IEC 13249 uses the prefix 'ST_' for user-defined type, attribute, SQL-invoked routine table and view names.

This part of ISO/IEC 13249 uses the prefix 'ST_Private' for names of certain attributes. The use of 'ST_Private' indicates that the attribute is not for public use.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

This part of ISO/IEC 13249 uses the symbols in Table 1 — Symbols.

Table 1 — Symbols

Symbols	Meaning
$\varnothing$	empty set
$\cap$	Intersection
$\cup$	union
—	difference
$\in$	is a member of
$\notin$	is not a member of
$\subset$	is a proper subset of
$\subseteq$	is a subset of
$\Leftrightarrow$	if and only if
$\Rightarrow$	implies
$\forall$	for all
{ *x* | ... }	set of all *x* such that ...
$\wedge$	and
$\vee$	or
$\neg$	not

3.3 Conventions

For the purposes of this part of ISO/IEC 13249, the conventions given in ISO/IEC 13249-1 apply.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

4 Concepts

4.1 Concepts provided in Part 1

For the purposes of this part of ISO/IEC 13249, the following concepts defined in ISO/IEC 13249-1 apply.

4.2 Geometry Types

The following geometry types are defined: ST_Geometry, ST_Point, ST_Curve, ST_LineString, ST_CircularString, ST_CompoundCurve, ST_Surface, ST_CurvePolygon, ST_Polygon, ST_Triangle, ST_PolyhdrlSurface, ST_TIN,ST_GeomCollection, ST_MultiPoint, ST_MultiCurve, ST_MultiLineString, ST_MultiSurface, and ST_MultiPolygon. ST_Geometry and its subtype family constitute the *geometry type hierarchy*, which is visually described in Annex E, "Geometry Type Hierarchy".

ST_Geometry, ST_Curve, and ST_Surface are not instantiable types. No constructor functions are defined for these types.

ST_Point, ST_LineString, ST_CircularString, ST_CompoundCurve, ST_CurvePolygon, ST_Polygon, ST_Triangle, ST_PolyhdrlSurface, ST_TIN, ST_GeomCollection, ST_MultiPoint, ST_MultiCurve, ST_MultiLineString, ST_MultiSurface, and ST_MultiPolygon are instantiable and have constructor functions.

Any geometry type can be used as the type for a column. Declaring a column to be of a particular type implies that any value of the type or of any of its subtypes can be stored in the column.

ST_TINElement is a support type. It is used by the ST_TIN geometry type to specify information about the TIN surface, other than just its triangles. Each ST_TINElement contains an ST_Geometry.

4.2.1 ST_Geometry

The ST_Geometry type is the maximal supertype of the geometry type hierarchy. The ST_Geometry type is not instantiable. The instantiable subtypes of the ST_Geometry type are 0-dimensional geometry, 1-dimensional geometry, and 2-dimensional geometry types that exist in two-dimensional coordinate space (R^2), three-dimensional coordinate space (R^3) or four-dimensional coordinate space (R^4). ST_Geometry values in R^2 have points with x and y coordinate values. ST_Geometry values in R^3 have points exclusively with x y and z coordinate values or exclusively with x, y and m coordinate values. ST_Geometry values in R^4 have points with x, y, z and m coordinate values.

The z coordinate is a coordinate of a point typically, but not necessarily, considered to represent altitude. The m coordinate is a coordinate of a point representing arbitrary measurement. ST_Geometry values that have the m coordinate value are key to supporting linear networking applications such as street routing, transportation, pipeline, telecommunications network, and utility management.

All instantiable types are defined so that all values are topologically closed (all ST_Geometry values include their boundary).

All locations in a geometry value are in the same spatial reference system.

In all routines, the geometric calculations are done in the spatial reference system of the first ST_Geometry value in the parameter list. If a routine returns an ST_Geometry value, then that value is in the spatial reference system of the first ST_Geometry value in the parameter list. Similarly, if the routine returns a measurement value such as length or area, then those values are returned in the spatial reference system of the first ST_Geometry value in the parameter list.

An implementation may define additional subtypes in the hierarchy that are outside the scope of this part of ISO/IEC 13249. An implementation shall preserve the subtype relationships between geometry types. Given two types *A* and *B* where *B* is an immediate subtype of *A*, an implementation may introduce another type *T* between types *A* and *B*, as an immediate supertype of *A*, or as an immediate subtype of *B.*

4.2.1.1 Methods on ST_Geometry

1) ST_Dimension: returns the dimension of an ST_Geometry value. The dimension of an ST_Geometry value is less than or equal to the coordinate dimension.
2) ST_CoordDim: returns the coordinate dimension of an ST_Geometry value. The coordinate dimension shall be the same as the coordinate dimension of the spatial reference system for the ST_Geometry value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3) ST_GeometryType: returns the type of the ST_Geometry value as a CHARACTER VARYING value.

4) ST_SRID: observes and mutates the spatial reference system identifier of an ST_Geometry value.

5) ST_Transform: returns the ST_Geometry value in the specified spatial reference system, considering z and m coordinate values in the calculations and including them in the resultant geometry.

6) ST_IsEmpty: tests if an ST_Geometry value corresponds to the empty set.

7) ST_IsSimple: tests if an ST_Geometry value has no anomalous geometric points, such as self intersection or self tangency, ignoring z coordinate values in the determination. Subtypes of ST_Geometry will define the specific conditions that cause a value to be classified as simple.

8) ST_3DIsSimple: tests if an ST_Geometry value has no anomalous geometric points, such as self intersection or self tangency, considering z coordinate values in the determination. Subtypes of ST_Geometry will define the specific conditions that cause a value to be classified as simple.

9) ST_IsValid: tests if an ST_Geometry value is well formed.

10) ST_Is3D: tests whether an ST_Geometry value has z coordinates.

11) ST_IsMeasured: tests whether an ST_Geometry value has m coordinate values.

12) ST_LocateAlong: returns a derived geometry value that matches the specified m coordinate value, ignoring z coordinate values in the calculations and not including them in the resultant geometry. See Subclause 4.2.1.7, "Measures on ST_Geometry" for more details.

13) ST_3DLocateAlong: returns a derived geometry value that matches the specified m coordinate value, considering z coordinate values in the calculations and including them in the resultant geometry. See Subclause 4.2.1.7, "Measures on ST_Geometry" for more details.

14) ST_LocateBetween: returns a derived geometry value that matches the specified range of m coordinate values inclusively, ignoring z coordinate values in the calculations and not including them in the resultant geometry. See Subclause 4.2.1.7, "Measures on ST_Geometry" for more details.

15) ST_3DLocateBetween: returns a derived geometry value that matches the specified range of m coordinate values inclusively, considering z coordinate values in the calculations and including them in the resultant geometry. See Subclause 4.2.1.7, "Measures on ST_Geometry" for more details.

16) ST_Boundary: returns the boundary of an ST_Geometry value, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

17) ST_3DBoundary: returns the boundary of an ST_Geometry value, considering z coordinate values in the calculations and including them in the resultant geometry.

18) ST_Envelope: returns the bounding rectangle of an ST_Geometry value, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

19) ST_ConvexHull: returns the convex hull of an ST_Geometry value, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

20) ST_Buffer: returns the ST_Geometry value that represents all points whose distance from any point of an ST_Geometry value is less than or equal to a specified distance, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

21) ST_Intersection: returns the ST_Geometry value that represents the point set intersection of two ST_Geometry values, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

    Given two geometries *a* and *b*, ST_Intersection is defined as:

    a.ST_Intersection(b) $\Leftrightarrow$ Closure($a \cap b$)

22) ST_3DIntersection: returns the ST_Geometry value that represents the point set intersection of two ST_Geometry values, considering z coordinate values in the calculations and including them in the resultant geometry.

23) ST_Union: returns the ST_Geometry value that represents the point set union of two ST_Geometry values, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Given two geometries *a* and *b*, ST_Union is defined as:

a.ST_Union(*b*) ⇔ Closure(*a* ∪ *b*)

24) ST_3DUnion: returns the ST_Geometry value that represents the point set union of two ST_Geometry values, considering z coordinate values in the calculations and including them in the resultant geometry.

25) ST_Difference: returns the ST_Geometry value that represents the point set difference of two ST_Geometry values, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

Given two geometries *a* and *b*, ST_Difference is defined as:

a.ST_Difference(*b*) ⇔ Closure(*a* — *b*)

26) ST_3DDifference: returns the ST_Geometry value that represents the point set difference of two ST_Geometry values, considering z coordinate values in the calculations and including them in the resultant geometry.

27) ST_SymDifference: returns the ST_Geometry value that represents the point set symmetric difference of two ST_Geometry values, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

Given two geometries *a* and *b*, ST_SymDifference is defined as:

a.ST_SymDifference(*b*) ⇔ Closure(*a* — *b*) ∪ Closure(*b* — *a*) ⇔
a.ST_Difference(*b*).ST_Union(*b*.ST_Difference(*a*))

28) ST_3DSymDifference: returns the ST_Geometry value that represents the point set symmetric difference of two ST_Geometry values, considering z coordinate values in the calculations and including them in the resultant geometry.

29) ST_Distance: returns the distance between two geometries, ignoring z and m coordinate values in the calculations.

30) ST_3DDistance: returns the distance between two geometries, considering z coordinate values in the calculations.

31) ST_WKTToSQL: returns the ST_Geometry value for the specified well-known text representation.

32) ST_AsText: returns the well-known text representation for the specified ST_Geometry value.

33) ST_WKBToSQL: returns the ST_Geometry value for the specified well-known binary representation.

34) ST_AsBinary: returns the well-known binary representation for the specified ST_Geometry value.

35) ST_GMLToSQL: returns the ST_Geometry value for the specified GML representation.

36) ST_AsGML: returns the GML representation for the specified ST_Geometry value.

4.2.1.2 Functions on ST_Geometry

1) ST_GeomFromText: returns an ST_Geometry value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_Geometry.

2) ST_GeomFromWKB: returns an ST_Geometry value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_Geometry.

3) ST_GeomFromGML: returns an ST_Geometry value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_Geometry.

4.2.1.3 Ordering on ST_Geometry

1) ST_OrderingEquals: is the equals only ordering definition for the ST_Geometry type.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

4.2.1.4 SQL Transforms on ST_Geometry

1) ST_WellKnownText: is the SQL Transform group that transforms an ST_Geometry value to and from a well-known text representation in a CHARACTER LARGE OBJECT value.
2) ST_WellKnownBinary: is the SQL Transform group that transforms an ST_Geometry value to and from a well-known binary representation in a BINARY LARGE OBJECT value.
3) ST_GML: is the SQL Transform group that transforms an ST_Geometry value to and from a GML representation in a CHARACTER LARGE OBJECT value.

4.2.1.5 Casts on ST_Geometry

The ST_Geometry type supports data conversion of a value from a source value (SV) of data type (SDT) to a target data type (TDT). SDT and TDT are instantiable subtypes of ST_Geometry. The supported types are described in Table 4 — Supported Casts. The SDT and TDT codes are data type codes (DT) defined in Table 2 — Data Type Codes. The cast codes used in Table 4 — Supported Casts are defined in Table 3 — Cast Codes.

Table 2 — Data Type Codes

DT	Data Type
P	ST_Point
C	ST_Curve
LS	ST_LineString
CS	ST_CircularString
CC	ST_CompoundCurve
S	ST_Surface
CP	ST_CurvePolygon
PY	ST_Polygon
T	ST_Triangle
PS	ST_PolyhdrlSurface
TN	ST_TIN
GC	ST_GeomCollection
MP	ST_MultiPoint
MC	ST_MultiCurve
MLS	ST_MultiLineString
MS	ST_MultiSurface
MPY	ST_MultiPolygon

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Table 3 — Cast Codes

Cast Code	Description
Y	The cast from the source type to the target type is supported
E	If the value is an empty set, then the cast from the source type to the target type is supported.
1-*DT*	If the value is an empty set or the value contains 1 element of type *DT*, then the cast from the source type to the target type is supported. The *DT* codes are defined in Table 2 — Data Type Codes.
1-*DT1* 4-*DT2*	If the value is an empty set or the value contains 1 element of type *DT1* which has 4 elements of type *DT2*, then the cast from the source type to the target type is supported. The *DT1* and *DT2* codes are defined DT codes in Table 2 — Data Type Codes.
n-*DT*	If the value is an empty set or the value contains 1 or more elements of type *DT*, then the cast from the source type to the target type is supported. The *DT* codes are defined in Table 2 — Data Type Codes.

Table 4 — Supported Casts

	CAST(SV AS TDT)														
TDT / SDT	P	LS	CS	CC	CP	PY	T	PS	TN	GC	MP	MC	MLS	MS	MPY
P	Y	E	E	E	E	E	E	E	E	Y	Y	E	E	E	E
LS	E	Y	E	Y	E	E	E	E	E	Y	E	Y	Y	E	E
CS	E	Y	Y	Y	E	E	E	E	E	Y	E	Y	Y	E	E
CC	E	Y	1-CS	Y	E	E	E	E	E	Y	E	Y	Y	E	E
CP	E	E	E	E	Y	Y	1-LS 4-P	Y	1-LS 4-P	Y	E	E	E	Y	Y
PY	E	E	E	E	Y	Y	1-LS 4-P	Y	1-LS 4-P	Y	E	E	E	Y	Y
T	E	E	E	E	Y	Y	Y	Y	Y	Y	E	E	E	Y	Y
PS	E	E	E	E	1-PY	1-PY	1-T	Y	E	Y	E	E	E	Y	1-PY
TN	E	E	E	E	1-T	1-T	1-T	Y	Y	Y	E	E	E	Y	1-T
GC	1-P	1-C	1-CS	1-C	1-CP	1-CP	1-T	1-PS	1-TN	Y	n-P	n-C	n-C	n-S	n-S
MP	1-P	E	E	E	E	E	E	E	E	Y	Y	E	E	E	E
MC	E	1-C	1-CS	1-C	E	E	E	E	E	Y	E	Y	Y	E	E
MLS	E	1-C	E	1-C	E	E	E	E	E	Y	E	Y	Y	E	E
MS	E	E	E	E	1-CP	1-CP	1-T	1-PS	1-TN	Y	E	E	E	Y	Y
MPY	E	E	E	E	1-PY	1-PY	1-T	1-PY	E	Y	E	E	E	Y	Y

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

4.2.1.6 Use of Z and M coordinate values

An ST_Point value may include a z coordinate value. The z coordinate value traditionally represents the third dimension (i.e. 3D). In a Geographic Information System (GIS) this may be height above or below sea level. For example: A map might have a point identifying the position of a mountain peak by its location on the earth, with the x and y coordinate values, and the height of the mountain, with the z coordinate value.

An ST_Point value may include an m coordinate value. The m coordinate value allows the application environment to associate some measure with the point values. For example: A stream network may be modeled as a multilinestring value with the m coordinate values measuring the distance from the mouth of the stream. ST_LocateBetween may be used to find all the parts of the stream that are between 10 and 12 kilometers from the mouth. There are no constraints on the m coordinate values in an ST_Geometry (e.g. the m coordinate values do not have to be continually increasing along an ST_LineString value).

As a general rule, unless explicity specified otherwise, when z or m coordinate values are present:

1) Observer methods that return ST_Point values shall include the z and m coordinate values.

© ISO/IEC 2011 – All rights reserved

2) Mutator methods shall maintain the ST_Point value z and m coordinate values.
3) Spatial operations shall not consider the z or m coordinate values in calculations (e.g. ST_Equals, ST_Length).
4) Routines which generate of new geometry values (eg. ST_Intersection) shall not consider the z or m coordinate values.
5) Cast and transform routines shall maintain the z and m coordinate values.
6) Routines converting to or from WKT, WKB, and GML shall honor the z and m coordinate values when possible.
7) Routines whose name begins with "ST_3D" (e.g., ST_3DEquals, ST_3DIntersection) shall consider the z coordinate values and if returning a geometry value, shall include z coordinate values in the returned geometry.

When this general rule may lead to ambiguous behavior, the behavior is made explicit in the Description section of the routine specification.

4.2.1.7 Measures on ST_Geometry

The ST_LocateAlong and ST_LocateBetween methods derive geometry values from the given geometry that match a measure *M* or a specific range of measures from the start measure, *SM,* to the end measure, *EM*. The ST_LocateAlong method is a variation of the ST_LocateBetween method where the *SM* and *EM* are equal to *M*.

4.2.1.7.1 Empty Sets

A null value is returned when ST_LocateAlong or ST_LocateBetween are invoked on an empty set.

4.2.1.7.2 Geometry values without m coordinate values.

An empty set of type ST_Point is returned for geometry values without m coordinate values.

4.2.1.7.3 0-dimensional geometry values

Only points in the 0-dimensional geometry values with m coordinate values between *SM* and *EM* inclusively are returned as a multipoint value. If no matching m coordinate values are found, then an empty set of type ST_Point is returned.

For example:

a) If ST_LocateAlong is invoked with an *M* value of 4 on an ST_MultiPoint value with well-known text representation:

   multipoint m(1 0 4, 1 1 1, 1 2 2, 3 1 4, 5 3 4)

   then the result is the following ST_MultiPoint value with well-known text representation:

   multipoint m(1 0 4, 3 1 4, 5 3 4)

b) If ST_LocateBetween is invoked with an *SM* value of 2 and an *EM* value of 4 on an ST_MultiPoint value with well-known text representation:

   multipoint m(1 0 4, 1 1 1, 1 2 2, 3 1 4, 5 3 5, 9 5 3, 7 6 7)

   then the result is the following ST_MultiPoint value with well-known text representation:

   multipoint m(1 0 4, 1 2 2, 3 1 4, 9 5 3)

c) If ST_LocateBetween is invoked with an *SM* value of 1 and an *EM* value of 4 on an ST_Point value with well-known text representation:

   point m(7 6 7)

   then the result is the following ST_Point value with well-known text representation:

   point m empty

d) If ST_LocateBetween is invoked with an *SM* value of 7 and an *EM* value of 7 on an ST_Point value with well-known text representation:

   point m(7 6 7)

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

then the result is the following ST_MultiPoint value with well-known text representation:

multipoint m(7 6 7)

4.2.1.7.4 1-dimensional geometry value

Interpolation is used to determine any points on the 1-dimensional geometry with an m coordinate value between SM and EM inclusively. The implementation-defined interpolation algorithm is used to estimate values between measured values, usually using a mathematical function. For example, given a measure of 6 and a 2-point linestring where the m coordinate value of the start point is 4 and the m coordinate value of the end point is 8, since 6 is halfway between 4 and 8, the interpolation algorithm would be a point on the linestring halfway between the start and end points. The interpolation is within a line segment and not across line segments in an ST_Curve. The interpolation is within an ST_Curve element and not across ST_Curve elements in an ST_MultiCurve.

The results are produced in a geometry collection. If there are consecutive points in the 1-dimensional geometry with an m coordinate value between SM and EM inclusively, then a curve value element is added to the geometry collection to represent the curve elements between these consecutive points. Any disconnected points in the 1-dimensional geometry value with m coordinate values between SM and EM inclusively are also added to the geometry collection. If no matching m coordinate values are found, then an empty set of type ST_Point is returned.

For example:

a) If ST_LocateAlong is invoked with an M value of 4 on an ST_LineString value with well-known text representation:

   linestring m(1 0 0, 3 1 4, 5 3 4, 5 5 1, 5 6 4, 7 8 4, 9 9 0)

   then the result is the following ST_MultiLineString value with well-known text representation:

   multilinestring m((3 1 4, 5 3 4), (5 6 4, 7 8 4))

b) If ST_LocateBetween is invoked with an SM value of 2 and an EM value of 4 on an ST_LineString value with well-known text representation:

   linestring m(1 0 0, 1 1 1, 1 2 2, 3 1 3, 5 3 4, 9 5 5, 7 6 6)

   then the result is the following ST_MultiLineString value with well-known text representation:

   multilinestring m((1 2 2, 3 1 3, 5 3 4))

c) If ST_LocateBetween is invoked with an SM value of 6 and an EM value of 9 on an ST_LineString value with well-known text representation:

   linestring m(1 0 0, 1 1 1, 1 2 2, 3 1 3, 5 3 4, 9 5 5, 7 6 6)

   then the result is the following ST_MultiPoint value with well-known text representation:

   multipoint m(7 6 6)

d) If ST_LocateBetween is invoked with an SM value of 1 and an EM value of 2 on an ST_LineString value with a well-known text representation:

   linestring m(0 0 1, 2 2 3, 4 4 2)

   then the result is the following ST_GeomCollection value with well-known text representation:

   geometrycollection m(linestring m(0 0 1, 1 1 2), point m(4 4 2))

e) If ST_LocateBetween is invoked with an SM value of 2 and an EM value of 4 on an ST_MultiLineString value with well-known text representation:

   multilinestring m((1 0 0, 1 1 1, 1 2 2, 3 1 3), (4 5 3, 5 3 4, 9 5 5, 7 6 6))

   then the result is the following ST_MultiLineString value with well-known text representation:

   multilinestring m((1 2 2, 3 1 3),(4 5 3, 5 3 4))

f) If ST_LocateBetween is invoked with an SM value of 1 and an EM value of 3 on an ST_LineString value with well-known text representation:

   linestring m(0 0 0, 2 2 2, 4 4 4)

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

then the result is the following ST_MultiLineString value with well-known text representation:

multilinestring m((1 1 1, 2 2 2, 3 3 3))

g) If ST_LocateBetween is invoked with an SM value of 7 and an EM value of 9 on an ST_MultiLineString value with well-known text representation:

multilinestring m((1 0 0, 1 1 1, 1 2 2, 3 1 3), (4 5 3, 5 3 4, 9 5 5, 7 6 6))

then the result is the following ST_Point value with well-known text representation:

point m empty

4.2.1.7.5 2-dimensional geometry value

The computation for 2-dimensioinal geometries is implementation-defined.

4.2.2 Spatial Relationships using ST_Geometry

The spatial relationships are methods that are used to test for the existence of a specified topological spatial relationship between two ST_Geometry values. The basic approach to comparing two ST_Geometry values is to make pair-wise tests of the intersections between the interiors, boundaries and exteriors of the two ST_Geometry values and to classify the relationship between the two ST_Geometry values based on the entries in the resulting intersection matrix.

The concepts of interior, boundary and exterior are well defined in general topology. These concepts can be applied in defining spatial relationships between 2-dimensional geometry values in *n*-dimensional coordinate space (R^n) where *n* is greater than 1 (one). In order to apply the concepts of interior, boundary and exterior to 0-dimensional geometry and 1-dimensional geometry values in R^n, a combinatorial topology approach is applied. This approach is based on the accepted definitions of the boundaries, interiors and exteriors for simplicial complexes and yields the following results.

The boundary of an ST_Geometry value is a set of ST_Geometry values of the next lower dimension. The boundary of an ST_Point value or an ST_MultiPoint value is the empty set. The boundary of a non-closed ST_Curve consists of the start and end ST_Point values; the boundary of a closed ST_Curve value is the empty set. The boundary of an ST_MultiCurve consists of those ST_Point values that are in the boundaries of an odd number of its element ST_Curve values. The boundary of an ST_Polygon value consists of its set of linear rings. The boundary of an ST_MultiPolygon value consists of the set of linear rings of its ST_Polygon values. The boundary of an arbitrary collection of geometries whose interiors are disjoint consists of geometries drawn from the boundaries of the element geometries by application of the mod 2 union rule.

The domain of ST_Geometry values considered consists of those values that are topologically closed. The interior of an ST_Geometry value consists of those points that are left when the boundary points are removed. The exterior of an ST_Geometry value consists of points not in the interior or boundary.

4.2.2.1 The Dimensionally Extended 9 Intersection Model

Given an ST_Geometry value *g*, let Interior(*g*), Boundary(*g*) and Exterior(*g*) represent the interior, boundary and exterior of *g* respectively. The intersection of any two of Interior(*g*), Boundary(*g*) and Exterior(*g*) can result in a set of ST_Geometry values, *x*, of mixed dimension. For example, the intersection of the boundaries of two ST_Polygon values may consist of an ST_Point value and an ST_LineString value. Let *x*.ST_Dimension() return the maximum dimension (-1, 0, 1, or 2) of *x*, with a value of -1 corresponding to the dimension of the empty set (∅). A dimensionally extended nine intersection matrix (DE-9IM) is specified in Table 5 — DE-9IM.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Table 5 — DE-9IM

	Interior	Boundary	Exterior
Interior	(Interior(*a*) ∩ Interior(*b*)). ST_Dimension()	(Interior(*a*) ∩ Boundary(*b*)). ST_Dimension()	(Interior(*a*) ∩ Exterior(*b*)). ST_Dimension()
Boundary	(Boundary(*a*) ∩ Interior(*b*)). ST_Dimension()	(Boundary(*a*) ∩ Boundary(*b*)). ST_Dimension()	(Boundary(*a*) ∩ Exterior(*b*)). ST_Dimension()
Exterior	(Exterior(*a*) ∩ Interior(*b*)). ST_Dimension()	(Exterior(*a*) ∩ Boundary(*b*)). ST_Dimension()	(Exterior(*a*) ∩ Exterior(*b*)). ST_Dimension()

For topologically closed input geometries computing the dimension of the intersection of the interior, boundary and exterior sets does not have as a prerequisite the explicit computation and representation of these sets. For example, to compute if the interiors of two ST_Polygon values intersect and to ascertain the dimension of this intersection, it is not necessary to explicitly represent the interior of the two polygons (which are topologically open sets) as separate ST_Geometry values.

In most cases the dimension of the intersection value at a cell is highly constrained given the type of the two ST_Geometry values. For example, in the case of comparing an ST_LineString value with an ST_Polygon: the only possible values for the Interior-Interior cell are drawn from { -1, 1 }. In the case of comparing an ST_Polygon with an ST_Polygon, the only possible values for the Interior-Interior cell are drawn from { -1, 2 }. In such cases, no work beyond detecting the intersection is required.

A spatial relationship predicate can be formulated on ST_Geometry values that takes as input a pattern matrix representing the set of acceptable values for the DE-9IM for the two geometries. If the spatial relationship between the two ST_Geometry values corresponds to one of the acceptable values as represented by the pattern matrix, then the predicate returns 1 (one). Otherwise, 0 (zero).

The pattern matrix consists of a set of 9 pattern-values, one for each cell in the matrix. Let *p* be a pattern value. The possible values of *p* are { T, F, 0, 1, 2, * } and their meanings for any cell where *x* is the intersection set for the cell are:

Case:

a) if *p* = T, then *x*.ST_Dimension() ∈ { 0, 1, 2 }, i.e. $x \neq \varnothing$

b) if *p* = F, then *x*.ST_Dimension() = -1, i.e. $x = \varnothing$

c) if *p* = 0, then *x*.ST_Dimension() = 0 (zero)

d) if *p* = 1, then *x*.ST_Dimension() = 1 (one)

e) if *p* = 2, then *x*.ST_Dimension() = 2

f) if *p* = *, then *x*.ST_Dimension() ∈ { -1, 0, 1, 2 }, i.e. any value

The pattern matrix can be represented as a <character string literal> with cardinality 9 representing the DE-9IM in row major order. See Subclause 5.3, "<literal>" in Part 2 of ISO/IEC 9075 for the definition of <character string literal>.

4.2.2.2 Common Spatial Relationships based on the DE-9IM

The ST_Relate method based on the pattern matrix enables a large number of spatial relationships to be tested and fine-tuned to the particular relationship being tested. However it is a low level building block and does not have a corresponding natural language equivalent. For this reason, commonly used spatial relationships have been specified: ST_Disjoint, ST_Intersects, ST_Touches, ST_Crosses, ST_Within, ST_Contains and ST_Overlaps. The theoretical basis for these relationships is limited to geometries having two coordinate dimensions. The z and m coordinate values will therefore be ignored when determining if these relationships hold between two geometry values. These spatial relationships have the following properties:

1) They are mutually exclusive.

2) They provide a complete covering of all topological cases.

3) They apply to spatial relationships between two geometries of either the same or different dimension.

4) Each predicate can be expressed in terms of a corresponding set of DE-9IM patterns.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5) Any realizable DE-9IM can be expressed as a Boolean expression over the 7 predicates, given the ST_Boundary method on the ST_Geometry type and the ST_StartPoint() and ST_EndPoint() method on the ST_Curve type.

4.2.2.3 Spatial Methods using ST_Geometry

1) ST_Equals: tests if an ST_Geometry value is spatially 2D equal to another ST_Geometry value. This test is for equivalence between two, possibly quite different, representations. The test may be limited by the resolution of the spatial reference system or the accuracy of the data. An implementation-defined tolerance may be provided such that two points are considered equal if the distance between the points is less that the tolerance.

   Given two geometries *a* and *b*, ST_Equals is defined as:

   a.ST_Equals(*b*) ⇔
   (*a* — *b*) ∪ (*b* — *a*) = ∅ ⇔
   a.ST_SymDifference(*b*).ST_IsEmpty()

   When determining if two geometries are equal, z and m coordinate values are ignored in the calculations.

2) ST_3DEquals: tests if an ST_Geometry value is spatially 3D equal to another ST_Geometry value. When determining if two geometries are 3D equal, z coordinate values are considered in the calculations.

3) ST_Relate: tests if an ST_Geometry value is spatially related to another ST_Geometry value by testing for intersections between the interior, boundary and exterior of the two ST_Geometry values as specified by the intersection matrix.

   When determining if two geometries are spatially related, z and m coordinate values are ignored in the calculations.

4) ST_Disjoint: tests if an ST_Geometry value is spatially disjoint from another ST_Geometry value.

   Given two geometries *a* and *b*, ST_Disjoint is defined as:

   a.ST_Disjoint(*b*) ⇔
   a ∩ *b* = ∅

   Expressed in terms of the DE-9IM:

   a.ST_Disjoint(*b*) ⇔
   (Interior(*a*) ∩ Interior(*b*) = ∅) ∧
   (Interior(*a*) ∩ Boundary(*b*) = ∅) ∧
   (Boundary(*a*) ∩ Interior(*b*) = ∅) ∧
   (Boundary(*a*) ∩ Boundary(*b*) = ∅) ⇔
   a.ST_Relate(*b*, 'FF*FF****')

   When determining if two geometries are disjoint, z and m coordinate values are ignored in the calculations.

5) ST_3DDisjoint: tests if an ST_Geometry value is spatially 3D disjoint from another ST_Geometry value. When determining if two geometries are 3D disjoint, z coordinate values are considered in the calculations.

6) ST_Intersects: tests if an ST_Geometry value spatially intersects another ST_Geometry value:

   Given two geometries *a* and *b*, ST_Intersects is defined as:

   a.ST_Intersects(*b*) ⇔
   Case *a*.ST_Disjoint(*b*) when 0 then 1 when 1 then 0 else NULL end

   When determining if two geometries intersect, z and m coordinate values are ignored in the calculations.

7) ST_3DIntersects: tests if an ST_Geometry value spatially 3D intersects another ST_Geometry value. When determining if two geometries 3D intersect, z coordinate values are considered in the calculations.

8) ST_Touches: tests if an ST_Geometry value spatially touches another ST_Geometry value.

   Given two geometries *a* and *b*, ST_Touches is defined as:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Case:

a) If *a*.ST_Dimension() = 0 (zero) and *b*.ST_Dimension() = 0 (zero), then the null value

b) Otherwise, *a*.ST_Touches(*b*) ⇔ (Interior(*a*) ∩ Interior(*b*) = ∅) ∧ (*a* ∩ *b*) ≠ ∅

Expressed in terms of the DE-9IM:

Case:

a) If *a*.ST_Dimension() = 0 (zero) and *b*.ST_Dimension() = 0 (zero), then the null value

b) Otherwise:

a.ST_Touches(*b*) ⇔
(Interior(*a*) ∩ Interior(*b*) = ∅) ∧
((Boundary(*a*) ∩ Interior(*b*) ≠ ∅) ∨
(Interior(*a*) ∩ Boundary(*b*) ≠ ∅) ∨
(Boundary(*a*) ∩ Boundary(*b*) ≠ ∅)) ⇔
Case *a*.ST_Relate(*b*, 'FT*******') = 1 OR
a.ST_Relate(*b*, 'F**T*****') = 1 OR
a.ST_Relate(*b*, 'F***T****') = 1 when TRUE then 1 when FALSE then 0 else NULL end

When determining if two geometries intersect, z and m coordinate values are ignored in the calculations.

9) ST_Crosses: tests if an ST_Geometry value spatially crosses another ST_Geometry value.

Given two geometries *a* and *b*, ST_Crosses is defined as:

Case:

a) If *a*.ST_Dimension() = 2 or *b*.ST_Dimension() = 0 (zero), then the null value

b) Otherwise:

a.ST_Crosses(*b*) ⇔
((Interior(*a*) ∩ Interior(*b*)).ST_Dimension() <
max(Interior(*a*).ST_Dimension(), Interior(*b*).ST_Dimension())) ∧ (*a* ∩ *b* ≠ *a*) ∧ (*a* ∩ *b* ≠ *b*)

Expressed in terms of the DE-9IM:

Case:

a) If *a*.ST_Dimension() = 2 or *b*.ST_Dimension() = 0 (zero), then the null value

b) If (*a*.ST_Dimension() = 0 (zero) and *b*.ST_Dimension() = 1 (one)) or
(*a*.ST_Dimension() = 0 (zero) and *b*.ST_Dimension() = 2) or
(*a*.ST_Dimension() = 1 (one) and *b*.ST_Dimension() = 2), then:

a.ST_Crosses(*b*) ⇔
(Interior(*a*) ∩ Interior(*b*) ≠ ∅) ∧ (Interior(*a*) ∩ Exterior(*b*) ≠ ∅) ⇔
a.ST_Relate(*b*, 'T*T******')

c) If *a*.ST_Dimension() = 1 (one) and *b*.ST_Dimension() = 1 (one), then:

a.ST_Crosses(*b*) ⇔
(Interior(*a*) ∩ Interior(*b*)).ST_Dimension() = 0 (zero) ⇔
a.ST_Relate(*b*, '0********')

When determining if two geometries intersect, z and m coordinate values are ignored in the calculations.

10) ST_Within: tests if an ST_Geometry value is spatially within another ST_Geometry value.

Given two geometries *a* and *b*, ST_Within is defined as:

a.ST_Within(*b*) ⇔ (*a* ∩ *b* = *a*) ∧ (Interior(*a*) ∩ Exterior(*b*) = ∅)

Expressed in terms of the DE-9IM:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

a.ST_Within(*b*) ⇔
(Interior(*a*) ∩ Interior(*b*) ≠ ∅) ∧
(Interior(*a*) ∩ Exterior(*b*) = ∅) ∧ (Boundary(*a*) ∩ Exterior(*b*) = ∅) ⇔
a.ST_Relate(*b*, 'T*F**F***')

When determining if two geometries intersect, z and m coordinate values are ignored in the calculations.

11) ST_Contains: tests if an ST_Geometry value spatially contains another ST_Geometry value.

Given two geometries *a* and *b*, ST_Contains is defined as:

a.ST_Contains(*b*) ⇔ *b*.ST_Within(*a*)

When determining if two geometries intersect, z and m coordinate values are ignored in the calculations.

12) ST_Overlaps: tests if an ST_Geometry value spatially overlaps another ST_Geometry value.

Given two geometries *a* and *b*, ST_Overlaps is defined as:

Case:

a) If (*a*.ST_Dimension() = 0 (zero) and *b*.ST_Dimension() = 0 (zero)) or
(*a*.ST_Dimension() = 1 (one) and *b*.ST_Dimension() = 1 (one)) or
(*a*.ST_Dimension() = 2 and *b*.ST_Dimension() = 2), then:

a.ST_Overlaps(*b*) ⇔
(Interior(*a*).ST_Dimension() = Interior(*b*).ST_Dimension() =
(Interior(*a*) ∩ Interior(*b*)).ST_Dimension()) ∧ (*a* ∩ *b* ≠ *a*) ∧ (*a* ∩ *b* ≠ *b*)

b) Otherwise, the null value

Expressed in terms of the DE-9IM:

Case:

a) If (*a*.ST_Dimension() = 0 (zero) and *b*.ST_Dimension() = 0 (zero)) or
(*a*.ST_Dimension() = 2 and *b*.ST_Dimension() = 2), then:

a.ST_Overlaps(*b*) ⇔
(Interior(*a*) ∩ Interior(*b*) ≠ ∅) ∧
(Interior(*a*) ∩ Exterior(*b*) ≠ ∅) ∧
(Exterior(*a*) ∩ Interior(*b*) ≠ ∅) ⇔
a.ST_Relate(*b*, 'T*T***T**')

b) If *a*.ST_Dimension() = 1 (one) and *b*.ST_Dimension() = 1 (one), then:

a.ST_Overlaps(*b*) ⇔
((Interior(*a*) ∩ Interior(*b*)).ST_Dimension() = 1 (one)) ∧
(Interior(*a*) ∩ Exterior(*b*) ≠ ∅) ∧ (Exterior(*a*) ∩ Interior(*b*) ≠ ∅) ⇔
a.ST_Relate(*b*, '1*T***T**')

c) Otherwise, the null value

When determining if two geometries intersect, z and m coordinate values are ignored in the calculations.

4.2.2.4 GML Support for ST_Geometry

An ST_Geometry value may be represented as an XML element in the GML representation. This support does not include support for an XML document using any or all of the XML schemas defined in GML. The provided routines may be used to create ST_Geometry values from specific XML elements from the Geometry Schemas and to generate XML elements from ST_Geometry values. The mapping is defined in Table 14 — Mapping between ST_Geometry values and GML representation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

4.2.3 ST_Point

The ST_Point type is a subtype of ST_Geometry. The ST_Point type is instantiable. An ST_Point value is a 0-dimensional geometry and represents a single location. An ST_Point has an x coordinate value, a y coordinate value, an optional z coordinate value, and an optional m coordinate value. The boundary of an ST_Point value is the empty set. ST_Point values are simple.

4.2.3.1 Methods on ST_Point

1) ST_Point: returns an ST_Point value constructed from either:
   a) the well-known text representation of an ST_Point value;
   b) the well-known binary representation of an ST_Point value;
   c) the GML representation of an ST_Point value;
   d) the specified coordinate values.
2) ST_X: observes and mutates the x coordinate value of an ST_Point value.
3) ST_Y: observes and mutates the y coordinate value of an ST_Point value.
4) ST_Z: observes and mutates the z coordinate value of an ST_Point value.
5) ST_M: observes and mutates the m coordinate value of an ST_Point value.
6) ST_ExplicitPoint: returns the coordinate values as a DOUBLE PRECISION ARRAY value.

4.2.3.2 Functions on ST_Point

1) ST_PointFromText: returns an ST_Point value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_Point.
2) ST_PointFromWKB: returns an ST_Point value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_Point.
3) ST_PointFromGML: returns an ST_Point value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_Point value.

4.2.4 ST_Curve

The ST_Curve type is a subtype of ST_Geometry. The ST_ Curve type is not instantiable. An ST_Curve value is a 1-dimensional geometry usually stored as a sequence of points. The subtype of ST_Curve specifies the form of the interpolation between points.

Topologically, an ST_Curve value is a 1-dimensional geometry that is the homomorphic image of a real, closed interval. An ST_Curve value is not simple if any interior point has the same location as another interior point or a point in the boundary.

An ST_Curve value is closed if its start point is equal to its end point. The boundary of a closed ST_Curve value is the empty set. An ST_Curve value that is simple and closed is called a *ring*. The boundary of a non-closed ST_Curve value consists of its start point and its end point. An ST_Curve value is defined to be topologically closed.

4.2.4.1 Methods on ST_Curve

1) ST_Length: returns the length of an ST_Curve value, ignoring z and m coordinate values in the calculations.
2) ST_3DLength: returns the length of an ST_Curve value, considering z coordinate values in the calculations.
3) ST_StartPoint: returns the ST_Point value that is the start point of an ST_Curve value including existing z and m coordinate values in the resultant geometry.
4) ST_EndPoint: returns the ST_Point value that is the end point of an ST_Curve value including existing z and m coordinate values in the resultant geometry.
5) ST_IsClosed: tests if an ST_Curve value is closed, ignoring z and m coordinate values in the calculations.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6) ST_3DIsClosed: tests if an ST_Curve value is closed, considering z and m coordinate values in the calculations.

7) ST_IsRing: tests if an ST_Curve value is a ring, ignoring z coordinate values in the calculations.

8) ST_3DIsRing: tests if an ST_Curve value is a ring, considering z coordinate values in the calculations.

9) ST_CurveToLine: returns the ST_LineString value approximating the ST_Curve value, considering z coordinate values in the calculations and including existing z coordinate values in the resultant geometry.

4.2.5 ST_LineString

The ST_LineString type is a subtype of ST_Curve. The ST_LineString type is instantiable. An ST_LineString value has linear interpolation between ST_Point values. Each consecutive pair of ST_Point values defines a line segment. A line is an ST_LineString value with exactly two points. A linear ring is an ST_LineString value that is both closed and simple.

4.2.5.1 Methods on ST_LineString

1) ST_LineString: returns an ST_LineString value constructed from either:
   a) the well-known text representation of an ST_LineString value;
   b) the well-known binary representation of an ST_LineString value;
   c) the GML representation of an ST_LineString value;
   d) the specified ST_Point values.
2) ST_Points: observes and mutates the ST_Point collection in the ST_LineString value.
3) ST_NumPoints: returns the cardinality of the ST_Point collection in the ST_LineString value.
4) ST_PointN: returns the specified element in the ST_Point collection in the ST_LineString value.

4.2.5.2 Functions on ST_LineString

1) ST_LineFromText: returns an ST_LineString value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_LineString.
2) ST_LineFromWKB: returns an ST_LineString value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_LineString.
3) ST_LineFromGML: returns an ST_LineString value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_LineString value.

4.2.6 ST_CircularString

The ST_CircularString type is a subtype of ST_Curve. The ST_CircularString type is instantiable. An ST_CircularString value has circular interpolation between ST_Point values. This subtype of ST_Curve consists of one or more circular arc segments connected end to end. The first three points define the first segment. The first point is the start point of the arc. The second point is any intermediate point on the arc other than the start or end point. The third point is the end point of the arc. Subsequent segments are defined by their intermediate and end points only, as the start point is implicitly defined as the previous segment's end point. In the special case where a segment is a complete circle, that is, the start and end points are coincident, then the intermediate point shall be the midpoint of the segment.

Let *CHORD1* be the line connecting the start point of a circular arc segment and the intermediate point on the segment. Let *CHORD2* be the line connecting the intermediate point with the end point of this arc segment. The center of the circular arc segment is located at the intersection of the perpendicular bisectors of *CHORD1* and *CHORD2*. In the case where the segment is a circle, then the center is located instead at the midpoint of the line connecting the start point with the intermediate point.

Let distance *R* be the radius of the circular arc segment, equal to the distance from the center of the circular arc segment to the start, intermediate, or end points.

The circular arc segment is the locus of points a distance *R* from the center of the arc, beginning at the start point and ending at the end point of the circular arc segment.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

If the start, intermediate, and end points of an arc segment are collinear, then the resultant arc segment degenerates to a straight line for which center and radius are not defined. In this case, the circular arc segment is the locus of points defined by the straight line connecting the start and end points.

An ST_CircularString value with exactly three points is a circular arc. A circular ring is an ST_CircularString value that is both closed and simple.

4.2.6.1 Methods on ST_CircularString

1) ST_CircularString: returns an ST_CircularString value constructed from either:
   a) the well-known text representation of an ST_CircularString value;
   b) the well-known binary representation of an ST_CircularString value;
   c) the GML representation of an ST_CircularString value;
   d) the specified ST_Point values.
2) ST_Points: observes and mutates the ST_Point collection in the ST_CircularString value.
3) ST_NumPoints: returns the cardinality of the ST_Point collection in the ST_CircularString value.
4) ST_PointN: returns the specified element in the ST_Point collection in the ST_CircularString value.
5) ST_MidPointRep: returns the array of points which identify an ST_CircularString value including start, mid, and end points for each curve segment.

4.2.6.2 Functions on ST_CircularString

1) ST_CircularFromTxt: returns an ST_CircularString value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_CircularString.
2) ST_CircularFromWKB: returns an ST_CircularString value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_CircularString.
3) ST_CircularFromGML: returns an ST_CircularString value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_CircularString.

4.2.7 ST_CompoundCurve

The ST_CompoundCurve type is a subtype of ST_Curve. The ST_CompoundCurve type is instantiable. The general notion of a compound curve is a sequence of contiguous curves such that adjacent curves are joined at their end points. The end point of each curve shall be equal to the start point of the next curve in the list.

4.2.7.1 Methods on ST_CompoundCurve

1) ST_CompoundCurve: returns an ST_CompoundCurve value constructed from either:
   a) the well-known text representation of an ST_CompoundCurve value;
   b) the well-known binary representation of an ST_CompoundCurve value;
   c) the GML representation of an ST_CompoundCurve value;
   c) the specified ST_Curve values.
2) ST_Curves: observes and mutates the ST_Curve collection in the ST_CompoundCurve value.
3) ST_NumCurves: returns the cardinality of the ST_Curve collection in the ST_CompoundCurve value.
4) ST_CurveN: returns the specified element in the ST_Curve collection in the ST_CompoundCurve value.

4.2.7.2 Functions on ST_CompoundCurve

1) ST_CompoundFromTxt: returns an ST_CompoundCurve value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_CompoundCurve.
2) ST_CompoundFromWKB: returns an ST_CompoundCurve value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_CompoundCurve.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3) ST_CompoundFromGML: returns an ST_CompoundCurve value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_CompoundCurve value.

4.2.8 ST_Surface

The ST_Surface type is a subtype of ST_Geometry. The ST_Surface type is not instantiable. An ST_Surface value is a 2-dimensional geometry that consists of a single connected interior that is associated with one exterior ring and zero or more interior rings. Surfaces in three-dimensional coordinate space are isomorphic to planar surfaces. Polyhedral surfaces are formed by stitching together simple surfaces along their boundaries, Polyhedral surfaces in three-dimensional coordinate space may not be planar.

The boundary of a simple surface is the set of closed curves corresponding to its exterior and interior rings.

4.2.8.1 Methods on ST_Surface

1) ST_Area: returns the area of an ST_Surface value, ignoring z and m coordinate values in the calculations.
2) ST_3DArea: returns the area of an ST_Surface value, considering z coordinate values in the calculations.
3) ST_Perimeter: returns the length of the boundary of an ST_Surface value, ignoring z and m coordinate values in the calculations.
4) ST_3DPerimeter: returns the length of the boundary of an ST_Surface value, considering z coordinate values in the calculations.
5) ST_Centroid: returns the ST_Point value that is the mathematical centroid of the ST_Surface value, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.
6) ST_3DCentroid: returns the ST_Point value that is the mathematical centroid of the ST_Surface value, considering z coordinate values in the calculations and including them in the resultant geometry.
7) ST_PointOnSurface: returns the ST_Point value that is guaranteed to intersect the ST_Surface value, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.
8) ST_3DPointOnSurf: returns the ST_Point value that is guaranteed to intersect the ST_Surface value, considering z coordinate values in the calculations and including them in the resultant geometry.
9) ST_IsWorld: test if the exterior of the ST_Surface value is the empty set, ignoring z and m coordinate values in the calculations.

4.2.9 ST_CurvePolygon

The ST_CurvePolygon type is a subtype of ST_Surface. The ST_CurvePolygon type is instantiable. An ST_CurvePolygon value is a planar surface consisting of a single patch, defined by one exterior boundary and zero or more interior boundaries. Each interior boundary defines a hole in the ST_CurvePolygon value.

ST_CurvePolygon values are topologically closed. The boundary of an ST_CurvePolygon consists of an exterior ring and zero or more interior rings. No two rings in the boundary cross. The rings in the boundary of an ST_CurvePolygon value may intersect at a point. An ST_CurvePolygon shall not have cut lines, spikes or punctures. The interior of every ST_CurvePolygon is a connected point set. The exterior of an ST_CurvePolygon with one or more holes is not connected. Each hole defines a disconnected component of the exterior.

ST_CurvePolygon values are simple.

4.2.9.1 Methods on ST_CurvePolygon

1) ST_CurvePolygon: returns an ST_CurvePolygon value constructed from either:
   a) the well-known text representation of an ST_CurvePolygon value;
   b) the well-known binary representation of an ST_CurvePolygon value;
   c) the GML representation of an ST_CurvePolygon value;
   d) the specified ST_Curve values.
2) ST_ExteriorRing: observes and mutates the exterior ring of an ST_CurvePolygon value.
3) ST_InteriorRings: observes and mutates the collection of interior rings of an ST_CurvePolygon value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

4) ST_NumInteriorRing: returns the cardinality of the collection of interior rings of an ST_CurvePolygon value.

5) ST_InteriorRingN: returns the specified element in the collection of interior rings of an ST_CurvePolygon value.

6) ST_CurvePolyToPoly: returns an ST_Polygon value approximating the ST_CurvePolygon value.

4.2.9.2 Functions on ST_CurvePolygon

1) ST_CPolyFromText: returns an ST_CurvePolygon value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_CurvePolygon.

2) ST_CPolyFromWKB: returns an ST_CurvePolygon value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_CurvePolygon.

3) ST_CPolyFromGML: returns an ST_CurvePolygon value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_CurvePolygon value.

4.2.10 ST_Polygon

The ST_Polygon type is a subtype of ST_CurvePolygon whose boundary is defined by linear rings. The ST_Polygon type is instantiable.

4.2.10.1 Methods on ST_Polygon

1) ST_Polygon: returns an ST_Polygon value constructed from either:

   a) the well-known text representation of an ST_Polygon value;

   b) the well-known binary representation of an ST_Polygon value;

   c) the GML representation of an ST_Polygon value;

   d) the specified ST_LineString values.

4.2.10.2 Functions on ST_Polygon

1) ST_PolyFromText: returns an ST_Polygon value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_Polygon.

2) ST_PolyFromWKB: returns an ST_Polygon value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_Polygon.

3) ST_PolyFromGML: returns an ST_Polygon value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_Polygon value.

4) ST_BdPolyFromText: returns an ST_Polygon value, which is built from a well-known text representation of an ST_MultiLineString.

5) ST_BdPolyFromWKB: returns an ST_Polygon value, which is built from a well-known binary representation of an ST_MultiLineString.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

4.2.11 ST_Triangle

The ST_Triangle type is a subtype of ST_Polygon with an exterior boundary having exactly four points (the last point being the same as the first point) and no interior boundaries. The ST_Triangle type is instantiable.

4.2.11.1 Methods on ST_Triangle

1) ST_Triangle: returns an ST_Triangle value constructed from either:
   a) the well-known text representation of an ST_Triangle value;
   b) the well-known binary representation of an ST_Triangle value;
   c) the GML representation of an ST_Triangle value;
   d) the specified ST_LineString value;
   e) the specified ST_Point values.
2) ST_Points: observes and mutates the four ST_Points in the ST_LineString exterior boundary of the ST_Triangle value.
3) ST_3DSlope: returns the slope of a triangle value.

4.2.11.2 Functions on ST_Triangle

1) ST_TriFromText: returns an ST_Triangle value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_Triangle.
2) ST_TriFromWKB: returns an ST_Triangle value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_Triangle.
3) ST_TriFromGML: returns an ST_Triangle value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_Triangle value.

4.2.12 ST_PolyhdrlSurface

The ST_PolyhdrlSurface type is a subtype of ST_Surface composed of contiguous polygon surfaces (ST_Polygon) connected along their common boundary curves. The ST_PolyhdrlSurface type is instantiable.

4.2.12.1 Methods on ST_PolyhdrlSurface

1) ST_PolyhdrlSurface: returns an ST_PolyhdrlSurface value constructed from either:
   a) the well-known text representation of an ST_PolyhdrlSurface value;
   b) the well-known binary representation of an ST_PolyhdrlSurface value;
   c) the GML representation of an ST_PolyhdrlSurface value;
   d) the specified ST_Polygon values.
2) ST_Patches: observes and mutates the ST_Polygon collection in the ST_PolyhdrlSurface value.
3) ST_NumPatches: returns the cardinality of the ST_Polygon collection in the ST_PolyhdrlSurface value.
4) ST_PatchN: returns the specified element in the ST_Polygon collection in the ST_PolyhdrlSurface value.

4.2.12.2 Functions on ST_PolyhdrlSurface

1) ST_PhSFromText: returns an ST_PolyhdrlSurface value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_PolyhdrlSurface.
2) ST_PhSFromWKB: returns an ST_PolyhdrlSurface value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_PolyhdrlSurface.
3) ST_PhSFromGML: returns an ST_PolyhdrlSurface value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_PolyhdrlSurface value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

4.2.13 ST_TIN

The ST_TIN type is a subtype of ST_PolyhdrlSurface composed only of triangles (ST_Triangle) that uses the Delaunay algorithm [3], or a similar implementation-defined algorithm, complemented with consideration for breaklines, soft breaks, control contours, break voids, drape voids, voids, holes, stop lines and maximum length of triangle sides. The ST_TIN type is instantiable.

4.2.13.1 Methods on ST_TIN

1) ST_TIN: returns an ST_TIN value constructed from either:
   a) the well-known text representation of an ST_TIN value;
   b) the well-known binary representation of an ST_TIN value;
   c) the GML representation of an ST_TIN value;
   d) the specified triangles (ST_Triangle values), TIN elements (ST_TINElement ARRAY value) and the DOUBLE PRECISION maximum allowable triangle side length;
   e) the specified TIN elements (ST_TINElement ARRAY value) and the DOUBLE PRECISION maximum allowable triangle side length.
2) ST_TINElements: observes and mutates the ST_TINElement ARRAY collection of TIN elements in the ST_TIN value.
3) ST_MaxSideLength: observes and mutates the DOUBLE PRECISION maximum allowable triangle side length in the ST_TIN value.
4) ST_TINTable: observes and mutates the ST_TIN value in table format with point references.
5) ST_Clip: returns that part of an ST_TIN value that is within the clipping boundary.
6) ST_Patches: observes and mutates the ST_Triangle collection in the ST_TIN value.

4.2.13.2 Functions on ST_TIN

1) ST_TINFromText: returns an ST_TIN value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_TIN.
2) ST_TINFromWKB: returns an ST_TIN value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_TIN.
3) ST_TINFromGML: returns an ST_TIN value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_TIN value.

4.2.14 ST_GeomCollection

The ST_GeomCollection type is a subtype of ST_Geometry. The ST_GeomCollection type is instantiable. An ST_GeomCollection is a collection of zero or more ST_Geometry values.

All the elements in an ST_GeomCollection are in the same spatial reference system. This is also the spatial reference system for the ST_GeomCollection value.

The ST_GeomCollection type places no other constraints on its elements. Subtypes of ST_GeomCollection may restrict membership based on dimension or place other constraints on the degree of spatial overlap between elements.

4.2.14.1 Methods on ST_GeomCollection

1) ST_GeomCollection: returns an ST_GeomCollection value constructed from either:
   a) the well-known text representation of an ST_GeomCollection value;
   b) the well-known binary representation of an ST_GeomCollection value;
   c) the GML representation of an ST_GeomCollection value;
   d) the specified ST_Geometry values.
2) ST_Geometries: observes and mutates the ST_Geometry collection in the ST_GeomCollection value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3) ST_NumGeometries: returns the cardinality of the ST_Geometry collection in the ST_GeomCollection value.

4) ST_GeometryN: returns the specified element in the ST_Geometry collection in the ST_GeomCollection value.

4.2.14.2 Functions on ST_GeomCollection

1) ST_GeomCollFromTxt: returns an ST_GeomCollection value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_GeomCollection.

2) ST_GeomCollFromWKB: returns an ST_GeomCollection value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_GeomCollection.

3) ST_GeomCollFromGML: returns an ST_GeomCollection value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_GeomCollection value.

4.2.15 ST_MultiPoint

The ST_MultiPoint type is a subtype of ST_GeomCollection. The ST_MultiPoint type is instantiable. An ST_MultiPoint value is a 0-dimensional geometry. The elements of an ST_MultiPoint value are restricted to ST_Point values. The ST_Point values are not connected or ordered. An ST_MultiPoint value is simple if and only if no two ST_Point values in the ST_MultiPoint value are equal. The boundary of an ST_MultiPoint is the empty set.

4.2.15.1 Methods on ST_MultiPoint

1) ST_MultiPoint returns an ST_MultiPoint value constructed from either:

   a) the well-known text representation of an ST_MultiPoint value;

   b) the well-known binary representation of an ST_MultiPoint value;

   c) the GML representation of an ST_MultiPoint value;

   d) the specified ST_Point values.

4.2.15.2 Functions on ST_MultiPoint

1) ST_MPointFromText: returns an ST_MultiPoint value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_MultiPoint.

2) ST_MPointFromWKB: returns an ST_MultiPoint value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_MultiPoint.

3) ST_MPointFromGML: returns an ST_MultiPoint value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_MultiPoint value.

4.2.16 ST_MultiCurve

The ST_MultiCurve type is a subtype of ST_GeomCollection. The ST_MultiCurve type may be instantiable. An ST_MultiCurve is a 1-dimensional geometry. The elements of an ST_MultiCurve value are restricted to ST_Curve values.

An ST_MultiCurve is simple if and only if all of its elements are simple and the only intersections between any two elements occur at points that are in the boundaries of both elements. The boundary of an ST_MultiCurve is obtained by applying the mod 2 union rule: an ST_Point value is in the boundary of an ST_MultiCurve if it is in the boundaries of an odd number of elements of the ST_MultiCurve value.

An ST_MultiCurve value is closed if all of its elements are closed. The boundary of a closed ST_MultiCurve is the empty set. An ST_MultiCurve value is defined to be topologically closed.

4.2.16.1 Methods on ST_MultiCurve

1) ST_MultiCurve: returns an ST_MultiCurve value constructed from either:

   a) the well-known text representation of an ST_MultiCurve value;

   b) the well-known binary representation of an ST_MultiCurve value;

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

c) the GML representation of an ST_MultiCurve value;

d) the specified ST_Curve values.

2) ST_IsClosed: tests if an ST_MultiCurve value is closed, ignoring z and m coordinate values in the calculations.

3) ST_3DIsClosed: tests if an ST_MultiCurve value is closed, considering z coordinate values in the calculations.

4) ST_Length: returns the length of an ST_MultiCurve value, ignoring z and m coordinate values in the calculations.

5) ST_3DLength: returns the 3D length of the ST_MultiCurve value, considering z coordinate values in the calculations.

4.2.16.2 Functions on ST_MultiCurve

1) ST_MCurveFromText: returns an ST_MultiCurve value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_MultiCurve.

2) ST_MCurveFromWKB: returns an ST_MultiCurve value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_MultiCurve.

3) ST_MCurveFromGML: returns an ST_MultiCurve value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_MultiCurve value.

4.2.17 ST_MultiLineString

The ST_MultiLineString type is a subtype of ST_MultiCurve. The ST_MultiLineString type is instantiable. The elements of an ST_MultiLineString value are restricted to ST_LineString values.

4.2.17.1 Methods on ST_MultiLineString

1) ST_MultiLineString: returns an ST_MultiLineString value constructed from either:

a) the well-known text representation of an ST_MultiLineString value;

b) the well-known binary representation of an ST_MultiLineString value;

c) the GML representation of an ST_MultiLineString value;

d) the specified ST_LineString values.

4.2.17.2 Functions on ST_MultiLineString

1) ST_MLineFromText: returns an ST_MultiLineString value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_MultiLineString.

2) ST_MLineFromWKB: returns an ST_MultiLineString value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_MultiLineString.

3) ST_MLineFromGML: returns an ST_MultiLineString value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_MultiLineString value.

4.2.18 ST_MultiSurface

The ST_MultiSurface type is a subtype of ST_GeomCollection. The ST_MultiSurface type may be instantiable. An ST_MultiSurface is a 2-dimensional geometry. The elements of an ST_MultiSurface value are restricted to ST_Surface values. The interiors of any two ST_Surface values in an ST_MultiSurface shall not intersect. The boundaries of any two elements in an ST_MultiSurface may intersect at a finite number of ST_Point values.

ST_MultiSurface values are simple.

4.2.18.1 Methods on ST_MultiSurface

1) ST_MultiSurface: returns an ST_MultiSurface value constructed from either:

a) the well-known text representation of an ST_MultiSurface value;

b) the well-known binary representation of an ST_MultiSurface value;

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

c) the GML representation of an ST_MultiSurface value;

d) the specified ST_Surface values.

2) ST_Area: returns the area of an ST_MultiSurface value, ignoring z and m coordinate values in the calculations.

3) ST_3DArea: returns the area of an ST_MultiSurface value, considering z coordinate values in the calculations.

4) ST_Perimeter: returns the length of the perimeter of an ST_MultiSurface value, ignoring z and m coordinate values in the calculations.

5) ST_3DPerimeter: returns the length of the perimeter of an ST_MultiSurface value, considering z coordinate values in the calculations.

6) ST_Centroid: returns the ST_Point value that is the mathematical centroid of the ST_MultiSurface value, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

7) ST_3DCentroid: returns the ST_Point value that is the mathematical centroid of the ST_MultiSurface value, considering z coordinate values in the calculations and including them in the resultant geometry.

8) ST_PointOnSurface: returns the ST_Point value that is guaranteed to intersect the ST_MultiSurface value, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

9) ST_3DPointOnSurf: returns the ST_Point value that is guaranteed to intersect the ST_MultiSurface value, considering z coordinate values in the calculations and including them in the resultant geometry.

4.2.18.2 Functions on ST_MultiSurface

1) ST_MSurfaceFromTxt: returns an ST_MultiSurface value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_MultiSurface.

2) ST_MSurfaceFromWKB: returns an ST_MultiSurface value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_MultiSurface.

3) ST_MSurfaceFromGML: returns an ST_MultiSurface value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_MultiSurface value.

4.2.19 ST_MultiPolygon

The ST_MultiPolygon type is a subtype of ST_MultiSurface. The ST_MultiPolygon type is instantiable. The elements of an ST_MultiPolygon value are restricted to ST_Polygon values. The interiors of any two ST_Polygon values that are elements of an ST_MultiPolygon shall not intersect. The boundaries of any two ST_Polygon values that are elements of an ST_MultiPolygon may touch at only a finite number of points.

An ST_MultiPolygon value shall not have cut lines, spikes or punctures. An ST_MultiPolygon value is a topologically closed point set. The interior of an ST_MultiPolygon value with more than one ST_Polygon value is not a connected point set. The number of disconnected components of the interior of an ST_MultiPolygon is equal to the number of ST_Polygon values in the ST_MultiPolygon. The boundary of an ST_MultiPolygon value is a set of linear rings corresponding to the boundaries of the ST_Polygon elements.

4.2.19.1 Methods on ST_MultiPolygon

1) ST_MultiPolygon: returns an ST_MultiPolygon value constructed from either:

a) the well-known text representation of an ST_MultiPolygon value;

b) the well-known binary representation of an ST_MultiPolygon value;

c) the GML representation of an ST_MultiPolygon value;

d) the specified ST_Polygon values.

4.2.19.2 Functions on ST_MultiPolygon

1) ST_MPolyFromText: returns an ST_MultiPolygon value, which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_MultiPolygon.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) ST_MPolyFromWKB: returns an ST_MultiPolygon value, which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_MultiPolygon.

3) ST_MPolyFromGML: returns an ST_MultiPolygon value, which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_MultiPolygon value.

4) ST_BdMPolyFromText: returns an ST_MultiPolygon value, which is built from a well-known text representation of an ST_MultiLineString.

5) ST_BdMPolyFromWKB: returns an ST_MultiPolygon value, which is built from a well-known binary representation of an ST_MultiLineString.

4.3 Topology-Geometry

A topology-geometry (Topo-Geo) is a model assuming full planar topology, comprised of nodes, edges, and faces. Multiple, independent topology-geometries can be defined for different geographic areas or for different, overlapping sets of features in the same geographic area, as for land parcels and wetlands. Topology-geometries are distinguished by unique names. This topology-geometry name is specified herein as <topology-name>. A separate SQL-schema is used for each topology-geometry and is named <topology-name>.

The following views are defined for each <topology-name> SQL-schema: <topology-name>.ST_NODE, <topology-name>.ST_EDGE, and <topology-name>.ST_FACE.

The rows in these views define topological primitives of type node, edge, and face, respectively for the Topo-Geo called <topology-name>. Nodes and edges have associated ST_Geometry value. Faces may have a minimum bounding rectangle geometry for spatial indexing. All geometry values for a given Topo-Geo shall have the same spatial reference system.

Each topological primitive has an ID, unique within the respective view, which allows the primitive to be referenced from another (e.g., feature or Topo-Geo) view.

The views provide the minimum information required to maintain a full planar topology. Support for additional information such as how these topological primitives are assigned to features or how real world attributes are assigned to topological primitives (e.g., edge weights) is not specified.

The ST_TOPO_GEO schema provides a suggested model of the base tables required to support the <topology-name> views. According to this model, all nodes from all topology-geometries reside in a single node base table, distinguished by a TOPOLOGY column containing <topology-name> values. Similarly, all edges reside in a single edge base table, and all faces reside in a single face base table, both with a distinguishing TOPOLOGY column.

4.3.1 <topology-name>.ST_NODE

The <topology-name>.ST_NODE view contains the node type of topological primitives (ST_Node) contained in the <topology-name> topology-geometry. An ST_Node has a known not nullable, unique node ID of type integer and a known not nullable geometry of type ST_Point. If ST_Node is an isolated node, it has a containing face, identified by a <topology-name>.ST_FACE.ID.

4.3.1.1 Routines on <topology-name>.ST_NODE only

1) ST_AddIsoNode: for the provided topology-name, optional face ID, and ST_Point geometry, inserts a row into the <topology-name>.ST_NODE view corresponding to an isolated node, returning the generated unique integer node ID. If a face ID is provided, the ST_Point geometry shall be within the geometry of the face or an exception is raised. If another node in the <topology-name>.ST_NODE view exists at the ST_Point location or if the geometry of an existing edge crosses the ST_Point location, an exception is raised.

2) ST_MoveIsoNode: for the provided topology-name, node ID, and ST_Point geometry, updates the existing ST_Point geometry value. If the node is a connected node or if another node in the <topology-name>.ST_NODE view exists at the new location or it the geometry of an existing edge crosses the new ST_Point location, an exception is raised.

3) ST_RemIsoNode: deletes the row for the isolated node identified by the provided topology-name and node ID. If the node is a connected node, an exception is raised.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

4.3.2 <topology-name>.ST_EDGE

The <topology-name>.ST_EDGE view contains the edge type of topological primitives (ST_Edge) contained in the <topology-name> Topology-geometry. An ST_Edge has a unique edge ID of type integer; node ID's of type integer for the start and end nodes; edge ID's of type integer for the next left face and next right face edges; face ID's of type integer for the left and right faces; and a geometry of type ST_Curve. All values are known not nullable. An isolated edge will have its containing face as both its left and right faces; the next right face edge will have an ID equal to the edge ID of the isolated edge and the next left face edge will have the negative of the isolated edge's ID.

Start and end node ID's are immutable. To change the start or end node of an edge, the edge shall be removed and a new one shall be created.

4.3.2.1 Routines on <topology-name>.ST_EDGE only

1) ST_AddIsoEdge: for the provided topology-name, start and end node IDs, and ST_Curve geometry, inserts a row into the <topology-name>.ST_EDGE view, returning the generated unique integer edge ID. The next right face edge is set equal to the new edge ID and the next left face edge is set equal to the negative of this value. The left and right face IDs are set equal to the containing face ID of the start and end node.

   An exception is raised for any of the following conditions:

   a) if the start and end node IDs do not correspond to existing isolated nodes,

   b) if the start and end node containing face IDs are not equal,

   c) if the ST_Point geometry of the start node does not equal the ST_StartPoint of the ST_Curve geometry of the edge,

   d) if the ST_Point geometry of the end node does not equal the ST_EndPoint of the ST_Curve geometry of the edge,

   e) if the ST_Curve geometry is not within the geometry of the containing face of the start and end nodes,

   f) if the ST_Curve geometry intersects the ST_Point geometry of any isolated node other than the start and end node,

   g) if the ST_Curve geometry intersects the ST_Curve geometry of any other edge in the <topology-name>.ST_EDGE view, or

   h) if the ST_Curve geometry is not simple.

2) ST_GetFaceEdges: for the provided topology-name and face ID, returns a table containing the integer edge IDs for the edges which bound the face, in counterclockwise order. Edge IDs will be negated in the query result if the face is right of the edge when looking in the direction of the edge from start to end node.

3) ST_ChangeEdgeGeom: for the provided topology-name, edge ID, and existing ST_Curve geometry, updates the ST_Curve geometry value.

   An exception is raised for any of the following conditions:

   a) if the ST_StartPoint of the new ST_Curve geometry is not equal to the ST_StartPoint of the existing ST_Curve geometry,

   b) if the ST_EndPoint of the new ST_Curve geometry is not equal to the ST_EndPoint of the existing ST_Curve geometry,

   c) if the interior of the new ST_Curve geometry intersects the ST_Point geometry of any isolated node in the <topology-name>.ST_NODE view,

   d) if the interior of the new ST_Curve geometry intersects the ST_Curve geometry of any other edge in the <topology-name>.ST_EDGE view, or

   e) if the ST_Curve geometry is not simple.

4) ST_RemIsoEdge: deletes the row for the isolated edge identified by the provided topology-name and edge ID. The start and end nodes are not removed from the <topology-name>.ST_NODE view. If the edge is a not an isolated edge, an exception is raised.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

4.3.2.2 Routines on <topology-name>.ST_NODE and <topology-name>.ST_EDGE

1) ST_NewEdgesSplit: splits an edge by creating a new node along an existing edge, deleting the original edge and replacing it with two new edges. For the provided topology-name, edge ID, and ST_Point geometry,

   a) inserts a row into the <topology-name>.ST_NODE view, with geometry equal to the input ST_Point value,

   b) returns the generated unique integer node ID,

   c) deletes the row in the <topology-name>.ST_EDGE view for the edge identified by the provided topology-name and edge ID, and

   d) inserts two rows into the <topology-name>.ST_EDGE view for the two new resultant edges, deriving appropriate node, edge, and face values from the deleted edge,

   e) creates ST_Curve geometries for the two new edges by splitting the geometry of the split edge at the ST_Point location,

   f) makes any necessary updates to the next left and right face edge IDs for any edges incident on the start and end nodes of the edge being split.

   To determine the two new edge IDs, query the <topology-name>.ST_EDGE view for edges with a start or end node equal to the returned node ID. Both new edges have the same direction as the edge being split. An exception is raised for any of the following conditions:

   a) if the edge identified by the edge ID does not exist in the <topology-name>.ST_EDGE view,

   b) if the ST_Point geometry is not within the ST_Curve geometry of the identified edge, or

   c) if a node already exists in the <topology-name>.ST_NODE view at the input ST_Point geometry location.

2) ST_ModEdgeSplit: splits an edge by creating a new node along an existing edge, modifying the original edge and adding a new edge. For the provided topology-name, edge ID, and ST_Point geometry,

   a) inserts a row into the <topology-name>.ST_NODE view, with geometry equal to the input ST_Point value,

   b) returns the generated unique integer node ID,

   c) modifies the row in the <topology-name>.ST_EDGE view for the edge identified by the provided topology-name and edge ID, deriving appropriate node, edge, and face values from the original edge and new node,

   d) inserts a new row into the <topology-name>.ST_EDGE view for the other new resultant edge, deriving appropriate node, edge, and face values from the original edge and new node,

   e) creates ST_Curve geometries for the new and modified edges by splitting the geometry of the original edge at the ST_Point location,

   f) makes any necessary updates to the next left and right face edge IDs for any edges incident on the start and end nodes of the edge being split.

   To determine the new edge ID, query the <topology-name>.ST_EDGE view for the edge with a start node equal to the returned node ID. The new and modified edges have the same direction as the original edge. An exception is raised for any of the following conditions:

   a) if the edge identified by the edge ID does not exist in the <topology-name>.ST_EDGE view,

   b) if the ST_Point geometry is not within the ST_Curve geometry of the identified edge, or

   c) if a node already exists in the <topology-name>.ST_NODE view at the input ST_Point geometry location.

3) ST_NewEdgeHeal: heals two edges by deleting the node connecting them, deleting both edges, and replacing them with a new edge whose direction is the same as the first edge provided. For the provided topology-name and two edge IDs,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

a) deletes the row in the <topology-name>.ST_NODE view corresponding to the node shared by the two identified edges,

b) deletes the two rows in the <topology-name>.ST_EDGE view identified by the input edge IDs,

c) inserts a new row into the <topology-name>.ST_EDGE view for the resultant edge, deriving appropriate node, edge, and face values from the deleted edges,

d) creates an ST_Curve geometry for the new edge from the geometries of the two deleted edges,

e) returns the generated unique integer edge ID for the new edge, and

f) makes any necessary updates to the next left and right face edge IDs for any edges incident on the start and end nodes of the edges being healed.

The direction of the new edge shall be the same as the direction of the first supplied edge. An exception is raised for any of the following conditions:

a) if either edge identified by the edge IDs does not exist in the <topology-name>.ST_EDGE view,

b) if the two edges do not share a common node, or

c) if additional edges also share the common node.

4) ST_ModEdgeHeal: heals two edges by deleting the node connecting them, modifying the first edge provided, and deleting the second edge. For the provided topology-name and two edge IDs,

a) deletes the row in the <topology-name>.ST_NODE view corresponding to the node shared by the two identified edges,

b) deletes the row in the <topology-name>.ST_EDGE view identified by the second input edge ID,

c) modifies the values in the row in the <topology-name>.ST_EDGE view for the other edge, deriving appropriate node, edge, and face values from the original edges,

d) creates an ST_Curve geometry for the modified edge from the geometries of the two original edges,

e) makes any necessary updates to the next left and right face edge IDs for any edges incident on the start and end nodes of the edges being healed.

An exception is raised for any of the following conditions:

a) if either edge identified by the edge IDs does not exist in the <topology-name>.ST_EDGE view,

b) if the two edges do not share a common node, or

c) if additional edges also share the common node.

4.3.3 <topology-name>.ST_FACE

The <topology-name>.ST_FACE view contains the face type of topological primitives (ST_Face) contained in the <topology-name> Topology-geometry. An ST_Face has a known not nullable, unique face ID of type integer and a possibly nullable MBR (minimum bounding rectangle) geometry of type ST_Polygon.

The <topology-name>.ST_FACE view contains a row for the universal face. The universal face contains everything else in the topology exterior to all other faces. This face has a face ID = 0 (zero). There is no geometry associated with the universal face.

4.3.3.1 Routines on <topology-name>.ST_EDGE and <topology-name>.ST_FACE

1) ST_AddEdgeNewFaces: adds a new edge and, if in doing so it splits a face, deletes the original face and replaces it with two new faces. For the provided topology-name, start and end node IDs, and ST_Curve geometry,

a) inserts a row into the <topology-name>.ST_EDGE view, with start and end nodes as specified, automatically determined next edges and left and right faces, and geometry equal to the input ST_Curve value,

b) returns the generated unique integer edge ID, and

c) if the new edge splits a face, then

i) deletes the row in the <topology-name>.ST_FACE view corresponding to the face being split,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) automatically generates two new unique integer face IDs,

iii) inserts two rows into the <topology-name>.ST_FACE view for the two new resultant faces,

iv) creates ST_Polygon MBR geometries for the two new faces, and

v) updates the appropriate next left and right face edge IDs and left and right face IDs for edges bounding the face being split.

To determine the two new face IDs, query the <topology-name>.ST_EDGE view for the left and right faces for the edge with the returned edge ID. An exception is raised for any of the following conditions:

a) if either the start or end nodes identified do not exist in the <topology-name>.ST_NODE view,

b) if the ST_StartPoint of the new ST_Curve geometry is not equal to the ST_Point value of the start node geometry,

c) if the ST_EndPoint of the new ST_Curve geometry is not equal to the ST_Point value of the end node geometry,

d) if the interior of the new ST_Curve geometry intersects the ST_Point geometry of any isolated node in the <topology-name>.ST_NODE view,

e) if the interior of the new ST_Curve geometry intersects the ST_Curve geometry of any other edge in the <topology-name>.ST_EDGE view,

f) if an edge already exists in the <topology-name>.ST_EDGE view with the same terminal nodes and geometry, or

g) if the ST_Curve geometry is not simple.

2) ST_AddEdgeModFace: adds a new edge and if in doing so it splits a face, modifies the original face and adds a new face. For the provided topology-name, start and end node IDs, and ST_Curve geometry,

a) inserts a row into the <topology-name>.ST_EDGE view, with start and end nodes as specified, automatically determined next edges and left and right faces, and geometry equal to the input ST_Curve value,

b) returns the generated unique integer edge ID, and

c) if the new edge splits a face, then

i) modifies the ST_Polygon MBR geometry in the <topology-name>.ST_FACE view for the face being split,

ii) inserts a new row into the <topology-name>.ST_FACE view for the other new resultant face,

iii) creates an ST_Polygon MBR geometry for the new face, and

iv) updates the appropriate next left and right face edge IDs and left and right face IDs for edges bounding the face being split.

To determine the new and modified face IDs, query the <topology-name>.ST_EDGE view for the left and right faces for the edge with the returned edge ID. An exception is raised for any of the following conditions:

a) if either the start or end nodes identified do not exist in the <topology-name>.ST_NODE view,

b) if the ST_StartPoint of the new ST_Curve geometry is not equal to the ST_Point value of the start node geometry,

c) if the ST_EndPoint of the new ST_Curve geometry is not equal to the ST_Point value of the end node geometry,

d) if the interior of the new ST_Curve geometry intersects the ST_Point geometry of any isolated node in the <topology-name>.ST_NODE view,

e) if the interior of the new ST_Curve geometry intersects the ST_Curve geometry of any other edge in the <topology-name>.ST_EDGE view,

f) if an edge already exists in the <topology-name>.ST_EDGE view with the same terminal nodes and geometry, or

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

g) if the ST_Curve geometry is not simple.

3) ST_RemEdgeNewFace: removes an edge and, if the removed edge separated two faces, deletes the original faces and replaces them with one new face. For the provided topology-name and edge ID,

a) deletes the row in the <topology-name>.ST_EDGE view identified by the edge ID, and

b) if the edge removal results in the healing of two faces, then

i) deletes the two rows in the <topology-name>.ST_FACE view corresponding to the faces being healed,

ii) inserts a new row into the <topology-name>.ST_FACE view for the new resultant face,

iii) creates an ST_Polygon MBR geometry for the new face,

iv) returns the generated unique integer face ID, and

v) updates the appropriate next left and right face edge IDs and left and right face IDs for edges bounding the faces being healed.

The start and end nodes of the deleted edge remain in the <topology-name>.ST_NODE view. An exception is raised if the edge identified by the edge ID does not exist in the <topology-name>.ST_EDGE view.

4) ST_RemEdgeModFace: removes an edge and, if the removed edge separated two faces, heals the two faces by modifying one of the faces and deleting the other. For the provided topology-name and edge ID,

a) deletes the row in the <topology-name>.ST_EDGE view identified by the edge ID, and

b) if the edge removal results in the healing of two faces, then

i) deletes the row in the <topology-name>.ST_FACE view corresponding to one of the faces being healed,

ii) creates a new ST_Polygon MBR geometry for the modified face,

iii) updates the appropriate next left and right face edge IDs and left and right face IDs for edges bounding the faces being healed.

The choice of which face to modify and which to delete is implementation-dependent.

The start and end nodes of the deleted edge remain in the <topology-name>.ST_NODE view. An exception is raised if the edge identified by the edge ID does not exist in the <topology-name>.ST_EDGE view.

5) ST_GetFaceGeometry: for the provided topology-name and face ID, returns the exact geometry of the face:

a) determines the edges in the <topology-name>.ST_Edge view which bound the face identified by the face ID

b) retrieves the ST_Curve geometries from the <topology-name>.ST_Edge view for each of these edges

c) returns a valid ST_Surface geometry value constructed from the edge geometries, if possible, or else an empty set of type ST_Surface.

An exception is raised if the face identified by the face ID does not exist in the <topology-name>.ST_FACE view, or if it is the universal face.

4.3.3.2 Routines on <topology-name>.ST_NODE, <topology-name>.ST_EDGE and <topology-name>.ST_FACE

1) ST_InitTopoGeo: for the provided topology-name, creates the <topology-name> schema, the ST_NODE, ST_EDGE, and ST_FACE views for this schema, and a row in the ST_FACE view contains the universal face. An exception is raised if a schema already exists with that name.

2) ST_CreateTopoGeo: for the provided topology-name, and ST_GeomCollection, populates the <topology-name>.ST_NODE, <topology-name>.ST_EDGE, and <topology-name>.ST_FACE views from the geometry values in the ST_GeomCollection. An exception is raised if any of these three views do not already exist or if they already contain any rows other than one for the universal face.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3) ST_ValidateTopoGeo: for the provided topology-name, returns a table containing possible topological inconsistencies.

4.4 Topology-Network

A topology-network (Topo-Net) is a model for linear applications including node and link topological primitives. Multiple, independent topology-networks can be defined for different geographic areas or for different, overlapping sets of features in the same geographic area, as for road and rail networks. Topology-networks are distinguished by unique names. This topology-network name is specified herein as <network-name>. A separate SQL-schema is used for each topology-network and is named <network-name>.

The following views are defined for each <network-name> SQL-schema: <network-name>.ST_NODE and <network-name>.ST_LINK.

The rows in these views define network primitives of type node and link, respectively for the Topo-Net called <network-name>. These network primitives may have associated ST_Geometry values. All geometry values for a given Topo-Net shall have the same spatial reference system.

Each topological primitive has an ID, unique within the respective view, which allows the primitive to be referenced from another (e.g., feature or Topo-Net) view.

The views provide the minimum information required to maintain a linear network. Support for additional information such as how these topological primitives are assigned to features or how real world attributes are assigned to topological primitives (e.g., link weights) is not specified.

The ST_TOPO_NET schema provides a suggested model of the base tables required to support the <network-name> views. According to this model, all nodes from all topology-networks reside in a single node base table, distinguished by a NETWORK column containing <network-name> values. Similarly, all links reside in a single link base table with a distinguishing NETWORK column.

4.4.1 <network-name>.ST_NODE

The <network-name>.ST_NODE view contains the node type of network primitives (ST_Node) contained in the <network-name> topology-network. An ST_Node has a known not nullable, unique node ID of type integer and a possibly nullable geometry of type ST_Point.

4.4.1.1 Routines on <network-name>.ST_NODE only

1) ST_AddIsoNetNode: for the provided network-name, and optional ST_Point geometry, inserts a row into the <network-name>.ST_NODE view, returning the generated unique integer node ID.
2) ST_MoveIsoNetNode: for the provided network-name, node ID, and ST_Point geometry, updates the ST_Point geometry value. If the node is a connected node exception is raised.
3) ST_RemIsoNetNode: deletes the row for the isolated node identified by the provided network-name and node ID. If the node is a connected node, an exception is raised.

4.4.2 <network-name>.ST_LINK

The <network-name>.ST_LINK view contains the link type of network primitives (ST_Link) contained in the <network-name> topology-network. An ST_Link has a known not nullable unique link ID of type integer; known not nullable node IDs of type integer for the start and end nodes; and a possibly nullable geometry of type ST_Curve.

Start and end node ID's are immutable. To change the start or end node of an link, the link shall be removed and a new one shall be created.

4.4.2.1 Routines on <network-name>.ST_LINK only

1) ST_AddLink: for the provided network-name, start and end node IDs, and optional ST_Curve geometry, inserts a row into the <network-name>.ST_LINK view, returning the generated unique integer link ID.

   An exception is raised for any of the following conditions:

   a) if the start and end nodes are not existing nodes in the <network-name>.ST_NODE view,

   b) if a non-null ST_Curve geometry value is provided, then

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

i) if the start node has a geometry and the location thereby specified does not equal the location of the ST_StartPoint of the proposed ST_Curve geometry of the link, or

ii) if the end node has a geometry and the location thereby specified does not equal the location of the ST_EndPoint of the proposed ST_Curve geometry of the link.

2) ST_ChangeLinkGeom: for the provided network-name, link ID, and ST_Curve geometry, updates the ST_Curve geometry value.

An exception is raised if a non-null ST_Curve geometry value is provided and any of the following are true:

a) the start node of the specified link has a null geometry value,

b) the start node of the specified link has a non-null geometry value and this location is not equal to the location of the start point of the ST_Curve value,

c) the end node of the specified link has a null geometry value, or

d) the end node of the specified link has a non-null geometry value and this location is not equal to the location of the end point of the ST_Curve value.

3) ST_RemoveLink: deletes the row for the link identified by the provided network-name and link ID. The start and end nodes are not removed from the <network-name>.ST_NODE view.

4.4.2.2 Routines on <network-name>.ST_NODE and <network-name>.ST_LINK

1) ST_InitTopoNet: for the provided network-name, creates the <network-name> schema and the ST_NODE and ST_LINK views for this schema. An exception is raised if a schema already exists with that name.

2) ST_NewLogLinkSplit: splits a link in a logical network by creating a new node along an existing link, deleting the original link and replacing it with two new links. For the provided network-name and link ID,

a) inserts a row into the <network-name>.ST_NODE view, with null values for Geometry,

b) returns the generated unique integer node ID,

c) deletes the row in the <network-name>.ST_LINK view for the link identified by the provided network-name and link ID, and

d) inserts two rows into the <network-name>.ST_LINK view for the two new resultant links, deriving appropriate start and end node IDs from the deleted link and newly generated node.

To determine the two new link IDs, query the <network-name>.ST_LINK view for links with a start or end node equal to the returned node ID. Both new links have the same direction as the link being split. An exception is raised if the link identified by the link ID does not exist in the <network-name>.ST_LINK view,

3) ST_ModLogLinkSplit: splits a logical network link by creating a new node along an existing link, modifying the original link and adding a new link. For the provided network-name and link ID,

a) inserts a row into the <network-name>.ST_NODE view, with null values for Geometry,

b) returns the generated unique integer node ID,

c) modifies the row in the <network-name>.ST_LINK view for the link identified by the provided network-name and link ID, deriving appropriate start and end node IDs from the original link and newly generated node, and

d) inserts a new row into the <network-name>.ST_LINK view for the other new resultant link, deriving appropriate start and end node IDs from the original link and newly generated node.

To determine the new link ID, query the <network-name>.ST_LINK view for the link with a start node equal to the returned node ID. The new and modified links have the same direction as the link being split. An exception is raised if the link identified by the link ID does not exist in the <network-name>.ST_LINK view,

4) ST_NewGeoLinkSplit: splits a link in a network with geometry by creating a new node along an existing link, deleting the original link and replacing it with two new links. For the provided network-name, link ID, and ST_Point geometry,

a) inserts a row into the <network-name>.ST_NODE, with geometry equal to the input ST_Point value,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

b) returns the generated unique integer node ID,

c) deletes the row in the <network-name>.ST_LINK for the link identified by the provided network-name and link ID,

d) inserts two rows into the <network-name>.ST_LINK for the two new resultant links, deriving appropriate start and end node IDs from the deleted link and newly generated node, and

e) creates ST_Curve geometries for the two new links by splitting the geometry of the split link at the ST_Point location.

To determine the two new link IDs, query the <network-name>.ST_LINK view for the link with a start node equal to the returned node ID. Both new links have the same direction as the link being split. An exception is raised for any of the following conditions:

a) if the link identified by the link ID does not exist in the <network-name>.ST_LINK,

b) if the ST_Point geometry is not within the ST_Curve geometry of the identified link,

c) if the link identified by the link ID contains a null geometry value.

5) ST_ ModGeoLinkSplit: splits a link in a network with geometry by creating a new node along an existing link, modifying the original link and adding a new link. For the provided network-name and link ID,

a) inserts a row into the <network-name>.ST_NODE view, with geometry equal to the input ST_Point value,

b) returns the generated unique integer node ID,

c) modifies the row in the <network-name>.ST_LINK view for the link identified by the provided network-name and link ID, deriving appropriate start and end node IDs from the original link and newly generated node,

d) inserts a new row into the <network-name>.ST_LINK view for the other new resultant link, deriving appropriate start and end node IDs from the original link and newly generated node, and

e) creates ST_Curve geometries for the new and modified links by splitting the geometry of the split link at the ST_Point location.

To determine the new link ID, query the <network-name>.ST_LINK view for links with a start or end node equal to the returned node ID. The new and modified links have the same direction as the link being split.

An exception is raised for any of the following conditions:

a) if the link identified by the link ID does not exist in the <network-name>.ST_LINK table,

b) if the ST_Point geometry is not within the ST_Curve geometry of the identified link,

c) if the link identified by the link ID contains a null geometry value.

6) ST_NewLinkHeal: heals two links by deleting the node connecting them, deleting both links, and replacing them with a new link, whose direction is the same as the first link provided. For the provided network-name and two link IDs,

a) if no other links start or end at the node shared by the two identified links, deletes the row in the <network-name>.ST_NODE view corresponding to the node shared by the two identified links,

b) deletes the two rows in the <network-name>.ST_LINK view identified by the input link IDs,

c) inserts a new row into the <network-name>.ST_LINK view for the resultant link, deriving appropriate start and end node IDs from the deleted links,

d) if the two links have geometry, creates an ST_Curve geometry for the new link from the geometries of the two deleted links,

e) returns the generated unique integer link ID for the new link.

The direction of the new link shall be the same as the direction of the first supplied link. An exception is raised for any of the following conditions:

a) if either link identified by the link IDs does not exist in the <network-name>.ST_LINK view, or

b) if the two links do not share a common node.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7) ST_ModLinkHeal: heals two links by deleting the node connecting them, modifying the first link and deleting the other. For the provided network-name and two link IDs,

   a) if no other links start or end at the node shared by the two identified links, deletes the row in the <network-name>.ST_NODE view corresponding to the node shared by the two identified links,

   b) deletes the row in the <network-name>.ST_LINK view identified by one of the input link IDs,

   c) modifies the values in the row in the <network-name>.ST_LINK view for the other link, deriving appropriate start and end node IDs from the original links,

   d) if the two links had geometry, creates a new ST_Curve geometry for the modified link from the geometries of the two original links.

   The original direction of the modified link is retained.

   An exception is raised for any of the following conditions:

   a) if either link identified by the link IDs does not exist in the <network-name>.ST_LINK view, or

   b) if the two links do not share a common node.

8) ST_LogiNetFromTGeo: for the provided network-name, and topology-name, creates a logical network by populating the <network-name>.ST_NODE and <network-name>.ST_LINK views from the Topo-Geo values identified by the provided topology-name. A Topo-Net node will be created for each Topo-Geo node. A Topo-Net link will be created for each Topo-Geo edge. A logical network will result, with nodes and links having geometry values set to the null value. An exception is raised if either of the two Topo-Net views do not already exist or if they already contain any rows or if any of the three Topo-Geo views do not exist.

9) ST_SpatNetFromTGeo: for the provided network-name, and topology-name, creates a spatial network by populating the <network-name>.ST_NODE and <network-name>.ST_LINK views from the Topo-Geo values identified by the provided topology-name. A Topo-Net node will be created for each Topo-Geo node. A Topo-Net link will be created for each Topo-Geo edge. A spatial network will result, with nodes and links having geometry values obtained from their corresponding Topo-Geo primitives. An exception is raised if either of the two Topo-Net views do not already exist or if they already contain any rows or if any of the three Topo-Geo views do not exist.

10) ST_SpatNetFromGeom: for the provided network-name, and ST_GeomCollection, creates a spatial network by populating the <network-name>.ST_NODE and <network-name>.ST_LINK views from the geometry values in the ST_GeomCollection. A node will be created wherever links start, end, or cross. A spatial network will result, with nodes and links having geometry values. An exception is raised if either of these two views do not already exist or if they already contain any rows.

11) ST_ValidLogicalNet: for the provided network-name, returns a table containing possible logical network inconsistencies.

12) ST_ValidSpatialNet: for the provided network-name, returns a table containing possible spatial network inconsistencies.

4.5 General Routines

4.5.1 ST_ShortestUndPath Function

A table function ST_ShortestUndPath calculates combinatorial geometric weighted distances between two specified points that are to be non-closed terminal points of an ST_Geometry value in a referenced table with undirected 1-dimensional simple geometry, and returns IDs of the shortest paths in form of a table.

4.5.2 ST_ShortestDirPath Function

A table function ST_ShortestDirPath calculates combinatorial geometric weighted distances between two specified points that are to be non-closed terminal points of an ST_Geometry value in a referenced table with directed 1-dimensional simple geometry, and returns IDs of the shortest paths in the form of a table.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

4.6 Spatial Reference System Type

4.6.1 ST_SpatialRefSys

The ST_SpatialRefSys type encapsulates all aspects of spatial reference systems.

4.6.1.1 Methods on ST_SpatialRefSys

1) ST_SpatialRefSys: returns the specified ST_SpatialRefSys value constructed from either:

a) the well-known text representation of spatial reference system;

b) a spatial reference system identifier.

2) ST_AsWKTSRS: returns the well-known text representation of a spatial reference system for the specified ST_SpatialRefSys value.

3) ST_WKTSRSToSQL: returns the ST_SpatialRefSys value represented by the specified well-known text representation of a spatial reference system.

4) ST_SRID: returns the integer identifier of an ST_SpatialRefSys value.

5) ST_Equals: tests if two ST_SpatialRefSys values are equal.

4.6.1.2 Ordering on ST_SpatialRefSys

1) ST_OrderingEquals: is the equals only ordering definition for the ST_SpatialRefSys type.

4.6.1.3 SQL Transforms on ST_SpatialRefSys

1) ST_WellKnownText: is the SQL Transform group that transforms an ST_SpatialRefSys value to and from a well-known text representation of a spatial reference system in a CHARACTER LARGE OBJECT value.

4.7 Angle and Direction Types

The following types are supported: ST_Angle and ST_Direction.

ST_Angle and ST_Direction are instantiable and have explicitly defined constructor functions.

Either of these types can be used as the type of a column. Declaring a column to be of a particular type implies that any value of the type or any of its subtypes can be used.

4.7.1 ST_Angle

The ST_Angle type is used to measure the degree of separation of two intersecting lines. The ST_Angle type is instantiable. The rotation (clockwise or counterclockwise) of the angle value represented by the ST_Angle type is not specified in order to maximize the applicability of this type across all disciplines.

4.7.1.1 Methods on ST_Angle

1) ST_Angle: returns a specified ST_Angle value from either:

a) radians, gradians, or degrees;

b) degrees and minutes;

c) degrees, minutes, and seconds;

d) three points, represented as ST_Point values, such that the angle is the lesser of the two possible rotations measured between the direction from the first point to the second point and the direction from the first point to the third point;

e) two directions, represented as ST_Direction values, where the angle is the lesser of the two possible rotations measured between the two directions;

f) two lines, represented as ST_LineString values, such that the angle specified is the lesser of the two possible rotations measured between the respective directions of the two lines, where the direction of a line is the direction from its start point to its end point;

g) a well-known text representation.

2) ST_Radians: observes and mutates the radians attribute of an ST_Angle value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3) ST_Degrees: observes and mutates the radians attribute of an ST_Angle value using decimal degrees.
4) ST_DegreeComponent: returns the integer value that represents the degrees part of the degrees, minutes, and seconds representation of the ST_Angle value.
5) ST_MinuteComponent: returns the integer value that represents the minutes part of the degrees, minutes, and seconds representation of the ST_Angle value.
6) ST_SecondComponent: returns the double precision value that represents the seconds part of the degrees, minutes, and seconds representation of the ST_Angle value.
7) ST_String: observes and mutates the radians attribute of an ST_Angle using a space separated string of degrees, minutes, and seconds.
8) ST_Gradians: observes and mutates the radians attribute of an ST_Angle value using gradians.
9) ST_Add: adds the value of an angle to the ST_Angle value.
10) ST_Subtract: subtracts the value of an angle from the ST_Angle value.
11) ST_Multiply: multiplies the ST_Angle value by a numeric value.
12) ST_Divide: divides the ST_Angle value by a non-zero, numeric value.
13) ST_AsText: returns the well-known tet representation of an ST_Angle value.

4.7.1.2 Ordering on ST_Angle

1) ST_OrderingCompare: defines full ordering for the ST_Angle type.

4.7.1.3 SQL Transforms on ST_Angle

1) ST_WellKnownText: is the SQL Transform group that transforms an ST_Angle value to and from a well-known text representation in a CHARACTER VARYING value.
2) ST_WellKnownBinary: is the SQL Transform group that transforms an ST_Angle value to and from a well-known binary representation in a DOUBLE PRECISION value.

4.7.2 ST_Direction

The ST_Direction type is used to express direction, expressible either as an azimuth or bearing. The ST_Direction type is instantiable.

4.7.2.1 Methods on ST_Direction

1) ST_Direction: returns a specified ST_Direction value from either:
   a) radians;
   b) 'N' (for North) or 'S' (for South), an angle, and 'E' (for East) or 'W' (for West);
   c) 'N' (for North) or 'S' (for South) and an angle;
   d) two points, represented as ST_Point values, such that the direction defined is the direction from the first point towards the second point;
   e) a line, represented as an ST_LineString value, such that the direction defined is the direction from the start point of the line to the end point of the line;
   f) a well-known text representation.
2) ST_Radians: returns the ST_Direction value as a DOUBLE PRECISION value in radians, representing clockwise rotation from True North.
3) ST_AngleNAzimuth: observes and mutates the ST_PrivateAngleNAzimuth attribute of an ST_Direction value.
4) ST_RadianBearing: observes the ST_Direction value represented as a bearing with its angle part expressed in radians.
5) ST_DegreesBearing: returns the ST_Direction value represented as a bearing with its angle part expressed in decimal degrees.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6) ST_DMSBearing: returns the ST_Direction value represented as a bearing with its angle part expressed in degrees, minutes, and seconds.
7) ST_RadianNAzimuth: returns the ST_Direction value represented as a North azimuth with its angle part expressed in radians.
8) ST_DegreesNAzimuth: returns the ST_Direction value represented as a North azimuth with its angle part expressed in decimal degrees.
9) ST_DMSNAzimuth: returns the ST_Direction value represented as a North azimuth with its angle part expressed in degrees, minutes, and seconds.
10) ST_RadianSAzimuth: returns the ST_Direction value represented as a South azimuth with its angle part expressed in radians.
11) ST_DegreesSAzimuth: returns the ST_Direction value represented as a South azimuth with its angle part expressed in decimal degrees.
12) ST_DMSSAzimuth: returns the ST_Direction value represented as a South azimuth with its angle part expressed in degrees, minutes, and seconds.
13) ST_AddAngle: mutates the ST_Direction value by adding an angle.
14) ST_SubtractAngle: mutates the ST_Direction value by subtracting an angle.
15) ST_AsText: returns the well-known tet representation of an ST_Direction value.

4.7.2.2 Ordering on ST_Direction

1) ST_OrderingCompare: defines full ordering for the ST_Direction type.

4.7.2.3 SQL Transforms on ST_Direction

1) ST_WellKnownText: is the SQL Transform group that transforms an ST_Direction value to and from a well-known text representation in a CHARACTER VARYING value.
2) ST_WellKnownBinary: is the SQL Transform group that transforms an ST_Direction value to and from a well-known binary representation in a DOUBLE PRECISION value.

4.8 Support Types

The following support types are defined: ST_TINElement.

ST_TINElement is instantiable and has constructor functions.

4.8.1 ST_TINElement

The ST_TINElement type is used to specify the information used to construct an ST_TIN surface. Element types include random points, group spot, boundary, breakline, soft break, control contour, break void, drape void, void, hole, stop line and user defined element types.

4.8.1.1 Methods on ST_TINElement

1) ST_TINElement: returns an ST_TINElement value constructed from:
   a) the specified element type, element ID, element tag and element geometry values.
2) ST_ElementType: observes and mutates the element type of an ST_TINElement value.
3) ST_ElementID: observes and mutates the element ID of an ST_TINElement value.
4) ST_ElementTag: observes and mutates the element tag of an ST_TINElement value.
5) ST_ElementGeometry: observes and mutates the element geometry of an ST_TINElement value.
6) ST_IsEmpty: tests if an ST_TINElement value corresponds to the empty set.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

4.9 Support Routines

4.9.1 ST_Geometry ARRAY Support Routines

4.9.1.1 Support Functions

1) ST_MaxDimension: returns the maximum geometric dimension value of the elements in an ST_Geometry ARRAY value.
2) ST_CheckSRID: if the elements in the ST_Geometry ARRAY value have mixed spatial reference systems, then raises an exception. Otherwise, the function returns the spatial reference system identifier of the elements of the ST_Geometry ARRAY value.
3) ST_GetCoordDim: checks the consistency of ST_Geometry values in an ST_Geometry ARRAY value with respect to the values returned by the ST_Is3D and ST_IsMeasured methods. If there is an inconsistency, then an exception is raised. Otherwise, all ST_Geometry values contained in the parameters have the same coordinate dimension, ST_Is3D value and ST_IsMeasured value. The ST_GetCoordDim function returns that coordinate dimension. If there are no ST_Geometry values in the parameter list, then the ST_GetCoordDim function returns the default value of 2.
4) ST_GetIs3D: returns the value for the ST_Is3D method which is consistent across all the ST_Geometry values in an ST_Geometry ARRAY value.
5) ST_GetIsMeasured: returns the value for the ST_IsMeasured method which is consistent across all the ST_Geometry values in an ST_Geometry ARRAY value.

4.9.1.2 Supporting Procedures

1) ST_CheckNulls: if an ST_Geometry ARRAY value is the null value, or if any of its elements is the null value or the empty set, then an exception is raised.
2) ST_CheckConsecDups: if an ST_Geometry ARRAY value has consecutive duplicate elements, then an exception is raised.

4.9.1.3 Supporting Cast Functions

1) ST_ToPointAry Cast: casts an ST_Geometry ARRAY value that contains only ST_Point valued elements to an ST_Point ARRAY value.
2) ST_ToCurveAry Cast: casts an ST_Geometry ARRAY value that contains only ST_Curve valued elements to an ST_Curve ARRAY value.
3) ST_ToLineStringAry Cast: casts an ST_Geometry ARRAY value that contains only ST_LineString valued elements to an ST_LineString ARRAY value.
4) ST_ToCircularAry Cast: casts an ST_Geometry ARRAY value that contains only ST_CircularString valued elements to an ST_CircularString ARRAY value.
5) ST_ToCompoundAry Cast: casts an ST_Geometry ARRAY value that contains only ST_CompoundCurve valued elements to an ST_CompoundCurve ARRAY value.
6) ST_ToSurfaceAry Cast: casts an ST_Geometry ARRAY value that contains only ST_Surface valued elements to an ST_Surface ARRAY value.
7) ST_ToCurvePolyAry Cast: casts an ST_Geometry ARRAY value that contains only ST_CurvePolygon valued elements to an ST_CurvePolygon ARRAY value.
8) ST_ToPolygonAry Cast: casts an ST_Geometry ARRAY value that contains only ST_Polygon valued elements to an ST_Polygon ARRAY value.
9) ST_ToTriangleAry Cast: casts an ST_Geometry ARRAY value that contains only ST_Triangle valued elements to an ST_Triangle ARRAY value.
10) ST_ToPolyhdrlAry Cast: casts an ST_Geometry ARRAY value that contains only ST_PolyhdrlSurface valued elements to an ST_PolyhdrlSurface ARRAY value.
11) ST_ToTINAry Cast: casts an ST_Geometry ARRAY value that contains only ST_TIN valued elements to an ST_TIN ARRAY value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

4.10 Tables with columns using geometry types

A table may be created with one or more columns that have a declared type of ST_Geometry or one of its subtypes. If a specific spatial reference system is associated with a column, then all geometry values in that column shall be in that specific spatial reference system. For base tables, constraints can be used to model this restriction. For derived tables, the derived table shall be defined in such a way as to ensure that all geometry values in the column will be in the spatial reference system that is associated with this column. The following is the general form of a constraint *CR* defined for a column *COL*, with a declared type of ST_Geometry or one of its subtypes, in the base table *TAB* in the schema *SCH* and the catalog *CAT*:

```
ALTER TABLE SCH.TAB
   ADD CONSTRAINT CR CHECK
      ( COL.ST_SRID() = COALESCE (
         ( SELECT SRS_ID
            FROM ST_INFORMTN_SCHEMA.ST_GEOMETRY_COLUMNS
            WHERE
               TABLE_CATALOG = CAT AND
               TABLE_SCHEMA  = SCH AND
               TABLE_NAME    = TAB AND
               COLUMN_NAME   = COL ),
           COL.ST_SRID()
         )
      )
```

4.11 The Spatial Information Schema

This part of ISO/IEC 13249 prescribes an Information Schema called ST_INFORMTN_SCHEMA. It contains views for the following purposes:

- a view ST_GEOMETRY_COLUMNS, which lists the columns whose declared type is ST_Geometry or one of its subtypes;
- a view ST_SPATIAL_REFERENCE_SYSTEMS, which lists the supported spatial reference systems;
- a view ST_UNITS_OF_MEASURE, which lists the supported units of measure;
- a view ST_SIZINGS, which lists implementation-defined meta-variables and their values.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5 Geometry Types

5.1 ST_Geometry Type and Routines

5.1.1 ST_Geometry Type

Purpose

The ST_Geometry type is the maximal supertype of the geometry type hierarchy. All subtypes have position specified in their attributes.

Definition

```
CREATE TYPE ST_Geometry
   AS (
      ST_PrivateDimension SMALLINT DEFAULT -1,
      ST_PrivateCoordinateDimension SMALLINT DEFAULT 2,
      ST_PrivateIs3D SMALLINT DEFAULT 0,
      ST_PrivateIsMeasured SMALLINT DEFAULT 0
   )
   NOT INSTANTIABLE
   NOT FINAL

   METHOD ST_Dimension()
      RETURNS SMALLINT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_CoordDim()
      RETURNS SMALLINT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_GeometryType()
      RETURNS CHARACTER VARYING(ST_MaxTypeNameLength)
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_SRID()
      RETURNS INTEGER
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_SRID
      (ansrid INTEGER)
      RETURNS ST_Geometry
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      CALLED ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_Transform
   (ansrid INTEGER)
   RETURNS ST_Geometry
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_IsEmpty()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_IsSimple()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DIsSimple()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_IsValid()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Is3D()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_IsMeasured()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_LocateAlong
   (measure DOUBLE PRECISION)
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_3DLocateAlong
   (measure DOUBLE PRECISION)
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DLocateBetween
   (start_measure DOUBLE PRECISION,
    end_measure DOUBLE PRECISION)
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_LocateBetween
   (start_measure DOUBLE PRECISION,
    end_measure DOUBLE PRECISION)
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Boundary()
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DBoundary()
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Envelope()
   RETURNS ST_Polygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_ConvexHull()
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Buffer
   (adistance DOUBLE PRECISION)
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_Buffer
    (adistance DOUBLE PRECISION,
     aunit CHARACTER VARYING(ST_MaxUnitNameLength))
    RETURNS ST_Geometry
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

METHOD ST_Intersection
    (ageometry ST_Geometry)
    RETURNS ST_Geometry
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

METHOD ST_3DIntersection
    (ageometry ST_Geometry)
    RETURNS ST_Geometry
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

METHOD ST_Union
    (ageometry ST_Geometry)
    RETURNS ST_Geometry
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

METHOD ST_3DUnion
    (ageometry ST_Geometry)
    RETURNS ST_Geometry
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

METHOD ST_Difference
    (ageometry ST_Geometry)
    RETURNS ST_Geometry
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

METHOD ST_3DDifference
    (ageometry ST_Geometry)
    RETURNS ST_Geometry
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_SymDifference
   (ageometry ST_Geometry)
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DSymDifference
   (ageometry ST_Geometry)
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Distance
   (ageometry ST_Geometry)
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Distance
   (ageometry ST_Geometry,
    aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DDistance
   (ageometry ST_Geometry)
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DDistance
   (ageometry ST_Geometry,
    aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Equals
   (ageometry ST_Geometry)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_3DEquals
    (ageometry ST_Geometry)
    RETURNS INTEGER
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

METHOD ST_Relate
    (ageometry ST_Geometry, amatrix CHARACTER(9))
    RETURNS INTEGER
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

METHOD ST_Disjoint
    (ageometry ST_Geometry)
    RETURNS INTEGER
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

METHOD ST_3DDisjoint
    (ageometry ST_Geometry)
    RETURNS INTEGER
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

METHOD ST_Intersects
    (ageometry ST_Geometry)
    RETURNS INTEGER
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

METHOD ST_3DIntersects
    (ageometry ST_Geometry)
    RETURNS INTEGER
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

METHOD ST_Touches
    (ageometry ST_Geometry)
    RETURNS INTEGER
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_Crosses
   (ageometry ST_Geometry)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Within
   (ageometry ST_Geometry)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Contains
   (ageometry ST_Geometry)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Overlaps
   (ageometry ST_Geometry)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_ToPoint()
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_ToLineString()
   RETURNS ST_LineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_ToCircular()
   RETURNS ST_CircularString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_ToCompound()
   RETURNS ST_CompoundCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_ToCurvePoly()
   RETURNS ST_CurvePolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_ToPolygon()
   RETURNS ST_Polygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_ToGeomColl()
   RETURNS ST_GeomCollection
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_ToMultiPoint()
   RETURNS ST_MultiPoint
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_ToMultiCurve()
   RETURNS ST_MultiCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_ToMultiLine()
   RETURNS ST_MultiLineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_ToMultiSurface()
   RETURNS ST_MultiSurface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_ToMultiPolygon()
   RETURNS ST_MultiPolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_WKTToSQL
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_AsText()
   RETURNS CHARACTER LARGE OBJECT(ST_MaxGeometryAsText)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_WKBToSQL
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_AsBinary()
   RETURNS BINARY LARGE OBJECT(ST_MaxGeometryAsBinary)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_GMLToSQL
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_AsGML()
   RETURNS CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

2) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

3) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

4) *ST_MaxTypeNameLength* is the implementation-defined maximum length used for the character string representation of a type name.

5) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

6) The attribute *ST_PrivateDimension* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivateDimension*.

© ISO/IEC 2011 – All rights reserved

7) The attribute *ST_PrivateCoordinateDimension* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivateCoordinateDimension*.

8) The attribute *ST_PrivateIs3D* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_Private3D*.

9) The attribute *ST_PrivateIsMeasured* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivateIsMeasured*.

Description

1) The *ST_Geometry* type provides for public use:
   a) a method *ST_Dimension()*,
   b) a method *ST_CoordDim()*,
   c) a method *ST_GeometryType()*,
   d) a method *ST_SRID()*,
   e) a method *ST_SRID(INTEGER)*,
   f) a method *ST_Transform(INTEGER)*,
   g) a method *ST_IsEmpty()*,
   h) a method *ST_IsSimple()*,
   i) a method *ST_3DIsSimple()*,
   j) a method *ST_IsValid()*,
   k) a method *ST_Is3D()*,
   l) a method *ST_IsMeasured()*,
   m) a method *ST_LocateAlong(DOUBLE PRECISION)*,
   n) a method *ST_3DLocateAlong(DOUBLE PRECISION)*,
   o) a method *ST_LocateBetween(DOUBLE PRECISION, DOUBLE PRECISION)*,
   p) a method *ST_3DLocateBetween(DOUBLE PRECISION, DOUBLE PRECISION)*,
   q) a method *ST_Boundary()*,
   r) a method *ST_3DBoundary()*,
   s) a method *ST_Envelope()*,
   t) a method *ST_ConvexHull()*,
   u) a method *ST_Buffer(DOUBLE PRECISION)*,
   v) a method *ST_Buffer(DOUBLE PRECISION, CHARACTER VARYING)*,
   w) a method *ST_Intersection(ST_Geometry)*,
   x) a method *ST_3DIntersection(ST_Geometry)*,
   y) a method *ST_Union(ST_Geometry)*,
   z) a method *ST_3DUnion(ST_Geometry)*,
   aa) a method *ST_Difference(ST_Geometry)*,
   ab) a method *ST_3DDifference(ST_Geometry)*,
   ac) a method *ST_SymDifference(ST_Geometry)*,
   ad) a method *ST_3DSymDifference(ST_Geometry)*,
   ae) a method *ST_Distance(ST_Geometry)*,
   af) a method *ST_Distance(ST_Geometry, CHARACTER VARYING)*,
   ag) a method *ST_3DDistance(ST_Geometry)*,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ah) a method *ST_3DDistance(ST_Geometry, CHARACTER VARYING)*,
ai) a method *ST_Equals(ST_Geometry)*,
aj) a method *ST_3DEquals(ST_Geometry)*,
ak) a method *ST_Relate(ST_Geometry, CHARACTER)*,
al) a method *ST_Disjoint(ST_Geometry)*,
am) a method *ST_3DDisjoint(ST_Geometry)*,
an) a method *ST_Intersects(ST_Geometry)*,
ao) a method *ST_3DIntersects(ST_Geometry)*,
ap) a method *ST_Touches(ST_Geometry)*,
aq) a method *ST_Crosses(ST_Geometry)*,
ar) a method *ST_Within(ST_Geometry)*,
as) a method *ST_Contains(ST_Geometry)*,
at) a method *ST_Overlaps(ST_Geometry)*,
au) a method *ST_WKTToSQL(CHARACTER LARGE OBJECT)*,
av) a method *ST_AsText()*,
aw) a method *ST_WKBToSQL(BINARY LARGE OBJECT)*,
ax) a method *ST_AsBinary()*,
ay) a method *ST_GMLToSQL(CHARACTER LARGE OBJECT)*,
az) a method *ST_AsGML()*,
ba) a function *ST_GeomFromText(CHARACTER LARGE OBJECT)*,
bb) a function *ST_GeomFromText(CHARACTER LARGE OBJECT, INTEGER)*,
bc) a function *ST_GeomFromWKB(BINARY LARGE OBJECT)*,
bd) a function *ST_GeomFromWKB(BINARY LARGE OBJECT, INTEGER)*,
be) a function *ST_GeomFromGML(CHARACTER LARGE OBJECT)*,
bf) a function *ST_GeomFromGML(CHARACTER LARGE OBJECT, INTEGER)*,
bg) an ordering function *ST_OrderingEquals*(*ST_Geometry, ST_Geometry)*,
bh) an SQL Transform group *ST_WellKnownText*,
bi) an SQL Transform group *ST_WellKnownBinary*,
bj) an SQL Transform group *ST_GML*,
bk) an implicit cast of an *ST_Geometry* value to an *ST_Point* value,
bl) an implicit cast of an *ST_Geometry* value to an *ST_LineString* value,
bm) an implicit cast of an *ST_Geometry* value to an *ST_CircularString* value,
bn) an implicit cast of an *ST_Geometry* value to an *ST_CompoundCurve* value,
bo) an implicit cast of an *ST_Geometry* value to an *ST_CurvePolygon* value,
bp) an implicit cast of an *ST_Geometry* value to an *ST_Polygon* value,
bq) an implicit cast of an *ST_Geometry* value to an *ST_GeomCollection* value,
br) an implicit cast of an *ST_Geometry* value to an *ST_MultiPoint* value,
bs) an implicit cast of an *ST_Geometry* value to an *ST_MultiCurve* value,
bt) an implicit cast of an *ST_Geometry* value to an *ST_MultiLineString* value,
bu) an implicit cast of an *ST_Geometry* value to an *ST_MultiSurface* value,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

bv) an implicit cast of an *ST_Geometry* value to an *ST_MultiPolygon* value.

2) The *ST_PrivateDimension* attribute contains the dimension of the *ST_Geometry* value:

   Case:

   a) If the *ST_Geometry* value corresponds to the empty set, then the dimension is -1.

   b) If the *ST_Geometry* value is a 0-dimensional geometry, then the dimension is 0 (zero).

   c) If the *ST_Geometry* value is a 1-dimensional geometry, then the dimension is 1 (one).

   d) If the *ST_Geometry* value is a 2-dimensional geometry, then the dimension is 2.

3) The *ST_PrivateCoordinateDimension* attribute contains the coordinate dimension of the *ST_Geometry* value.

4) The *ST_PrivateCoordinateDimension* attribute shall be:

   Case:

   a) 2 for *ST_Geometry* values in two-dimensional coordinate space (R^2).

   b) 3 for *ST_Geometry* values in three-dimensional coordinate space where:

      i) all points in the *ST_Geometry* value have x, y and z coordinate values, or

      ii) all points in the *ST_Geometry* value have x, y and m coordinate values.

   c) 4 for *ST_Geometry* values in four-dimensional coordinate space where all points in the *ST_Geometry* value have x, y, z, and m coordinate values.

5) Case:

   a) If all the points in the *ST_Geometry* value have z coordinate values, then *ST_PrivateIs3D* is 1 (one).

   b) Otherwise, the *ST_PrivateIs3D* is 0 (zero).

6) Case:

   a) If all the points in the *ST_Geometry* value have m coordinate values, then *ST_PrivateIsMeasured* is 1 (one).

   b) Otherwise, the *ST_PrivateIsMeasured* is 0 (zero).

7) The value of the *ST_PrivateDimension* attribute shall be less than or equal to the value of the *ST_PrivateCoordinateDimension* attribute.

8) All instantiable *ST_Geometry* subtypes are defined so that simple values of the geometry type are topologically closed.

9) The coordinate dimension shall be the same as the coordinate dimension of the spatial reference system for the *ST_Geometry* value.

10) An *ST_Geometry* value has an associated spatial reference system specified by a spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.2 ST_Dimension Method

Purpose

Return the dimension of the ST_Geometry value.

Definition

```
CREATE METHOD ST_Dimension()
   RETURNS SMALLINT
   FOR ST_Geometry
   RETURN SELF.ST_PrivateDimension
```

Description

1) The method *ST_Dimension()* has no input parameters.
2) The null-call method *ST_Dimension()* returns the value of the *ST_PrivateDimension* attribute.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.3 ST_CoordDim Method

Purpose

Return the coordinate dimension of the ST_Geometry value.

Definition

```
CREATE METHOD ST_CoordDim()
   RETURNS SMALLINT
   FOR ST_Geometry
   RETURN SELF.ST_PrivateCoordinateDimension
```

Description

1) The method *ST_CoordDim()* has no input parameters.
2) The null-call method *ST_CoordDim()* returns the value of the *ST_PrivateCoordinateDimension* attribute.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.4 ST_GeometryType Method

Purpose

Return the geometry type of the ST_Geometry value.

Definition

```
CREATE METHOD ST_GeometryType()
   RETURNS CHARACTER VARYING(ST_MaxTypeNameLength)
   FOR ST_Geometry
   RETURN
      CASE
         WHEN SELF IS OF (ST_Point) THEN
            'ST_Point'
         WHEN SELF IS OF (ST_LineString) THEN
            'ST_LineString'
         WHEN SELF IS OF (ST_CircularString) THEN
            'ST_CircularString'
         WHEN SELF IS OF (ST_CompoundCurve) THEN
            'ST_CompoundCurve'
         WHEN SELF IS OF (ST_Triangle) THEN
            'ST_Triangle'
         WHEN SELF IS OF (ST_Polygon) THEN
            'ST_Polygon'
         WHEN SELF IS OF (ST_CurvePolygon) THEN
            'ST_CurvePolygon'
         WHEN SELF IS OF (ST_TIN) THEN
            'ST_TIN'
         WHEN SELF IS OF (ST_PolyhdrlSurface) THEN
            'ST_PolyhdrlSurface'
         WHEN SELF IS OF (ST_GeomCollection) THEN
            CASE
               WHEN SELF IS OF (ST_MultiPoint) THEN
                  'ST_MultiPoint'
               WHEN SELF IS OF (ST_MultiLineString) THEN
                  'ST_MultiLineString'
               WHEN SELF IS OF (ST_MultiCurve) THEN
                  'ST_MultiCurve'
               WHEN SELF IS OF (ST_MultiPolygon) THEN
                  'ST_MultiPolygon'
               WHEN SELF IS OF (ST_MultiSurface) THEN
                  'ST_MultiSurface'
               ELSE
                  'ST_GeomCollection'
               END
         -- ELSE
            --
            -- See Description
            --
      END
```

Definitional Rules

1) *ST_MaxTypeNameLength* is the implementation-defined maximum length used for the character string representation of a type name.

Description

1) The method *ST_GeometryType()* has no input parameters.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) For the null-call method *ST_GeometryType()*:

Case:

a) If SELF is of type *ST_Point*, then return the value: 'ST_Point'.

b) If SELF is of type *ST_LineString*, then return the value: 'ST_LineString'.

c) If SELF is of type *ST_CircularString*, then the value 'ST_CircularString'.

d) If SELF is of type *ST_CompoundCurve*, then the value 'ST_CompoundCurve'.

e) If SELF is of type *ST_Triangle*, then return the value: 'ST_Triangle'.

f) If SELF is of type *ST_Polygon*, then return the value: 'ST_Polygon'.

g) If SELF is of type *ST_CurvePolygon*, then return the value: 'ST_CurvePolygon'.

h) If SELF is of type *ST_TIN*, then return the value: 'ST_TIN'.

i) If SELF is of type *ST_PolyhdrlSurface*, then return the value: 'ST_PolyhdrlSurface'.

j) If SELF is of type *ST_GeomCollection*, then:

Case:

i) If SELF is of type *ST_MultiPoint*, then return the value 'ST_MultiPoint'.

ii) If SELF is of type *ST_MultiLineString*, then return the value 'ST_MultiLineString'.

iii) If SELF is of type *ST_MultiCurve*, then return the value 'ST_MultiCurve'.

iv) If SELF is of type *ST_MultiPolygon*, then return the value 'ST_MultiPolygon'.

v) If SELF is of type *ST_MultiSurface*, then return the value 'ST_MultiSurface'.

vi) Otherwise, return the value: 'ST_GeomCollection'.

k) Otherwise, the method *ST_GeometryType()* returns an implementation-defined CHARACTER VARYING value for a user-defined type not defined in this part of ISO/IEC 13249.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.5 ST_SRID Methods

Purpose

Observe and mutate the spatial reference system identifier of the ST_Geometry value.

Definition

```
CREATE METHOD ST_SRID()
   RETURNS INTEGER
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END

CREATE METHOD ST_SRID
   (ansrid INTEGER)
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_SRID()* has no input parameters.
2) The null-call method *ST_SRID()* returns the spatial reference system identifier for the *ST_Geometry* value.
3) The method *ST_SRID(INTEGER)* takes the following input parameters:
   a) an INTEGER value *ansrid*.
4) The parameter *ansrid* is a spatial reference system identifier.
5) The type preserving method *ST_SRID(INTEGER)* returns an *ST_Geometry* value with the spatial reference system identifier set to *ansrid*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.6 ST_Transform Method

Purpose

Return an ST_Geometry value transformed to the specified spatial reference system, considering z and m coordinate values in the calculations and including them in the resultant geometry.

Definition

```
CREATE METHOD ST_Transform
   (ansrid INTEGER)
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_Transform(INTEGER)* takes the following input parameters:
   a) an INTEGER value *ansrid*.
2) The parameter *ansrid* is a spatial reference system identifier.
3) For the null-call type preserving method *ST_Transform(INTEGER)*:

   Case:

   a) If the spatial reference system identifier of SELF is equal to *ansrid*, then return SELF.
   b) If SELF is an empty set, then return SELF with the spatial reference system identifier equal to *ansrid*.
   c) If SELF cannot be transformed to the spatial reference system specified by *ansrid*, then an exception condition is raised: *SQL/MM Spatial exception – failed to transform geometry*
   d) Otherwise, return an *ST_Geometry* value as the result of an implementation-defined transform of SELF from the spatial reference system of SELF to the spatial reference system specified by *ansrid*. The value returned has the spatial reference system identifier equal to *ansrid*.
4) If SELF.ST_SRID() supports z or m coordinate values and the geometry of SELF contains ST_Point values with z or m coordinate values, then the spatial reference system specified by *ansrid* must also support z or m coordinate values or else SELF cannot be transformed to the spatial reference system specified by *ansrid*.
5) If SELF.ST_Is3D() is equal to 1 (one), then:
   a) The z coordinate values are considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Geometry* value include the z coordinate values that were included when SELF was specified.
6) If SELF.ST_IsMeasured() is equal to 1 (one), then:
   a) The m coordinate values are considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Geometry* value include the m coordinate values that were included when SELF was specified.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.7 ST_IsEmpty Method

Purpose

Test if an ST_Geometry value corresponds to the empty set.

Definition

```
CREATE METHOD ST_IsEmpty()
   RETURNS INTEGER
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_IsEmpty()* has no input parameters.
2) For the null-call method *ST_IsEmpty()*:

   Case:

   a) If the *ST_Geometry* value corresponds to the empty set, then return 1 (one).

   b) Otherwise, return 0 (zero).
3) The Description sections in the subclauses defining the subtypes of *ST_Geometry* include the specific conditions that cause a value of these types to correspond to the empty set.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.8 ST_IsSimple Method

Purpose

Test if an ST_Geometry value has no anomalous geometric points, such as self intersection, ignoring z coordinate values in the determination.

Definition

```
CREATE METHOD ST_IsSimple()
   RETURNS INTEGER
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_IsSimple()* has no input parameters.
2) For the null-call method *ST_IsSimple()*:

   Case:

   a) If SELF is an empty set, then return 1 (one).
   b) If the *ST_Geometry* value is simple, then return 1 (one).
   c) Otherwise, return 0 (zero).
3) The Description sections in the subclauses defining the subtypes of *ST_Geometry* include the specific conditions that cause a value of these types to be classified as simple.
4) If SELF.*ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the determination.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.9 ST_3DIsSimple Method

Purpose

Test if an ST_Geometry value has no anomalous geometric points, such as self intersection, considering z coordinate values in the determination.

Definition

```
CREATE METHOD ST_3DIsSimple()
   RETURNS INTEGER
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_3DIsSimple()* has no input parameters.
2) For the null-call method *ST_3DIsSimple()*:

   Case:

   a) If SELF is an empty set, then return 1 (one).
   b) If the *ST_Geometry* value is simple, then return 1 (one).
   c) Otherwise, return 0 (zero).
3) The Description sections in the subclauses defining the subtypes of *ST_Geometry* include the specific conditions that cause a value of these types to be classified as simple.
4) If SELF.ST_Is3D() is equal to 1 (one), then the z coordinate values are considered in the determination.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.10 ST_IsValid Method

Purpose

Test if an ST_Geometry value is well formed.

Definition

```
CREATE METHOD ST_IsValid()
   RETURNS INTEGER
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_IsValid()* has no input parameters.
2) For the null-call method *ST_IsValid()*:

   Case:

   a) If SELF is an empty *set*, then return 1 (one).
   b) If the *ST_Geometry* value is well formed, then return 1 (one).
   c) Otherwise, return 0 (zero).
3) The Description sections in the subclauses defining the subtypes of *ST_Geometry* include the specific conditions that cause a value of these types to be classified as well formed.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.11 ST_Is3D Method

Purpose

Test if an ST_Geometry value has z coordinate values.

Definition

```
CREATE METHOD ST_Is3D()
   RETURNS INTEGER
   FOR ST_Geometry
   RETURN SELF.ST_PrivateIs3D
```

Description

1) The method *ST_Is3D()* has no input parameters.
2) The null-call method *ST_Is3D()* returns the value of the *ST_PrivateIs3D* attribute.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.12 ST_IsMeasured Method

Purpose

Test if an ST_Geometry value has m coordinate values.

Definition

```
CREATE METHOD ST_IsMeasured()
   RETURNS INTEGER
   FOR ST_Geometry
   RETURN SELF.ST_PrivateIsMeasured
```

Description

1) The method *ST_IsMeasured()* has no input parameters.
2) The null-call method *ST_IsMeasured()* returns the value of the *ST_PrivateIsMeasured* attribute.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.13 ST_LocateAlong Method

Purpose

Return a derived geometry value that matches the specified measure, ignoring z coordinate values in the calculations and not including them in the resultant geometry.

Definition

```
CREATE METHOD ST_LocateAlong
   (measure DOUBLE PRECISION)
   RETURNS ST_Geometry
   FOR ST_Geometry
   RETURN ST_LocateBetween(measure, measure)
```

Description

1) The method *ST_LocateAlong(DOUBLE PRECISION)* takes the following input parameters:

   a) a DOUBLE PRECISION value *measure*.

2) The null-call method *ST_LocateAlong(DOUBLE PRECISION)* returns the result of the value expression: *ST_LocateBetween(measure, measure)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.14 ST_3DLocateAlong Method

Purpose

Return a derived geometry value that matches the specified measure, considering z coordinate values in the calculations and including them in the resultant geometry.

Definition

```
CREATE METHOD ST_3DLocateAlong
   (measure DOUBLE PRECISION)
   RETURNS ST_Geometry
   FOR ST_Geometry
   RETURN ST_3DLocateBetween(measure, measure)
```

Description

1) The method *ST_3DLocateAlong(DOUBLE PRECISION)* takes the following input parameters:
   a) a DOUBLE PRECISION value *measure*.
2) The null-call method *ST_3DLocateAlong(DOUBLE PRECISION)* returns the result of the value expression: *ST_3DLocateBetween(measure, measure)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.15 ST_LocateBetween Method

Purpose

Return a derived geometry collection value with elements that match the specified range of measures inclusively, ignoring z coordinate values in the calculations and not including them in the resultant geometry.

Definition

```
CREATE METHOD ST_LocateBetween
   (start_measure DOUBLE PRECISION,
    end_measure DOUBLE PRECISION)
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_LocateBetween(DOUBLE PRECISION, DOUBLE PRECISION)* takes the following input parameters:

   a) a DOUBLE PRECISION value *start_measure*,

   b) a DOUBLE PRECISION value *end_measure*.

2) For the null-call method *ST_LocateBetween(DOUBLE PRECISION, DOUBLE PRECISION)*:

   Case:

   a) If *end_measure* is less than *start_measure*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

   b) If SELF is an empty set, then return the null value.

   c) If *SELF.ST_IsMeasured*() is equal to 0 (zero), then return an empty *ST_Point* value.

   d) If *SELF.ST_Dimension()* is equal to 0 (zero), then

      Case:

      i) If SELF contains any *ST_Point* values with m coordinate values between *start_measure* and *end_measure* inclusively, then return an *ST_MultiPoint* value that contains the *ST_Point* values with m coordinate values between *start_measure* and *end_measure* inclusively.

      ii) Otherwise, return an empty *ST_Point* value.

   e) If *SELF.ST_Dimension()* is equal to 1, then use the implementation-defined interpolation algorithm to estimate values between *start_measure* and *end_measure* inclusively.

      Case:

      i) If the result does not contain any points, then return an empty *ST_Point* value.

      ii) If the *ST_GeomCollection* type is instantiable, then:

         Case:

         1) If the result only contains consecutive points with m coordinate values between *start_measure* and *end_measure* inclusively, then return an *ST_MultiCurve* value with elements of type *ST_Curve* are constructed to represent the curve elements comprised of these consecutive points.

         2) If the result only contains disconnected points with m coordinate values between *start_measure* and *end_measure* inclusively, then return an *ST_MultiPoint* value with elements of type *ST_Point* constructed to represent these points.

         3) Otherwise:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

A) Return an *ST_GeomCollection* value containing:

I) elements of type *ST_Curve* constructed to represent the curve elements comprised of consecutive points with m coordinate values between *start_measure* and *end_measure* inclusively.

II) elements of type *ST_Point* are constructed to represent disconnected points with m coordinate values between *start_measure* and *end_measure* inclusively.

iii) Otherwise:

Case:

1) If the result contains consecutive points with m coordinate values between *start_measure* and *end_measure* inclusively, then:

A) If the result also contains disconnected points with m coordinate values between *start_measure* and *end_measure* inclusively, then it is implementation-defined whether or not the following completion condition is raised: *SQL/MM Spatial warning – disconnected points not included in result*

B) Return an *ST_MultiCurve* value containing elements of type *ST_Curve* constructed to represent the curve elements between these consecutive points.

2) Otherwise, the result contains disconnected points with m coordinate values between *start_measure* and *end_measure* inclusively, then an *ST_MultiPoint* value containing elements of type *ST_Point* are constructed to represent these points.

f) If *SELF.ST_Dimension()* is equal to 2, then the operation is implementation-defined.

3) If *SELF.ST_Is3D()* is equal to 1 (one), then he z coordinate values are not considered in the calculation and they are not included in the resultant geometry.

4) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.16 ST_3DLocateBetween Method

Purpose

Return a derived geometry collection value with elements that match the specified range of measures inclusively, considering z coordinate values in the calculations and including them in the resultant geometry.

Definition

```
CREATE METHOD ST_3DLocateBetween
   (start_measure DOUBLE PRECISION,
    end_measure DOUBLE PRECISION)
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_3DLocateBetween(DOUBLE PRECISION, DOUBLE PRECISION)* takes the following input parameters:

   a) a DOUBLE PRECISION value *start_measure*,

   b) a DOUBLE PRECISION value *end_measure*.

2) For the null-call method *ST_3DLocateBetween(DOUBLE PRECISION, DOUBLE PRECISION)*:

   Case:

   a) If *end_measure* is less than *start_measure*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

   b) If SELF is an empty set, then return the null value.

   c) If *SELF.ST_IsMeasured*() is equal to 0 (zero), then return an empty *ST_Point* value.

   d) If *SELF.ST_Dimension()* is equal to 0 (zero), then

      Case:

      i) If SELF contains any *ST_Point* values with m coordinate values between *start_measure* and *end_measure* inclusively, then return an *ST_MultiPoint* value that contains the *ST_Point* values with m coordinate values between *start_measure* and *end_measure* inclusively.

      ii) Otherwise, return an empty *ST_Point* value.

   e) If *SELF.ST_Dimension()* is equal to 1, then use the implementation-defined interpolation algorithm to estimate values between *start_measure* and *end_measure* inclusively.

      Case:

      i) If the result does not contain any points, then return an empty *ST_Point* value.

      ii) If the *ST_GeomCollection* type is instantiable, then:

         Case:

         1) If the result only contains consecutive points with m coordinate values between *start_measure* and *end_measure* inclusively, then return an *ST_MultiCurve* value with elements of type *ST_Curve* are constructed to represent the curve elements comprised of these consecutive points.

         2) If the result only contains disconnected points with m coordinate values between *start_measure* and *end_measure* inclusively, then return an *ST_MultiPoint* value with elements of type *ST_Point* constructed to represent these points.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3) Otherwise:

A) Return an *ST_GeomCollection* value containing:

I) elements of type *ST_Curve* constructed to represent the curve elements comprised of consecutive points with m coordinate values between *start_measure* and *end_measure* inclusively.

II) elements of type *ST_Point* are constructed to represent disconnected points with m coordinate values between *start_measure* and *end_measure* inclusively.

iii) Otherwise:

Case:

1) If the result contains consecutive points with m coordinate values between *start_measure* and *end_measure* inclusively, then:

A) If the result also contains disconnected points with m coordinate values between *start_measure* and *end_measure* inclusively, then it is implementation-defined whether or not the following completion condition is raised: *SQL/MM Spatial warning – disconnected points not included in result*

B) Return an *ST_MultiCurve* value containing elements of type *ST_Curve* constructed to represent the curve elements between these consecutive points.

2) Otherwise, the result contains disconnected points with m coordinate values between *start_measure* and *end_measure* inclusively, then an *ST_MultiPoint* value containing elements of type *ST_Point* are constructed to represent these points.

f) If *SELF.ST_Dimension()* is equal to 2, then the operation is implementation-defined.

3) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are considered in the calculation and included in the resultant geometry.

4) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.17 ST_Boundary Method

Purpose

Return the boundary of the ST_Geometry value, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

Definition

```
CREATE METHOD ST_Boundary()
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_Boundary()* has no input parameters.
2) For the null-call method *ST_Boundary()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the closure of the boundary of the *ST_Geometry* value: *Closure(Boundary(SELF))*.
3) If SELF.*ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation and they are not included in the resultant geometry.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one), then:

   a) The m coordinate values are not considered in the calculation.

   b) The *ST_Point* values contained in the returned *ST_Geometry* value include the m coordinate values that were included when SELF was specified.
5) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.18 ST_3DBoundary Method

Purpose

Return the boundary of the ST_Geometry value, considering z coordinate values in the calculations and including them in the resultant geometry.

Definition

```
CREATE METHOD ST_3DBoundary()
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_3DBoundary()* has no input parameters.
2) For the null-call method *ST_3DBoundary()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the closure of the boundary of the *ST_Geometry* value: *Closure(Boundary(SELF))*.
3) If *SELF.ST_Is3D()* is equal to 1 (one), then:

   a) The z coordinate values are considered in the calculation.

   b) The *ST_Point* values contained in the returned *ST_Geometry* value include the z coordinate values that were included when SELF was specified.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one), then:

   a) The m coordinate values are not considered in the calculation.

   b) The *ST_Point* values contained in the returned *ST_Geometry* value include the m coordinate values that were included when SELF was specified.
5) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.19 ST_Envelope Method

Purpose

Return the bounding rectangle for the ST_Geometry value, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

Definition

```
CREATE METHOD ST_Envelope()
   RETURNS ST_Polygon
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_Envelope()* has no input parameters.
2) For the null-call method *ST_Envelope()* returns the bounding rectangle the *ST_Geometry* value:
   a) If SELF is an empty set, then return the null value.
   b) Let *MINX* be the minimum x coordinate value in the *ST_Geometry* value. Let *MINY* be the minimum y coordinate value in the *ST_Geometry* value. Let *MAXX* be the maximum x coordinate value in the *ST_Geometry* value. Let *MAXY* be the minimum y coordinate value in the *ST_Geometry* value.
   c) Let *ETOL* be an implementation-defined envelope tolerance. *ETOL* shall be greater than zero.
   d) If *MINX* is equal to *MAXX*, then set *MINX* to *MINX* - *ETOL* and set *MAXX* to *MAXX* + *ETOL*.
   e) If *MINY* is equal to *MAXY*, then set *MINY* to *MINY* - *ETOL* and set *MAXY* to *MAXY* + *ETOL*.
   f) Return the bounding rectangle constructed as follows:

```
NEW ST_Polygon(
   NEW ST_LineString(
      ARRAY[
         NEW ST_Point(MINX, MINY, SELF.ST_SRID()),
         NEW ST_Point(MAXX, MINY, SELF.ST_SRID()),
         NEW ST_Point(MAXX, MAXY, SELF.ST_SRID()),
         NEW ST_Point(MINX, MAXY, SELF.ST_SRID()),
         NEW ST_Point(MINX, MINY, SELF.ST_SRID())],
      SELF.ST_SRID()),
   SELF.ST_SRID())
```

3) If *SELF.ST_Is3D()* is equal to 1 (one), then:
   a) The z coordinate values are not considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Polygon* value do not include z coordinate values.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one), then:
   a) The m coordinate values are not considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Polygon* value do not include m coordinate values.
5) The spatial reference system identifier of the returned *ST_Polygon* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.20 ST_ConvexHull Method

Purpose

Return the convex hull of the ST_Geometry value, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

Definition

```
CREATE METHOD ST_ConvexHull()
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_ConvexHull()* has no input parameters.
2) For the null-call method *ST_ConvexHull()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return an *ST_Geometry* value representing the convex hull of the *ST_Geometry* value.
3) If *SELF.ST_Is3D()* is equal to 1 (one), then:

   a) The z coordinate values are not considered in the calculation.

   b) The *ST_Point* values contained in the returned *ST_Geometry* value do not include z coordinate values.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one), then:

   a) The m coordinate values are not considered in the calculation.

   b) The *ST_Point* values contained in the returned *ST_Geometry* value do not include m coordinate values.
5) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.21 ST_Buffer Methods

Purpose

Return the ST_Geometry value that represents all points whose distance from any point of an ST_Geometry value is less than or equal to a specified distance, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

Definition

```
CREATE METHOD ST_Buffer
   (adistance DOUBLE PRECISION)
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END

CREATE METHOD ST_Buffer
   (adistance DOUBLE PRECISION,
    aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The method *ST_Buffer(DOUBLE PRECISION)* takes the following input parameters:
   a) a DOUBLE PRECISION value *adistance*.
2) For the null-call method *ST_Buffer(DOUBLE PRECISION)*:
   a) The parameter *adistance* is measured in an implementation-defined linear unit of measure in the spatial reference system of SELF.
   b) Case:
      i) If SELF is an empty set, then return the null value.
      ii) Otherwise, return an *ST_Geometry* value that represents all points whose distance from SELF is less than or equal to *adistance*.
   c) If *SELF.ST_Is3D()* is equal to 1 (one), then:
      i) The z coordinate values are not considered in the calculation.
      ii) The *ST_Point* values contained in the returned *ST_Geometry* value do not include z coordinate values.
   d) If *SELF.ST_IsMeasured()* is equal to 1 (one), then:
      i) The m coordinate values are not considered in the calculation.
      ii) The *ST_Point* values contained in the returned *ST_Geometry* value do not include m coordinate values.
   e) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3) The method *ST_Buffer(DOUBLE PRECISION, CHARACTER VARYING)* takes the following input parameters:
   a) a DOUBLE PRECISION value *adistance*,
   b) a CHARACTER VARYING value *aunit*.
4) For the null-call method *ST_Buffer(DOUBLE PRECISION, CHARACTER VARYING)*:
   a) The DOUBLE PRECISION value *adistance* is measured in the units indicated by *aunit*.
   b) The values for *aunit* shall be a supported <unit name>.
   c) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.
   d) If the unit specified by *aunit* is not supported by the implementation to compute the *ST_Geometry* value that represents all points whose distance from SELF is less than or equal to *adistance*, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.
   e) Case:
      i) If SELF is an empty set, then return the null value.
      ii) Otherwise, return an *ST_Geometry* value that represents all points whose distance from SELF is less than or equal to *adistance.*
   f) If *SELF.ST_Is3D()* is equal to 1 (one), then:
      i) The z coordinate values are not considered in the calculation.
      ii) The *ST_Point* values contained in the returned *ST_Geometry* value do not include z coordinate values.
   g) If *SELF.ST_IsMeasured()* is equal to 1 (one), then:
      i) The m coordinate values are not considered in the calculation.
      ii) The *ST_Point* values contained in the returned *ST_Geometry* value do not include m coordinate values.
   h) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.22 ST_Intersection Method

Purpose

Return an ST_Geometry value that represents the point set intersection of two ST_Geometry values, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

Definition

```
CREATE METHOD ST_Intersection
   (ageometry ST_Geometry)
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_Intersection(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) The null-call method *ST_Intersection(ST_Geometry)* returns an *ST_Geometry* value that represents the point set intersection: *Closure(SELF ∩ ageometry)*.

NOTE For the list of subtypes returned by ST_Intersection(ST_Geometry), see Table 8 — Return Type Matrix for the ST_Intersection Method in Subclause 5.1.30, "Return Types from ST_Intersection, ST_Union, ST_Difference, and ST_SymDifference".

3) If *SELF.ST_Is3D()* is equal to 1 (one) or *ageometry.ST_Is3D()* is equal to 1 (one), then:
   a) The z coordinate values are not considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Geometry* value do not include z coordinate values.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one) or *ageometry.ST_IsMeasured()* is equal to 1 (one), then:
   a) The m coordinate values are not considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Geometry* value do not include m coordinate values.
5) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.23 ST_3DIntersection Method

Purpose

Return an ST_Geometry value that represents the point set intersection of two ST_Geometry values, considering z coordinate values in the calculations and including them in the resultant geometry.

Definition

```
CREATE METHOD ST_3DIntersection
   (ageometry ST_Geometry)
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_3DIntersection(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) The null-call method *ST_3DIntersection(ST_Geometry)* returns an *ST_Geometry* value that represents the point set intersection: *Closure(SELF ∩ ageometry)*.

NOTE For the list of subtypes returned by *ST_3DIntersection(ST_Geometry)*, see Table 8 — Return Type Matrix for the ST_Intersection Method in Subclause 5.1.22, "Return Types from ST_Intersection, ST_Union, ST_Difference, and ST_SymDifference".

3) Case:
   a) If *SELF.ST_Is3D()* is equal to 1 (one) and *ageometry.ST_Is3D()* is equal to 1 (one), then:
      i) The z coordinate values are considered in the calculation.
      ii) The *ST_Point* values contained in the returned *ST_Geometry* value include z coordinate values.
   b) Otherwise, an exception condition is raised: *SQL/MM Spatial Exception – both geometries must be 3D*.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one) or *ageometry.ST_IsMeasured()* is equal to 1 (one), then:
   a) The m coordinate values are not considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Geometry* value do not include m coordinate values.
5) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.24 ST_Union Method

Purpose

Return an ST_Geometry value that represents the point set union of two ST_Geometry values, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

Definition

```
CREATE METHOD ST_Union
   (ageometry ST_Geometry)
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_Union(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) The null-call method *ST_Union(ST_Geometry)* returns an *ST_Geometry* value that represents the point set union: *Closure(SELF ∪ ageometry)*.

NOTE For the list of subtypes returned by *ST_Union(ST_Geometry)*, see Table 9 — Return Type Matrix for the ST_Union Method in Subclause 5.1.30, "Return Types from ST_Intersection, ST_Union, ST_Difference, and ST_SymDifference".

3) If *SELF.ST_Is3D()* is equal to 1 (one) or *ageometry.ST_Is3D()* is equal to 1 (one) or *ageometry.ST_Is3D()* is equal to 1 (one), then:
   a) The z coordinate values are not considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Geometry* value do not include z coordinate values.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one) or *ageometry.ST_IsMeasured()* is equal to 1 (one), then:
   a) The m coordinate values are not considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Geometry* value do not include m coordinate values.
5) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.25 ST_3DUnion Method

Purpose

Return an ST_Geometry value that represents the point set union of two ST_Geometry values, considering z coordinate values in the calculations and including them in the resultant geometry.

Definition

```
CREATE METHOD ST_3DUnion
   (ageometry ST_Geometry)
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_3DUnion(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) The null-call method *ST_3DUnion(ST_Geometry)* returns an *ST_Geometry* value that represents the point set union: *Closure(SELF ∪ ageometry)*.

NOTE For the list of subtypes returned by *ST_3DUnion(ST_Geometry)*, see Table 9 — Return Type Matrix for the ST_Union Method in Subclause 5.1.22, "Return Types from ST_Intersection, ST_Union, ST_Difference, and ST_SymDifference".

3) Case:
   a) If *SELF.ST_Is3D()* is equal to 1 (one) and *ageometry.ST_Is3D()* is equal to 1 (one), then:
      i) The z coordinate values are considered in the calculation.
      ii) The *ST_Point* values contained in the returned *ST_Geometry* value include z coordinate values.
   b) Otherwise, an exception condition is raised: *SQL/MM Spatial Exception – both geometries must be 3D*.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one) or *ageometry.ST_IsMeasured()* is equal to 1 (one), then:
   a) The m coordinate values are not considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Geometry* value do not include m coordinate values.
5) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.26 ST_Difference Method

Purpose

Return an ST_Geometry value that represents the point set difference of two ST_Geometry values, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

Definition

```
CREATE METHOD ST_Difference
   (ageometry ST_Geometry)
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_Difference(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) The null-call method *ST_Difference(ST_Geometry)* returns an *ST_Geometry* value that represents the point set difference: *Closure(SELF — ageometry)*.

NOTE For the list of subtypes returned by *ST_Difference(ST_Geometry)*, see Table 10 — Return Type Matrix for the ST_Difference Method in Subclause 5.1.30, "Return Types from ST_Intersection, ST_Union, ST_Difference, and ST_SymDifference".

3) If *SELF.ST_Is3D()* is equal to 1 (one) or *ageometry.ST_Is3D()* is equal to 1 (one), then:
   a) The z coordinate values are not considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Geometry* value do not include z coordinate values.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one) or *ageometry.ST_IsMeasured()* is equal to 1 (one), then:
   a) The m coordinate values are not considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Geometry* value do not include m coordinate values.
5) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.27 ST_3DDifference Method

Purpose

Return an ST_Geometry value that represents the point set difference of two ST_Geometry values, considering z coordinate values in the calculations and including them in the resultant geometry.

Definition

```
CREATE METHOD ST_3DDifference
   (ageometry ST_Geometry)
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_3DDifference(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) The null-call method *ST_3DDifference(ST_Geometry)* returns an *ST_Geometry* value that represents the point set difference: *Closure(SELF — ageometry)*.

NOTE For the list of subtypes returned by *ST_3DDifference(ST_Geometry)*, see Table 10 — Return Type Matrix for the ST_Difference Method in Subclause 5.1.22, "Return Types from ST_Intersection, ST_Union, ST_Difference, and ST_SymDifference".

3) Case:
   a) If *SELF.ST_Is3D()* is equal to 1 (one) and *ageometry.ST_Is3D()* is equal to 1 (one), then:
      i) The z coordinate values are considered in the calculation.
      ii) The *ST_Point* values contained in the returned *ST_Geometry* value include z coordinate values.
   b) Otherwise, an exception condition is raised: *SQL/MM Spatial Exception – both geometries must be 3D*.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one) or *ageometry.ST_IsMeasured()* is equal to 1 (one), then:
   a) The m coordinate values are not considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Geometry* value do not include m coordinate values.
5) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.28 ST_SymDifference Method

Purpose

Return an ST_Geometry value that represents the point set symmetric difference of two ST_Geometry values, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

Definition

```
CREATE METHOD ST_SymDifference
   (ageometry ST_Geometry)
   RETURNS ST_Geometry
   FOR ST_Geometry
   RETURN SELF.ST_Difference(ageometry).
      ST_Union(ageometry.ST_Difference(SELF))
```

Description

1) The method *ST_SymDifference(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) The null-call method *ST_SymDifference(ST_Geometry)* returns an *ST_Geometry* value that represents the point set symmetric difference: *Closure(Closure(SELF — ageometry) ∪ Closure(ageometry — SELF))*.

NOTE For the list of subtypes returned by *ST_SymDifference(ST_Geometry)*, see Table 11 — Return Type Matrix for the ST_SymDifference Method in Subclause 5.1.30, "Return Types from ST_Intersection, ST_Union, ST_Difference, and ST_SymDifference".

3) If *SELF.ST_Is3D()* is equal to 1 (one) or *ageometry.ST_Is3D()* is equal to 1 (one), then:
   a) The z coordinate values are not considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Geometry* value do not include z coordinate values.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one) or *ageometry.ST_IsMeasured()* is equal to 1 (one), then:
   a) The m coordinate values are not considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Geometry* value do not include m coordinate values.
5) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.29 ST_3DSymDifference Method

Purpose

Return an ST_Geometry value that represents the point set symmetric difference of two ST_Geometry values, considering z coordinate values in the calculations and including them in the resultant geometry.

Definition

```
CREATE METHOD ST_3DSymDifference
   (ageometry ST_Geometry)
   RETURNS ST_Geometry
   FOR ST_Geometry
   RETURN SELF.ST_3DDifference(ageometry).
      ST_3DUnion(ageometry.ST_3DDifference(SELF))
```

Description

1) The method *ST_3DSymDifference(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) The null-call method *ST_3DSymDifference(ST_Geometry)* returns an *ST_Geometry* value that represents the point set symmetric difference: *Closure(Closure(SELF — ageometry) ∪ Closure(ageometry — SELF))*.

NOTE For the list of subtypes returned by *ST_3DSymDifference(ST_Geometry)*, see Table 11 — Return Type Matrix for the ST_SymDifference Method in Subclause 5.1.22, "Return Types from ST_Intersection, ST_Union, ST_Difference, and ST_SymDifference"

3) Case:
   a) If *SELF.ST_Is3D()* is equal to 1 (one) and *ageometry.ST_Is3D()* is equal to 1 (one), then:
      i) The z coordinate values are considered in the calculation.
      ii) The *ST_Point* values contained in the returned *ST_Geometry* value include z coordinate values.
   b) Otherwise, an exception condition is raised: *SQL/MM Spatial Exception – both geometries must be 3D*.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one) or *ageometry.ST_IsMeasured()* is equal to 1 (one), then:
   a) The m coordinate values are not considered in the calculation.
   b) The *ST_Point* values contained in the returned *ST_Geometry* value do not include m coordinate values.
5) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.30 Return Types from ST_Intersection, ST_Union, ST_Difference, and ST_SymDifference

The return types from the *ST_Intersection*, *ST_Union*, *ST_Difference*, and *ST_SymDifference* methods on the *ST_Geometry* type are defined in this subclause. These methods take two *ST_Geometry* values, the subject parameter and an additional parameter and return *ST_Geometry* values. The parameter type code for the possible parameter types is described in Table 6 — Parameter Types. For any given method, the type of the return value shall be one of the possible subtypes of *ST_Geometry* as specified in the return type set. The return type set codes are described in

Table 7 — Return Type Sets.

A matrix for each method is presented with the parameter type code of the subject parameter down the column and the parameter type code of the additional parameter across the row. Each cell of the matrix contains the return set code for the two parameter types. The matrix for the *ST_Intersection* method is in Table 8 — Return Type Matrix for the ST_Intersection Method. The matrix for the *ST_Union* method is in Table 9 — Return Type Matrix for the ST_Union Method. The matrix for the *ST_Difference* method is in Table 10 — Return Type Matrix for the ST_Difference Method. The matrix for the *ST_SymDifference* method is in Table 11 — Return Type Matrix for the ST_SymDifference Method.

The elements in return values of type *ST_GeomCollection* have no implied order.

Table 6 — Parameter Types

Parameter Type	
Code	Type
∅	empty set of any type
P	*ST_Point*
C	*ST_Curve*
S	*ST_Surface*
MP	*ST_MultiPoint*
MC	*ST_MultiCurve*
MS	*ST_MultiSurface*
GC	*ST_GeomCollection*

Table 7 — Return Type Sets

Return Type Sets	
Code	Set of Types
R00	empty set of type *ST_Point*
R01	*ST_Point*
R02	*ST_Curve*
R03	*ST_Surface*
R04	*ST_MultiPoint*,
R05	*ST_MultiCurve*
R06	*ST_MultiSurface*
R07	*ST_GeomCollection*
R08	empty set of type *ST_Point*, *ST_Curve*, *ST_MultiCurve*
R09	empty set of type *ST_Point*, *ST_Point*
R10	empty set of type *ST_Point*, *ST_MultiPoint*

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Return Type Sets	
Code	Set of Types
R11	empty set of type *ST_Point*, *ST_Point*, *ST_Curve*, *ST_MultiPoint*, *ST_MultiCurve*, *ST_GeomCollection* of *ST_Point* and *ST_Curve* values
R12	empty set of type *ST_Point*, *ST_Point*, *ST_Curve*, *ST_Surface*, *ST_MultiPoint*, *ST_MultiCurve*, *ST_MultiSurface*, *ST_GeomCollection*
R13	empty set of type *ST_Point*, *ST_Point*, *ST_MultiPoint*
R14	empty set of type *ST_Point*, *ST_Surface*, *ST_MultiSurface*
R15	*ST_Curve*, *ST_GeomCollection* of *ST_Point* and *ST_Curve* values
R16	*ST_Curve*, *ST_MultiCurve*
R17	*ST_MultiCurve*, *ST_GeomCollection* of *ST_Point* and *ST_Curve* values
R18	*ST_MultiSurface*, *ST_GeomCollection* of *ST_Curve* and *ST_Surface* values
R19	*ST_MultiSurface*, *ST_GeomCollection* of *ST_Point* and *ST_Surface* values
R20	*ST_Point*, *ST_MultiPoint*
R21	*ST_Surface*, *ST_GeomCollection* of *ST_Curve* and *ST_Surface* values
R22	*ST_Surface*, *ST_GeomCollection* of *ST_Point* and *ST_Surface* values
R23	*ST_Surface*, *ST_MultiSurface*

Table 8 — Return Type Matrix for the ST_Intersection Method

a ∩ b								
a \ b	∅	P	C	S	MP	MC	MS	GC
∅	R00	R00	R00	R00	R00	R00	R00	R00
P	R00	R09	R09	R09	R09	R09	R09	R09
C	R00	R09	R11	R11	R13	R11	R11	R11
S	R00	R09	R11	R12	R13	R11	R12	R12
MP	R00	R09	R13	R13	R13	R13	R13	R13
MC	R00	R09	R11	R11	R13	R11	R11	R11
MS	R00	R09	R11	R12	R13	R11	R12	R12
GC	R00	R09	R11	R12	R13	R11	R12	R12

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Table 9 — Return Type Matrix for the ST_Union Method

a ∪ b								
a \ b	∅	P	C	S	MP	MC	MS	GC
∅	R00	R01	R02	R03	R04	R05	R06	R07
P	R01	R20	R15	R22	R04	R17	R19	R07
C	R02	R15	R16	R21	R15	R16	R18	R07
S	R03	R22	R21	R23	R22	R21	R23	R07
MP	R04	R04	R15	R22	R04	R17	R19	R07
MC	R05	R17	R16	R21	R17	R16	R18	R07
MS	R06	R19	R18	R23	R19	R18	R23	R07
GC	R07	R07	R07	R07	R07	R07	R07	R07

Table 10 — Return Type Matrix for the ST_Difference Method

a — b								
a \ b	∅	P	C	S	MP	MC	MS	GC
∅	R00	R00	R00	R00	R00	R00	R00	R00
P	R01	R09	R09	R09	R09	R09	R09	R09
C	R02	R02	R08	R08	R02	R08	R08	R08
S	R03	R03	R03	R14	R03	R03	R14	R14
MP	R04	R13	R13	R13	R13	R13	R13	R13
MC	R05	R05	R08	R08	R05	R08	R08	R08
MS	R06	R06	R06	R14	R06	R06	R14	R14
GC	R07	R12	R12	R12	R12	R12	R12	R12

Table 11 — Return Type Matrix for the ST_SymDifference Method

(a — b) ∪ (b — a)								
a \ b	∅	P	C	S	MP	MC	MS	GC
∅	R00	R01	R02	R03	R04	R05	R06	R07
P	R01	R10	R15	R22	R10	R17	R19	R12
C	R02	R15	R08	R21	R15	R08	R18	R12
S	R03	R22	R21	R14	R22	R21	R14	R12
MP	R04	R10	R15	R22	R13	R17	R19	R12
MC	R05	R17	R08	R21	R17	R08	R18	R12
MS	R06	R19	R18	R14	R19	R18	R14	R12
GC	R07	R12	R12	R12	R12	R12	R12	R12

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.31 Return Types from ST_3DIntersection, ST_3DUnion, ST_3DDifference, and ST_3DSymDifference

The return types from the *ST_3DIntersection*, *ST_3DUnion*, *ST_3DDifference*, and *ST_3DSymDifference* methods on the *ST_Geometry* type are the same as their 2D counterparts (*ST_Intersection*, *ST_Union*, *ST_Difference*, and *ST_SymDifference*, respectively).

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.32 ST_Distance Methods

Purpose

Return the distance between two geometry values, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_Distance
   (ageometry ST_Geometry)
   RETURNS DOUBLE PRECISION
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END

CREATE METHOD ST_Distance
   (ageometry ST_Geometry,
    aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The method *ST_Distance(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) For the null-call method *ST_Distance(ST_Geometry)*:
   a) Case:
      i) If SELF is an empty set, then return the null value.
      ii) If *ageometry* is an empty set, then return the null value.
      iii) If SELF and *ageometry* spatially intersect, then return 0 (zero).
      iv) Otherwise, return the distance between two geometries, SELF and *ageometry*, calculated in the spatial reference system of SELF. The distance between two points is calculated using an implementation-defined algorithm such that z and m coordinate values are not considered in the calculation.
   b) Case:
      i) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_Distance(ST_Geometry)* is in the linear unit of measure identified by <linear unit>.
      ii) Otherwise, the value returned by *ST_Distance(ST_Geometry)* is in an implementation-defined unit of measure.
3) The method *ST_Distance(ST_Geometry, CHARACTER VARYING)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*;
   b) a CHARACTER VARYING value *aunit*.
4) For the null-call method *ST_Distance(ST_Geometry, CHARACTER VARYING)*:
   a) The values for *aunit* shall be a supported <unit name>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

b) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.

c) If the unit specified by *aunit* is not supported by the implementation to compute the distance between SELF and ageometry, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

d) Case:

   i) If SELF is an empty set, then return the null value.

   ii) If *ageometry* is an empty set, then return the null value.

   iii) If SELF and *ageometry* spatially intersect, then return 0 (zero).

   iv) Otherwise, return the distance between two geometries, SELF and *ageometry*, calculated in the spatial reference system of SELF. The distance between two points is calculated using an implementation-defined algorithm such that z and m coordinate values are not considered in the calculation.

e) The returned value is measured in the units indicated by *aunit*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.33 ST_3DDistance Methods

Purpose

Return the distance between two geometry values, considering z coordinate values in the calculations.

Definition

```
CREATE METHOD ST_3DDistance
   (ageometry ST_Geometry)
   RETURNS DOUBLE PRECISION
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END

CREATE METHOD ST_3DDistance
   (ageometry ST_Geometry,
    aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The method *ST_3DDistance(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) For the null-call method *ST_3DDistance(ST_Geometry)*:
   a) Case:
      i) If SELF is an empty set, then return the null value.
      ii) If *ageometry* is an empty set, then return the null value.
      iii) If *SELF.ST_Is3D()* is equal to 0 (zero) or *ageometry.ST_Is3D()* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial Exception – both geometries must be 3D*.
      iv) If SELF and *ageometry* spatially 3D intersect in 3D, then return 0 (zero).
      v) Otherwise, return the distance between two geometries, SELF and *ageometry*, calculated in the spatial reference system of SELF. The distance between two points is calculated using an implementation-defined algorithm such that z (but not m) coordinate values are considered in the calculation.
   b) Case:
      i) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_3DDistance(ST_Geometry)* is in the linear unit of measure identified by <linear unit>.
      ii) Otherwise, the value returned by *ST_3DDistance(ST_Geometry)* is in an implementation-defined unit of measure.
3) The method *ST_3DDistance(ST_Geometry, CHARACTER VARYING)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*;

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

b) a CHARACTER VARYING value *aunit*.

4) For the null-call method *ST_3DDistance(ST_Geometry, CHARACTER VARYING)*:

a) The values for *aunit* shall be a supported <unit name>.

b) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.

c) If the unit specified by *aunit* is not supported by the implementation to compute the distance between SELF and ageometry, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

d) Case:

i) If SELF is an empty set, then return the null value.

ii) If *ageometry* is an empty set, then return the null value.

iii) If *SELF.ST_Is3D()* is equal to 0 (zero) or *ageometry.ST_Is3D()* is equal to 0 (zero), then an exception condition is raised: SQL/MM *Spatial Exception – both geometries must be 3D*.

iv) If SELF and *ageometry* spatially 3D intersect, then return 0 (zero).

v) Otherwise, return the distance between two geometries, SELF and *ageometry*, calculated in the spatial reference system of SELF. The distance between two points is calculated using an implementation-defined algorithm such that z (but not m) coordinate values are considered in the calculation.

e) The returned value is measured in the units indicated by *aunit*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.34 ST_Equals Method

Purpose

Test if an ST_Geometry value is spatially 2D equal to another ST_Geometry value, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_Equals
   (ageometry ST_Geometry)
   RETURNS INTEGER
   FOR ST_Geometry
   RETURN SELF.ST_SymDifference(ageometry).ST_IsEmpty()
```

Description

1) The method *ST_Equals(ST_Geometry)* takes the following input parameters:

   a) an *ST_Geometry* value *ageometry*.

2) The null-call method *ST_Equals(ST_Geometry)* returns the result of the value expression: *SELF.ST_SymDifference*(*ageometry*).*ST_IsEmpty()*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.35 ST_3DEquals Method

Purpose

Test if an ST_Geometry value is spatially 3D equal to another ST_Geometry value, considering z coordinate values in the calculations.

Definition

```
CREATE METHOD ST_3DEquals
   (ageometry ST_Geometry)
   RETURNS INTEGER
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_3DEquals(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) Case:
   a) If *SELF.ST_Is3D()* is equal to 0 (zero) or *ageometry.ST_Is3D()* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial Exception – both geometries must be 3D*.
   b) Otherwise, the null-call method *ST_3DEquals(ST_Geometry)* returns the result of the value expression: *SELF.ST_3DSymDifference(ageometry).ST_IsEmpty()*.
3) Z coordinate values are considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.36 ST_Relate Method

Purpose

Test if an ST_Geometry value is spatially 2D related to another ST_Geometry value, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_Relate
   (ageometry ST_Geometry,
    amatrix CHARACTER(9))
   RETURNS INTEGER
   FOR ST_Geometry
   BEGIN
      DECLARE counter INTEGER;
      DECLARE intersectdim INTEGER;

      -- If any value in amatrix is not the list of
      -- possible values: 'T', 'F', '0', '1', '2', '*', then
      -- raise an exception.
      SET counter = 1;
      WHILE counter <= 9 DO
         IF SUBSTRING(amatrix FROM counter FOR 1)
            NOT IN ( 'T', 'F', '0', '1', '2', '*' ) THEN
            SIGNAL SQLSTATE '2FF04'
               SET MESSAGE_TEXT = 'invalid intersection matrix';
         END IF;
         SET counter = counter + 1;
      END WHILE;
      -- Process each of the 9 intersections
      SET counter = 1;
      WHILE counter <= 9 DO
         -- Set intersectdim to the dimension of the current intersection
         CASE counter
            WHEN 1 THEN
               SET intersectdim =
                  Intersection(Interior(SELF), Interior(ageometry)).
                  ST_Dimension(); -- See Description
            WHEN 2 THEN
               SET intersectdim =
                  Intersection(Interior(SELF), Boundary(ageometry)).
                  ST_Dimension(); -- See Description
            WHEN 3 THEN
               SET intersectdim =
                  Intersection(Interior(SELF), Exterior(ageometry)).
                  ST_Dimension(); -- See Description
            WHEN 4 THEN
               SET intersectdim =
                  Intersection(Boundary(SELF), Interior(ageometry)).
                  ST_Dimension(); -- See Description
            WHEN 5 THEN
               SET intersectdim =
                  Intersection(Boundary(SELF), Boundary(ageometry)).
                  ST_Dimension(); -- See Description
            WHEN 6 THEN
               SET intersectdim =
                  Intersection(Boundary(SELF), Exterior(ageometry)).
                  ST_Dimension(); -- See Description
            WHEN 7 THEN
               SET intersectdim =
                  Intersection(Exterior(SELF), Interior(ageometry)).
                  ST_Dimension(); -- See Description
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
            WHEN 8 THEN
               SET intersectdim =
                  Intersection(Exterior(SELF), Boundary(ageometry)).
                  ST_Dimension(); -- See Description
            WHEN 9 THEN
               SET intersectdim =
                  Intersection(Exterior(SELF), Exterior(ageometry)).
                  ST_Dimension(); -- See Description
         END CASE;
         -- If intersectdim is not in the result set as defined by the
         -- current amatrix position, then return 0 (zero).
         CASE SUBSTRING(amatrix FROM counter FOR 1)
            WHEN 'T' THEN
               IF intersectdim NOT IN ( 0, 1, 2 ) THEN
                  RETURN 0;
               END IF;
            WHEN 'F' THEN
               IF intersectdim <> -1 THEN
                  RETURN 0;
               END IF;
            WHEN '0' THEN
               IF intersectdim <> 0 THEN
                  RETURN 0;
               END IF;
            WHEN '1' THEN
               IF intersectdim <> 1 THEN
                  RETURN 0;
               END IF;
            WHEN '2' THEN
               IF intersectdim <> 2 THEN
                  RETURN 0;
               END IF;
            WHEN '*' THEN
               IF intersectdim NOT IN ( -1, 0, 1, 2 ) THEN
                  RETURN 0;
               END IF;
         END CASE;
         SET counter = counter + 1;
      END WHILE;
      -- If the dimension of each intersection matches the amatrix, then
      -- return 1 (one).
      RETURN 1;
   END
```

Definitional Rules

1) Let *G1* and *G2* be *ST_Geometry* values.
2) *Interior(G1)* represents the interior of *G1*.
3) *Boundary(G1)* represents the boundary of *G1*.
4) *Exterior(G1)* represents the exterior of *G1*.
5) *Intersection(G1, G2)* returns the point set intersection of *G1* and *G2*: *(G1 ∩ G2)*

NOTE interior, boundary, exterior and point set intersection are described in Subclause 4.2.2.1, "The Dimensionally Extended 9 Intersection Model".

Description

1) The method *ST_Relate(ST_Geometry, CHARACTER)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*,
   b) a CHARACTER value *amatrix*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) For null-call method *ST_Relate(ST_Geometry, CHARACTER)*:

Case:

a) If SELF is an empty set, then return the null value.

b) If *ageometry* is an empty set, then return the null value.

c) If any character in *amatrix* is not 'T', 'F', '0', '1', '2', or '*', then an exception condition is raised: *SQL/MM Spatial exception – invalid intersection matrix*.

d) Otherwise:

i) Each character in *amatrix* corresponds to a cell in the DE9IM pattern matrix. This mapping is defined in Table 12 — DE-9IM .

Table 12 — DE-9IM Mapping

Position	DE-9IM Cell
1	*(Interior(SELF) ∩ Interior(ageometry)).ST_Dimension()*
2	*(Interior(SELF) ∩ Boundary(ageometry)).ST_Dimension()*
3	*(Interior(SELF) ∩ Exterior(ageometry)).ST_Dimension()*
4	*(Boundary(SELF) ∩ Interior(ageometry)).ST_Dimension()*
5	*(Boundary(SELF) ∩ Boundary(ageometry)).ST_Dimension()*
6	*(Boundary(SELF) ∩ Exterior(ageometry)).ST_Dimension()*
7	*(Exterior(SELF) ∩ Interior(ageometry)).ST_Dimension()*
8	*(Exterior(SELF) ∩ Boundary(ageometry)).ST_Dimension()*
9	*(Exterior(SELF) ∩ Exterior(ageometry)).ST_Dimension()*

See Subclause 4.2.2.1, "The Dimensionally Extended 9 Intersection Model" for a detailed description of the DE-9IM.

ii) Each character value in *amatrix* specifies the set of acceptable values for an intersection at a given cell. The meaning for any cell is described in Table 13 — Cell Values.

Table 13 — Cell Values

Cell Value	Intersection Set Results
'T'	{ 0, 1, 2 }
'F'	{ -1 }
'0'	{ 0 }
'1'	{ 1 }
'2'	{ 2 }
'*'	{ -1, 0, 1, 2 }

iii) Let *RESULT* be the value returned by this method. Set *RESULT* to 1 (one).

iv) For *COUNTER* varying from 1 (one) to 9:

1) Let *INTERSECTDIM* be the result of the DE-9IM Intersection at position *COUNTER*.

2) Let *SVI* be the character value at *COUNTER* and let *SRI* be the intersection set results corresponding to *SVI*.

3) If *INTERSECTDIM* is not in the set *SRI*, then set *RESULT* to 0 (zero).

v) Return *RESULT*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3) If *SELF.ST_Is3D()* is equal to 1 (one) or ageometry.ST_Is3D() is equal to 1 (one), then the z coordinate values are not considered in the calculation.

4) If *SELF.ST_IsMeasured*() is equal to 1 (one) or ageometry.ST_IsMeasured() is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.37 ST_Disjoint Method

Purpose

Test if an ST_Geometry value is spatially 2D disjoint from another ST_Geometry value, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_Disjoint
   (ageometry ST_Geometry)
   RETURNS INTEGER
   FOR ST_Geometry
   RETURN SELF.ST_Relate(ageometry, 'FF*FF****')
```

Description

1) The method *ST_Disjoint(ST_Geometry)* takes the following input parameters:

   a) an *ST_Geometry* value *ageometry*.

2) The null-call method *ST_Disjoint(ST_Geometry)* returns the result of the value expression: *SELF.ST_Relate(ageometry, 'FF*FF****')*.

3) If *SELF.ST_Is3D()* is equal to 1 (one) or *ageometry.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation.

4) If *SELF.ST_IsMeasured()* is equal to 1 (one) or *ageometry.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.38 ST_3DDisjoint Method

Purpose

Test if an ST_Geometry value is spatially 3D disjoint from another ST_Geometry value, considering z coordinate values in the calculations.

Definition

```
CREATE METHOD ST_3DDisjoint
   (ageometry ST_Geometry)
   RETURNS INTEGER
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_3DDisjoint(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) For the null-call method *ST_3DDisjoint(ST_Geometry)*:

   Case:

   a) If *SELF.ST_Is3D()* is equal to 0 (zero) or *ageometry.ST_Is3D()* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial Exception – both geometries must be 3D*.
   b) If SELF and *ageometry* have no points in common (SELF ∩ *ageometry* = ∅), then return 1 (one).
   c) Otherwise, return 0 (zero).
3) Z coordinate values are considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.39 ST_Intersects Method

Purpose

Test if an ST_Geometry value spatially 2D intersects another ST_Geometry value, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_Intersects
   (ageometry ST_Geometry)
   RETURNS INTEGER
   FOR ST_Geometry
   RETURN
      CASE SELF.ST_Disjoint(ageometry)
         WHEN 1 THEN
            0
         WHEN 0 THEN
            1
         ELSE
            NULL
      END
```

Description

1) The method *ST_Intersects(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) For the null-call method *ST_Intersects(ST_Geometry)*:

   Case:
   a) If *SELF.ST_Disjoint*(*ageometry*) is equal to 1 (one), then return 0 (zero).
   b) If *SELF.ST_Disjoint*(*ageometry*) is equal to 0 (zero), then return 1 (one).
   c) Otherwise, return the null value.
3) If *SELF.ST_Is3D()* is equal to 1 (one) or *ageometry.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one) or *ageometry.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.40 ST_3DIntersects Method

Purpose

Test if an ST_Geometry value spatially 3D intersects another ST_Geometry value, considering z coordinate values in the calculations.

Definition

```
CREATE METHOD ST_3DIntersects
   (ageometry ST_Geometry)
   RETURNS INTEGER
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_3DIntersects(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) For the null-call method *ST_3DIntersects(ST_Geometry)*:

   Case:

   a) If *SELF.ST_Is3D()* is equal to 0 (zero) or *ageometry.ST_Is3D()* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial Exception – both geometries must be 3D*.
   b) If SELF and *ageometry* have any points in common (SELF $\cap$ *ageometry* $\neq \varnothing$), then return 1 (one).
   c) Otherwise, return 0 (zero).
3) Z coordinate values are considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.41 ST_Touches Method

Purpose

Test if an ST_Geometry value spatially 2D touches another ST_Geometry value, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_Touches
   (ageometry ST_Geometry)
   RETURNS INTEGER
   FOR ST_Geometry
   RETURN
      CASE
         WHEN (SELF.ST_Dimension() = 0 AND
               ageometry.ST_Dimension() = 0) THEN
            NULL
         ELSE
            -- Use ST_Relate to determine result of the touch operation
            -- on SELF and ageometry
            CASE (SELF.ST_Relate(ageometry, 'FT*******') = 1 OR
                  SELF.ST_Relate(ageometry, 'F**T*****') = 1 OR
                  SELF.ST_Relate(ageometry, 'F***T****') = 1)
               WHEN TRUE THEN
                  1
               WHEN FALSE THEN
                  0
               ELSE
                  NULL
            END
      END
```

Description

1) The method *ST_Touches(ST_Geometry)* takes the following input parameters:

   a) an *ST_Geometry* value *ageometry*.

2) For the null-call method *ST_Touches(ST_Geometry)*:

   Case:

   a) If the dimension of SELF is equal to 0 (zero) and the dimension of *ageometry* is equal to 0 (zero), then return the null value.

   b) Otherwise,

      i) Let *BVE* be the result of the value expression: *SELF.ST_Relate(ageometry, 'FT*******') = 1 OR SELF.ST_Relate(ageometry, 'F**T*****') = 1 OR SELF.ST_Relate(ageometry, 'F***T****') = 1*.

      ii) Case:

         1) If *BVE* is *True*, then return 1 (one).

         2) If *BVE* is *False*, then return 0 (zero).

         3) Otherwise, return the null value.

3) If *SELF.ST_Is3D()* is equal to 1 (one) or *ageometry.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation.

4) If *SELF.ST_IsMeasured()* is equal to 1 (one) or *ageometry.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.42 ST_Crosses Method

Purpose

Test if an ST_Geometry value spatially 2D crosses another ST_Geometry value, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_Crosses
   (ageometry ST_Geometry)
   RETURNS INTEGER
   FOR ST_Geometry
   RETURN
      CASE
         WHEN (SELF.ST_Dimension() = 0 AND
               ageometry.ST_Dimension() = 1) THEN
            SELF.ST_Relate(ageometry, 'T*T******')
         WHEN (SELF.ST_Dimension() = 0 AND
               ageometry.ST_Dimension() = 2) THEN
            SELF.ST_Relate(ageometry, 'T*T******')
         WHEN (SELF.ST_Dimension() = 1 AND
               ageometry.ST_Dimension() = 1) THEN
            SELF.ST_Relate(ageometry, '0********')
         WHEN (SELF.ST_Dimension() = 1 AND
               ageometry.ST_Dimension() = 2) THEN
            SELF.ST_Relate(ageometry, 'T*T******')
         ELSE
            NULL
      END
```

Description

1) The method *ST_Crosses(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) For the null-call method *ST_Crosses(ST_Geometry)*:

   Case:

   a) If the dimension of SELF is equal to 0 (zero) and the dimension of *ageometry* is equal to 1 (one), then return the result of the value expression: *SELF.ST_Relate(ageometry, 'T*T******')*.
   b) If the dimension of SELF is equal to 0 (zero) and the dimension of *ageometry* is equal to 2, then return the result of the value expression: *SELF.ST_Relate(ageometry, 'T*T******')*.
   c) If the dimension of SELF is equal to 1 (one) and the dimension of *ageometry* is equal to 1 (one), then return the result of the value expression: *SELF.ST_Relate(ageometry, '0********')*.
   d) If the dimension of SELF is equal to 1 (one) and the dimension of *ageometry* is equal to 2, then return the result of the value expression: *SELF.ST_Relate(ageometry, 'T*T******')*.
   e) Otherwise, return the null value.
3) If *SELF.ST_Is3D()* is equal to 1 (one) or *ageometry.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one) or *ageometry.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.43 ST_Within Method

Purpose

Test if an ST_Geometry value is spatially 2D within another ST_Geometry value, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_Within
   (ageometry ST_Geometry)
   RETURNS INTEGER
   FOR ST_Geometry
   RETURN SELF.ST_Relate(ageometry, 'T*F**F***')
```

Description

1) The method *ST_Within(ST_Geometry)* takes the following input parameters:

   a) an *ST_Geometry* value *ageometry*.

2) The null-call method *ST_Within(ST_Geometry)* returns the result of the value expression: *SELF.ST_Relate(ageometry, 'T*F**F***')*.

3) If *SELF.ST_Is3D()* is equal to 1 (one) or *ageometry.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation.

4) If *SELF.ST_IsMeasured()* is equal to 1 (one) or *ageometry.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.44 ST_Contains Method

Purpose

Test if an ST_Geometry value spatially 2D contains another ST_Geometry value, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_Contains
   (ageometry ST_Geometry)
   RETURNS INTEGER
   FOR ST_Geometry
   RETURN ageometry.ST_Within(SELF)
```

Description

1) The method *ST_Contains(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) The null-call method *ST_Contains(ST_Geometry)* returns the result of the value expression: *ageometry.ST_Within(SELF)*.
3) If *SELF.ST_Is3D()* is equal to 1 (one) or *ageometry.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one) or *ageometry.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.45 ST_Overlaps Method

Purpose

Test if an ST_Geometry value spatially 2D overlaps another ST_Geometry value, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_Overlaps
   (ageometry ST_Geometry)
   RETURNS INTEGER
   FOR ST_Geometry
   RETURN
      CASE
         WHEN (SELF.ST_Dimension() = 0 AND
               ageometry.ST_Dimension() = 0) THEN
            SELF.ST_Relate(ageometry, 'T*T***T**')
         WHEN (SELF.ST_Dimension() = 1 AND
               ageometry.ST_Dimension() = 1) THEN
            SELF.ST_Relate(ageometry, '1*T***T**')
         WHEN (SELF.ST_Dimension() = 2 AND
               ageometry.ST_Dimension() = 2) THEN
            SELF.ST_Relate(ageometry, 'T*T***T**')
         ELSE
            NULL
      END
```

Description

1) The method *ST_Overlaps(ST_Geometry)* takes the following input parameters:

   a) an *ST_Geometry* value *ageometry*.

2) For the null-call method *ST_Overlaps(ST_Geometry)*:

   Case:

   a) If the dimension of SELF is equal to 0 (zero) and the dimension of *ageometry* is equal to 0 (zero), then return the result of the value expression: *SELF.ST_Relate(ageometry, 'T*T***T**')*.

   b) If the dimension of SELF is equal to 1 (one) and the dimension of *ageometry* is equal to 1 (one), then return the result of the value expression: *SELF.ST_Relate(ageometry, '1*T***T**')*.

   c) If the dimension of SELF is equal to 2 and the dimension of *ageometry* is equal to 2, then return the result of the value expression: *SELF.ST_Relate(ageometry, 'T*T***T**')*.

   d) Otherwise, return the null value.

3) If *SELF.ST_Is3D()* is equal to 1 (one) or *ageometry.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation.

4) If *SELF.ST_IsMeasured()* is equal to 1 (one) or *ageometry.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.46 Cast

Purpose

Cast an ST_Geometry value to a specific instantiable subtype of ST_Geometry.

Definition

```
CREATE METHOD ST_ToPoint()
   RETURNS ST_Point
   FOR ST_Geometry
   BEGIN
      IF SELF IS OF (ST_Point) THEN
         RETURN TREAT(SELF AS ST_Point);
      ELSEIF SELF IS OF (ST_GeomCollection) THEN
         IF (SELF AS ST_GeomCollection).ST_NumGeometries() = 1 THEN
            IF (SELF AS ST_GeomCollection).ST_GeometryN(1)
                  IS OF (ST_Point) THEN
               RETURN CAST((SELF AS ST_GeomCollection).ST_GeometryN(1)
                  AS ST_Point);
            END IF;
         END IF;
      END IF;
      IF SELF.ST_IsEmpty() = 0 THEN
         SIGNAL SQLSTATE '2FF16'
            SET MESSAGE_TEXT = 'not an empty set';
      END IF;
      RETURN NEW ST_Point().ST_SRID(SELF.ST_SRID());
   END

CREATE CAST(ST_Geometry AS ST_Point)
   WITH METHOD ST_ToPoint()
   AS ASSIGNMENT

CREATE METHOD ST_ToLineString()
   RETURNS ST_LineString
   FOR ST_Geometry
   BEGIN
      IF SELF IS OF (ST_LineString) THEN
         RETURN TREAT(SELF AS ST_LineString);
      ELSEIF SELF IS OF (ST_CircularString, ST_CompoundCurve) THEN
         RETURN (SELF AS ST_Curve).ST_CurveToLine();
      ELSEIF SELF IS OF (ST_GeomCollection) THEN
         IF (SELF AS ST_GeomCollection).ST_NumGeometries() = 1 THEN
            IF (SELF AS ST_GeomCollection).ST_GeometryN(1)
                  IS OF (ST_Curve) THEN
               RETURN CAST((SELF AS ST_GeomCollection).ST_GeometryN(1)
                  AS ST_LineString);
            END IF;
         END IF;
      END IF;
      IF SELF.ST_IsEmpty() = 0 THEN
         SIGNAL SQLSTATE '2FF16'
            SET MESSAGE_TEXT = 'not an empty set';
      END IF;
      RETURN NEW ST_LineString().ST_SRID(SELF.ST_SRID());
   END

CREATE CAST(ST_Geometry AS ST_LineString)
   WITH METHOD ST_ToLineString()
   AS ASSIGNMENT
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CREATE METHOD ST_ToCircular()
   RETURNS ST_CircularString
   FOR ST_Geometry
   BEGIN
      IF SELF IS OF (ST_CircularString) THEN
         RETURN TREAT(SELF AS ST_CircularString);
      ELSEIF SELF IS OF (ST_CompoundCurve) THEN
         IF (SELF AS ST_CompoundCurve).ST_NumCurves() = 1 THEN
            IF (SELF AS ST_CompoundCurve).ST_CurveN(1)
                  IS OF (ST_CircularString) THEN
               RETURN CAST((SELF AS ST_CompoundCurve).ST_CurveN(1)
                  AS ST_CircularString);
            END IF;
         END IF;
      ELSEIF SELF IS OF (ST_GeomCollection) THEN
         IF (SELF AS ST_GeomCollection).ST_NumGeometries() = 1 THEN
            IF (SELF AS ST_GeomCollection).ST_GeometryN(1)
                  IS OF (ST_CircularString) THEN
               RETURN
                  CAST((SELF AS ST_GeomCollection).ST_GeometryN(1)
                     AS ST_CircularString);
            END IF;
         END IF;
      END IF;
      IF SELF.ST_IsEmpty() = 0 THEN
         SIGNAL SQLSTATE '2FF16'
            SET MESSAGE_TEXT = 'not an empty set';
      END IF;
      RETURN NEW ST_CircularString().ST_SRID(SELF.ST_SRID());
   END

CREATE CAST(ST_Geometry AS ST_CircularString)
   WITH METHOD ST_ToCircular()
   AS ASSIGNMENT

CREATE METHOD ST_ToCompound()
   RETURNS ST_CompoundCurve
   FOR ST_Geometry
   BEGIN
      IF SELF IS OF (ST_CompoundCurve) THEN
         RETURN TREAT(SELF AS ST_CompoundCurve);
      ELSEIF SELF IS OF (ST_LineString, ST_CircularString) THEN
         RETURN NEW ST_CompoundCurve(ARRAY[TREAT(SELF AS ST_Curve)],
            SELF.ST_SRID());
      ELSEIF SELF IS OF (ST_GeomCollection) THEN
         IF (SELF AS ST_GeomCollection).ST_NumGeometries() = 1 THEN
            IF (SELF AS ST_GeomCollection).ST_GeometryN(1)
                  IS OF (ST_Curve) THEN
               RETURN
                  CAST((SELF AS ST_GeomCollection).ST_GeometryN(1)
                     AS ST_CompoundCurve);
            END IF;
         END IF;
      END IF;
      IF SELF.ST_IsEmpty() = 0 THEN
         SIGNAL SQLSTATE '2FF16'
            SET MESSAGE_TEXT = 'not an empty set';
      END IF;
      RETURN NEW ST_CompoundCurve().ST_SRID(SELF.ST_SRID());
   END
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CREATE CAST(ST_Geometry AS ST_CompoundCurve)
   WITH METHOD ST_ToCompound()
   AS ASSIGNMENT

CREATE METHOD ST_ToCurvePoly()
   RETURNS ST_CurvePolygon
   FOR ST_Geometry
   BEGIN
      IF SELF IS OF (ST_CurvePolygon) THEN
         RETURN TREAT(SELF AS ST_CurvePolygon);
      ELSEIF SELF IS OF (ST_GeomCollection) THEN
         IF (SELF AS ST_GeomCollection).ST_NumGeometries() = 1 THEN
            IF (SELF AS ST_GeomCollection).ST_GeometryN(1)
                  IS OF (ST_CurvePolygon) THEN
               RETURN
                  CAST((SELF AS ST_GeomCollection).ST_GeometryN(1)
                     AS ST_CurvePolygon);
            END IF;
         END IF;
      END IF;
      IF SELF.ST_IsEmpty() = 0 THEN
         SIGNAL SQLSTATE '2FF16'
            SET MESSAGE_TEXT = 'not an empty set';
      END IF;
      RETURN NEW ST_CurvePolygon().ST_SRID(SELF.ST_SRID());
   END

CREATE CAST(ST_Geometry AS ST_CurvePolygon)
   WITH METHOD ST_ToCurvePoly()
   AS ASSIGNMENT

CREATE METHOD ST_ToPolygon()
   RETURNS ST_Polygon
   FOR ST_Geometry
   BEGIN
      IF SELF IS OF (ST_Polygon) THEN
         RETURN TREAT(SELF AS ST_Polygon);
      ELSEIF SELF IS OF (ST_CurvePolygon) THEN
         RETURN (SELF AS ST_CurvePolygon).ST_CurvePolyToPoly();
      ELSEIF SELF IS OF (ST_GeomCollection) THEN
         IF (SELF AS ST_GeomCollection).ST_NumGeometries() = 1 THEN
            IF (SELF AS ST_GeomCollection).ST_GeometryN(1)
                  IS OF (ST_CurvePolygon) THEN
               RETURN
                  CAST((SELF AS ST_GeomCollection).ST_GeometryN(1)
                     AS ST_Polygon);
            END IF;
         END IF;
      END IF;
      IF SELF.ST_IsEmpty() = 0 THEN
         SIGNAL SQLSTATE '2FF16'
            SET MESSAGE_TEXT = 'not an empty set';
      END IF;
      RETURN NEW ST_Polygon().ST_SRID(SELF.ST_SRID());
   END

CREATE CAST(ST_Geometry AS ST_Polygon)
   WITH METHOD ST_ToPolygon()
   AS ASSIGNMENT
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CREATE METHOD ST_ToTriangle()
   RETURNS ST_Triangle
   FOR ST_Geometry
   BEGIN
      IF SELF IS OF (ST_Triangle) THEN
         RETURN TREAT(SELF AS ST_Triangle);
      ELSEIF SELF IS OF (ST_GeomCollection) THEN
         IF (SELF AS ST_GeomCollection).ST_NumGeometries() = 1 THEN
            IF (SELF AS ST_GeomCollection).ST_GeometryN(1)
                  IS OF (ST_Triangle) THEN
               RETURN
                  CAST((SELF AS ST_GeomCollection).ST_GeometryN(1)
                     AS ST_Triangle);
            END IF;
         END IF;
      END IF;
      IF SELF.ST_IsEmpty() = 0 THEN
         SIGNAL SQLSTATE '2FF16'
            SET MESSAGE_TEXT = 'not an empty set';
      END IF;
      RETURN NEW ST_Triangle().ST_SRID(SELF.ST_SRID());
   END

CREATE CAST(ST_Geometry AS ST_PolyhdrlSurface)
   WITH METHOD ST_ToPolyhdrlSurf ()
   AS ASSIGNMENT

CREATE METHOD ST_ToPolyhdrlSurf()
   RETURNS ST_PolyhdrlSurface
   FOR ST_Geometry
   BEGIN
      IF SELF IS OF (ST_PolyhdrlSurface) THEN
         RETURN TREAT(SELF AS ST_PolyhdrlSurface);
      ELSEIF SELF IS OF (ST_GeomCollection) THEN
         IF (SELF AS ST_GeomCollection).ST_NumGeometries() = 1 THEN
            IF (SELF AS ST_GeomCollection).ST_GeometryN(1)
                  IS OF (ST_PolyhdrlSurface) THEN
               RETURN
                  CAST((SELF AS ST_GeomCollection).ST_GeometryN(1)
                     AS ST_PolyhdrlSurface);
            END IF;
         END IF;
      END IF;
      IF SELF.ST_IsEmpty() = 0 THEN
         SIGNAL SQLSTATE '2FF16'
            SET MESSAGE_TEXT = 'not an empty set';
      END IF;
      RETURN NEW ST_PolyhdrlSurface().ST_SRID(SELF.ST_SRID());
   END

CREATE CAST(ST_Geometry AS ST_PolyhdrlSurface)
   WITH METHOD ST_ToPolyhdrlSurf()
   AS ASSIGNMENT

CREATE METHOD ST_ToTIN()
   RETURNS ST_TIN
   FOR ST_Geometry
   BEGIN
      IF SELF IS OF (ST_TIN) THEN
         RETURN TREAT(SELF AS ST_TIN);
      ELSEIF SELF IS OF (ST_GeomCollection) THEN
         IF (SELF AS ST_GeomCollection).ST_NumGeometries() = 1 THEN
            IF (SELF AS ST_GeomCollection).ST_GeometryN(1)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
                IS OF (ST_TIN) THEN
              RETURN
                CAST((SELF AS ST_GeomCollection).ST_GeometryN(1)
                  AS ST_TIN);
            END IF;
          END IF;
        END IF;
        IF SELF.ST_IsEmpty() = 0 THEN
          SIGNAL SQLSTATE '2FF16'
            SET MESSAGE_TEXT = 'not an empty set';
        END IF;
        RETURN NEW ST_TIN().ST_SRID(SELF.ST_SRID());
    END

CREATE CAST(ST_Geometry AS ST_TIN)
    WITH METHOD ST_ToTIN()
    AS ASSIGNMENT

CREATE METHOD ST_ToGeomColl()
    RETURNS ST_GeomCollection
    FOR ST_Geometry
    BEGIN
        IF SELF IS OF (ST_GeomCollection) THEN
          RETURN TREAT(SELF AS ST_GeomCollection);
        ELSEIF SELF IS OF (ST_Point, ST_Curve, ST_Surface) THEN
          RETURN NEW ST_GeomCollection(ARRAY[SELF], SELF.ST_SRID());
        END IF;
        IF SELF.ST_IsEmpty() = 0 THEN
          SIGNAL SQLSTATE '2FF16'
            SET MESSAGE_TEXT = 'not an empty set';
        END IF;
        RETURN NEW ST_GeomCollection().ST_SRID(SELF.ST_SRID());
    END

CREATE CAST(ST_Geometry AS ST_GeomCollection)
    WITH METHOD ST_ToGeomColl()
    AS ASSIGNMENT

CREATE METHOD ST_ToMultiPoint()
    RETURNS ST_MultiPoint
    FOR ST_Geometry
    BEGIN
        IF SELF IS OF (ST_MultiPoint) THEN
          RETURN TREAT(SELF AS ST_MultiPoint);
        ELSEIF SELF IS OF (ST_GeomCollection) THEN
          RETURN NEW ST_MultiPoint(
            (SELF AS ST_GeomCollection).ST_Geometries(), SELF.ST_SRID());
        ELSEIF SELF IS OF (ST_Point) THEN
          RETURN NEW ST_MultiPoint(ARRAY[CAST(SELF AS ST_Point)],
            SELF.ST_SRID());
        END IF;
        IF SELF.ST_IsEmpty() = 0 THEN
          SIGNAL SQLSTATE '2FF16'
            SET MESSAGE_TEXT = 'not an empty set';
        END IF;
        RETURN NEW ST_MultiPoint().ST_SRID(SELF.ST_SRID());
    END

CREATE CAST(ST_Geometry AS ST_MultiPoint)
    WITH METHOD ST_ToMultiPoint()
    AS ASSIGNMENT

CREATE METHOD ST_ToMultiCurve()
    RETURNS ST_MultiCurve
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
    FOR ST_Geometry
    BEGIN
       IF SELF IS OF (ST_MultiCurve) THEN
          RETURN TREAT(SELF AS ST_MultiCurve);
       ELSEIF SELF IS OF (ST_GeomCollection) THEN
          RETURN NEW ST_MultiCurve(
             (SELF AS ST_GeomCollection).ST_Geometries(), SELF.ST_SRID());
       ELSEIF SELF IS OF (ST_Curve) THEN
          RETURN NEW ST_MultiCurve(ARRAY[TREAT(SELF AS ST_Curve)],
             SELF.ST_SRID());
       END IF;
       IF SELF.ST_IsEmpty() = 0 THEN
          SIGNAL SQLSTATE '2FF16'
             SET MESSAGE_TEXT = 'not an empty set';
       END IF;
       RETURN NEW ST_MultiCurve().ST_SRID(SELF.ST_SRID());
    END

CREATE CAST(ST_Geometry AS ST_MultiCurve)
    WITH METHOD ST_ToMultiCurve()
    AS ASSIGNMENT

CREATE METHOD ST_ToMultiLine()
    RETURNS ST_MultiLineString
    FOR ST_Geometry
    BEGIN
       IF SELF IS OF (ST_MultiLineString) THEN
          RETURN TREAT(SELF AS ST_MultiLineString);
       ELSEIF SELF IS OF (ST_GeomCollection) THEN
          RETURN NEW ST_MultiLineString(
             (SELF AS ST_GeomCollection).ST_Geometries(), SELF.ST_SRID());
       ELSEIF SELF IS OF (ST_Curve) THEN
          RETURN NEW
             ST_MultiLineString(ARRAY[CAST(SELF AS ST_LineString)],
                SELF.ST_SRID());
       END IF;
       IF SELF.ST_IsEmpty() = 0 THEN
          SIGNAL SQLSTATE '2FF16'
             SET MESSAGE_TEXT = 'not an empty set';
       END IF;
       RETURN NEW ST_MultiLineString().ST_SRID(SELF.ST_SRID());
    END

CREATE CAST(ST_Geometry AS ST_MultiLineString)
    WITH METHOD ST_ToMultiLine()
    AS ASSIGNMENT

CREATE METHOD ST_ToMultiSurface()
    RETURNS ST_MultiSurface
    FOR ST_Geometry
    BEGIN
       IF SELF IS OF (ST_MultiSurface) THEN
          RETURN TREAT(SELF AS ST_MultiSurface);
       ELSEIF SELF IS OF (ST_GeomCollection) THEN
          RETURN NEW ST_MultiSurface(
             (SELF AS ST_GeomCollection).ST_Geometries(), SELF.ST_SRID());
       ELSEIF SELF IS OF (ST_Surface) THEN
          RETURN NEW ST_MultiSurface(ARRAY[TREAT(SELF AS ST_Surface)],
             SELF.ST_SRID());
       END IF;
       IF SELF.ST_IsEmpty() = 0 THEN
          SIGNAL SQLSTATE '2FF16'
             SET MESSAGE_TEXT = 'not an empty set';
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
      END IF;
      RETURN NEW ST_MultiSurface().ST_SRID(SELF.ST_SRID());
   END

CREATE CAST(ST_Geometry AS ST_MultiSurface)
   WITH METHOD ST_ToMultiSurface()
   AS ASSIGNMENT

CREATE METHOD ST_ToMultiPolygon()
   RETURNS ST_MultiPolygon
   FOR ST_Geometry
   BEGIN
      IF SELF IS OF (ST_MultiPolygon) THEN
         RETURN TREAT(SELF AS ST_MultiPolygon);
      ELSEIF SELF IS OF (ST_GeomCollection) THEN
         RETURN NEW ST_MultiPolygon(
            (SELF AS ST_GeomCollection).ST_Geometries(), SELF.ST_SRID());
      ELSEIF SELF IS OF (ST_CurvePolygon) THEN
         RETURN NEW ST_MultiPolygon(ARRAY[CAST(SELF AS ST_Polygon)],
            SELF.ST_SRID());
      END IF;
      IF SELF.ST_IsEmpty() = 0 THEN
         SIGNAL SQLSTATE '2FF16'
            SET MESSAGE_TEXT = 'not an empty set';
      END IF;
      RETURN NEW ST_MultiPolygon().ST_SRID(SELF.ST_SRID());
   END

CREATE CAST(ST_Geometry AS ST_MultiPolygon)
   WITH METHOD ST_ToMultiPolygon()
   AS ASSIGNMENT
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

Description

1) The method *ST_ToPoint()* has no input parameters.
2) For the null-call method *ST_ToPoint()*:

   Case:

   a) If SELF is of type *ST_Point*, then return SELF.

   b) If SELF is of type *ST_GeomCollection* and SELF has only one element of type *ST_Point*, then return the element cast to an *ST_Point* value.

   c) If SELF is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

   d) Otherwise, return an empty set of type *ST_Point* with the spatial reference system identifier set to *SELF.ST_SRID*().
3) Use the method *ST_ToPoint()* to define an implicitly invocable cast function to cast an *ST_Geometry* value to an *ST_Point* value.
4) The method *ST_ToLineString()* has no input parameters.
5) For the null-call method *ST_ToLineString()*:

   Case:

   a) If SELF is of type *ST_LineString*, then return SELF.

   b) If SELF is of type *ST_CircularString* or *ST_CompoundCurve*, then return *SELF.ST_CurveToLine()*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

c) If SELF is of type *ST_GeomCollection* and SELF has only one element of type *ST_Curve*, then return the element cast to an *ST_LineString* value.

d) If SELF is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

e) Otherwise, return an empty set of type *ST_LineString* with the spatial reference system identifier set to *SELF.ST_SRID()*.

6) Use the method *ST_ToLineString()* to define an implicitly invocable cast function to cast an *ST_Geometry* value to an *ST_LineString* value.

7) The method *ST_ToCircular()* has no input parameters.

8) For the null-call method *ST_ToCircular()*:

Case:

a) If SELF is of type *ST_CircularString*, then return SELF.

b) If SELF is of type *ST_CompoundCurve* and SELF has only one element of type *ST_CircularString*, then return the element cast to an *ST_CircularString* value.

c) If SELF is of type *ST_GeomCollection* and SELF has only one element of type *ST_CircularString*, then return the element cast to an *ST_CircularString* value.

d) If SELF is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

e) Otherwise, return an empty set of type *ST_CircularString* with the spatial reference system identifier set to *SELF.ST_SRID()*.

9) Use the method *ST_ToCircular()* to define an implicitly invocable cast function to cast an *ST_Geometry* value to an *ST_CircularString* value.

10) The method *ST_ToCompound()* has no input parameters.

11) For the null-call method *ST_ToCompound()*:

Case:

a) If SELF is of type *ST_CompoundCurve*, then return SELF.

b) If SELF is of type *ST_LineString* or *ST_CircularString*, then return an *ST_CompoundCurve* value with the spatial reference system identifier set to *SELF.ST_SRID()* and containing one element, SELF.

c) If SELF is of type *ST_GeomCollection* and SELF has only one element of type *ST_Curve*, then return the element cast as an *ST_CompoundCurve* value.

d) If SELF is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

e) Otherwise, return an empty set of type *ST_CompoundCurve* with the spatial reference system identifier set to *SELF.ST_SRID()*.

12) Use the method *ST_ToCompound()* to define an implicitly invocable cast function to cast an *ST_Geometry* value to an *ST_CompoundCurve* value.

13) The method *ST_ToCurvePoly()* has no input parameters.

14) For the null-call method *ST_ToCurvePoly()*:

Case:

a) If SELF is of type *ST_CurvePolygon*, then return SELF.

b) If SELF is of type *ST_GeomCollection* and SELF has one element of type *ST_CurvePolygon*, then return the element cast as an *ST_CurvePolygon* value.

c) If SELF is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

d) Otherwise, return an empty set of type *ST_CurvePolygon* with the spatial reference system identifier set to *SELF.ST_SRID()*.

15) Use the method *ST_ToCurvePoly()* to define an implicitly invocable cast function to cast an *ST_Geometry* value to an *ST_CurvePolygon* value.

16) The method *ST_ToPolygon()* has no input parameters.

17) For the null-call method *ST_ToPolygon()*:

Case:

a) If SELF is of type *ST_Polygon*, then return SELF.

b) If SELF is of type *ST_CurvePolygon*, then return *SELF.ST_CurvePolyToPoly()*

c) If SELF is of type *ST_GeomCollection* and SELF has one element of type *ST_CurvePolygon*, then return the element cast as an *ST_Polygon* value.

d) If SELF is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

e) Otherwise, return an empty set of type *ST_Polygon* with the spatial reference system identifier set to *SELF.ST_SRID()*.

18) Use the method *ST_ToPolygon()* to define an implicitly invocable cast function to cast an *ST_Geometry* value to an *ST_Polygon* value.

19) The method *ST_ToTriangle()* has no input parameters.

20) For the null-call method *ST_ToTriangle()*:

Case:

a) If SELF is of type *ST_Triangle*, then return SELF.

b) If SELF is of type *ST_GeomCollection* and SELF has one element of type *ST_Triangle*, then return the element cast as an *ST_Triangle* value.

c) If SELF is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

d) Otherwise, return an empty set of type *ST_Triangle* with the spatial reference system identifier set to *SELF.ST_SRID()*.

21) Use the method *ST_ToTriangle()* to define an implicitly invocable cast function to cast an *ST_Geometry* value to an *ST_Triangle* value.

22) The method *ST_ToPolyhdrlSurf ()* has no input parameters.

23) For the null-call method *ST_ToPolyhdrlSurf()*:

Case:

a) If SELF is of type *ST_PolyhdrlSurface*, then return SELF.

b) If SELF is of type *ST_GeomCollection* and SELF has one element of type *ST_PolyhdrlSurface*, then return the element cast as an *ST_PolyhdrlSurface* value.

c) If SELF is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

d) Otherwise, return an empty set of type *ST_PolyhdrlSurface* with the spatial reference system identifier set to *SELF.ST_SRID()*.

24) Use the method *ST_ToPolyhdrlSurf ()* to define an implicitly invocable cast function to cast an *ST_Geometry* value to an *ST_PolyhdrlSurface* value.

25) The method *ST_ToTIN()* has no input parameters.

26) For the null-call method *ST_ToTIN()*:

Case:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

a) If SELF is of type *ST_TIN*, then return SELF.

b) If SELF is of type *ST_GeomCollection* and SELF has one element of type *ST_TIN*, then return the element cast as an *ST_TIN* value.

c) If SELF is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

d) Otherwise, return an empty set of type *ST_TIN* with the spatial reference system identifier set to *SELF.ST_SRID()*.

27) Use the method *ST_ToTIN()* to define an implicitly invocable cast function to cast an *ST_Geometry* value to an *ST_TIN* value.

28) The method *ST_ToGeomColl()* has no input parameters.

29) For the null-call method *ST_ToGeomColl()*:

Case:

a) If SELF is of type *ST_GeomCollection*, then return SELF.

b) If SELF is of type *ST_Point*, *ST_Curve* or *ST_Surface*, then return an *ST_GeomCollection* value with the spatial reference system identifier set to *SELF.ST_SRID()* and containing one element, SELF.

c) If SELF is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

d) Otherwise, return an empty set of type *ST_GeomCollection* with the spatial reference system identifier set to *SELF.ST_SRID()*.

30) Use the method *ST_ToGeomColl()* to define an implicitly invocable cast function to cast an *ST_Geometry* value to an *ST_GeomCollection* value.

31) The method *ST_ToMultiPoint()* has no input parameters.

32) For the null-call method *ST_ToMultiPoint()*:

Case:

a) If SELF is of type *ST_MultiPoint*, then return SELF.

b) If SELF is of type *ST_GeomCollection*, then return an *ST_MultiPoint* value with the spatial reference system identifier set to *SELF.ST_SRID()* and containing the elements of SELF.

c) If SELF is of type *ST_Point*, then return an *ST_MultiPoint* value with the spatial reference system identifier set to *SELF.ST_SRID()* and containing one element, SELF, cast to an *ST_Point* value.

d) If SELF is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

e) Otherwise, return an empty set of type *ST_MultiPoint* with the spatial reference system identifier set to *SELF.ST_SRID()*.

33) Use the method *ST_ToMultiPoint()* to define an implicitly invocable cast function to cast an *ST_Geometry* value to an *ST_MultiPoint* value.

34) The method *ST_ToMultiCurve()* has no input parameters.

35) For the null-call method *ST_ToMultiCurve()*:

Case:

a) If SELF is of type *ST_MultiCurve*, then return SELF.

b) If SELF is of type *ST_GeomCollection*, then return an *ST_MultiCurve* value with the spatial reference system identifier set to *SELF.ST_SRID()* and containing the elements of SELF.

c) If SELF is of type *ST_Curve* then return an *ST_MultiCurve* value with the spatial reference system identifier set to *SELF.ST_SRID()* and containing one element, SELF, treated aw an *ST_Curve* value.

d) If SELF is not an empty set value, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

e) Otherwise, return an empty set of type *ST_MultiCurve* with the spatial reference system identifier set to *SELF.ST_SRID()*.

36) Use the method *ST_ToMultiCurve()* to define an implicitly invocable cast function to cast an *ST_Geometry* value to an *ST_MultiCurve* value.

37) The method *ST_ToMultiLine()* has no input parameters.

38) For the null-call method *ST_ToMultiLine()*:

Case:

a) If SELF is of type *ST_MultiLineString*, then return SELF.

b) If SELF is of type *ST_GeomCollection*, then return an *ST_MultiLineString* value with the spatial reference system identifier set to *SELF.ST_SRID()* and containing the elements of SELF.

c) If SELF is of type *ST_LineString*, then return an *ST_MultiLineString* value with the spatial reference system identifier set to *SELF.ST_SRID()* and containing one element, SELF, cast to an *ST_LineString* value.

d) If SELF is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

e) Otherwise, return an empty set of type *ST_MultiLineString* with the spatial reference system identifier set to *SELF.ST_SRID()*.

39) Use the method *ST_ToMultiLine()* to define an implicitly invocable cast function to cast an *ST_Geometry* value to an *ST_MultiLineString* value.

40) The method *ST_ToMultiSurface()* has no input parameters.

41) For the null-call method *ST_ToMultiSurface()*:

Case:

a) If SELF is of type *ST_MultiSurface*, then return SELF.

b) If SELF is of type *ST_GeomCollection*, then return an *ST_MultiSurface* value with the spatial reference system identifier set to *SELF.ST_SRID()* and containing the elements of SELF.

c) If SELF is of type *ST_Surface*, then return an *ST_MultiSurface* value with the spatial reference system identifier set to *SELF.ST_SRID()* and containing one element, SELF, treated as an *ST_Surface* value.

d) If SELF is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

e) Otherwise, return an empty set of type *ST_MultiSurface* with the spatial reference system identifier set to *SELF.ST_SRID()*.

42) Use the method *ST_ToMultiSurface()* to define an implicitly invocable cast function to cast an *ST_Geometry* value to an *ST_MultiSurface* value.

43) The method *ST_ToMultiPolygon()* has no input parameters.

44) For the null-call method *ST_ToMultiPolygon()*:

Case:

a) If SELF is of type *ST_MultiPolygon*, then return SELF.

b) If SELF is of type *ST_GeomCollection*, then return an *ST_MultiPolygon* value with the spatial reference system identifier set to *SELF.ST_SRID()* and containing the elements of SELF.

c) If SELF is of type *ST_Polygon*, then return an *ST_MultiPolygon* value with the spatial reference system identifier set to *SELF.ST_SRID()* and containing one element, SELF cast to an *ST_Polygon* value.

d) If SELF is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

e) Otherwise, return an empty set of type *ST_MultiPolygon* with the spatial reference system identifier set to *SELF.ST_SRID()*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

45) Use the method *ST_ToMultiPolygon()* to define an implicitly invocable cast function to cast an *ST_Geometry* value to an *ST_MultiPolygon* value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.47 ST_WKTToSQL Method

Purpose

Return an ST_Geometry value for a given well-known text representation.

Definition

```
CREATE METHOD ST_WKTToSQL
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The method *ST_WKTToSQL(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The parameter *awkt* is the well-known text representation of an *ST_Geometry* value. If *awkt* is not producible in the BNF for <well-known text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
3) The null-call method *ST_WKTToSQL(CHARACTER LARGE OBJECT)* returns an *ST_Geometry* value represented by *awkt*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.48 ST_AsText Method

Purpose

Return the well-known text representation of an ST_Geometry value.

Definition

```
CREATE METHOD ST_AsText()
   RETURNS CHARACTER LARGE OBJECT(ST_MaxGeometryAsText)
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The method *ST_AsText()* has no input parameters.
2) The null-call method *ST_AsText()* returns a CHARACTER LARGE OBJECT value containing the well-known text representation of SELF. Values shall be produced in the BNF for <well-known text representation>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.49 ST_WKBToSQL Method

Purpose

Return an ST_Geometry value for a given well-known binary representation.

Definition

```
CREATE METHOD ST_WKBToSQL
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The method *ST_WKBToSQL(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The parameter *awkb* is the well-known binary representation of an *ST_Geometry* value. If *awkb* is not producible in the BNF for <well-known binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
3) The null-call method *ST_WKBToSQL(BINARY LARGE OBJECT)* returns an *ST_Geometry* value represented by *awkb*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.50 ST_AsBinary Method

Purpose

Return the well-known binary representation of an ST_Geometry value.

Definition

```
CREATE METHOD ST_AsBinary()
   RETURNS BINARY LARGE OBJECT(ST_MaxGeometryAsBinary)
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The method *ST_AsBinary()* has no input parameters.
2) The null-call method *ST_AsBinary()* returns a BINARY LARGE OBJECT value containing the well-known binary representation of SELF. Values shall be produced in the BNF for <well-known binary representation>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.51 ST_GMLToSQL Method

Purpose

Return an ST_Geometry value for a given GML representation.

Definition

```
CREATE METHOD ST_GMLToSQL
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
   RETURNS ST_Geometry
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The method *ST_GMLToSQL(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The parameter *agml* is the GML representation of an *ST_Geometry* value. If *agml* does not contain an XML element as defined in Table 14 — Mapping between ST_Geometry values and GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.

Table 14 — Mapping between ST_Geometry values and GML representation

XML Element in the GML representation	ST_Geometry values
Point	ST_Point
LineString	ST_LineString
Circle	ST_CircularString representing a circle
ArcString	ST_CircularString not representing a circle
CompositeCurve	ST_CompoundCurve
Surface containing a PolygonPatch	ST_CurvePolygon
Polygon	ST_Polygon
Triangle	ST_Triangle
PolyhedralSurface	ST_PolyhdrlSurface
Tin	ST_TIN
MultiGeometry	ST_GeomCollection
MultiPoint	ST_MultiPoint
MultiCurve	ST_MultiCurve
MultiLineString	ST_MultiLineString
MultiSurface	ST_MultiSurface
MultiPolygon	ST_MultiPolygon

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3) The x coordinate value of an *ST_Point* value is represented as either the X XML element of a coord XML element, the first coordinate value of a coordinates XML element or the first coordinate value of a pos XML element.

4) The y coordinate value of an *ST_Point* value is represented as either the Y XML element of a coord XML element, the second coordinate value of a coordinates XML element or the second coordinate value of a pos XML element.

5) The z coordinate value of an *ST_Point* value is represented as either the Z XML element of a coord XML element, the third coordinate value of coordinates XML element or the third coordinate value of a pos XML element.

   Case:

   a) If all the coord XML elements contain the Z XML element, all the coordinates XML elements contain at least 3 coordinate values and the all the pos XML elements have at least 3 coordinate values, then all the *ST_Point* values contained in the resulting *ST_Geometry* value will have x, y, and z coordinate values.

   b) Otherwise, all the *ST_Point* values contained in the resulting *ST_Geometry* value will have only x and y coordinate values.

6) The *ST_Point* values contained in the resulting *ST_Geometry* value will not have m coordinate values.

7) Let *S* be the spatial reference system identifier for the resulting *ST_Geometry* value.

   Case:

   a) If the srsname XML attribute is not specified, then set *S* to 0 (zero).

   b) Otherwise,

      Case:

      i) If the value of srsname XML attribute is producible in the BNF for <spatial reference system>, then

         1) Set *SRT* to the spatial reference system text in the srsname XML attribute.

         2) Select the row in the SPATIAL_REF_SYS view where the SRTEXT column is equal to *SRT*.

            Case:

            A) If the row is not found, then the following exception condition is raised: *SQL/MM Spatial Exception – unknown spatial reference system*.

            B) Otherwise, set *S* to the value of the SRID column in the returned row.

      ii) If the value of the srsname XML attribute is in the form: *ON*:*OI* where *ON* is the organization name and *OI* is organization assigned identifier, then

         1) Let *AN* be the organization name *ON* and *AI* be the organization assigned identifier *OI*.

         2) Select the row in the SPATIAL_REF_SYS view where the AUTH_NAME column is equal to *AN* and AUTH_ID column is equal to *AI*.

            Case:

            A) If the row is not found, then the following exception condition is raised: *SQL/MM Spatial Exception – unknown spatial reference system*.

            B) Otherwise, set *S* to the value of the SRID column in the returned row.

      iii) Otherwise, the following exception condition is raised: *SQL/MM Spatial Exception – unknown spatial reference system*.

8) The null-call method *ST_GMLToSQL(CHARACTER LARGE OBJECT)* returns an *ST_Geometry* value represented by *agml*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.52 ST_AsGML Method

Purpose

Return the GML representation of an ST_Geometry value.

Definition

```
CREATE METHOD ST_AsGML()
   RETURNS CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML)
   FOR ST_Geometry
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The method *ST_AsGML()* has no input parameters.
2) The null-call method *ST_AsGML()* returns a CHARACTER LARGE OBJECT value containing a GML representation. The instantiable subtypes of ST_Geometry are mapped to XML elements in the GML representation as defined in Table 14 — Mapping between ST_Geometry values and GML representation.
3) The srsname XML attribute of the XML element identifies its spatial reference system. Select the row in the SPATIAL_REF_SYS view where the *srid* is equal to *SELF.ST_SRID()*. For the selected row, let *AN* be the value of the AUTH_NAME column, *AI* be the value of the AUTH_ID column and *SRT* be the value of the SRTEXT column.

   Case:

   a) If the *AN* is not the null value and *AI* is not the null value then the srsname XML attribute attribute is specified as:

      srsname='*AN*:*AI*'

   b) Otherwise, the srsname XML attribute is specified as:

      srsname='*SRT*'

4) If SELF.*ST_IsMeasured()* is equal to 1 (one), then the GML representation does not contain the m coordinate values.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.53 ST_GeomFromText Functions

Purpose

Return an ST_Geometry value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_Geometry value.

Definition

```
CREATE FUNCTION ST_GeomFromText
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
    RETURNS ST_Geometry
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    RETURN ST_GeomFromText(awkt, 0)

CREATE FUNCTION ST_GeomFromText
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
     ansrid INTEGER)
    RETURNS ST_Geometry
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    BEGIN
       --
       -- See Description
       --
    END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_GeomFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_GeomFromText(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_GeomFromText(awkt, 0)*.
3) The function *ST_GeomFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_GeomFromText(CHARACTER LARGE OBJECT, INTEGER)*:
   a) The parameter *awkt* is the well-known text representation of an *ST_Geometry* value. If *awkt* is not producible in the BNF for <well-known text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
   b) Return an *ST_Geometry* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.54 ST_GeomFromWKB Functions

Purpose

Return an ST_Geometry value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_Geometry value.

Definition

```
CREATE FUNCTION ST_GeomFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_GeomFromWKB(awkb, 0)

CREATE FUNCTION ST_GeomFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_GeomFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_GeomFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_GeomFromWKB(awkb, 0)*.
3) The function *ST_GeomFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_GeomFromWKB(BINARY LARGE OBJECT, INTEGER)*:
   a) The parameter *awkb* is the well-known binary representation of an *ST_Geometry* value. If *awkb* is not producible in the BNF for <well-known binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
   b) Return an *ST_Geometry* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.55 ST_GeomFromGML Functions

Purpose

Return an ST_Geometry value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representationof an ST_Geometry.

Definition

```
CREATE FUNCTION ST_GeomFromGML
    (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
    RETURNS ST_Geometry
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    RETURN ST_GeomFromGML(agml, 0)

CREATE FUNCTION ST_GeomFromGML
    (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
     ansrid INTEGER)
    RETURNS ST_Geometry
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    BEGIN
       --
       -- See Description
       --
    END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_GeomFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_GeomFromGML(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_GeomFromGML(agml, 0)*.
3) The function *ST_GeomFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_GeomFromGML(CHARACTER LARGE OBJECT, INTEGER)*:
   a) If the parameter *agml* does not contain an XML element as defined in Table 14 — Mapping between ST_Geometry values and GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Return an *ST_Geometry* value represented by *agml* with the spatial reference system identifier set to *ansrid*.
   c) The x coordinate value of an *ST_Point* value is represented as either the X XML element of a coord XML element, the first coordinate value of a coordinates XML element or the first coordinate value of a pos XML element.
   d) The y coordinate value of an *ST_Point* value is represented as either the Y XML element of a coord XML element, the second coordinate value of a coordinates XML element or the second coordinate value of a pos XML element.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

e) The z coordinate value of an *ST_Point* value is represented as either the Z XML element of a coord XML element, the third coordinate value of a coordinates XML element or the third coordinate value of a pos XML element.

   Case:

   i) If all the coord XML elements contain the Z XML element, all the coordinates XML elements contain at least 3 coordinate values and all the pos XML elements contain at least 3 pos values, then all the *ST_Point* values contained in the resulting *ST_Geometry* value will have x, y, and z coordinate values.

   ii) Otherwise, all the *ST_Point* values contained in the resulting *ST_Geometry* value will have only x and y coordinate values.

f) The *ST_Point* values contained in the resulting *ST_Geometry* value will not have m coordinate values.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.56 ST_Geometry Ordering Definition

Purpose

Provide the equals only ordering definition for the ST_Geometry type.

Definition

```
CREATE FUNCTION ST_OrderingEquals
   (ageometry ST_Geometry,
    anothergeometry ST_Geometry)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN
      CASE
         WHEN ageometry.ST_Equals(anothergeometry) = 1 THEN
            0
         ELSE
            1
      END

CREATE ORDERING FOR ST_Geometry
   EQUALS ONLY BY RELATIVE WITH
      FUNCTION ST_OrderingEquals(ST_Geometry, ST_Geometry)
```

Description

1) The function *ST_OrderingEquals(ST_Geometry, ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*,
   b) an *ST_Geometry* value *anothergeometry*.
2) For the null-call function *ST_OrderingEquals(ST_Geometry, ST_Geometry)*:
   Case:
   a) If the value expression: *ageometry.ST_Equals(anothergeometry)* is 1 (one), then return 0 (zero).
   b) Otherwise, return 1 (one).
3) Use the function *ST_OrderingEquals(ST_Geometry, ST_Geometry)* to define ordering for the *ST_Geometry* type.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.57 SQL Transform Functions

Purpose

Define SQL transform functions for the ST_Geometry type.

Definition

```
CREATE TRANSFORM FOR ST_Geometry
   ST_WellKnownText
      (TO SQL WITH METHOD ST_WKTToSQL
         (CHARACTER LARGE OBJECT(ST_MaxGeometryAsText)),
       FROM SQL WITH METHOD ST_AsText())
   ST_WellKnownBinary
      (TO SQL WITH METHOD ST_WKBToSQL
         (BINARY LARGE OBJECT(ST_MaxGeometryAsBinary)),
       FROM SQL WITH METHOD ST_AsBinary())
   ST_GML
      (TO SQL WITH METHOD ST_GMLToSQL
         (CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML)),
       FROM SQL WITH METHOD ST_AsGML())
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) Use the method *ST_WKTToSQL(CHARACTER LARGE OBJECT)* and the method *ST_AsText()* to define the transform group *ST_WellKnownText*.
2) Use the method *ST_WKBToSQL(BINARY LARGE OBJECT)* and the method ST_AsBinary*()* to define the transform group *ST_WellKnownBinary*.
3) Use the method *ST_GMLToSQL(CHARACTER LARGE OBJECT)* and the method *ST_AsGML()* to define the transform group *ST_GML*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.58 <well-known text representation>

Purpose

This subclause contains the definition of <well-known text representation>.

Description

1) The well-known tet representation of an *ST_Geometry* value is defined by the following BNF for <well-known text representation>.

```
<well-known text representation> ::=
      <point text representation>
    | <curve text representation>
    | <surface text representation>
    | <collection text representation>

<point text representation> ::=
      POINT [ <z m> ] <point text>

<curve text representation> ::=
      <linestring text representation>
    | <circularstring text representation>
    | <compoundcurve text representation>

<linestring text representation> ::=
      LINESTRING [ <z m> ] <linestring text body>

<circularstring text representation> ::=
      CIRCULARSTRING [ <z m> ] <circularstring text>

<compoundcurve text representation> ::=
      COMPOUNDCURVE [ <z m> ] <compoundcurve text>

<surface text representation> ::=
      <curvepolygon text representation>
    | <polyhedralsurface text representation>

<curvepolygon text representation> ::=
      CURVEPOLYGON [ <z m> ] <curvepolygon text body>
    | <polygon text representation>

<polygon text representation> ::=
      POLYGON [ <z m> ] <polygon text body>
    | <triangle text representation>

<triangle text representation> ::=
      TRIANGLE [ <z m> ] <triangle text body>

<polyhedralsurface text representation> ::=
      POLYHEDRALSURFACE [ <z m> ] <polyhedralsurface text body>
    | <tin text representation>

<tin text representation> ::=
      TIN [ <z m> ] <tin text body>

<collection text representation> ::=
      <multipoint text representation>
    | <multicurve text representation>
    | <multisurface text representation>
    | <geometrycollection text representation>

<multipoint text representation> ::=
      MULTIPOINT [ <z m> ] <multipoint text>

<multicurve text representation> ::=
      MULTICURVE [ <z m> ] <multicurve text>
    | <multilinestring text representation>

<multilinestring text representation> ::=
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
     MULTILINESTRING [ <z m> ] <multilinestring text>

<multisurface text representation> ::=
     MULTISURFACE [ <z m> ] <multisurface text>
   | <multipolygon text representation>

<multipolygon text representation> ::=
     MULTIPOLYGON [ <z m> ] <multipolygon text>

<geometrycollection text representation> ::=
     GEOMETRYCOLLECTION [ <z m> ] <geometrycollection text>

<linestring text body> ::=
     <linestring text>

<curvepolygon text body> ::=
     <curvepolygon text>

<polygon text body> ::=
     <polygon text>

<triangle text body> ::=
     <triangle text>

<polyhedralsurface text body> ::=
     <polyhedralsurface text>

<tin text body> ::=
     <tin text>

<point text> ::=
     <empty set>
   | <left paren> <point> <right paren>

<point> ::= <x> <y> [ <z> ] [ <m> ]

<x> ::= <number>

<y> ::= <number>

<z> ::= <number>

<m> ::= <number>

<linestring text> ::=
     <empty set>
   | <left paren> <point>
        { <comma> <point> }... <right paren>

<circularstring text> ::=
     <empty set>
   | <left paren> <point>
        { <comma> <point> }... <right paren>

<compoundcurve text> ::=
     <empty set>
   | <left paren> <single curve text>
        { <comma> <single curve text> }... <right paren>

<single curve text> ::=
     <linestring text body>
   | <circularstring text representation>

<curve text> ::=
     <linestring text body>
   | <circularstring text representation>
   | <compoundcurve text representation>
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
<ring text> ::=
     <linestring text body>
   | <circularstring text representation>
   | <compoundcurve text representation>

<surface text> ::=
     CURVEPOLYGON <curvepolygon text body>
   | <polygon text body>

<curvepolygon text> ::=
     <empty set>
   | <left paren> <ring text>
        { <comma> <ring text> }... <right paren>

<polygon text> ::=
     <empty set>
   | <left paren> <linestring text>
        { <comma> <linestring text> }... <right paren>

<triangle text> ::=
     <empty set>
   | <left paren> <point> <comma> <point> <comma> <point> <right paren>

<polyhedralsurface text> ::=
     <empty set>
   | <left paren> PATCHES <polygonpatches text> <right paren>

<polygonpatches text> ::=
     <left paren> <polygon text representation>
        { <comma> <polygon text representation> }... <right paren>

<tin text> ::=
     <empty set>
   | <left paren> PATCHES <trianglepatches text>
        [ ELEMENTS <tinelement list> ]
        [ <maxsidelength> ]
        <right paren>

<trianglepatches text> ::=
     <left paren> <triangle text body>
        { <comma> <triangle text body> }... <right paren>

<tinelement list> ::=
     <left paren> <tinelementtype text>
        { <comma> <tinelementtype text> }... <right paren>

<tinelementtype text> ::=
     <randompoints representation>
   | <groupspot representation>
   | <boundary representation>
   | <breakline representation>
   | <softbreak representation>
   | <controlcontour representation>
   | <breakvoid representation>
   | <drapevoid representation>
   | <void representation>
   | <hole representation>
   | <stopline representation>

<randompoints representation> ::=
     POINTS <elementlabel text> <multipoint text representation>

<groupspot representation> ::=
     GROUPSPOT <elementlabel text> <multipoint text representation>
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
<boundary representation> ::=
     BOUNDARY <elementlabel text> <polygon text representation>

<breakline representation> ::=
     BREAKLINE <elementlabel text> <linestring text representation>

<softbreak representation> ::=
     SOFTBREAK <elementlabel text> <linestring text representation>

<controlcontour representation> ::=
     CONTROLCONTOUR <elementlabel text> <linestring text representation>

<breakvoid representation> ::=
     BREAKVOID <elementlabel text> <polygon text representation>

<drapevoid representation> ::=
     DRAPEVOID <elementlabel text> <polygon text representation>

<void representation> ::=
     VOID <elementlabel text> <polygon text representation>

<hole representation> ::=
     HOLE <elementlabel text> <polygon text representation>

<stopline representation ::=
     STOPLINE <elementlabel text> <linestring text representation>

<elementlabel text> ::=
     [ ID <element id> ] [ TAG <element tag> ]

<multipoint text> ::=
     <empty set>
   | <left paren> <point text>
        { <comma> <point text > }... <right paren>

<multicurve text> ::=
     <empty set>
   | <left paren> <curve text>
        { <comma> <curve text> }... <right paren>

<multilinestring text> ::=
     <empty set>
   | <left paren> <linestring text body>
        { <comma> <linestring text body> }... <right paren>

<multisurface text> ::=
     <empty set>
   | <left paren> <surface text>
        { <comma> <surface text> }... <right paren>

<multipolygon text> ::=
     <empty set>
   | <left paren> <polygon text body>
        { <comma> <polygon text body> }... <right paren>

<geometrycollection text> ::=
     <empty set>
   | <left paren> <well-known text representation>
        { <comma> <well-known text representation> }... <right paren>

<empty set> ::= EMPTY

<z m> ::=
     ZM
   | Z
   | M

<double quote> ::=
     !! See Subclause 13.1.9, "<spatial reference system>"
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
<letters> ::=
     !! See Subclause 13.1.9, "<spatial reference system>"

<element id> ::=
     <signed integer>

<element tag> ::=
     <double quote> <letters> <double quote>

<maxsidelength> ::=
     MAXSIDELENGTH <number>

<signed integer> ::=
     !! See Subclause 5.3, "<literal>", in Part 2 of ISO/IEC 9075
```

a) Case:

   i) If <well-known text representation> immediately contains a <point text representation>, then <well-known text representation> produces an *ST_Point* value specified by the immediately contained <point text representation>.

   ii) If <well-known text representation> immediately contains a <curve text representation>, then <well-known text representation> produces an *ST_Curve* value specified by the immediately contained <curve text representation>.

   iii) If <well-known text representation> immediately contains a <surface text representation>, then <well-known text representation> produces an *ST_Surface* value specified by the immediately contained <surface text representation>.

   iv) Otherwise, <well-known text representation> produces an *ST_GeomCollection* value specified by the immediately contained <collection text representation>.

b) <point text representation> is the well-known text representation for an *ST_Point* value that is produced by <point text>.

c) Case:

   i) If <curve text representation> immediately contains a <linestring text representation>, then <curve text representation> produces an *ST_LineString* value specified by the immediately contained <linestring text representation>.

   ii) If <curve text representation> immediately contains a <circularstring text representation>, then <curve text representation> produces an *ST_CircularString* value specified by the immediately contained <circularstring text representation>.

   iii) Otherwise, <curve text representation> produces an *ST_CompoundCurve* value specified by the immediately contained <compoundcurve text representation>.

d) <linestring text representation> is the well-known text representation for an *ST_LineString* value. <linestring text representation> produces an *ST_LineString* value specified by the immediately contained <linestring text body>.

e) <circularstring text representation> is the well-known text representation for an *ST_CircularString* value. Let *APA* be the *ST_Point* ARRAY value produced by a <circularstring text>.

   Case:

   i) If the cardinality of *APA* is 0 (zero), then <circularstring text representation> produces an empty set of type *ST_CircularString*.

   ii) Otherwise, <circularstring text representation> produces an *ST_CircularString* value as the result of the value expression: *NEW ST_CircularString(APA)*.

f) <compoundcurve text representation> is the well-known text representation for an *ST_CompoundCurve* value. Let *ACA* be the *ST_Curve* ARRAY value produced by a <compoundcurve text>.

   Case:

   i) If the cardinality of *ACA* is 0 (zero), then <compoundcurve text representation> produces an empty set of type *ST_CompoundCurve*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) Otherwise, <compoundcurve text representation> produces an *ST_CompoundCurve* value as the result of the value expression: *NEW ST_CompoundCurve(ACA)*.

g) Case:

i) If <surface text representation> immediately contains a <curvepolygon text representation>, then <surface text representation> produces an *ST_CurvePolygon* value specified by the immediately contained <curvepolygon text representation>.

ii) Otherwise, <surface text representation> produces an *ST_PolyhdrlSurface* value specified by the immediately contained <polyhedralsurface text representation>.

h) <curvepolygon text representation> is the well-known text representation for an *ST_CurvePolygon* value.

Case:

i) If <curvepolygon text representation> immediately contains a <curvepolygon text body>, then <curvepolygon text representation> produces an *ST_CurvePolygon* value specified by the immediately contained <curvepolygon text body>.

ii) Otherwise, <curvepolygon text representation> produces an *ST_Polygon* value specified by the immediately contained <polygon text representation>.

i) Case:

i) If <polygon text representation> immediately contains a <polygon text body>, then <polygon text representation> produces an *ST_Polygon* value specified by the immediately contained <polygon text body>.

ii) Otherwise, <polygon text representation> produces an *ST_Triangle* value specified by the immediately contained <triangle text representation>.

j) <triangle text representation> is the well-known text representation for an *ST_Triangle* value. <triangle text representation> produces an *ST_Triangle* value specified by the immediately contained <triangle text body>.

k) <polyhedralsurface text representation> is the well-known text representation for an *ST_PolyhdrlSurface* value.

Case:

i) If <polyhedralsurface text representation> immediately contains a <polyhedralsurface text body>, then <polyhedralsurface text representation> produces an *ST_PolyhdrlSurface* value specified by the immediately contained <polyhedralsurface text body>.

ii) Otherwise, <polyhedralsurface text representation> produces an *ST_TIN* value specified by the immediately contained <tin text body>.

l) <tin text representation> is the well-known text representation for an *ST_TIN* value. <tin text representation> produces an *ST_TIN* value specified by the immediately contained <tin text body>.

m) Case:

i) If <collection text representation> immediately contains a <multipoint text representation>, then <collection text representation> produces an *ST_MultiPoint* value specified by the immediately contained <multipoint text representation>.

ii) If <collection text representation> immediately contains a <multicurve text representation>, then <collection text representation> produces an *ST_MultiCurve* value specified by the immediately contained <multicurve text representation>.

iii) If <collection text representation> immediately contains a <multisurface text representation>, then <collection text representation> produces an *ST_MultiSurface* value specified by the immediately contained <multisurface text representation>.

iv) Otherwise, <collection text representation> produces an *ST_GeomCollection* value specified by the immediately contained <geometrycollection text representation>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

n) <multipoint text representation> is the well-known text representation for an *ST_MultiPoint* value. Let *APA* be the *ST_Point* ARRAY value produced by a <multipoint text>.

Case:

i) If the cardinality of *APA* is 0 (zero), then <multipoint text representation> produces an empty set of type *ST_MultiPoint*.

ii) Otherwise, <multipoint text representation> produces an *ST_MultiPoint* value as the result of the value expression: *NEW ST_MultiPoint(APA)*.

o) Case:

i) If <multicurve text representation> immediately contains a <multicurve text>, then <multicurve text representation> produces an *ST_MultiCurve* value. Let *ACA* be the *ST_Curve* ARRAY value produced by a <multicurve text>.

Case:

1) If the cardinality of *ACA* is 0 (zero), then <multicurve text representation> produces an empty set of type *ST_MultiCurve*.

2) Otherwise, <multicurve text representation> produces an *ST_MultiCurve* value as the result of the value expression: *NEW ST_MultiCurve(ACA)*.

ii) Otherwise, <multicurve text representation> produces an *ST_MultiLineString* value specified by the immediately contained <multilinestring text representation>.

p) <multilinestring text representation> is the well-known text representation for an *ST_MultiLineString* value. Let *ALSA* be the *ST_LineString* ARRAY value produced by a <multilinestring text>.

Case:

i) If the cardinality of *ALSA* is 0 (zero), then <multilinestring text representation> produces an empty set of type *ST_MultiLineString*.

ii) Otherwise, <multilinestring text representation> produces an *ST_MultiLineString* value as the result of the value expression: *NEW ST_MultiLineString(ALSA)*.

q) Case:

i) If <multisurface text representation> immediately contains a <multisurface text>, then <multisurface text representation> produces an *ST_MultiSurface* value. Let *ASA* be the *ST_Surface* ARRAY value produced by a <multisurface text>.

Case:

1) If the cardinality of *ASA* is 0 (zero), then <multisurface text representation> produces an empty set of type *ST_MultiSurface*.

2) Otherwise, <multisurface text representation> produces an *ST_MultiSurface* value as the result of the value expression: *NEW ST_MultiSurface(ASA)*.

ii) Otherwise, <multisurface text representation> produces an *ST_MultiPolygon* value specified by the immediately contained <multipolygon text representation>.

r) <multipolygon text representation> is the well-known text representation for an *ST_MultiPolygon* value. Let *APA* be the *ST_Polygon* ARRAY value produced by a <multipolygon text>.

Case:

i) If the cardinality of *APA* is 0 (zero), then <multipolygon text representation> produces an empty set of type *ST_MultiPolygon*.

ii) Otherwise, <multipolygon text representation> produces an *ST_MultiPolygon* value as the result of the value expression: *NEW ST_MultiPolygon(APA)*.

s) <geometrycollection text representation> is the well-known text representation for an *ST_GeomCollection*. Let *AGA* be the *ST_Geometry* ARRAY value produced by a <geometrycollection text>.

Case:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

i) If the cardinality of *AGA* is 0 (zero), then <geometrycollection text representation> produces an empty set of type *ST_GeomCollection*.

ii) Otherwise, <geometrycollection text representation> produces an *ST_GeomCollection* value as the result of the value expression: *NEW ST_GeomCollection(AGA)*.

t) Let *APA* be the *ST_Point* ARRAY value produced by a <linestring text> in <linestring text body>.

Case:

i) If the cardinality of *APA* is 0 (zero), then <linestring text body> produces an empty set of type *ST_LineString*.

ii) Otherwise, <linestring text body> produces an *ST_LineString* value as the result of the value expression: *NEW ST_LineString(APA)*.

u) Let *ACA* be the *ST_Curve* ARRAY value produced by a <curvepolygon text> in <curvepolygon text body>.

Case:

i) If the cardinality of *ACA* is 0 (zero), then <curvepolygon text body> produces an empty set of type *ST_CurvePolygon*.

ii) If the cardinality of *ACA* is 1 (one), then let *AER* be the element of *ACA*. <curvepolygon text body> produces an *ST_CurvePolygon* value as the result of the value expression: *NEW ST_CurvePolygon(AER)*.

iii) Otherwise, let *AER* be the first element in *ACA* and let *AIR* be the sublist of *ACA* containing the other elements of *ACA*. <curvepolygon text body>< produces an *ST_CurvePolygon* value as the result of the value expression: *NEW ST_CurvePolygon(AER, AIR)*.

v) Let *ALSA* be the *ST_LineString* ARRAY value produced by a <polygon text> in <polygon text body>.

Case:

i) If the cardinality of *ALSA* is 0 (zero), then <polygon text body> produces an empty set of type *ST_Polygon*.

ii) If the cardinality of *ALSA* is 1 (one), then let *ALS* be the element of *ALSA*. <polygon text body> produces an *ST_Polygon* value as the result of the value expression: *NEW ST_Polygon*(*ALS*).

iii) Otherwise, let *AER* be the first element in *ALSA* and let *AIR* be the sublist of *ALSA* containing the other elements of *ALSA*. <polygon text body> produces an *ST_Polygon* value as the result of the value expression: *NEW ST_Polygon(AER, AIR)*.

w) Let *P1, P2 and P3* be the *ST_Point* values produced by a <triangle text> in <triangle text body>.

Case:

i) If *P1, P2, P3* values are not provided, then <polygon text body> produces an empty set of type *ST_Triangle*.

ii) Otherwise, let *APA* be an ST_Point ARRAY containing four elements: *P1, P2, P3, P1*. <triangle text body> produces an *ST_Triangle* value as the result of the value expression: *NEW ST_Triangle*(*APA*).

x) Let *APA* be the *ST_Polygon* ARRAY value produced by a <polygonpatches text> in <polyhedralsurface text>.

Case:

i) If the cardinality of *APA* is 0 (zero), then <polyhedralsurface text> produces an empty set of type *ST_PolyhdrlSurface*.

ii) Otherwise, <polyhedralsurface text> produces an *ST_PolyhdrlSurface* value as the result of the value expression: *NEW ST_PolyhdrlSurface(APA)*.

y) Let *ATA* be the *ST_Triangle* ARRAY value produced by the <trianglepatches text> and *AEA* be the ST_TINElement ARRAY value produced by the <tinelement list> and MSL be the DOUBLE PRECISION value produced by the <maxsidelength>, all in the same <tin text>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Case:

i) If the cardinality of *ATA* is 0 (zero), then <tin text> produces an empty set of type *ST_TIN*.

ii) Otherwise, <tin text> produces an *ST_TIN* value as the result of the value expression: *NEW ST_TIN(ATA, AEA, MSL)*.

z) Case:

i) If any <well-known text representation> *WKT1* contains <z m>, then every <well-known text representation> *WKT2* contained in *WKT1* shall contain <z m> with the same value.

ii) Otherwise, an exception condition is raised: *SQL/MM Spatial exception – mixed coordinate dimensions*.

aa) Case:

i) If <z m> is specified, then

Case:

1) If <z m> immediately contains ZM, then let *ZORM* be ZM.

2) If <z m> immediately contains Z, then let *ZORM* be Z

3) If <z m> immediately contains M, then let *ZORM* be M.

ii) Otherwise, let *ZORM* be 2D.

ab) Case:

i) If <point text> immediately contains an <empty set>, then:

Case:

1) If *ZORM* is ZM, then <point text> produces an empty set of type *ST_Point* as the result of the value expression: *NEW ST_Point(NULL, NULL, NULL, NULL)*.

2) If *ZORM* is Z, then <point text> produces an empty set of type *ST_Point* as the result of the value expression: *NEW ST_Point(NULL, NULL, NULL)*.

3) If *ZORM* is M, then <point text> produces an empty set of type *ST_Point* as the result of the value expression: *NEW ST_Point(NULL, NULL, NULL, 0)*.

4) Otherwise, <point text> produces an empty set of type *ST_Point* as the result of the value expression: *NEW ST_Point()*.

ii) Otherwise, <point text> produces the *ST_Point* value from <point>.

ac) Let *XC* be the DOUBLE PRECISION value specified by <x> in <point> and *YC* be the DOUBLE PRECISION value specified by <y> in <point>.

Case:

i) If *ZORM* is ZM then,

1) If <point> does not contain <z> or <point> does not contain <m>, then an exception condition is raised: *SQL/MM Spatial exception – mixed coordinate dimensions*.

2) Let *ZC* be the DOUBLE PRECISION value specified by <z> in <point>.

3) Let *MC* be the DOUBLE PRECISION value specified by <m> in <point>.

4) <point> produces an *ST_Point* value as the result of the value expression: *NEW ST_Point(XC, YC, ZC, MC)*.

ii) If *ZORM* is Z then,

1) If <point> does not contain <z> or <point> contains <m>, then an exception condition is raised: *SQL/MM Spatial exception – mixed coordinate dimensions*.

2) Let *ZC* be the DOUBLE PRECISION value specified by <z> in <point>.

3) <point> produces an *ST_Point* value as the result of the value expression: *NEW ST_Point(XC, YC, ZC)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

iii) If *ZORM* is M then,

1) If <point> contains <z> or <point> does not contain <m>, then an exception condition is raised: *SQL/MM Spatial exception – mixed coordinate dimensions*.

2) Let *MC* be the DOUBLE PRECISION value specified by <m> in <point>.

3) <point> produces an *ST_Point* value as the result of the value expression: *NEW ST_Point(XC, YC, NULL, MC)*.

iv) Otherwise,

1) If <point> contains <z> or <point> contains <m>, then an exception condition is raised: *SQL/MM Spatial exception – mixed coordinate dimensions*.

2) <point> produces an *ST_Point* value as the result of the value expression: *NEW ST_Point(XC, YC)*.

ad) Case:

i) If <linestring text> immediately contains an <empty set>, then <linestring text> produces an empty *ST_Point* ARRAY value.

ii) Otherwise, <linestring text> produces an *ST_Point* ARRAY value that contains the *ST_Point* values specified by the immediately contained <point>s.

ae) Case:

i) If <circularstring text> immediately contains an <empty set>, then <circularstring text> produces an empty *ST_Point* ARRAY value.

ii) Otherwise, <circularstring text> produces an *ST_Point* ARRAY value that contains the *ST_Point* values specified by the immediately contained <point>s.

af) Case:

i) If <compoundcurve text> immediately contains an <empty set>, then <compoundcurve text> produces an empty *ST_Curve* ARRAY value.

ii) Otherwise, <compoundcurve text> produces an *ST_Curve* ARRAY value that contains the *ST_Curve* values specified by the immediately contained <single curve text>s.

ag) Case:

i) If <single curve text> immediately contains a <linestring text body>, then <single curve text> produces an *ST_LineString* value specified by the immediately contained <linestring text body>.

ii) Otherwise, <single curve text> produces an *ST_CircularString* value specified by the immediately contained <circularstring text representation>.

ah) Case:

i) If <curve text> immediately contains a <linestring text body>, then <curve text> produces an *ST_LineString* value specified by the immediately contained <linestring text body>.

ii) If <curve text> immediately contains a <circularstring text representation>, then <curve text> produces an *ST_CircularString* value specified by the immediately contained <circularstring text>.

iii) Otherwise, <curve text> produces an *ST_CompoundCurve* value specified by the immediately contained <compoundcurve text representation>.

ai) Case:

i) If <ring text> immediately contains a <linestring text body>, then <ring text> produces an *ST_LineString* value specified by the immediately contained <linestring text body>.

ii) If <ring text> immediately contains a <circularstring text representation>, then <ring text> produces an *ST_CircularString* value specified by the immediately contained <circularstring text representation>.

iii) Otherwise, <ring text> produces an *ST_CompoundCurve* value specified by the immediately contained <compoundcurve text representation>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

aj) Case:

i) If <surface text> immediately contains a <curvepolygon text body>, then <surface text> produces an *ST_CurvePolygon* value specified by the immediately contained <curvepolygon text body>.

ii) Otherwise, <surface text> produces an *ST_Polygon* value specified by the immediately contained <polygon text body>.

ak) Case:

i) If <curvepolygon text> immediately contains an <empty set>, then <curvepolygon text> produces an empty *ST_Curve* ARRAY value.

ii) Otherwise, <curvepolygon text> produces an *ST_Curve* ARRAY value that contains the *ST_Curve* values specified by the immediately contained <ring text>s.

al) Case:

i) If <polygon text> immediately contains an <empty set>, then <polygon text> produces an empty *ST_LineString* ARRAY value.

ii) Otherwise, <polygon text> produces an *ST_LineString* ARRAY value that contains the *ST_LineString* values specified by the immediately contained <linestring text>s.

am) Case:

i) If <triangle text> immediately contains an <empty set>, then <triangle text> produces an empty *ST_Point* ARRAY value.

ii) Otherwise, <triangle text> produces an *ST_Point* ARRAY value that contains the three *ST_Point* values specified by the immediately contained <point>s.

an) Case:

i) If <polyhedralsurface text> immediately contains an <empty set>, then <polyhedralsurface text> produces an empty *ST_Polygon ARRAY* value.

ii) Otherwise, <polyhedralsurface text> produces an *ST_Polygon ARRAY* value that contains the *ST_Polygon* values specified by the immediately contained <polygonpatches text>s.

ao) Case:

i) If <tin text> immediately contains an <empty set>, then <polyhedralsurface text> produces an empty *ST_Triangle ARRAY* value.

ii) Otherwise, <tin text> produces an *ST_Triangle ARRAY* value that contains the *ST_Triangle* values specified by the immediately contained <trianglepatches text>s.

ap) Case:

i) If <multipoint text> immediately contains an <empty set>, then <multipoint text> produces an empty *ST_Point* ARRAY value.

ii) Otherwise, <multipoint text> produces an *ST_Point* ARRAY value that contains the *ST_Point* values specified by the immediately contained <point text>s.

aq) Case:

i) If <multicurve text> immediately contains an <empty set>, then <multicurve text> produces an empty *ST_Curve* ARRAY value.

ii) Otherwise, <multicurve text> produces an *ST_Curve* ARRAY value that contains the *ST_Curve* values specified by the immediately contained <curve text>s.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ar) Case:

i) If <multilinestring text> immediately contains an <empty set>, then <multilinestring text> produces an empty *ST_LineString* ARRAY value.

ii) Otherwise, <multilinestring text> produces an *ST_LineString* ARRAY value that contains the ST_LineString values specified by the immediately contained <linestring text body>s.

as) Case:

i) If <multisurface text> immediately contains an <empty set>, then <multisurface text> produces an empty *ST_Surface* ARRAY value.

ii) Otherwise, <multisurface text> produces an *ST_Surface* ARRAY value that contains the ST_Surface values from the immediately contained <surface text>s.

at) Case:

i) If <multipolygon text> immediately contains an <empty set>, then <multipolygon text> produces an empty *ST_Polygon* ARRAY value.

ii) Otherwise, <multipolygon text> produces an *ST_Polygon* ARRAY value that contains the ST_Polygon values from the immediately contained <polygon text body>s.

au) Case:

i) If <geometrycollection text> immediately contains an <empty set>, then <geometrycollection text> produces an empty *ST_Geometry* ARRAY value.

ii) Otherwise, an ST_Geometry ARRAY value that contains the ST_Geometry values from the immediately contained <well-known text representation>s.

av) The list of keywords are BOUNDARY, BREAKLINE, BREAKVOID, CIRCULARSTRING, COMPOUNDCURVE, CONTROLCONTOUR, CURVEPOLYGON, ELEMENTS, EMPTY, DRAPEVOID, GEOMETRYCOLLECTION, GROUPSPOT, HOLE, ID, LINESTRING, M, MAXSIDELENGTH, MULTICURVE, MULTILINESTRING, MULTIPOINT, MULTIPOLYGON, MULTISURFACE, POINT, POINTS, POLYGON, POLYHEDRALSURFACE, SOFTBREAK, STOPLINE, TAG, TIN, TRIANGLE, VOID, Z, and ZM.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5.1.59 <well-known binary representation>

Purpose

This subclause contains the definition of <well-known binary representation>.

Description

1) The well-known binary representation of an *ST_Geometry* value is defined by the following BNF for <well-known binary representation>.

```
<well-known binary representation> ::=
     <well-knownzm binary representation>
   | <well-knownz binary representation>
   | <well-knownm binary representation>
   | <well-known2d binary representation>

<well-knownzm binary representation> ::=
     <pointzm binary representation>
   | <curvezm binary representation>
   | <surfacezm binary representation>
   | <collectionzm binary representation>

<well-knownz binary representation> ::=
     <pointz binary representation>
   | <curvez binary representation>
   | <surfacez binary representation>
   | <collectionz binary representation>

<well-knownm binary representation> ::=
     <pointm binary representation>
   | <curvem binary representation>
   | <surfacem binary representation>
   | <collectionm binary representation>

<well-known2d binary representation> ::=
     <point binary representation>
   | <curve binary representation>
   | <surface binary representation>
   | <collection binary representation>

<pointzm binary representation> ::=
     <byte order> <wkbpointzm> [ <wkbpointzm binary> ]

<pointz binary representation> ::=
     <byte order> <wkbpointz> [ <wkbpointz binary> ]

<pointm binary representation> ::=
     <byte order> <wkbpointm> [ <wkbpointm binary> ]

<point binary representation> ::=
     <byte order> <wkbpoint> [ <wkbpoint binary> ]

<curvezm binary representation> ::=
     <linestringzm binary representation>
   | <circularstringzm binary representation>
   | <compoundcurvezm binary representation>

<curvez binary representation> ::=
     <linestringz binary representation>
   | <circularstringz binary representation>
   | <compoundcurvez binary representation>

<curvem binary representation> ::=
     <linestringm binary representation>
   | <circularstringm binary representation>
   | <compoundcurvem binary representation>

<curve binary representation> ::=
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
      <linestring binary representation>
    | <circularstring binary representation>
    | <compoundcurve binary representation>

<linestringzm binary representation> ::=
      <byte order> <wkblinestringzm> [ <num> <wkbpointzm binary>... ]

<linestringz binary representation> ::=
      <byte order> <wkblinestringz> [ <num> <wkbpointz binary>... ]

<linestringm binary representation> ::=
      <byte order> <wkblinestringm> [ <num> <wkbpointm binary>... ]

<linestring binary representation> ::=
      <byte order> <wkblinestring> [ <num> <wkbpoint binary>... ]

<circularstringzm binary representation> ::=
      <byte order> <wkbcircularstringzm> [ <num> <wkbpointzm binary>... ]

<circularstringz binary representation> ::=
      <byte order> <wkbcircularstringz> [ <num> <wkbpointz binary>... ]

<circularstringm binary representation> ::=
      <byte order> <wkbcircularstringm> [ <num> <wkbpointm binary>... ]

<circularstring binary representation> ::=
      <byte order> <wkbcircularstring> [ <num> <wkbpoint binary>... ]

<compoundcurvezm binary representation> ::=
      <byte order> <wkbcompoundcurvezm> [ <num> <wkbcurvezm binary>... ]

<compoundcurvez binary representation> ::=
      <byte order> <wkbcompoundcurvez> [ <num> <wkbcurvez binary>... ]

<compoundcurvem binary representation> ::=
      <byte order> <wkbcompoundcurvem> [ <num> <wkbcurvem binary>... ]

<compoundcurve binary representation> ::=
      <byte order> <wkbcompoundcurve> [ <num> <wkbcurve binary>... ]

<surfacezm binary representation> ::=
      <curvezmpolygonzm binary representation>
    | <polyhedralsurfacezm binary representation>

<surfacez binary representation> ::=
      <curvepolygonz binary representation>
    | <polyhedralsurfacez binary representation>

<surfacem binary representation> ::=
      <curvepolygonm binary representation>
    | <polyhedralsurfacem binary representation>

<surface binary representation> ::=
      <curvepolygon binary representation>
    | <polyhedralsurface binary representation>

<curvepolygonzm binary representation> ::=
      <byte order> <wkbcurvepolygonzm> [ <num> <wkbringzm binary>... ]
    | <polygonzm binary representation>

<curvepolygonz binary representation> ::=
      <byte order> <wkbcurvepolygonz> [ <num> <wkbringz binary>... ]
    | <polygonz binary representation>

<curvepolygonm binary representation> ::=
      <byte order> <wkbcurvepolygonm> [ <num> <wkbringm binary>... ]
    | <polygonm binary representation>
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
<curvepolygon binary representation> ::=
     <byte order> <wkbcurvepolygon> [ <num> <wkbring binary>... ]
   | <polygon binary representation>

<polygonzm binary representation> ::=
     <byte order> <wkbpolygonzm> [ <num> <wkblinearringzm binary>... ]
   | <trianglezm binary representation>

<polygonz binary representation> ::=
     <byte order> <wkbpolygonz> [ <num> <wkblinearringz binary>... ]
   | <trianglez binary representation>

<polygonm binary representation> ::=
     <byte order> <wkbpolygonm> [ <num> <wkblinearringm binary>... ]
   | <trianglem binary representation>

<polygon binary representation> ::=
     <byte order> <wkbpolygon> [ <num> <wkblinearring binary>... ]
   | <triangle binary representation>

<trianglezm binary representation> ::=
     <byte order> <wkbtrianglezm>
       [ <wkbpointzm binary> <wkbpointzm binary> <wkbpointzm binary> ]

<trianglez binary representation> ::=
     <byte order> <wkbtrianglez>
       [ <wkbpointz binary> <wkbpointz binary> <wkbpointz binary> ]

<trianglem binary representation> ::=
     <byte order> <wkbtrianglem>
       [ <wkbpointm binary> <wkbpointm binary> <wkbpointm binary> ]

<triangle binary representation> ::=
     <byte order> <wkbtriangle>
       [ <wkbpoint binary> <wkbpoint binary> <wkbpoint binary> ]

<polyhedralsurfacezm binary representation> ::=
     <byte order> <wkbpolyhedralsurfacezm>
       [ <num> <wkbpolygonpatchzm binary>... ]
   | <tinzm binary representation>

<polyhedralsurfacez binary representation> ::=
     <byte order> <wkbpolyhedralsurfacez>
       [ <num> <wkbpolygonpatchz binary>... ]
   | <tinz binary representation>

<polyhedralsurfacem binary representation> ::=
     <byte order> <wkbpolyhedralsurfacem>
       [ <num> <wkbpolygonpatchm binary>... ]
   | <tinm binary representation>

<polyhedralsurface binary representation> ::=
     <byte order> <wkbpolyhedralsurface>
       [ <num> <wkbpolygonpatch binary>... ]
   | <tin binary representation>

<tinzm binary representation> ::=
     <byte order> <wkbtinzm>
       [ <num> <wkbtrianglepatchzm binary>...
         <nume> <wkbtinelement binary>...
         <wkbmaxsidelength> ]

<tinz binary representation> ::=
     <byte order> <wkbtinz>
       [ <num> <wkbtrianglepatchz binary>...
         <nume> <wkbtinelement binary>...
         <wkbmaxsidelength> ]
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
<tinm binary representation> ::=
     <byte order> <wkbtinm>
       [ <num> <wkbtrianglepatchm binary>...
         <nume> <wkbtinelement binary>...
         <wkbmaxsidelength> ]

<tin binary representation> ::=
     <byte order> <wkbtin>
       [ <num> <wkbtrianglepatch binary>...
         <nume> <wkbtinelement binary>...
         <wkbmaxsidelength> ]

<tinelement binary representation> ::=
   <byte order> <tinelement element type> <tinelement element id>
   <tinelement element tag> <well-known binary representation>

<tinelement element type> ::=
     <byte> <letters>

<tinelement element id> ::=
     <signed integer>

<tinelement element tag> ::=
     <byte> <letters>

<collectionzm binary representation> ::=
     <multipointzm binary representation>
   | <multicurvezm binary representation>
   | <multisurfacezm binary representation>
   | <geometrycollectionzm binary representation>

<collectionz binary representation> ::=
     <multipointz binary representation>
   | <multicurvez binary representation>
   | <multisurfacez binary representation>
   | <geometrycollectionz binary representation>

<collectionm binary representation> ::=
     <multipointm binary representation>
   | <multicurvem binary representation>
   | <multisurfacem binary representation>
   | <geometrycollectionm binary representation>

<collection binary representation> ::=
     <multipoint binary representation>
   | <multicurve binary representation>
   | <multisurface binary representation>
   | <geometrycollection binary representation>

<multipointzm binary representation> ::=
     <byte order> <wkbmultipointzm>
        [ <num> <pointzm binary representation>... ]

<multipointz binary representation> ::=
     <byte order> <wkbmultipointz>
        [ <num> <pointz binary representation>... ]

<multipointm binary representation> ::=
     <byte order> <wkbmultipointm>
        [ <num> <pointm binary representation>... ]

<multipoint binary representation> ::=
     <byte order> <wkbmultipoint>
        [ <num> <point binary representation>... ]
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
<multicurvezm binary representation> ::=
     <byte order> <wkbmulticurvezm>
        [ <num> <curvezm binary representation>... ]
   | <multilinestringzm binary representation>

<multicurvez binary representation> ::=
     <byte order> <wkbmulticurvez>
        [ <num> <curvez binary representation>... ]
   | <multilinestringz binary representation>

<multicurvem binary representation> ::=
     <byte order> <wkbmulticurvem>
        [ <num> <curvem binary representation>... ]
   | <multilinestringm binary representation>

<multicurve binary representation> ::=
     <byte order> <wkbmulticurve>
        [ <num> <curve binary representation>... ]
   | <multilinestring binary representation>

<multilinestringzm binary representation> ::=
     <byte order> <wkbmultilinestringzm>
        [ <num> <linestringzm binary representation>... ]

<multilinestringz binary representation> ::=
     <byte order> <wkbmultilinestringz>
        [ <num> <linestringz binary representation>... ]

<multilinestringm binary representation> ::=
     <byte order> <wkbmultilinestringm>
        [ <num> <linestringm binary representation>... ]

<multilinestring binary representation> ::=
     <byte order> <wkbmultilinestring>
        [ <num> <linestring binary representation>... ]

<multisurfacezm binary representation> ::=
     <byte order> <wkbmultisurfacezm>
        [ <num> <surfacezm binary representation>... ]
   | <multipolygonzm binary representation>

<multisurfacez binary representation> ::=
     <byte order> <wkbmultisurfacez>
        [ <num> <surfacez binary representation>... ]
   | <multipolygonz binary representation>

<multisurfacem binary representation> ::=
     <byte order> <wkbmultisurfacem>
        [ <num> <surfacem binary representation>... ]
   | <multipolygonm binary representation>

<multisurface binary representation> ::=
     <byte order> <wkbmultisurface>
        [ <num> <surface binary representation>... ]
   | <multipolygon binary representation>

<multipolygonzm binary representation> ::=
     <byte order> <wkbmultipolygonzm>
        [ <num> <polygonzm binary representation>... ]

<multipolygonz binary representation> ::=
     <byte order> <wkbmultipolygonz>
        [ <num> <polygonz binary representation>... ]

<multipolygonm binary representation> ::=
     <byte order> <wkbmultipolygonm>
        [ <num> <polygonm binary representation>... ]
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
<multipolygon binary representation> ::=
     <byte order> <wkbmultipolygon>
        [ <num> <polygon binary representation>... ]

<geometrycollectionzm binary representation> ::=
     <byte order> <wkbgeometrycollectionzm>
        [ <num> <well-knownzm binary representation>... ]

<geometrycollectionz binary representation> ::=
     <byte order> <wkbgeometrycollectionz>
        [ <num> <well-knownz binary representation>... ]

<geometrycollectionm binary representation> ::=
     <byte order> <wkbgeometrycollectionm>
        [ <num> <well-knownm binary representation>... ]

<geometrycollection binary representation> ::=
     <byte order> <wkbgeometrycollection>
        [ <num> <well-known binary representation>... ]

<wkbpolygonpatchzm binary> ::=
     <polygonzm binary representation>

<wkbpolygonpatchz binary> ::=
     <polygonz binary representation>

<wkbpolygonpatchm binary> ::=
     <polygonm binary representation>

<wkbpolygonpatch binary> ::=
     <polygon binary representation>

<wkbtrianglepatchzm binary> ::=
     <trianglezm binary representation>

<wkbtrianglepatchz binary> ::=
     <trianglez binary representation>

<wkbtrianglepatchm binary> ::=
     <trianglem binary representation>

<wkbtrianglepatch binary> ::=
     <triangle binary representation>

<wkbtinelement binary> ::=
     <tinelement binary representation>

<wkbcurvezm binary> ::=
     <linestringzm binary representation>
   | <circularstringzm binary representation>

<wkbcurvez binary> ::=
     <linestringz binary representation>
   | <circularstringz binary representation>

<wkbcurvem binary> ::=
     <linestringm binary representation>
   | <circularstringm binary representation>

<wkbcurve binary> ::=
     <linestring binary representation>
   | <circularstring binary representation>

<wkbringzm binary> ::=
     <linestringzm binary representation>
   | <circularstringzm binary representation>
   | <compoundcurvezm binary representation>
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
<wkbringz binary> ::=
     <linestringz binary representation>
   | <circularstringz binary representation>
   | <compoundcurvez binary representation>

<wkbringm binary> ::=
     <linestringm binary representation>
   | <circularstringm binary representation>
   | <compoundcurvem binary representation>

<wkbring binary> ::=
     <linestring binary representation>
   | <circularstring binary representation>
   | <compoundcurve binary representation>

<wkbpointzm binary> ::= <wkbx> <wkby> <wkbz> <wkbm>

<wkbpointz binary> ::= <wkbx> <wkby> <wkbz>

<wkbpointm binary> ::= <wkbx> <wkby> <wkbm>

<wkbpoint binary> ::= <wkbx> <wkby>

<wkbx> ::= <double>

<wkby> ::= <double>

<wkbz> ::= <double>

<wkbm> ::= <double>

<num> ::= <uint32>

<nume> ::= <uint32>

<wkbmaxsidelength> ::= <double>

<wkblinearringzm> ::= <num> <wkbpointzm binary>...

<wkblinearringz> ::= <num> <wkbpointz binary>...

<wkblinearringm> ::= <num> <wkbpointm binary>...

<wkblinearring> ::= <num> <wkbpoint binary>...

<wkbpointzm> ::= <uint32>

<wkbpointz> ::= <uint32>

<wkbpointm> ::= <uint32>

<wkbpoint> ::= <uint32>

<wkblinestringzm> ::= <uint32>

<wkblinestringz> ::= <uint32>

<wkblinestringm> ::= <uint32>

<wkblinestring> ::= <uint32>

<wkbcircularstringzm> ::= <uint32>

<wkbcircularstringz> ::= <uint32>

<wkbcircularstringm> ::= <uint32>

<wkbcircularstring> ::= <uint32>

<wkbcompoundcurvezm> ::= <uint32>

<wkbcompoundcurvez> ::= <uint32>

<wkbcompoundcurvem> ::= <uint32>

<wkbcompoundcurve> ::= <uint32>
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
<wkbtrianglezm> ::= <uint32>
<wkbtrianglez> ::= <uint32>
<wkbtrianglem> ::= <uint32>
<wkbtriangle> ::= <uint32>
<wkbpolygonzm> ::= <uint32>
<wkbpolygonz> ::= <uint32>
<wkbpolygonm> ::= <uint32>
<wkbpolygon> ::= <uint32>
<wkbcurvepolygonzm> ::= <uint32>
<wkbcurvepolygonz> ::= <uint32>
<wkbcurvepolygonm> ::= <uint32>
<wkbcurvepolygon> ::= <uint32>
<wkbtinzm> ::= <uint32>
<wkbtinz> ::= <uint32>
<wkbtinm> ::= <uint32>
<wkbtin> ::= <uint32>
<wkbpolyhedralsurfacezm> ::= <uint32>
<wkbpolyhedralsurfacez> ::= <uint32>
<wkbpolyhedralsurfacem> ::= <uint32>
<wkbpolyhedralsurface> ::= <uint32>
<wkbmultipointzm> ::= <uint32>
<wkbmultipointz> ::= <uint32>
<wkbmultipointm> ::= <uint32>
<wkbmultipoint> ::= <uint32>
<wkbmultilinestringzm> ::= <uint32>
<wkbmultilinestringz> ::= <uint32>
<wkbmultilinestringm> ::= <uint32>
<wkbmultilinestring> ::= <uint32>
<wkbmulticurvezm> ::= <uint32>
<wkbmulticurvez> ::= <uint32>
<wkbmulticurvem> ::= <uint32>
<wkbmulticurve> ::= <uint32>
<wkbmultisurfacezm> ::= <uint32>
<wkbmultisurfacez> ::= <uint32>
<wkbmultisurfacem> ::= <uint32>
<wkbmultisurface> ::= <uint32>
<wkbmultipolygonzm> ::= <uint32>
<wkbmultipolygonz> ::= <uint32>
<wkbmultipolygonm> ::= <uint32>
<wkbmultipolygon> ::= <uint32>
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
<wkbgeometrycollectionzm> ::= <uint32>

<wkbgeometrycollectionz> ::= <uint32>

<wkbgeometrycollectionm> ::= <uint32>

<wkbgeometrycollection> ::= <uint32>

<byte order> ::=
     <big endian>
   | <little endian>

<big endian> ::= !! See Description

<little endian> ::= !! See Description

<byte> ::= !! See Description

<uint32> ::= !! See Description

<double> ::= !! See Description
```

a) Case:

   i) If <well-known binary representation> immediately contains a <well-knownzm binary representation>, then <well-known binary representation> produces an ST_Geometry value specified by the immediately contained <well-knownzm binary representation>.

   ii) If <well-known binary representation> immediately contains a <well-knownz binary representation>, then <well-known binary representation> produces an ST_Geometry value specified by the immediately contained <well-knownz binary representation>.

   iii) If <well-known binary representation> immediately contains a <well-knownm binary representation>, then <well-known binary representation> produces an ST_Geometry value specified by the immediately contained <well-knownm binary representation>.

   iv) Otherwise, <well-known binary representation> produces an ST_Geometry value specified by the immediately contained <well-known2d binary representation>.

b) Case:

   i) If <well-knownzm binary representation> immediately contains a <pointzm binary representation>, then <well-knownzm binary representation> produces an *ST_Point* value specified by the immediately contained <pointzm binary representation>.

   ii) If <well-knownzm binary representation> immediately contains a <curvezm binary representation>, then <well-knownzm binary representation> produces an *ST_Curve* value specified by the immediately contained <curvezm binary representation>.

   iii) If <well-knownzm binary representation> immediately contains a <surfacezm binary representation>, then <well-knownzm binary representation> produces an *ST_Surface* value specified by the immediately contained <surfacezm binary representation>.

   iv) Otherwise, <well-knownzm binary representation> produces an *ST_GeomCollection* value specified by the immediately contained <collectionzm binary representation>.

c) Case:

   i) If <well-knownz binary representation> immediately contains a <pointz binary representation>, then <well-knownz binary representation> produces an *ST_Point* value specified by the immediately contained <pointz binary representation>.

   ii) If <well-knownz binary representation> immediately contains a <curvez binary representation>, then <well-knownz binary representation> produces an *ST_Curve* value specified by the immediately contained <curvez binary representation>.

   iii) If <well-knownz binary representation> immediately contains a <surfacez binary representation>, then <well-knownz binary representation> produces an *ST_Surface* value specified by the immediately contained <surfacez binary representation>.

   iv) Otherwise, <well-knownz binary representation> produces an *ST_GeomCollection* value specified by the immediately contained <collectionz binary representation>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

d) Case:

i) If <well-knownm binary representation> immediately contains a <pointm binary representation>, then <well-knownm binary representation> produces an *ST_Point* value specified by the immediately contained <pointm binary representation>.

ii) If <well-knownm binary representation> immediately contains a <curvem binary representation>, then <well-knownm binary representation> produces an *ST_Curve* value specified by the immediately contained <curvem binary representation>.

iii) If <well-knownm binary representation> immediately contains a <surfacem binary representation>, then <well-knownm binary representation> produces an *ST_Surface* value specified by the immediately contained <surfacem binary representation>.

iv) Otherwise, <well-knownm binary representation> produces an *ST_GeomCollection* value specified by the immediately contained <collectionm binary representation>.

e) Case:

i) If <well-known2d binary representation> immediately contains a <point binary representation>, then <well-known2d binary representation> produces an *ST_Point* value specified by the immediately contained <point binary representation>.

ii) If <well-known2d binary representation> immediately contains a <curve binary representation>, then <well-known2d binary representation> produces an *ST_Curve* value specified by the immediately contained <curve binary representation>.

iii) If <well-known2d binary representation> immediately contains a <surface binary representation>, then <well-known2d binary representation> produces an *ST_Surface* value specified by the immediately contained <surface binary representation>.

iv) Otherwise, <well-known2d binary representation> produces an *ST_GeomCollection* value specified by the immediately contained <collection binary representation>.

f) Case:

i) If <pointzm binary representation> immediately contains a <wkbpointzm binary>, then <pointzm binary representation> is the well-known binary representation for an *ST_Point* value that is produced by <wkbpointzm binary>.

ii) Otherwise, <pointzm binary representation> produces an empty set of type ST_Point

g) Case:

i) If <pointz binary representation> immediately contains a <wkbpointz binary>, then <pointz binary representation> is the well-known binary representation for an *ST_Point* value that is produced by <wkbpointz binary>.

ii) Otherwise, <pointz binary representation> produces an empty set of type ST_Point

h) Case:

i) If <pointm binary representation> immediately contains a <wkbpointm binary>, then <pointm binary representation> is the well-known binary representation for an *ST_Point* value that is produced by <wkbpointm binary>.

ii) Otherwise, <pointm binary representation> produces an empty set of type *ST_Point*

i) Case:

i) If <point binary representation> immediately contains a <wkbpoint binary>, then <point binary representation> is the well-known binary representation for an *ST_Point* value that is produced by <wkbpoint binary>.

ii) Otherwise, <point binary representation> produces an empty set of type *ST_Point*

j) Case:

i) If <curvezm binary representation> immediately contains a <linestringzm binary representation>, then <curvezm binary representation> produces an *ST_LineString* value specified by the immediately contained <linestringzm binary representation>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) If <curvezm binary representation> immediately contains a <circularstringzm binary representation>, then <curvezm binary representation> produces an *ST_CircularString* value specified by the immediately contained <circularstringzm binary representation>.

iii) Otherwise, <curvezm binary representation> produces an *ST_CompoundCurve* value specified by the immediately contained <compoundcurvezm binary representation>.

k) Case:

i) If <curvez binary representation> immediately contains a <linestringz binary representation>, then <curvez binary representation> produces an *ST_LineString* value specified by the immediately contained <linestringz binary representation>.

ii) If <curvez binary representation> immediately contains a <circularstringz binary representation>, then <curvez binary representation> produces an *ST_CircularString* value specified by the immediately contained <circularstringz binary representation>.

iii) Otherwise, <curvez binary representation> produces an *ST_CompoundCurve* value specified by the immediately contained <compoundcurvez binary representation>.

l) Case:

i) If <curvem binary representation> immediately contains a <linestringm binary representation>, then <curvem binary representation> produces an *ST_LineString* value specified by the immediately contained <linestringm binary representation>.

ii) If <curvem binary representation> immediately contains a <circularstringm binary representation>, then <curvem binary representation> produces an *ST_CircularString* value specified by the immediately contained <circularstringm binary representation>.

iii) Otherwise, <curvem binary representation> produces an *ST_CompoundCurve* value specified by the immediately contained <compoundcurvem binary representation>.

m) Case:

i) If <curve binary representation> immediately contains a <linestring binary representation>, then <curve binary representation> produces an *ST_LineString* value specified by the immediately contained <linestring binary representation>.

ii) If <curve binary representation> immediately contains a <circularstring binary representation>, then <curve binary representation> produces an *ST_CircularString* value specified by the immediately contained <circularstring binary representation>.

iii) Otherwise, <curve binary representation> produces an *ST_CompoundCurve* value specified by the immediately contained <compoundcurve binary representation>.

n) Case:

i) If <linestringzm binary representation> immediately contains <num>, then <linestringzm binary representation> is the well-known binary representation for an *ST_LineString* value. Let *APA* be an *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <wkbpointzm binary>s. <linestringzm binary representation> produces an *ST_LineString* value as the result of the value expression: *NEW ST_LineString(APA)*.

ii) Otherwise, <linestringzm binary representation> produces an empty set of type *ST_LineString*.

o) Case:

i) If <linestringz binary representation> immediately contains <num>, then <linestringz binary representation> is the well-known binary representation for an *ST_LineString* value. Let *APA* be an *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <wkbpointz binary>s. <linestringz binary representation> produces an *ST_LineString* value as the result of the value expression: *NEW ST_LineString(APA)*.

ii) Otherwise, <linestringz binary representation> produces an empty set of type *ST_LineString*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

p) Case:

i) If <linestringm binary representation> immediately contains <num>, then <linestringm binary representation> is the well-known binary representation for an *ST_LineString* value. Let *APA* be an *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <wkbpointm binary>s. <linestringm binary representation> produces an *ST_LineString* value as the result of the value expression: *NEW ST_LineString(APA)*.

ii) Otherwise, <linestringm binary representation> produces an empty set of type *ST_LineString*.

q) Case:

i) If <linestring binary representation> immediately contains <num>, then <linestring binary representation> is the well-known binary representation for an *ST_LineString* value. Let *APA* be an *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <wkbpoint binary>s. <linestring binary representation> produces an *ST_LineString* value as the result of the value expression: *NEW ST_LineString(APA)*.

ii) Otherwise, <linestring binary representation> produces an empty set of type *ST_LineString*.

r) Case:

i) If <circularstringzm binary representation> immediately contains <num>, then <circularstringzm binary representation> is the well-known binary representation for an *ST_CircularString* value. Let *APA* be an *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <wkbpointzm binary>s. <linestringzm binary representation> produces an *ST_CircularString* value as the result of the value expression: *NEW ST_CircularString(APA)*.

ii) Otherwise, <circularstringzm binary representation> produces an empty set of type *ST_CircularString*.

s) Case:

i) If <circularstringz binary representation> immediately contains <num>, then <circularstringz binary representation> is the well-known binary representation for an *ST_CircularString* value. Let *APA* be an *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <wkbpointz binary>s. <linestringz binary representation> produces an *ST_CircularString* value as the result of the value expression: *NEW ST_CircularString(APA)*.

ii) Otherwise, <circularstringz binary representation> produces an empty set of type *ST_CircularString*.

t) Case:

i) If <circularstringm binary representation> immediately contains <num>, then <circularstringm binary representation> is the well-known binary representation for an *ST_CircularString* value. Let *APA* be an *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <wkbpointm binary>s. <linestringm binary representation> produces an *ST_CircularString* value as the result of the value expression: *NEW ST_CircularString(APA)*.

ii) Otherwise, <circularstringm binary representation> produces an empty set of type *ST_CircularString*.

u) Case:

i) If <circularstring binary representation> immediately contains <num>, then <circularstring binary representation> is the well-known binary representation for an *ST_CircularString* value. Let *APA* be an *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <wkbpoint binary>s. <linestring binary representation> produces an *ST_CircularString* value as the result of the value expression: *NEW ST_CircularString(APA)*.

ii) Otherwise, <circularstring binary representation> produces an empty set of type *ST_CircularString*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

v) Case:

i) If <compoundcurvezm binary representation> immediately contains <num>, then <compoundcurvezm binary representation> is the well-known binary representation for an *ST_CompoundCurve* value. Let *ACA* be an *ST_Curve* ARRAY value with cardinality of <num> that contains the *ST_Curve* values specified by the immediately contained <wkbcurvezm binary>s. <compoundcurvezm binary representation> produces an *ST_CompoundCurve* value as the result of the value expression: *NEW ST_CompoundCurve(ACA)*.

ii) Otherwise, <compoundcurvezm binary representation> produces an empty set of type *ST_CompoundCurve*.

w) Case:

i) If <compoundcurvez binary representation> immediately contains <num>, then <compoundcurvez binary representation> is the well-known binary representation for an *ST_CompoundCurve* value. Let *ACA* be an *ST_Curve* ARRAY value with cardinality of <num> that contains the *ST_Curve* values specified by the immediately contained <wkbcurvez binary>s. <compoundcurvez binary representation> produces an *ST_CompoundCurve* value as the result of the value expression: *NEW ST_CompoundCurve(ACA)*.

ii) Otherwise, <compoundcurvez binary representation> produces an empty set of type *ST_CompoundCurve*.

x) Case:

i) If <compoundcurvem binary representation> immediately contains <num>, then <compoundcurvem binary representation> is the well-known binary representation for an *ST_CompoundCurve* value. Let *ACA* be an *ST_Curve* ARRAY value with cardinality of <num> that contains the *ST_Curve* values specified by the immediately contained <wkbcurvem binary>s. <compoundcurvem binary representation> produces an *ST_CompoundCurve* value as the result of the value expression: *NEW ST_CompoundCurve(ACA)*.

ii) Otherwise, <compoundcurvem binary representation> produces an empty set of type *ST_CompoundCurve*.

y) Case:

i) If <compoundcurve binary representation> immediately contains <num>, then <compoundcurve binary representation> is the well-known binary representation for an *ST_CompoundCurve* value. Let *ACA* be an *ST_Curve* ARRAY value with cardinality of <num> that contains the *ST_Curve* values specified by the immediately contained <wkbcurve binary>s. <compoundcurve binary representation> produces an *ST_CompoundCurve* value as the result of the value expression: *NEW ST_CompoundCurve(ACA)*.

ii) Otherwise, <compoundcurve binary representation> produces an empty set of type *ST_CompoundCurve*.

z) Case:

i) If <surfacezm binary representation> immediately contains a <curvepolygonzm binary representation>, then <surfacezm binary representation> produces an *ST_CurvePolygon* value specified by the immediately contained <curvepolygonzm binary representation>.

ii) Otherwise, <surfacezm binary representation> produces an *ST_PolyhdrlSurface* value specified by the immediately contained <polyhedralsurfacezm binary representation>.

aa) Case:

i) If <surfacez binary representation> immediately contains a <curvepolygonz binary representation>, then <surfacez binary representation> produces an *ST_CurvePolygon* value specified by the immediately contained <curvepolygonz binary representation>.

ii) Otherwise, <surfacez binary representation> produces an *ST_PolyhdrlSurface* value specified by the immediately contained <polyhedralsurfacez binary representation>.

ab) Case:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

i) If <surfacem binary representation> immediately contains a <curvepolygonm binary representation>, then <surfacem binary representation> produces an *ST_CurvePolygon* value specified by the immediately contained <curvepolygonm binary representation>.

ii) Otherwise, <surfacem binary representation> produces an *ST_PolyhdrlSurface* value specified by the immediately contained <polyhedralsurfacem binary representation>.

ac) Case:

i) If <surface binary representation> immediately contains a <curvepolygon binary representation>, then <surface binary representation> produces an *ST_CurvePolygon* value specified by the immediately contained <curvepolygonm binary representation>.

ii) Otherwise, <surface binary representation> produces an *ST_PolyhdrlSurface* value specified by the immediately contained <polyhedralsurface binary representation>.

ad) Case:

i) If <curvepolygonzm binary representation> immediately contains a <num>, then <curvepolygonzm binary representation> produces an *ST_CurvePolygon*. Let *ACA* be an *ST_Curve* ARRAY value with cardinality of <num> that contains the *ST_Curve* values specified by the immediately contained <wkbringzm binary>s.

Case:

1) If the cardinality of *ACA* is 0 (zero), then <curvepolygonzm binary representation> produces an empty set of type *ST_CurvePolygon*.

2) If the cardinality of *ACA* is 1 (one), then let *AER* be the element of *ACA*. <curvepolygonzm binary representation> produces an *ST_CurvePolygon* value as the result of the value expression: *NEW ST_CurvePolygon(AER)*.

3) Otherwise, let *AER* be the first element in *ACA* and let *AIR* be the sublist of *ACA* containing the other elements of *ACA*. <curvepolygonzm binary representation> produces an *ST_CurvePolygon* value as the result of the value expression: *NEW ST_CurvePolygon(AER, AIR)*.

ii) If <curvepolygonzm binary representation> immediately contains a <polygonzm binary representation>, then <curvepolygonzm binary representation> produces an *ST_Polygon* value specified by the immediately contained <polygonzm binary representation>.

iii) Otherwise, <curvepolygonzm binary representation> produces an empty set of type *ST_CurvePolygon*.

ae) Case:

i) If <curvepolygonz binary representation> immediately contains a <num>, then <curvepolygonz binary representation> produces an *ST_CurvePolygon*. Let *ACA* be an *ST_Curve* ARRAY value with cardinality of <num> that contains the *ST_Curve* values specified by the immediately contained <wkbringz binary>s.

Case:

1) If the cardinality of *ACA* is 0 (zero), then <curvepolygonz binary representation> produces an empty set of type *ST_CurvePolygon*.

2) If the cardinality of *ACA* is 1 (one), then let *AER* be the element of *ACA*. <curvepolygonz binary representation> produces an *ST_CurvePolygon* value as the result of the value expression: *NEW ST_CurvePolygon(AER)*.

3) Otherwise, let *AER* be the first element in *ACA* and let *AIR* be the sublist of *ACA* containing the other elements of *ACA*. <curvepolygonz binary representation> produces an *ST_CurvePolygon* value as the result of the value expression: *NEW ST_CurvePolygon(AER, AIR)*.

ii) If <curvepolygonz binary representation> immediately contains a <polygonz binary representation>, then <curvepolygonz binary representation> produces an *ST_Polygon* value specified by the immediately contained <polygonz binary representation>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

iii) Otherwise, <curvepolygonz binary representation> produces an empty set of type *ST_CurvePolygon*.

af) Case:

i) If <curvepolygonm binary representation> immediately contains a <num>, then <curvepolygonm binary representation> produces an *ST_CurvePolygon*. Let *ACA* be an *ST_Curve* ARRAY value with cardinality of <num> that contains the *ST_Curve* values specified by the immediately contained <wkbringm binary>s.

Case:

1) If the cardinality of *ACA* is 0 (zero), then <curvepolygonm binary representation> produces an empty set of type *ST_CurvePolygon*.

2) If the cardinality of *ACA* is 1 (one), then let *AER* be the element of *ACA*. <curvepolygonm binary representation> produces an *ST_CurvePolygon* value as the result of the value expression: *NEW ST_CurvePolygon(AER)*.

3) Otherwise, let *AER* be the first element in *ACA* and let *AIR* be the sublist of *ACA* containing the other elements of *ACA*. <curvepolygonm binary representation> produces an *ST_CurvePolygon* value as the result of the value expression: *NEW ST_CurvePolygon(AER, AIR)*.

ii) If <curvepolygonm binary representation> immediately contains a <polygonm binary representation>, then <curvepolygonm binary representation> produces an *ST_Polygon* value specified by the immediately contained <polygonm binary representation>.

iii) Otherwise, <curvepolygonm binary representation> produces an empty set of type *ST_CurvePolygon*.

ag) Case:

i) If <curvepolygon binary representation> immediately contains a <num>, then <curvepolygon binary representation> produces an *ST_CurvePolygon*. Let *ACA* be an *ST_Curve* ARRAY value with cardinality of <num> that contains the *ST_Curve* values specified by the immediately contained <wkbring binary>s.

Case:

1) If the cardinality of *ACA* is 0 (zero), then <curvepolygon binary representation> produces an empty set of type *ST_CurvePolygon*.

2) If the cardinality of *ACA* is 1 (one), then let *AER* be the element of *ACA*. <curvepolygon binary representation> produces an *ST_CurvePolygon* value as the result of the value expression: *NEW ST_CurvePolygon(AER)*.

3) Otherwise, let *AER* be the first element in *ACA* and let *AIR* be the sublist of *ACA* containing the other elements of *ACA*. <curvepolygon binary representation> produces an *ST_CurvePolygon* value as the result of the value expression: *NEW ST_CurvePolygon(AER, AIR)*.

ii) If <curvepolygon binary representation> immediately contains a <polygon binary representation>, then <curvepolygon binary representation> produces an *ST_Polygon* value specified by the immediately contained <polygon binary representation>.

iii) Otherwise, <curvepolygon binary representation> produces an empty set of type *ST_CurvePolygon*.

ah) Case:

i) If <polygonzm binary representation> immediately contains <num>, then <polygonzm binary representation> is the well-known binary representation for an *ST_Polygon* value. Let *ALSA* be an *ST_LineString* ARRAY value with cardinality of <num> that contains the *ST_LineString* values specified by the immediately contained <wkblinearringzm binary>s.

Case:

1) If the cardinality of *ALSA* is 0 (zero), then <polygonzm binary representation> produces an empty set of type *ST_Polygon*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) If the cardinality of *ALSA* is 1 (one), then let *ALS* be the element of *ALSA*. <polygonzm binary representation> produces an *ST_Polygon* value as the result of the value expression: *NEW ST_Polygon(ALS)*.

3) Otherwise, let *AER* be the first element in *ALSA* and let *AIR* be the sublist of *ALSA* containing the other elements of *ALSA*. <polygonzm binary representation> produces an *ST_Polygon* value as the result of the value expression: *NEW ST_Polygon(AER, AIR)*.

ii) If <polygonzm binary representation> immediately contains a <trianglezm binary representation>, then <polygonzm binary representation> produces an *ST_Triangle* value specified by the immediately contained <trianglezm binary representation>.

iii) Otherwise, <polygonzm binary representation> produces an empty set of type *ST_Polygon*.

ai) Case:

i) If <polygonz binary representation> immediately contains <num>, then <polygonz binary representation> is the well-known binary representation for an *ST_Polygon* value. Let *ALSA* be an *ST_LineString* ARRAY value with cardinality of <num> that contains the *ST_LineString* values specified by the immediately contained <wkblinearringz binary>s.

Case:

1) If the cardinality of *ALSA* is 0 (zero), then <polygonz binary representation> produces an empty set of type *ST_Polygon*.

2) If the cardinality of *ALSA* is 1 (one), then let *ALS* be the element of *ALSA*. <polygonz binary representation> produces an *ST_Polygon* value as the result of the value expression: *NEW ST_Polygon(ALS)*.

3) Otherwise, let *AER* be the first element in *ALSA* and let *AIR* be the sublist of *ALSA* containing the other elements of *ALSA*. <polygonz binary representation> produces an *ST_Polygon* value as the result of the value expression: *NEW ST_Polygon(AER, AIR)*.

ii) If <polygonz binary representation> immediately contains a <trianglez binary representation>, then <polygonz binary representation> produces an *ST_Triangle* value specified by the immediately contained <trianglez binary representation>.

iii) Otherwise, <polygonz binary representation> produces an empty set of type *ST_Polygon*.

aj) Case:

i) If <polygonm binary representation> immediately contains <num>, then <polygonm binary representation> is the well-known binary representation for an *ST_Polygon* value. Let *ALSA* be an *ST_LineString* ARRAY value with cardinality of <num> that contains the *ST_LineString* values specified by the immediately contained <wkblinearringm binary>s.

Case:

1) If the cardinality of *ALSA* is 0 (zero), then <polygonm binary representation> produces an empty set of type *ST_Polygon*.

2) If the cardinality of *ALSA* is 1 (one), then let *ALS* be the element of *ALSA*. <polygonm binary representation> produces an *ST_Polygon* value as the result of the value expression: *NEW ST_Polygon(ALS)*.

3) Otherwise, let *AER* be the first element in *ALSA* and let *AIR* be the sublist of *ALSA* containing the other elements of *ALSA*. <polygonm binary representation> produces an *ST_Polygon* value as the result of the value expression: *NEW ST_Polygon(AER, AIR)*.

ii) If <polygonm binary representation> immediately contains a <trianglem binary representation>, then <polygonm binary representation> produces an *ST_Triangle* value specified by the immediately contained <trianglem binary representation>.

iii) Otherwise, <polygonm binary representation> produces an empty set of type *ST_Polygon*.

ak) Case:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

i) If <polygon binary representation> immediately contains <num>, then <polygon binary representation> is the well-known binary representation for an *ST_Polygon* value. Let *ALSA* be an *ST_LineString* ARRAY value with cardinality of <num> that contains the *ST_LineString* values specified by the immediately contained <wkblinearring binary>s.

Case:

1) If the cardinality of *ALSA* is 0 (zero), then <polygon binary representation> produces an empty set of type *ST_Polygon*.

2) If the cardinality of *ALSA* is 1 (one), then let *ALS* be the element of *ALSA*. <polygon binary representation> produces an *ST_Polygon* value as the result of the value expression: *NEW ST_Polygon(ALS)*.

3) Otherwise, let *AER* be the first element in *ALSA* and let *AIR* be the sublist of *ALSA* containing the other elements of *ALSA*. <polygon binary representation> produces an *ST_Polygon* value as the result of the value expression: *NEW ST_Polygon*(*AER, AIR*).

ii) If <polygon binary representation> immediately contains a <triangle binary representation>, then <polygon binary representation> produces an *ST_Triangle* value specified by the immediately contained <triangle binary representation>.

iii) Otherwise, <polygon binary representation> produces an empty set of type *ST_Polygon*.

al) Case:

i) If <trianglezm binary representation> immediately contains <wkbpointzm binary>, then <trianglezm binary representation> is the well-known binary representation for an *ST_Triangle* value. Let *APA* be an *ST_Point* ARRAY value with cardinality of three that contains the *ST_Point* values specified by the immediately contained <wkbpointzm binary>s. Then <trianglezm binary representation>> produces an *ST_Triangle* value as the result of the value expression*: NEW ST_Triangle(APA)*.

ii) Otherwise, <trianglezm binary representation> produces an empty set of type *ST_Triangle*.

am) Case:

i) If <trianglez binary representation> immediately contains <wkbpointz binary>, then <trianglez binary representation> is the well-known binary representation for an *ST_Triangle* value. Let *APA* be an *ST_Point* ARRAY value with cardinality of three that contains the *ST_Point* values specified by the immediately contained <wkbpointz binary>s. Then <trianglez binary representation>> produces an *ST_Triangle* value as the result of the value expression: *NEW ST_Triangle(APA)*.

ii) Otherwise, <trianglez binary representation> produces an empty set of type *ST_Triangle*.

an) Case:

i) If <trianglem binary representation> immediately contains <wkbpointm binary>, then <trianglem binary representation> is the well-known binary representation for an *ST_Triangle* value. Let *APA* be an *ST_Point* ARRAY value with cardinality of three that contains the *ST_Point* values specified by the immediately contained <wkbpointm binary>s. Then <trianglem binary representation>> produces an *ST_Triangle* value as the result of the value expression: *NEW ST_Triangle(APA)*.

ii) Otherwise, <trianglem binary representation> produces an empty set of type *ST_Triangle*.

ao) Case:

i) If <triangle binary representation> immediately contains three <wkbpoint binary>s, then <triangle binary representation> is the well-known binary representation for an *ST_Triangle* value. Let *APA* be an *ST_Point* ARRAY value with cardinality of three that contains the *ST_Point* values specified by the three immediately contained <wkbpoint binary>s. Then <triangle binary representation>> produces an *ST_Triangle* value as the result of the value expression: *NEW ST_Triangle(APA)*).

ii) Otherwise, <triangle binary representation> produces an empty set of type *ST_Triangle*.

ap) Case:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

i) If <polyhedralsurfacezm binary representation> immediately contains <num>, then <polyhedralsurfacezm binary representation> is the well-known binary representation for an ST_*PolyhdrlSurface* value. Let *APA* be an *ST_Polygon* ARRAY value with cardinality of <num> that contains the *ST_Polygon* values specified by the immediately contained <wkbpolygonpatchzm binary>s.

Case:

1) If the cardinality of *APA* is 0 (zero), then <polyhedralsurfacezm binary representation> produces an empty set of type ST_*PolyhdrlSurface*.

2) Otherwise, <polyhedralsurfacezm binary representation> produces an ST_*PolyhdrlSurface* value as the result of the value expression: *NEW ST_PolyhdrlSurface(APA)*.

ii) If <polyhedralsurfacezm binary representation> immediately contains a <tinzm binary representation>, then <polyhedralsurfacezm binary representation> produces an *ST_TIN* value specified by the immediately contained <tinzm binary representation>.

iii) Otherwise, <polyhedralsurfacezm binary representation> produces an empty set of type ST_*PolyhdrlSurface*.

aq) Case:

i) If <polyhedralsurfacez binary representation> immediately contains <num>, then <polyhedralsurfacez binary representation> is the well-known binary representation for an ST_*PolyhdrlSurface* value. Let *APA* be an *ST_Polygon* ARRAY value with cardinality of <num> that contains the *ST_Polygon* values specified by the immediately contained <wkbpolygonpatchz binary>s.

Case:

1) If the cardinality of *APA* is 0 (zero), then <polyhedralsurfacez binary representation> produces an empty set of type ST_*PolyhdrlSurface*.

2) Otherwise, <polyhedralsurfacez binary representation> produces an ST_*PolyhdrlSurface* value as the result of the value expression: *NEW ST_PolyhdrlSurface(APA)*.

ii) If <polyhedralsurfacez binary representation> immediately contains a <tinz binary representation>, then <polyhedralsurfacez binary representation> produces an *ST_TIN* value specified by the immediately contained <tinz binary representation>.

iii) Otherwise, <polyhedralsurfacez binary representation> produces an empty set of type ST_*PolyhdrlSurface*.

ar) Case:

i) If <polyhedralsurfacem binary representation> immediately contains <num>, then <polyhedralsurfacem binary representation> is the well-known binary representation for an ST_*PolyhdrlSurface* value. Let *APA* be an *ST_Polygon* ARRAY value with cardinality of <num> that contains the *ST_Polygon* values specified by the immediately contained <wkbpolygonpatchm binary>s.

Case:

1) If the cardinality of *APA* is 0 (zero), then <polyhedralsurfacem binary representation> produces an empty set of type ST_*PolyhdrlSurface*.

2) Otherwise, <polyhedralsurfacem binary representation> produces an ST_*PolyhdrlSurface* value as the result of the value expression: *NEW ST_PolyhdrlSurface(APA)*.

ii) If <polyhedralsurfacem binary representation> immediately contains a <tinm binary representation>, then <polyhedralsurfacem binary representation> produces an *ST_TIN* value specified by the immediately contained <tinm binary representation>.

iii) Otherwise, <polyhedralsurfacem binary representation> produces an empty set of type ST_*PolyhdrlSurface*.

as) Case:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

i) If <polyhedralsurface binary representation> immediately contains <num>, then <polyhedralsurface binary representation> is the well-known binary representation for an ST_*PolyhdrlSurface* value. Let *APA* be an *ST_Polygon* ARRAY value with cardinality of <num> that contains the *ST_Polygon* values specified by the immediately contained <wkbpolygonpatch binary>s.

Case:

1) If the cardinality of *APA* is 0 (zero), then <polyhedralsurface binary representation> produces an empty set of type ST_*PolyhdrlSurface*.

2) Otherwise, <polyhedralsurface binary representation> produces an ST_*PolyhdrlSurface* value as the result of the value expression: *NEW ST_PolyhdrlSurface(APA)*.

ii) If <polyhedralsurface binary representation> immediately contains a <tin binary representation>, then <polyhedralsurface binary representation> produces an *ST_TIN* value specified by the immediately contained <tin binary representation>.

iii) Otherwise, <polyhedralsurface binary representation> produces an empty set of type ST_*PolyhdrlSurface*.

at) Case:

i) If <tinzm binary representation> immediately contains <num>, then <tinzm binary representation> is the well-known binary representation for an *ST_TIN* value. Let *ATA* be an *ST_Triangle* ARRAY value with cardinality of <num> that contains the *ST_Triangle* values specified by the immediately contained <wkbtrianglepatchzm binary>s. Let *AEA* be an *ST_TINElement* ARRAY value with cardinality of <nume> that contains the *ST_TINElement* values specified by the immediately contained <wkbtinelement binary>s. Let *M* be a DOUBLE PRECISION value specified by the immediately contained <wkbmaxsidelength>.

Case:

1) If the cardinality of *ATA* is 0 (zero), then <tinzm binary representation> produces an empty set of type *ST_TIN*.

2) Otherwise, <tinzm binary representation> produces an *ST_TIN* value as the result of the value expression: *NEW ST_TIN(APA, AEA, M)*.

ii) Otherwise, <tinzm binary representation> produces an empty set of type *ST_TIN*.

au) Case:

i) If <tinz binary representation> immediately contains <num>, then <tinz binary representation> is the well-known binary representation for an *ST_TIN* value. Let *ATA* be an *ST_Triangle* ARRAY value with cardinality of <num> that contains the *ST_Triangle* values specified by the immediately contained <wkbtrianglepatchz binary>s. Let *AEA* be an *ST_TINElement* ARRAY value with cardinality of <nume> that contains the *ST_TINElement* values specified by the immediately contained <wkbtinelement binary>s. Let *M* be a DOUBLE PRECISION value specified by the immediately contained <wkbmaxsidelength>.

Case:

1) If the cardinality of *ATA* is 0 (zero), then <tinz binary representation> produces an empty set of type *ST_TIN*.

2) Otherwise, <tinz binary representation> produces an *ST_TIN* value as the result of the value expression: *NEW ST_TIN(APA, AEA, M)*.

ii) Otherwise, <tinz binary representation> produces an empty set of type *ST_TIN*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

av) Case:

i) If <tinm binary representation> immediately contains <num>, then <tinm binary representation> is the well-known binary representation for an *ST_TIN* value. Let *ATA* be an *ST_Triangle* ARRAY value with cardinality of <num> that contains the *ST_Triangle* values specified by the immediately contained <wkbtrianglepatchm binary>s. Let *AEA* be an *ST_TINElement* ARRAY value with cardinality of <nume> that contains the *ST_TINElement* values specified by the immediately contained <wkbtinelement binary>s. Let *M* be a DOUBLE PRECISION value specified by the immediately contained <wkbmaxsidelength>.

Case:

1) If the cardinality of *ATA* is 0 (zero), then <tinm binary representation> produces an empty set of type *ST_TIN*.

2) Otherwise, <tinm binary representation> produces an *ST_TIN* value as the result of the value expression: *NEW ST_TIN(APA, AEA, M)*.

ii) Otherwise, <tinm binary representation> produces an empty set of type *ST_TIN*.

aw) Case:

i) If <tin binary representation> immediately contains <num>, then <tin binary representation> is the well-known binary representation for an *ST_TIN* value. Let *ATA* be an *ST_Triangle* ARRAY value with cardinality of <num> that contains the *ST_Triangle* values specified by the immediately contained <wkbtrianglepatch binary>s. Let *AEA* be an *ST_TINElement* ARRAY value with cardinality of <nume> that contains the *ST_TINElement* values specified by the immediately contained <wkbtinelement binary>s. Let *M* be a DOUBLE PRECISION value specified by the immediately contained <wkbmaxsidelength>.

Case:

1) If the cardinality of *ATA* is 0 (zero), then <tin binary representation> produces an empty set of type *ST_TIN*.

2) Otherwise, <tin binary representation> produces an *ST_TIN* value as the result of the value expression: *NEW ST_TIN(APA, AEA, M)*.

ii) Otherwise, <tin binary representation> produces an empty set of type ST_TIN.

ax) Let *TET* be the value of <tinelement element type> in <tinelement binary representation>, let *TEID* be the value of <tinelement element id>, let *TETAG* be the value of <tinelement element tag> and let *TEGEOM* be the value of <well-known binary representation>. <tinelement binary representation> produces an *ST_TINElement* value as the result of the value expression: *NEW ST_TINElement(TET, TEID, TETAG, TEGEOM)*.

ay) <tinelement element type> produces a CHARACTER VARYING(30) value of length <byte> containing the characters in <letters>.

az) <tinelement element id> produces an INTEGER value of <signed integer>.

ba) <tinelement element tag> produces a CHARACTER VARYING(64) value of length <byte> containing the characters in <letters>.

bb) Case:

i) If <collectionzm binary representation> immediately contains a <multipointzm binary representation>, then <collectionzm binary representation> produces an *ST_MultiPoint* value specified by the immediately contained <multipointzm binary representation>.

ii) If <collectionzm binary representation> immediately contains a <multicurvezm binary representation>, then <collectionzm binary representation> produces an *ST_MultiCurve* value specified by the immediately contained <multicurvezm binary representation>.

iii) If <collectionzm binary representation> immediately contains a <multisurfacezm binary representation>, then <collectionzm binary representation> produces an *ST_MultiSurface* value specified by the immediately contained <multisurfacezm binary representation>.

iv) Otherwise, <collectionzm binary representation> produces an *ST_GeomCollection* value specified by the immediately contained <geometrycollection binary representation>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

bc) Case:

i) If <collectionz binary representation> immediately contains a <multipointz binary representation>, then <collectionz binary representation> produces an *ST_MultiPoint* value specified by the immediately contained <multipointz binary representation>.

ii) If <collectionz binary representation> immediately contains a <multicurvez binary representation>, then <collectionz binary representation> produces an *ST_MultiCurve* value specified by the immediately contained <multicurvez binary representation>.

iii) If <collectionz binary representation> immediately contains a <multisurfacez binary representation>, then <collectionz binary representation> produces an *ST_MultiSurface* value specified by the immediately contained <multisurfacez binary representation>.

iv) Otherwise, <collectionz binary representation> produces an *ST_GeomCollection* value specified by the immediately contained <geometrycollection binary representation>.

bd) Case:

i) If <collectionm binary representation> immediately contains a <multipointm binary representation>, then <collectionm binary representation> produces an *ST_MultiPoint* value specified by the immediately contained <multipointm binary representation>.

ii) If <collectionm binary representation> immediately contains a <multicurvem binary representation>, then <collectionm binary representation> produces an *ST_MultiCurve* value specified by the immediately contained <multicurvem binary representation>.

iii) If <collectionm binary representation> immediately contains a <multisurfacem binary representation>, then <collectionm binary representation> produces an *ST_MultiSurface* value specified by the immediately contained <multisurfacem binary representation>.

iv) Otherwise, <collectionm binary representation> produces an *ST_GeomCollection* value specified by the immediately contained <geometrycollection binary representation>.

be) Case:

i) If <collection binary representation> immediately contains a <multipoint binary representation>, then <collection binary representation> produces an *ST_MultiPoint* value specified by the immediately contained <multipoint binary representation>.

ii) If <collection binary representation> immediately contains a <multicurve binary representation>, then <collection binary representation> produces an *ST_MultiCurve* value specified by the immediately contained <multicurve binary representation>.

iii) If <collection binary representation> immediately contains a <multisurface binary representation>, then <collection binary representation> produces an *ST_MultiSurface* value specified by the immediately contained <multisurface binary representation>.

iv) Otherwise, <collection binary representation> produces an *ST_GeomCollection* value specified by the immediately contained <geometrycollection binary representation>.

bf) Case:

i) If <multipointzm binary representation> immediately contains <num>, then <multipointzm binary representation> is the well-known binary representation for an *ST_MultiPoint* value. Let *APA* be the *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <pointzm binary representation>s. <multipointzm binary representation> produces an *ST_MultiPoint* value as the result of the value expression: *NEW ST_MultiPoint(APA)*.

ii) Otherwise, <multipointzm binary representation> produces an empty set of type *ST_MultiPoint*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

bg) Case:

i) If <multipointz binary representation> immediately contains <num>, then <multipointz binary representation> is the well-known binary representation for an *ST_MultiPoint* value. Let *APA* be the *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <pointz binary representation>s. <multipointz binary representation> produces an *ST_MultiPoint* value as the result of the value expression: *NEW ST_MultiPoint(APA)*.

ii) Otherwise, <multipointz binary representation> produces an empty set of type *ST_MultiPoint*.

bh) Case:

i) If <multipointm binary representation> immediately contains <num>, then <multipointm binary representation> is the well-known binary representation for an *ST_MultiPoint* value. Let *APA* be the *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <pointm binary representation>s. <multipointm binary representation> produces an *ST_MultiPoint* value as the result of the value expression: *NEW ST_MultiPoint(APA)*.

ii) Otherwise, <multipointm binary representation> produces an empty set of type *ST_MultiPoint*.

bi) Case:

i) If <multipoint binary representation> immediately contains <num>, then <multipoint binary representation> is the well-known binary representation for an *ST_MultiPoint* value. Let *APA* be the *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <point binary representation>s. <multipoint binary representation> produces an *ST_MultiPoint* value as the result of the value expression: *NEW ST_MultiPoint(APA)*.

ii) Otherwise, <multipoint binary representation> produces an empty set of type *ST_MultiPoint*.

bj) Case:

i) If <multicurvezm binary representation> immediately contains a <num>, then <multicurvezm binary representation> produces an *ST_MultiCurve* value. Let *ACA* be the *ST_Curve* ARRAY value with cardinality of <num> that contains the *ST_Curve* values specified by the immediately contained <curvezm binary representation>s. <multicurvezm binary representation> produces an *ST_MultiCurve* value as the result of the value expression: *NEW ST_MultiCurve(ACA)*.

ii) If <multicurvezm binary representation> immediately contains a <multilinestringzm binary representation>, then <multicurvezm binary representation> produces an *ST_MultiLineString* value specified by the immediately contained <multilinestringzm binary representation>.

iii) Otherwise, <multicurvezm binary representation> produces an empty set of type *ST_MultiCurve*.

bk) Case:

i) If <multicurvez binary representation> immediately contains a <num>, then <multicurvez binary representation> produces an *ST_MultiCurve* value. Let *ACA* be the *ST_Curve* ARRAY value with cardinality of <num> that contains the *ST_Curve* values specified by the immediately contained <curvez binary representation>s. <multicurvez binary representation> produces an *ST_MultiCurve* value as the result of the value expression: *NEW ST_MultiCurve(ACA)*.

ii) If <multicurvez binary representation> immediately contains a <multilinestringz binary representation>, then <multicurvez binary representation> produces an *ST_MultiLineString* value specified by the immediately contained <multilinestringz binary representation>.

iii) Otherwise, <multicurvez binary representation> produces an empty set of type *ST_MultiCurve*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

bl) Case:

i) If <multicurvem binary representation> immediately contains a <num>, then <multicurvem binary representation> produces an *ST_MultiCurve* value. Let *ACA* be the *ST_Curve* ARRAY value with cardinality of <num> that contains the *ST_Curve* values specified by the immediately contained <curvem binary representation>s. <multicurvem binary representation> produces an *ST_MultiCurve* value as the result of the value expression: *NEW ST_MultiCurve(ACA)*.

ii) If <multicurvem binary representation> immediately contains a <multilinestringm binary representation>, then <multicurvem binary representation> produces an *ST_MultiLineString* value specified by the immediately contained <multilinestringm binary representation>.

iii) Otherwise, <multicurvem binary representation> produces an empty set of type *ST_MultiCurve*.

bm) Case:

i) If <multicurve binary representation> immediately contains a <num>, then <multicurve binary representation> produces an *ST_MultiCurve* value. Let *ACA* be the *ST_Curve* ARRAY value with cardinality of <num> that contains the *ST_Curve* values specified by the immediately contained <curve binary representation>s. <multicurve binary representation> produces an *ST_MultiCurve* value as the result of the value expression: *NEW ST_MultiCurve(ACA)*.

ii) If <multicurve binary representation> immediately contains a <multilinestring binary representation>, then <multicurve binary representation> produces an *ST_MultiLineString* value specified by the immediately contained <multilinestring binary representation>.

iii) Otherwise, <multicurve binary representation> produces an empty set of type *ST_MultiCurve*.

bn) Case:

i) If <multilinestringzm binary representation> immediately contains <num>, then <multilinestringzm binary representation> is the well-known binary representation for an *ST_MultiLineString* value. Let *ALSA* be the *ST_LineString* ARRAY value with cardinality of <num> that contains the *ST_LineString* values specified by the immediately contained <linestringzm binary representation>s. <multilinestringzm binary representation> produces an *ST_MultiLineString* value as the result of the value expression: *NEW ST_MultiLineString(ALSA)*.

ii) Otherwise, <multilinestringzm binary representation> produces an empty set of type *ST_MultiLineString*.

bo) Case:

i) If <multilinestringz binary representation> immediately contains <num>, then <multilinestringz binary representation> is the well-known binary representation for an *ST_MultiLineString* value. Let *ALSA* be the *ST_LineString* ARRAY value with cardinality of <num> that contains the *ST_LineString* values specified by the immediately contained <linestringz binary representation>s. <multilinestringz binary representation> produces an *ST_MultiLineString* value as the result of the value expression: *NEW ST_MultiLineString(ALSA)*.

ii) Otherwise, <multilinestringz binary representation> produces an empty set of type *ST_MultiLineString*.

bp) Case:

i) If <multilinestringm binary representation> immediately contains <num>, then <multilinestringm binary representation> is the well-known binary representation for an *ST_MultiLineString* value. Let *ALSA* be the *ST_LineString* ARRAY value with cardinality of <num> that contains the *ST_LineString* values specified by the immediately contained <linestringm binary representation>s. <multilinestringm binary representation> produces an *ST_MultiLineString* value as the result of the value expression: *NEW ST_MultiLineString(ALSA)*.

ii) Otherwise, <multilinestringm binary representation> produces an empty set of type *ST_MultiLineString*.

bq) Case:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

i) If <multilinestring binary representation> immediately contains <num>, then <multilinestring binary representation> is the well-known binary representation for an *ST_MultiLineString* value. Let *ALSA* be the *ST_LineString* ARRAY value with cardinality of <num> that contains the *ST_LineString* values specified by the immediately contained <linestring binary representation>s. <multilinestring binary representation> produces an *ST_MultiLineString* value as the result of the value expression: *NEW ST_MultiLineString(ALSA)*.

ii) Otherwise, <multilinestring binary representation> produces an empty set of type *ST_MultiLineString*.

br) Case:

i) If <multisurfacezm binary representation> immediately contains a <num>, then <multisurfacezm binary representation> produces an *ST_MultiSurface* value. Let *ASA* be the *ST_Surface* ARRAY value with cardinality of <num> that contains the *ST_Surface* values specified by the immediately contained <surfacezm binary representation>s. <multisurfacezm binary representation> produces an *ST_MultiSurface* value as the result of the value expression: *NEW ST_MultiSurface(ASA)*.

ii) If <multisurfacezm binary representation> immediately contains a <multipolygonzm binary representation>, then <multisurfacezm binary representation> produces an *ST_MultiPolygon* value specified by the immediately contained <multipolygonzm binary representation>.

iii) Otherwise, <multisurfacezm binary representation> produces an empty set of type *ST_MultiSurface*.

bs) Case:

i) If <multisurfacez binary representation> immediately contains a <num>, then <multisurfacez binary representation> produces an *ST_MultiSurface* value. Let *ASA* be the *ST_Surface* ARRAY value with cardinality of <num> that contains the *ST_Surface* values specified by the immediately contained <surfacez binary representation>s. <multisurfacez binary representation> produces an *ST_MultiSurface* value as the result of the value expression: *NEW ST_MultiSurface(ASA)*.

ii) If <multisurfacez binary representation> immediately contains a <multipolygonz binary representation>, then <multisurfacez binary representation> produces an *ST_MultiPolygon* value specified by the immediately contained <multipolygonz binary representation>.

iii) Otherwise, <multisurfacez binary representation> produces an empty set of type *ST_MultiSurface*.

bt) Case:

i) If <multisurfacem binary representation> immediately contains a <num>, then <multisurfacem binary representation> produces an *ST_MultiSurface* value. Let *ASA* be the *ST_Surface* ARRAY value with cardinality of <num> that contains the *ST_Surface* values specified by the immediately contained <surfacem binary representation>s. <multisurfacem binary representation> produces an *ST_MultiSurface* value as the result of the value expression: *NEW ST_MultiSurface(ASA)*.

ii) If <multisurfacem binary representation> immediately contains a <multipolygonm binary representation>, then <multisurfacem binary representation> produces an *ST_MultiPolygon* value specified by the immediately contained <multipolygonm binary representation>.

iii) Otherwise, <multisurfacem binary representation> produces an empty set of type *ST_MultiSurface*.

bu) Case:

i) If <multisurface binary representation> immediately contains a <num>, then <multisurface binary representation> produces an *ST_MultiSurface* value. Let *ASA* be the *ST_Surface* ARRAY value with cardinality of <num> that contains the *ST_Surface* values specified by the immediately contained <surface binary representation>s. <multisurface binary representation> produces an *ST_MultiSurface* value as the result of the value expression: *NEW ST_MultiSurface(ASA)*.

ii) If <multisurface binary representation> immediately contains a <multipolygon binary representation>, then <multisurface binary representation> produces an *ST_MultiPolygon* value specified by the immediately contained <multipolygon binary representation>.

iii) Otherwise, <multisurface binary representation> produces an empty set of type *ST_MultiSurface*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

bv) Case:

i) If <multipolygonzm binary representation> immediately contains <num>, then <multipolygonzm binary representation> is the well-known binary representation for an *ST_MultiPolygon* value. Let *APA* be the *ST_Polygon* ARRAY value with cardinality of <num> that contains the *ST_Polygon* values specified by the immediately contained <polygonzm binary representation>s. <multipolygonzm binary representation> produces an *ST_MultiPolygon* value as the result of the value expression: *NEW ST_MultiPolygon(APA)*.

ii) Otherwise, <multipolygonzm binary representation> produces an empty set of type *ST_MultiPolygon*.

bw) Case:

i) If <multipolygonz binary representation> immediately contains <num>, then <multipolygonz binary representation> is the well-known binary representation for an *ST_MultiPolygon* value. Let *APA* be the *ST_Polygon* ARRAY value with cardinality of <num> that contains the *ST_Polygon* values specified by the immediately contained <polygonz binary representation>s. <multipolygonz binary representation> produces an *ST_MultiPolygon* value as the result of the value expression: *NEW ST_MultiPolygon*(*APA*).

ii) Otherwise, <multipolygonz binary representation> produces an empty set of type *ST_MultiPolygon*.

bx) Case:

i) If <multipolygonm binary representation> immediately contains <num>, then <multipolygonm binary representation> is the well-known binary representation for an *ST_MultiPolygon* value. Let *APA* be the *ST_Polygon* ARRAY value with cardinality of <num> that contains the *ST_Polygon* values specified by the immediately contained <polygonm binary representation>s. <multipolygonm binary representation> produces an *ST_MultiPolygon* value as the result of the value expression: *NEW ST_MultiPolygon(APA)*.

ii) Otherwise, <multipolygonm binary representation> produces an empty set of type *ST_MultiPolygon*.

by) Case:

i) If <multipolygon binary representation> immediately contains <num>, then <multipolygon binary representation> is the well-known binary representation for an *ST_MultiPolygon* value. Let *APA* be the *ST_Polygon* ARRAY value with cardinality of <num> that contains the *ST_Polygon* values specified by the immediately contained <polygon binary representation>s. <multipolygon binary representation> produces an *ST_MultiPolygon* value as the result of the value expression: *NEW ST_MultiPolygon(APA)*.

ii) Otherwise, <multipolygon binary representation> produces an empty set of type *ST_MultiPolygon*.

bz) Case:

i) If <geometrycollectionzm binary representation> immediately contains <num>, then <geometrycollectionzm binary representation> is the well-known binary representation for an *ST_GeomCollection*. Let *AGA* be the *ST_Geometry* ARRAY value with cardinality of <num> that contains the *ST_Geometry* values specified by the immediately contained <well-knownzm binary representation>s. <geometrycollectionzm binary representation> produces an *ST_GeomCollection* value as the result of the value expression: *NEW ST_GeomCollection(AGA)*.

ii) Otherwise, <geometrycollectionzm binary representation> produces an empty set of type *ST_GeomCollection*.

ca) Case:

i) If <geometrycollectionz binary representation> immediately contains <num>, then <geometrycollectionz binary representation> is the well-known binary representation for an *ST_GeomCollection*. Let *AGA* be the *ST_Geometry* ARRAY value with cardinality of <num> that contains the *ST_Geometry* values specified by the immediately contained <well-knownz binary representation>s. <geometrycollectionz binary representation> produces an *ST_GeomCollection* value as the result of the value expression: *NEW ST_GeomCollection(AGA)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) Otherwise, <geometrycollectionz binary representation> produces an empty set of type *ST_GeomCollection*.

cb) Case:

i) If <geometrycollectionm binary representation> immediately contains <num>, then <geometrycollectionm binary representation> is the well-known binary representation for an *ST_GeomCollection*. Let *AGA* be the *ST_Geometry* ARRAY value with cardinality of <num> that contains the *ST_Geometry* values specified by the immediately contained <well-known binarym representation>s. <geometrycollectionm binary representation> produces an *ST_GeomCollection* value as the result of the value expression: *NEW ST_GeomCollection(AGA)*.

ii) Otherwise, <geometrycollectionm binary representation> produces an empty set of type *ST_GeomCollection*.

cc) Case:

i) If <geometrycollection binary representation> immediately contains <num>, then <geometrycollection binary representation> is the well-known binary representation for an *ST_GeomCollection*. Let *AGA* be the *ST_Geometry* ARRAY value with cardinality of <num> that contains the *ST_Geometry* values specified by the immediately contained <well-known2d binary representation>s. <geometrycollection binary representation> produces an *ST_GeomCollection* value as the result of the value expression: *NEW ST_GeomCollection(AGA)*.

ii) Otherwise, <geometrycollection binary representation> produces an empty set of type *ST_GeomCollection*.

cd) <wkbpolygonpatchzm binary> produces an *ST_Polygon* value specified by the immediately contained <polygonzm binary representation>.

ce) <wkbpolygonpatchz binary> produces an *ST_Polygon* value specified by the immediately contained <polygonz binary representation>.

cf) <wkbpolygonpatchm binary> produces an *ST_Polygon* value specified by the immediately contained <polygonm binary representation>.

cg) <wkbpolygonpatch binary> produces an *ST_Polygon* value specified by the immediately contained <polygon binary representation>.

ch) <wkbtrianglepatchzm binary> produces an *ST_Triangle* value specified by the immediately contained <trianglezm binary representation>.

ci) <wkbtrianglepatchz binary> produces an *ST_Triangle* value specified by the immediately contained <trianglez binary representation>.

cj) <wkbtrianglepatchm binary> produces an *ST_Triangle* value specified by the immediately contained <trianglem binary representation>.

ck) <wkbtrianglepatch binary> produces an *ST_Triangle* value specified by the immediately contained <triangle binary representation>.

cl) <wkbtinelement binary> produces an *ST_TINElement* value specified by the immediately contained <tinelement binary representation>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

cm) Case:

i) If <wkbcurvezm binary> immediately contains a <linestringzm binary representation>, then <wkbcurvezm binary> produces an *ST_LineString* value specified by the immediately contained <linestringzm binary representation>.

ii) Otherwise, <wkbcurvezm binary> produces an *ST_CircularString* value specified by the immediately contained <circularstringzm binary representation>.

cn) Case:

i) If <wkbcurvez binary> immediately contains a <linestringz binary representation>, then <wkbcurvez binary> produces an *ST_LineString* value specified by the immediately contained <linestringz binary representation>.

ii) Otherwise, <wkbcurvez binary> produces an *ST_CircularString* value specified by the immediately contained <circularstringz binary representation>.

co) Case:

i) If <wkbcurvem binary> immediately contains a <linestringm binary representation>, then <wkbcurvem binary> produces an *ST_LineString* value specified by the immediately contained <linestringm binary representation>.

ii) Otherwise, <wkbcurvem binary> produces an *ST_CircularString* value specified by the immediately contained <circularstringm binary representation>.

cp) Case:

i) If <wkbcurve binary> immediately contains a <linestring binary representation>, then <wkbcurve binary> produces an *ST_LineString* value specified by the immediately contained <linestring binary representation>.

ii) Otherwise, <wkbcurve binary> produces an *ST_CircularString* value specified by the immediately contained <circularstring binary representation>.

cq) Case:

i) If <wkbringzm binary> immediately contains a <linestringzm binary representation>, then <wkbringzm binary> produces an *ST_LineString* value specified by the immediately contained <linestringzm binary representation>.

ii) If <wkbringzm binary> immediately contains a <circularstringzm binary representation>, then <wkbringzm binary> produces an *ST_CircularString* value specified by the immediately contained <circularstringzm binary representation>.

iii) Otherwise, <wkbringzm binary> produces an *ST_CompoundCurve* value specified by the immediately contained <compoundcurvezm binary representation>.

cr) Case:

i) If <wkbringz binary> immediately contains a <linestringz binary representation>, then <wkbringz binary> produces an *ST_LineString* value specified by the immediately contained <linestringz binary representation>.

ii) If <wkbringz binary> immediately contains a <circularstringz binary representation>, then <wkbringz binary> produces an *ST_CircularString* value specified by the immediately contained <circularstringz binary representation>.

iii) Otherwise, <wkbringz binary> produces an *ST_CompoundCurve* value specified by the immediately contained <compoundcurvez binary representation>.

cs) Case:

i) If <wkbringm binary> immediately contains a <linestringm binary representation>, then <wkbringm binary> produces an *ST_LineString* value specified by the immediately contained <linestringm binary representation>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) If <wkbringm binary> immediately contains a <circularstringm binary representation>, then <wkbringm binary> produces an *ST_CircularString* value specified by the immediately contained <circularstringm binary representation>.

iii) Otherwise, <wkbringm binary> produces an *ST_CompoundCurve* value specified by the immediately contained <compoundcurvem binary representation>.

ct) Case:

i) If <wkbring binary> immediately contains a <linestring binary representation>, then <wkbring binary> produces an *ST_LineString* value specified by the immediately contained <linestring binary representation>.

ii) If <wkbring binary> immediately contains a <circularstring binary representation>, then <wkbring binary> produces an *ST_CircularString* value specified by the immediately contained <circularstring binary representation>.

iii) Otherwise, <wkbring binary> produces an *ST_CompoundCurve* value specified by the immediately contained <compoundcurve binary representation>.

cu) Let *XC* be the DOUBLE PRECISION value specified by <wkbx> in <wkbpointzm binary>, *YC* be the DOUBLE PRECISION value specified by <wkby> in <wkbpointzm binary>, *ZC* be the DOUBLE PRECISION value specified by <wkbz> in <wkbpointzm binary>, and *MC* be the DOUBLE PRECISION value specified by <wkbm> in <wkbpointzm binary>. <wkbpointzm binary> produces an *ST_Point* value as the result of the value expression: *NEW ST_Point*(*XC, YC, ZC, MC*).

cv) Let *XC* be the DOUBLE PRECISION value specified by <wkbx> in <wkbpointz binary>, *YC* be the DOUBLE PRECISION value specified by <wkby> in <wkbpointz binary>, and *ZC* be the DOUBLE PRECISION value specified by <wkbz> in <wkbpointz binary>. <wkbpointz binary> produces an *ST_Point* value as the result of the value expression: *NEW ST_Point(XC, YC, ZC*).

cw) Let *XC* be the DOUBLE PRECISION value specified by <wkbx> in <wkbpointm binary>, *YC* be the DOUBLE PRECISION value specified by <wkby> in <wkbpointm binary>, and *MC* be the DOUBLE PRECISION value specified by <wkbm> in <wkbpointm binary>. <wkbpointm binary> produces an *ST_Point* value as the result of the value expression: *NEW ST_Point(XC, YC, NULL, MC*).

cx) Let *XC* be the DOUBLE PRECISION value specified by <wkbx> in <wkbpoint binary> and *YC* be the DOUBLE PRECISION value specified by <wkby> in <wkbpoint binary>. <wkbpoint binary> produces an *ST_Point* value as the result of the value expression: *NEW ST_Point(XC, YC)*.

cy) <wkbx> is a <double> representing the x coordinate value of an *ST_Point* value.

cz) <wkby> is a <double> representing the y coordinate value of an *ST_Point* value.

da) <wkbz> is a <double> representing the z coordinate value of an *ST_Point* value.

db) <wkbm> is a <double> representing the m coordinate value of an *ST_Point* value.

dc) <num> is an <uint32> that represents the number of elements in a repeating group.

dd) <nume> is an <uint32> that represents the number of elements in a repeating group.

de) <wkbmaxsidelength> is a <double> representing the maxsidelength value of an *ST_TIN* value.

df) <wkblinearringzm binary> produces an *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <wkbpointzm binary>s.

dg) <wkblinearringz binary> produces an *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <wkbpointz binary>s.

dh) <wkblinearringm binary> produces an *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <wkbpointm binary>s.

di) <wkblinearring binary> produces an *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <wkbpoint binary>s.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

dj) The <well-known binary representation> <uint32> values are defined in Table 15 — <well-known binary representation> <uint32> Values.

Table 15 — <well-known binary representation> <uint32> Values

<well-known binary representation>	<uint32> Value
<wkbpoint>	1 (one)
<wkblinestring>	2
<wkbpolygon>	3
<wkbmultipoint>	4
<wkbmultilinestring>	5
<wkbmultipolygon>	6
<wkbgeometrycollection>	7
<wkbcircularstring>	8 or 1000001
<wkbcompoundcurve>	9 or 1000002
<wkbcurvepolygon>	10 or 1000003
<wkbmulticurve>	11 or 1000004
<wkbmultisurface>	12 or 1000005
<wkbpolyhedralsurface>	15
<wkbtin>	16
<wkbtriangle>	17
<wkbpointz>	1001
<wkblinestringz>	1002
<wkbpolygonz>	1003
<wkbmultipointz>	1004
<wkbmultilinestringz>	1005
<wkbmultipolygonz>	1006
<wkbgeometrycollectionz>	1007
<wkbcircularstringz>	1008
<wkbcompoundcurvez>	1009
<wkbcurvepolygonz>	1010
<wkbmulticurvez>	1011
<wkbmultisurfacez>	1012
<wkbpolyhedralsurfacez>	1015
<wkbtinz>	1016
<wkbtrianglez>	1017

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

<well-known binary representation>	<uint32> Value
<wkbpointm>	2001
<wkblinestringm>	2002
<wkbpolygonm>	2003
<wkbmultipointm>	2004
<wkbmultilinestringm>	2005
<wkbmultipolygonm>	2006
<wkbgeometrycollectionm>	2007
<wkbcircularstringm>	2008
<wkbcompoundcurvem>	2009
<wkbcurvepolygonm>	2010
<wkbmulticurvem>	2011
<wkbmultisurfacem>	2012
<wkbpolyhedralsurfacem>	2015
<wkbtinm>	2016
<wkbtrianglem>	2017
<wkbpointzm>	3001
<wkblinestringzm>	3002
<wkbpolygonzm>	3003
<wkbmultipointzm>	3004
<wkbmultilinestringzm>	3005
<wkbmultipolygonzm>	3006
<wkbgeometrycollectionzm>	3007
<wkbcircularstringzm>	3008
<wkbcompoundcurvezm>	3009
<wkbcurvepolygonzm>	3010
<wkbmulticurvezm>	3011
<wkbmultisurfacezm>	3012
<wkbpolyhedralsurfacezm>	3015
<wkbtinzm>	3016
<wkbtrianglezm>	3017

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

dk) <byte order> indicates the binary representation of <uint32> and <double> values that follow <byte order>.

dl) <big endian> is a <byte order> represented by a <byte> with the value 0 (zero). <uint32> is Big Endian (most significant octet first). <double> is Big Endian (sign bit is in the first octet).

dm) <little endian> is a <byte order> represented by a <byte> with the value 1 (one). <uint32> is Little Endian (most significant octet last). <double> is Little Endian (sign bit is in the last octet).

dn) <byte> is an 8 bit (1 (one) octet) data type that encodes an unsigned integer in the range [0, 255].

© ISO/IEC 2011 – All rights reserved

do) <uint32> s a 32 bit (4 octets) data type that encodes an unsigned integer in the range [0, 4294967295].

dp) <double> is a 64 bit (8 octets) double precision data type that encodes a double precision format using the IEC 559:1989.

dq) <well-known binary representation> provides a portable representation of a geometry value as a contiguous stream of octets in a BINARY LARGE OBJECT value. The serialized *ST_Geometry* is either represented in Big Endian format or Little Endian format. Conversion between Big Endian format or Little Endian format is a simple operation involving reversing the order of octets within each <uint32> or <double> value in the BINARY LARGE OBJECT.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6 Point Types

6.1 ST_Point Type and Routines

6.1.1 ST_Point Type

Purpose

The ST_Point type is a 0-dimensional geometry and represents a single location.

Definition

```
CREATE TYPE ST_Point
   UNDER ST_Geometry
   AS (
      ST_PrivateX DOUBLE PRECISION DEFAULT NULL,
      ST_PrivateY DOUBLE PRECISION DEFAULT NULL,
      ST_PrivateZ DOUBLE PRECISION DEFAULT NULL,
      ST_PrivateM DOUBLE PRECISION DEFAULT NULL
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_Point
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
      RETURNS ST_Point
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_Point
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
       ansrid INTEGER)
      RETURNS ST_Point
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_Point
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
      RETURNS ST_Point
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_Point
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
       ansrid INTEGER)
      RETURNS ST_Point
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_Point
   (xcoord DOUBLE PRECISION,
    ycoord DOUBLE PRECISION)
   RETURNS ST_Point
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Point
   (xcoord DOUBLE PRECISION,
    ycoord DOUBLE PRECISION,
    ansrid INTEGER)
   RETURNS ST_Point
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Point
   (xcoord DOUBLE PRECISION,
    ycoord DOUBLE PRECISION,
    zcoord DOUBLE PRECISION)
   RETURNS ST_Point
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Point
   (xcoord DOUBLE PRECISION,
    ycoord DOUBLE PRECISION,
    zcoord DOUBLE PRECISION,
    ansrid INTEGER)
   RETURNS ST_Point
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Point
   (xcoord DOUBLE PRECISION,
    ycoord DOUBLE PRECISION,
    zcoord DOUBLE PRECISION,
    mcoord DOUBLE PRECISION)
   RETURNS ST_Point
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_Point
   (xcoord DOUBLE PRECISION,
    ycoord DOUBLE PRECISION,
    zcoord DOUBLE PRECISION,
    mcoord DOUBLE PRECISION,
    ansrid INTEGER)
   RETURNS ST_Point
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_X()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_X
   (xcoord DOUBLE PRECISION)
   RETURNS ST_Point
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_Y()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Y
   (ycoord DOUBLE PRECISION)
   RETURNS ST_Point
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_Z()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Z
   (zcoord DOUBLE PRECISION)
   RETURNS ST_Point
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_M()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_M
   (mcoord DOUBLE PRECISION)
   RETURNS ST_Point
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_ExplicitPoint()
   RETURNS DOUBLE PRECISION ARRAY[4]
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
2) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.
3) The attribute *ST_PrivateX* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivateX*.
4) The attribute *ST_PrivateY* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivateY*.
5) The attribute *ST_PrivateZ* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivateZ*.
6) The attribute *ST_PrivateM* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivateM*.

Description

1) The *ST_Point* type provides for public use:
   a) a method *ST_Point(CHARACTER LARGE OBJECT)*,
   b) a method *ST_Point(CHARACTER LARGE OBJECT, INTEGER)*,
   c) a method *ST_Point(BINARY LARGE OBJECT)*,
   d) a method *ST_Point(BINARY LARGE OBJECT, INTEGER)*,
   e) a method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION)*,
   f) a method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, INTEGER)*,
   g) a method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION)*,
   h) a method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION, INTEGER)*,
   i) a method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION)*,
   j) a method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION, INTEGER)*,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

k) a method *ST_X()*,

l) a method *ST_X(DOUBLE PRECISION)*,

m) a method *ST_Y()*,

n) a method *ST_Y(DOUBLE PRECISION)*,

o) a method *ST_Z()*,

p) a method *ST_Z(DOUBLE PRECISION)*,

q) a method *ST_M()*,

r) a method *ST_M(DOUBLE PRECISION)*,

s) a method *ST_ExplicitPoint()*,

t) a function *ST_PointFromText(CHARACTER LARGE OBJECT)*,

u) a function *ST_PointFromText(CHARACTER LARGE OBJECT, INTEGER)*,

v) a function *ST_PointFromWKB(BINARY LARGE OBJECT)*,

w) a function *ST_PointFromWKB(BINARY LARGE OBJECT, INTEGER)*,

x) a function *ST_PointFromGML(CHARACTER LARGE OBJECT)*,

y) a function *ST_PointFromGML(CHARACTER LARGE OBJECT, INTEGER)*.

2) The *ST_PrivateX* attribute contains the x coordinate value.

3) The *ST_PrivateY* attribute contains the y coordinate value.

4) The *ST_PrivateZ* attribute contains the z coordinate value.

5) The *ST_PrivateM* attribute contains the m coordinate value.

6) An *ST_Point* value is a 0-dimensional geometry that represents a single location.

7) The dimension of an *ST_Point* value is 0 (zero).

8) The coordinate dimension of an *ST_Point* value is the number of coordinate values associated with the position.

9) The boundary of an *ST_Point* value is the empty set.

10) An *ST_Point* value is simple.

11) An *ST_Point* value returned by the constructor function corresponds to the empty set.

12) An *ST_Point* value is not well formed if either:

a) the *ST_PrivateX* attribute is the null value and the *ST_PrivateY* attribute is not the null value,

b) the *ST_PrivateY* attribute is the null value and the *ST_PrivateX* attribute is not the null value,

c) the *ST_PrivateIs3D* attribute is 0 (one) and the *ST_PrivateZ* attribute is not the null value,

d) the *ST_PrivateIs3D* attribute is 1 (one), the *ST_PrivateX* attribute is the null value, and the *ST_PrivateZ* attribute is not the null value,

e) the *ST_PrivateIs3D* attribute is 1 (one), the *ST_PrivateX* attribute is not the null value, and the *ST_PrivateZ* attribute is the null value,

f) the *ST_PrivateIsMeasured* attribute is 0 (one) and the *ST_PrivateM* attribute is not the null value,

g) the *ST_PrivateIsMeasured* attribute is 1 (one), the *ST_PrivateX* attribute is the null value, and the *ST_PrivateM* attribute is not the null value,

h) the *ST_PrivateIsMeasured* attribute is 1 (one), the *ST_PrivateX* attribute is not the null value, and the *ST_PrivateM* attribute is the null value, or

i) the *ST_PrivateCoordinateDimension* attribute is not equal to value expression: *2 + ST_PrivateIs3D + ST_PrivateIsMeasured*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6.1.2 ST_Point Methods

Purpose

Return an ST_Point value constructed from either the well-known text representation, the well-known binary representation, the GML representation, or the specified coordinate values.

Definition

```
CREATE CONSTRUCTOR METHOD ST_Point
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_Point
   FOR ST_Point
   RETURN NEW ST_Point(awktorgml, 0)

CREATE CONSTRUCTOR METHOD ST_Point
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_Point
   FOR ST_Point
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_Point
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_Point
   FOR ST_Point
   RETURN NEW ST_Point(awkb, 0)

CREATE CONSTRUCTOR METHOD ST_Point
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_Point
   FOR ST_Point
   RETURN ST_PointFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_Point
   (xcoord DOUBLE PRECISION,
    ycoord DOUBLE PRECISION)
   RETURNS ST_Point
   FOR ST_Point
   RETURN NEW ST_Point(xcoord, ycoord, 0)

CREATE CONSTRUCTOR METHOD ST_Point
   (xcoord DOUBLE PRECISION,
    ycoord DOUBLE PRECISION,
    ansrid INTEGER)
   RETURNS ST_Point
   FOR ST_Point
   RETURN SELF.                        -- Return an ST_Point value with
      ST_PrivateDimension(0).             -- dimension = 0,
      ST_PrivateCoordinateDimension(2). -- coordinate dimension = 2,
      ST_PrivateIs3D(0).                  -- is not 3D,
      ST_PrivateIsMeasured(0).            -- is not measured,
      ST_SRID(ansrid).                     -- SRID = ansrid,
      ST_X(xcoord).                       -- x coordinate = xcoord,
      ST_Y(ycoord)                        -- y coordinate = ycoord
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CREATE CONSTRUCTOR METHOD ST_Point
   (xcoord DOUBLE PRECISION,
    ycoord DOUBLE PRECISION,
    zcoord DOUBLE PRECISION)
   RETURNS ST_Point
   FOR ST_Point
   RETURN NEW ST_Point(xcoord, ycoord, zcoord, 0)

CREATE CONSTRUCTOR METHOD ST_Point
   (xcoord DOUBLE PRECISION,
    ycoord DOUBLE PRECISION,
    zcoord DOUBLE PRECISION,
    ansrid INTEGER)
   RETURNS ST_Point
   FOR ST_Point
   BEGIN
      IF ( xcoord IS NULL AND ycoord IS NOT NULL ) OR
         ( xcoord IS NOT NULL AND ycoord IS NULL ) THEN
         -- if not well-formed, raise an exception
         SIGNAL SQLSTATE '2FF03'
            SET MESSAGE_TEXT = 'null argument';
      END IF;
      IF xcoord IS NULL THEN
         -- If xcoord is the null value, assume an empty
         -- point value is being created, check zcoord is null.
         IF zcoord IS NOT NULL THEN
            SIGNAL SQLSTATE '2FF16'
               SET MESSAGE_TEXT = 'not an empty set';
         END IF;
      ELSE
         -- Otherwise, check zcoord is not null.
         IF zcoord IS NULL THEN
            SIGNAL SQLSTATE '2FF03'
               SET MESSAGE_TEXT = 'null argument';
         END IF;
      END IF;
      RETURN SELF.                    -- Return an ST_Point value with
         ST_PrivateDimension(0).      -- dimension = 0,
         ST_PrivateCoordinateDimension(
            CASE
               WHEN zcoord IS NOT NULL THEN
                                       -- if z coordinate is not the null
                                       -- value, then
                  3                    --   coordinate dimension = 3
               ELSE                    -- otherwise,
                  2                    --   coordinate dimension = 2
            END).
         ST_PrivateIs3D(
            CASE
               WHEN zcoord IS NOT NULL THEN
                                       -- if z coordinate is not the null
                  1                    --   value, then, is 3D
               ELSE                    -- otherwise,
                  0                    --   is not 3D
            END).
         ST_PrivateIsMeasured(0).      -- not measured,
         ST_SRID(ansrid).               -- SRID = ansrid,
         ST_PrivateX(xcoord).          -- x coordinate = xcoord,
         ST_PrivateY(ycoord).          -- y coordinate = ycoord,
         ST_PrivateZ(zcoord);          -- z coordinate = zcoord
   END
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CREATE CONSTRUCTOR METHOD ST_Point
   (xcoord DOUBLE PRECISION,
    ycoord DOUBLE PRECISION,
    zcoord DOUBLE PRECISION,
    mcoord DOUBLE PRECISION)
   RETURNS ST_Point
   FOR ST_Point
   RETURN NEW ST_Point(xcoord, ycoord, zcoord, mcoord, 0)

CREATE CONSTRUCTOR METHOD ST_Point
   (xcoord DOUBLE PRECISION,
    ycoord DOUBLE PRECISION,
    zcoord DOUBLE PRECISION,
    mcoord DOUBLE PRECISION,
    ansrid INTEGER)
   RETURNS ST_Point
   FOR ST_Point
   BEGIN
      IF ( xcoord IS NULL AND ycoord IS NOT NULL ) OR
         ( xcoord IS NOT NULL AND ycoord IS NULL ) THEN
         -- if not well-formed, raise an exception
         SIGNAL SQLSTATE '2FF03'
            SET MESSAGE_TEXT = 'null argument';
      END IF;
      IF xcoord IS NULL THEN
         -- If xcoord is the null value, assume an empty
         -- point value is being created, check zcoord and mcoord is null.
         IF zcoord IS NOT NULL THEN
            SIGNAL SQLSTATE '2FF16'
               SET MESSAGE_TEXT = 'not an empty set';
         END IF;
      END IF;
      RETURN SELF.                        -- Return an ST_Point value with
         ST_PrivateDimension(0).             -- dimension = 0,
         ST_PrivateCoordinateDimension(
            CASE
               WHEN (zcoord IS NOT NULL AND mcoord IS NOT NULL) THEN
                                          -- if z coordinate is not the null
                                          -- value and mcoord is not the null
                                          -- value, then
                  4                       --   coordinate dimension = 4
               WHEN ((zcoord IS NOT NULL AND mcoord IS NULL) OR
                     (zcoord IS NULL AND mcoord IS NOT NULL)) THEN
                                          -- if z coordinate is not the null
                                          -- value and mcoord is the null
                                          -- value or if z coordinate is
                                          -- the null value and m is not
                                          -- the null value, then
                  3                       --   coordinate dimension = 3
               ELSE                       -- otherwise,
                  2                       --   coordinate dimension = 2
            END).
         ST_PrivateIs3D(
            CASE
               WHEN zcoord IS NOT NULL THEN
                                          -- if z coordinate is not the null
                  1                       --   value, then is 3D
               ELSE                       -- otherwise,
                  0                       --   is not 3D
            END).
         ST_PrivateIsMeasured(
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
            CASE
               WHEN mcoord IS NOT NULL THEN
                                        -- if m coordinate value is
                                        -- not the null value, then
                  1                     --   is measured
               ELSE                     -- otherwise,
                  0                     --   is not measured
            END).
         ST_SRID(ansrid).                -- SRID = ansrid,
         ST_PrivateX(xcoord).           -- x coordinate = xcoord,
         ST_PrivateY(ycoord).           -- y coordinate = ycoord,
         ST_PrivateZ(zcoord).           -- z coordinate = zcoord
         ST_PrivateM(
            CASE
               WHEN ( xcoord IS NULL AND
                      ycoord IS NULL AND
                      zcoord IS NULL ) THEN
                                        -- if x, y and z coordinate value
                                        -- is the null value, then
                  NULL                  --   set m coordinate value to
                                        --   the null value
               ELSE                     -- otherwise,
                  mcoord                --   set m coordinate value
                                        -- to mcoord
            END);
END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

2) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The method *ST_Point(CHARACTER LARGE OBJECT)* takes the following input parameter:

   a) a CHARACTER LARGE OBJECT value *awktorgml*.

2) The null-call type preserving SQL-invoked constructor method *ST_Point(CHARACTER LARGE OBJECT)* returns the result of the value expression: *NEW ST_Point(awktorgml, 0)*.

3) The method *ST_Point(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a CHARACTER LARGE OBJECT value *awktorgml*,

   b) an INTEGER value *ansrid*.

4) For the null-call type preserving SQL-invoked constructor method *ST_Point(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If *awktorgml* contains a Point XML element in the GML representation, then return the result of the value expression: *ST_PointFromGML(awktorgml, ansrid)*.

   b) Otherwise, return the result of the value expression: *ST_PointFromText(awktorgml, ansrid)*.

5) The method *ST_Point(BINARY LARGE OBJECT)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *awkb*.

6) The null-call type preserving SQL-invoked constructor method *ST_Point(BINARY LARGE OBJECT)* returns the result of the value expression: *NEW ST_Point(awkb, 0)*.

7) The method *ST_Point(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

a) a BINARY LARGE OBJECT value *awkb*,

b) an INTEGER value *ansrid*.

8) The null-call type preserving SQL-invoked constructor method *ST_Point(BINARY LARGE OBJECT, INTEGER)* returns the result of the value expression: *ST_PointFromWKB(awkb, ansrid)*.

9) The method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION)* takes the following input parameters:

a) a DOUBLE PRECISION value *xcoord*,

b) a DOUBLE PRECISION value *ycoord*.

10) The null-call type preserving SQL-invoked constructor method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION)* returns the result of the value expression: *NEW ST_Point(xcoord, ycoord, 0)*.

11) The method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, INTEGER)* takes the following input parameters:

a) a DOUBLE PRECISION value *xcoord*,

b) a DOUBLE PRECISION value *ycoord*,

c) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, INTEGER)* returns an *ST_Point* value with:

a) The dimension set to 0 (zero).

b) The coordinate dimension value set to 2.

c) The *ST_PrivateIs3D* attribute set to 0 (zero).

d) The *ST_PrivateIsMeasured* attribute set to 0 (zero).

e) The spatial reference system identifier set to *ansrid*.

f) Using the method *ST_X(DOUBLE PRECISION)*, the x coordinate value is set to *xcoord*.

g) Using the method *ST_Y(DOUBLE PRECISION)*, the y coordinate value is set to *ycoord*.

h) The z coordinate value set to NULL by default clause.

i) The m coordinate value set to NULL by default clause.

13) The method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION)* takes the following input parameters:

a) a DOUBLE PRECISION value *xcoord*,

b) a DOUBLE PRECISION value *ycoord*,

c) a DOUBLE PRECISION value *zcoord*.

14) The null-call type preserving SQL-invoked constructor method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION)* returns the result of the value expression: *NEW ST_Point(xcoord, ycoord, zcoord, 0)*.

15) The method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION, INTEGER)* takes the following input parameters:

a) a DOUBLE PRECISION value *xcoord*,

b) a DOUBLE PRECISION value *ycoord*,

c) a DOUBLE PRECISION value *zcoord*,

d) an INTEGER value *ansrid*.

16) For the null-call type preserving SQL-invoked constructor method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION, INTEGER)*:

Case:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

a) If *xcoord* is the null value and *ycoord* is not the null value, *xcoord* is not the null value and *ycoord* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.

b) If *xcoord* is the null value and *zcoord* is not the null value, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

c) If *xcoord* is not the null value and *zcoord* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.

d) Otherwise, return an *ST_Point* value with:

i) The dimension set to 0 (zero).

ii) Case:

1) If *zcoord* is not the null value, then the coordinate dimension value set to 3.

2) Otherwise, the coordinate dimension value set to 2.

iii) Case:

1) If *zcoord* is not the null value, then the *ST_PrivateIs3D* attribute set to 1 (one).

2) Otherwise, the *ST_PrivateIs3D* attribute set to 0 (zero).

iv) The *ST_PrivateIsMeasured* attribute set to 0 (zero).

v) The spatial reference system identifier set to *ansrid*.

vi) The x coordinate value is set to *xcoord*.

vii) The y coordinate value is set to *ycoord*.

viii) The z coordinate value is set to *zcoord*.

ix) The m coordinate value set to NULL by default clause.

17) The method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION)* takes the following input parameters:

a) a DOUBLE PRECISION value *xcoord*,

b) a DOUBLE PRECISION value *ycoord*,

c) a DOUBLE PRECISION value *zcoord*,

d) a DOUBLE PRECISION value *mcoord*.

18) The null-call type preserving SQL-invoked constructor method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION)* returns the result of the value expression: *NEW ST_Point(xcoord, ycoord, zcoord, mcoord, 0)*.

19) The method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION, INTEGER)* takes the following input parameters:

a) a DOUBLE PRECISION value *xcoord*,

b) a DOUBLE PRECISION value *ycoord*,

c) a DOUBLE PRECISION value *zcoord*,

d) a DOUBLE PRECISION value *mcoord*,

e) an INTEGER value *ansrid*.

20) For the null-call type preserving SQL-invoked constructor method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION, DOUBLE PRECISION, INTEGER)*:

Case:

a) If *xcoord* is the null value and *ycoord* is not the null value, *xcoord* is not the null value and *ycoord* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.

b) If *xcoord* is the null value and *zcoord* is not the null value, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

c) Otherwise, return an ST_Point value with:

i) The dimension set to 0 (zero).

ii) Case:

1) If *zcoord* is not the null value and *mcoord* is not the null value, then the coordinate dimension value set to 4.

2) If *zcoord* is not the null value and *mcoord* is the null value or if *zcoord* is the null value and *mcoord* is not the null value, then the coordinate dimension value set to 3.

3) Otherwise, the coordinate dimension value set to 2.

iii) Case:

1) If *zcoord* is not the null value, then the *ST_PrivateIs3D* attribute set to 1 (one).

2) Otherwise, the *ST_PrivateIs3D* attribute set to 0 (zero).

iv) Case:

1) If *mcoord* is not the null value, then the *ST_PrivateIsMeasured* attribute set to 1 (one).

2) Otherwise, the *ST_PrivateIsMeasured* attribute set to 0 (zero).

v) The spatial reference system identifier set to *ansrid*.

vi) The x coordinate value is set to *xcoord*.

vii) The y coordinate value is set to *ycoord*.

viii) The z coordinate value is set to *zcoord*.

ix) Case:

1) If *xcoord* is the null value and *ycoord* is the null value and *zcoord* is the null value, then the m coordinate value is set to the null value.

2) Otherwise, the m coordinate value is set to *mcoord*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6.1.3 ST_X Methods

Purpose

Observe and mutate the x coordinate value of an ST_Point value.

Definition

```
CREATE METHOD ST_X()
   RETURNS DOUBLE PRECISION
   FOR ST_Point
   RETURN SELF.ST_PrivateX

CREATE METHOD ST_X
   (xcoord DOUBLE PRECISION)
   RETURNS ST_Point
   FOR ST_Point
   BEGIN
      IF xcoord IS NULL THEN
         SIGNAL SQLSTATE '2FF03'
            SET MESSAGE_TEXT = 'null argument';
      ELSE
         RETURN
            CASE
               WHEN SELF IS NULL THEN
                  NULL
               ELSE
                  SELF.ST_PrivateX(xcoord)
            END;
      END IF;
   END
```

Description

1) The method *ST_X()* has no input parameters.
2) The null-call method *ST_X()* returns the value of the *ST_PrivateX* attribute.
3) The method *ST_X(DOUBLE PRECISION)* takes the following input parameters:
   a) a DOUBLE PRECISION value *xcoord*.
4) For the type preserving method *ST_X(DOUBLE PRECISION)*:

   Case:

   a) If *xcoord* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If SELF is the null value, then return the null value.
   c) Otherwise, return the result of the value expression: *SELF.ST_PrivateX(xcoord)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6.1.4 ST_Y Methods

Purpose

Observe and mutate the y coordinate value of an ST_Point value.

Definition

```
CREATE METHOD ST_Y()
   RETURNS DOUBLE PRECISION
   FOR ST_Point
   RETURN SELF.ST_PrivateY

CREATE METHOD ST_Y
   (ycoord DOUBLE PRECISION)
   RETURNS ST_Point
   FOR ST_Point
   BEGIN
      IF ycoord IS NULL THEN
         SIGNAL SQLSTATE '2FF03'
            SET MESSAGE_TEXT = 'null argument';
      ELSE
         RETURN
            CASE
               WHEN SELF IS NULL THEN
                  NULL
               ELSE
                  SELF.ST_PrivateY(ycoord)
            END;
      END IF;
   END
```

Description

1) The method *ST_Y()* has no input parameters.
2) The null-call method *ST_Y()* returns the value of the *ST_PrivateY* attribute.
3) The method *ST_Y(DOUBLE PRECISION)* takes the following input parameters:
   a) a DOUBLE PRECISION value *ycoord*.
4) For the type preserving method *ST_Y(DOUBLE PRECISION)*:

   Case:

   a) If *ycoord* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If SELF is the null value, then return the null value.
   c) Otherwise, return the result of the value expression: *SELF.ST_PrivateY(ycoord)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6.1.5 ST_Z Methods

Purpose

Observe and mutate the z coordinate value of an ST_Point value.

Definition

```
CREATE METHOD ST_Z()
   RETURNS DOUBLE PRECISION
   FOR ST_Point
   RETURN SELF.ST_PrivateZ

CREATE METHOD ST_Z
   (zcoord DOUBLE PRECISION)
   RETURNS ST_Point
   FOR ST_Point
   BEGIN
      IF SELF.ST_IsEmpty() = 0 AND zcoord IS NULL THEN
         SIGNAL SQLSTATE '2FF03'
            SET MESSAGE_TEXT = 'null argument';
      END IF;
      IF SELF.ST_IsEmpty() = 1 AND zcoord IS NOT NULL THEN
         SIGNAL SQLSTATE '2FF16'
            SET MESSAGE_TEXT = 'not an empty set';
      END IF;
      RETURN
         CASE
            WHEN SELF IS NULL THEN NULL
            ELSE SELF.ST_PrivateZ(zcoord)
         END;
   END
```

Description

1) The method *ST_Z()* has no input parameters.
2) The null-call method *ST_Z()* returns the value of the *ST_PrivateZ* attribute.
3) The method *ST_Z(DOUBLE PRECISION)* takes the following input parameters:
   a) a DOUBLE PRECISION value *zcoord*.
4) For the type preserving method *ST_Z(DOUBLE PRECISION)*:

   Case:

   a) If SELF is an empty set and *zcoord* is not the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument.*
   b) If SELF is not an empty set and *zcoord* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set.*
   c) If SELF is the null value, then return the null value.
   d) Otherwise, return the result of the value expression: *SELF.ST_PrivateZ(zcoord)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6.1.6 ST_M Methods

Purpose

Observe and mutate the m coordinate value of an ST_Point value.

Definition

```
CREATE METHOD ST_M()
   RETURNS DOUBLE PRECISION
   FOR ST_Point
   RETURN SELF.ST_PrivateM

CREATE METHOD ST_M
   (mcoord DOUBLE PRECISION)
   RETURNS ST_Point
   FOR ST_Point
   BEGIN
      IF SELF.ST_IsEmpty() = 0 AND mcoord IS NULL THEN
         SIGNAL SQLSTATE '2FF03'
            SET MESSAGE_TEXT = 'null argument';
      END IF;
      IF SELF.ST_IsEmpty() = 1 AND mcoord IS NOT NULL THEN
         SIGNAL SQLSTATE '2FF16'
            SET MESSAGE_TEXT = 'not an empty set';
      END IF;
      RETURN
         CASE
            WHEN SELF IS NULL THEN NULL
            ELSE SELF.ST_PrivateM(mcoord)
         END;
   END
```

Description

1) The method *ST_M()* has no input parameters.
2) The null-call method *ST_M()* returns the value of the *ST_PrivateM* attribute.
3) The method *ST_M(DOUBLE PRECISION)* takes the following input parameters:
   a) a DOUBLE PRECISION value *mcoord*.
4) For the type preserving method *ST_M(DOUBLE PRECISION)*:

   Case:

   a) If SELF is an empty set and *mcoord* is not the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument.*
   b) If SELF is not an empty set and *mcoord* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set.*
   c) If SELF is the null value, then return the null value.
   d) Otherwise, return the result of the value expression: *SELF.ST_PrivateM(mcoord)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6.1.7 ST_ExplicitPoint Method

Purpose

Return the coordinate values as a DOUBLE PRECISION ARRAY value.

Definition

```
CREATE METHOD ST_ExplicitPoint()
   RETURNS DOUBLE PRECISION ARRAY[4]
   FOR ST_Point
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         WHEN (SELF.ST_Z() IS NOT NULL AND
               SELF.ST_M() IS NOT NULL) THEN
            ARRAY[SELF.ST_X(), SELF.ST_Y(), SELF.ST_Z(), SELF.ST_M()]
         WHEN (SELF.ST_Z() IS NOT NULL AND
               SELF.ST_M() IS NULL) THEN
            ARRAY[SELF.ST_X(), SELF.ST_Y(), SELF.ST_Z()]
         WHEN (SELF.ST_Z() IS NULL AND
               SELF.ST_M() IS NOT NULL) THEN
            ARRAY[SELF.ST_X(), SELF.ST_Y(), SELF.ST_M()]
         ELSE
            ARRAY[SELF.ST_X(), SELF.ST_Y()]
      END
```

Description

1) The method *ST_ExplicitPoint()* has no input parameters.
2) For the null-call method *ST_ExplicitPoint()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) If the z coordinate value is not the null value and the m coordinate value is not the null value, then return an array of type DOUBLE PRECISION with the first element representing the x coordinate value, the second element representing the y coordinate value, the third element representing the z coordinate value, and the forth element representing the m coordinate value.
   c) If the z coordinate value is not the null value and the m coordinate value is the null value, then return an array of type DOUBLE PRECISION with the first element representing the x coordinate value, the second element representing the y coordinate value, and the third element representing the z coordinate value.
   d) If the z coordinate value is the null value and the m coordinate value is not the null value, then return an array of type DOUBLE PRECISION with the first element representing the x coordinate value, the second element representing the y coordinate value, and the third element representing the m coordinate value.
   e) Otherwise, return an array of type DOUBLE PRECISION with the first element representing the x coordinate value and the second element representing the y coordinate value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6.1.8 ST_PointFromText Functions

Purpose

Return an ST_Point value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_Point value.

Definition

```
CREATE FUNCTION ST_PointFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN PointFromText(awkt, 0)

CREATE FUNCTION ST_PointFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_PointFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_PointFromText(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_PointFromText(awkt, 0)*.
3) The function *ST_PointFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_PointFromText(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkt* is the well-known text representation of an *ST_Point* value.

      If *awkt* is not producible in the BNF for <point text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromText(awkt, ansrid) AS ST_Point)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6.1.9 ST_PointFromWKB Functions

Purpose

Return an ST_Point value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_Point value.

Definition

```
CREATE FUNCTION ST_PointFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_PointFromWKB(awkb, 0)

CREATE FUNCTION ST_PointFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_PointFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_PointFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_PointFromWKB(awkb, 0)*.
3) The function *ST_PointFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_PointFromWKB(BINARY LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkb* is the well-known binary representation of an *ST_Point* value.

      If *awkb* is not producible in the BNF for <point binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_Point)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6.1.10 ST_PointFromGML Functions

Purpose

Return an ST_Point value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representationof an ST_Point.

Definition

```
CREATE FUNCTION ST_PointFromGML
    (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
    RETURNS ST_Point
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    RETURN ST_PointFromGML(agml, 0)

CREATE FUNCTION ST_PointFromGML
    (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
     ansrid INTEGER)
    RETURNS ST_Point
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    BEGIN
       --
       -- See Description
       --
    END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_PointFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_PointFromGML(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_PointFromGML(agml, 0)*.
3) The function *ST_PointFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_PointFromGML(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If the parameter *agml* does not contain a Point XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromGML(agml, ansrid) AS ST_Point)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7 Curve Types

7.1 ST_Curve Type and Routines

7.1.1 ST_Curve Type

Purpose

The ST_Curve type is a supertype for 1-dimensional geometry types and represents a continuous locus of points from the start point to the end point. Subtypes of ST_Curve specify the form of interpolation between points.

Definition

```
CREATE TYPE ST_Curve
   UNDER ST_Geometry
   NOT INSTANTIABLE
   NOT FINAL

   METHOD ST_Length()
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_Length
      (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_3DLength()
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_3DLength
      (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_StartPoint()
      RETURNS ST_Point
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_EndPoint()
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_IsClosed()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DIsClosed()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_IsRing()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DIsRing()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_CurveToLine()
   RETURNS ST_LineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
```

Definitional Rules

1) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The *ST_Curve* type provides for public use:

   a) a method *ST_Length()*,

   b) a method *ST_Length(CHARACTER VARYING)*,

   c) a method *ST_3DLength()*,

   d) a method *ST_3DLength(CHARACTER VARYING)*,

   e) a method *ST_StartPoint()*,

   f) a method *ST_EndPoint()*,

   g) a method *ST_IsClosed()*,

   h) a method *ST_3DIsClosed()*,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

i) a method *ST_IsRing()*,

j) a method *ST_3DIsRing()*,

k) a method *ST_CurveToLine()*.

2) An *ST_Curve* value is a 1-dimensional geometry that is defined as a sequence of *ST_Point* values.

3) Subtypes of the *ST_Curve* type specifies the form of interpolation between *ST_Point* values.

4) An *ST_Curve* value is defined to be topologically closed.

5) An *ST_Curve* value is the homomorphic image of a real, closed interval:

Domain = [a, b] = { x $\in$ R | a <= x <= b } under a mapping f:[a, b] $\rightarrow$ R^2.

6) An *ST_Curve* value is not simple if any interior point has the same location as another interior point or a point in the boundary:

$\forall$ c $\in$ *ST_Curve*, [a, b] = c.Domain,

c.*ST_IsSimple()* $\Leftrightarrow$
($\forall$ x_1, x_2 $\in$ (a, b] $x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2)$) $\wedge$ ($\forall$ x_1, x_2 $\in$ [a, b) $x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2)$)

a) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation of *ST_IsSimple*.

b) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are considered in the calculation of *ST_3DIsSimple*.

c) If *SELF.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

7) The dimension of an *ST_Curve* value is 1 (one).

8) The start point of an *ST_Curve* value is returned by the method *ST_StartPoint()*.

9) The end point of an *ST_Curve* value is returned by the method *ST_EndPoint()*.

10) If the start point of an *ST_Curve* value is equal to the end point of the *ST_Curve* value, then the *ST_Curve* value is closed.

a) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation of *ST_IsClosed*.

b) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are considered in the calculation of *ST_3DIsClosed*.

c) If *SELF.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

11) The boundary of a closed *ST_Curve* value is the empty set.

12) The boundary of an *ST_Curve* value that is not closed consists of the start point and end point of the *ST_Curve* value.

13) If an *ST_Curve* value is simple and closed, then it is called a *ring*.

a) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation of *ST_IsRing*.

b) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are considered in the calculation of *ST_3DIsRing*.

c) If *SELF.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.1.2 ST_Length Methods

Purpose

Return the length measurement of an ST_Curve value, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_Length()
   RETURNS DOUBLE PRECISION
   FOR ST_Curve
   BEGIN
      --
      -- See Description
      --
   END

CREATE METHOD ST_Length
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   FOR ST_Curve
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The method *ST_Length()* has no input parameters.
2) For the null-call method *ST_Length()*:
   a) Case:
      i) If SELF is an empty set, then return the null value.
      ii) Otherwise, return the implementation-defined length of SELF, such that z and m coordinate values are not considered in the calculation, as measured in its spatial reference system.
   b) Case:
      i) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_Length()* is in the linear unit of measure identified by <linear unit>.
      ii) Otherwise, the value returned by *ST_Length()* is in an implementation-defined unit of measure.
3) The method *ST_Length(CHARACTER VARYING)* takes the following input parameter:
   a) a CHARACTER VARYING value *aunit*.
4) For the null-call method *ST_Length(CHARACTER VARYING)*:
   a) The values for *aunit* shall be a supported <unit name>.
   b) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.
   c) If the unit specified by *aunit* is not supported by the implementation to compute the length of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.
   d) Case:
      i) If SELF is an empty set, then return the null value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) Otherwise, return the implementation-defined length of SELF, such that z and m coordinate values are not considered in the calculation, as measured in its spatial reference system.

e) The returned value is in the units indicated by *aunit*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.1.3 ST_3DLength Methods

Purpose

Return the length measurement of an ST_Curve value, considering z coordinate values in the calculations.

Definition

```
CREATE METHOD ST_3DLength()
   RETURNS DOUBLE PRECISION
   FOR ST_Curve
   BEGIN
      --
      -- See Description
      --
   END

CREATE METHOD ST_3DLength
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   FOR ST_Curve
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The method *ST_3DLength()* has no input parameters.
2) For the null-call method *ST_3DLength()*:
   a) Case:
      i) If SELF is an empty set, then return the null value.
      ii) Otherwise, return the implementation-defined length of SELF, such that z (but not m) coordinate values are considered in the calculation, as measured in its spatial reference system.
   b) Case:
      i) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_3DLength()* is in the linear unit of measure identified by <linear unit>.
      ii) Otherwise, the value returned by *ST_Length()* is in an implementation-defined unit of measure.
3) The method *ST_3DLength(CHARACTER VARYING)* takes the following input parameter:
   a) a CHARACTER VARYING value *aunit*.
4) For the null-call method *ST_3DLength(CHARACTER VARYING)*:
   a) The values for *aunit* shall be a supported <unit name>.
   b) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.
   c) If the unit specified by *aunit* is not supported by the implementation to compute the length of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.
   d) Case:
      i) If SELF is an empty set, then return the null value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) Otherwise, return the implementation-defined length of SELF, such that z (but not m) coordinate values are considered in the calculation, as measured in its spatial reference system.

e) The returned value is in the units indicated by *aunit*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.1.4 ST_StartPoint Method

Purpose

Return an ST_Point value that is the start point of an ST_Curve value including existing z and m coordinate values in the resultant geometry.

Definition

```
CREATE METHOD ST_StartPoint()
   RETURNS ST_Point
   FOR ST_Curve
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_StartPoint()* has no input parameters.
2) For the null-call method *ST_StartPoint()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return an *ST_Point* value that is the start point of SELF.
3) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate value is included in the resultant geometry.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.1.5 ST_EndPoint Method

Purpose

Return an ST_Point value that is the end point of an ST_Curve value including existing z and m coordinate values in the resultant geometry.

Definition

```
CREATE METHOD ST_EndPoint()
   RETURNS ST_Point
   FOR ST_Curve
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_EndPoint()* has no input parameters.
2) For the null-call method *ST_EndPoint()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise, return the *ST_Point* value that is the end point of SELF.
3) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate value is included in the resultant geometry.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.1.6 ST_IsClosed Method

Purpose

Test if an ST_Curve value is closed, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_IsClosed()
   RETURNS INTEGER
   FOR ST_Curve
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            0
         ELSE
            SELF.ST_Boundary().ST_IsEmpty()
      END
```

Description

1) The method *ST_IsClosed()* has no input parameters.
2) For the null-call method *ST_IsClosed()*:

   Case:

   a) If SELF is an empty set, then return 0 (zero).
   b) If the boundary of the *ST_Curve* value is the empty set, then 1 (one).
   c) Otherwise, 0 (zero).
3) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.1.7 ST_3DIsClosed Method

Purpose

Test if an ST_Curve value is closed, considering z coordinate values in the calculations.

Definition

```
CREATE METHOD ST_3DIsClosed()
   RETURNS INTEGER
   FOR ST_Curve
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            0
         ELSE
            SELF.ST_3DBoundary().ST_IsEmpty()
      END
```

Description

1) The method *ST_3DIsClosed()* has no input parameters.
2) For the null-call method *ST_3DIsClosed()*:

   Case:

   a) If SELF is an empty set, then return 0 (zero).
   b) If the boundary of the *ST_Curve* value is the empty set, then 1 (one).
   c) Otherwise, 0 (zero).
3) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are considered in the calculation.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.1.8 ST_IsRing Method

Purpose

Test if an ST_Curve value is a ring, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_IsRing()
   RETURNS INTEGER
   FOR ST_Curve
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            0
         WHEN (SELF.ST_IsSimple() = 1 AND SELF.ST_IsClosed() = 1) THEN
            1
         ELSE
            0
      END
```

Description

1) The method *ST_IsRing()* has no input parameters.
2) For the null-call method *ST_IsRing()*:

   Case:

   a) If SELF is an empty set, then return 0 (zero).

   b) If SELF is simple and SELF is closed, then return 1 (one).

   c) Otherwise 0 (zero).
3) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.1.9 ST_3DIsRing Method

Purpose

Test if an ST_Curve value is a ring, considering z coordinate values in the calculations.

Definition

```
CREATE METHOD ST_3DIsRing()
   RETURNS INTEGER
   FOR ST_Curve
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            0
         WHEN (SELF.ST_3DIsSimple() = 1 AND SELF.ST_3DIsClosed() = 1) THEN
            1
         ELSE
            0
      END
```

Description

1) The method *ST_3DIsRing()* has no input parameters.
2) For the null-call method *ST_3DIsRing()*:

   Case:

   a) If SELF is an empty set, then return 0 (zero).
   b) If SELF is simple and SELF is closed, then return 1 (one).
   c) Otherwise 0 (zero).
3) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are considered in the calculation.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.1.10 ST_CurveToLine Method

Purpose

Return the ST_LineString value approximation of an ST_Curve value, considering z coordinate values in the calculations and including existing z coordinate values in the resultant geometry.

Definition

```
CREATE METHOD ST_CurveToLine()
   RETURNS ST_LineString
   FOR ST_Curve
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_CurveToLine()* has no input parameters.
2) For the null-call method *ST_CurveToLine()*:

   Case:

   a) If SELF is an empty set, then return an empty set of type *ST_LineString*.

   b) Otherwise, return the implementation-defined *ST_LineString* value approximation of the *ST_Curve* value.
3) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are considered in the calculation and are included in the resultant geometry.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.2 ST_LineString Type and Routines

7.2.1 ST_LineString Type

Purpose

The ST_LineString type is a subtype of the ST_Curve type and represents a continuous locus of points from the start point to the end point with a linear interpolation between points.

Definition

```
CREATE TYPE ST_LineString
   UNDER ST_Curve
   AS (
      ST_PrivatePoints ST_Point
         ARRAY[ST_MaxGeometryArrayElements] DEFAULT ARRAY[]
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_LineString
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
      RETURNS ST_LineString
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_LineString
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
       ansrid INTEGER)
      RETURNS ST_LineString
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_LineString
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
      RETURNS ST_LineString
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_LineString
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
       ansrid INTEGER)
      RETURNS ST_LineString
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_LineString
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_LineString
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_LineString
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_LineString
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Points()
   RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements]
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Points
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_LineString
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_NumPoints()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_PointN
   (aposition INTEGER)
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_StartPoint()
   RETURNS ST_Point,

OVERRIDING METHOD ST_EndPoint()
   RETURNS ST_Point
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

4) The attribute *ST_PrivatePoints* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivatePoints*.

Description

1) The *ST_LineString* type provides for public use:
   a) a method *ST_LineString(CHARACTER LARGE OBJECT)*,
   b) a method *ST_LineString(CHARACTER LARGE OBJECT, INTEGER)*,
   c) a method *ST_LineString(BINARY LARGE OBJECT)*,
   d) a method *ST_LineString(BINARY LARGE OBJECT, INTEGER)*,
   e) a method *ST_LineString(ST_Point ARRAY)*,
   f) a method *ST_LineString(ST_Point ARRAY, INTEGER)*,
   g) a method *ST_Points()*,
   h) a method *ST_Points(ST_Point ARRAY)*,
   i) a method *ST_NumPoints()*,
   j) a method *ST_PointN(INTEGER)*,
   k) an overriding method *ST_StartPoint()*,
   l) an overriding method *ST_EndPoint()*,
   m) a function *ST_LineFromText(CHARACTER LARGE OBJECT)*,
   n) a function *ST_LineFromText(CHARACTER LARGE OBJECT, INTEGER)*,
   o) a function *ST_LineFromWKB(BINARY LARGE OBJECT)*,
   p) a function *ST_LineFromWKB(BINARY LARGE OBJECT, INTEGER)*,
   q) a function *ST_LineFromGML(CHARACTER LARGE OBJECT)*,
   r) a function *ST_LineFromGML(CHARACTER LARGE OBJECT, INTEGER)*.

2) The *ST_PrivatePoints* attribute contains the collection of *ST_Point* values.

3) The *ST_PrivatePoints* attribute shall not be the null value. The elements in the *ST_PrivatePoints* attribute shall not be the null value.

4) If the cardinality of the *ST_PrivatePoints* attribute is greater than or equal to two, then the *ST_LineString* value is well formed.

5) All the *ST_Point* values in the *ST_PrivatePoints* attribute shall be in the same spatial reference system as the *ST_LineString* value.

6) The coordinate dimension of an *ST_LineString* value is the number of coordinate values associated with the position.

7) The type *ST_LineString* is a subtype of *ST_Curve* with linear interpolation between points. Each consecutive pair of points defines a *line segment*.

8) If the cardinality of the *ST_PrivatePoints* attribute is two, then the *ST_LineString* value is called a *line*.

9) If an *ST_LineString* value is simple and closed, then it is called a *linear ring*.

10) An *ST_LineString* value returned by the constructor function corresponds to the empty set.

11) An *ST_LineString* value with the cardinality of the *ST_PrivatePoints* attribute equal to 0 (zero) corresponds to the empty set.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.2.2 ST_LineString Methods

Purpose

Return an ST_LineString value constructed from either the well-known text representation, the well-known binary representation, the GML representation, or the specified ST_Point values.

Definition

```
CREATE CONSTRUCTOR METHOD ST_LineString
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_LineString
   FOR ST_LineString
   RETURN NEW ST_LineString(awktorgml, 0)

CREATE CONSTRUCTOR METHOD ST_LineString
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_LineString
   FOR ST_LineString
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_LineString
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_LineString
   FOR ST_LineString
   RETURN NEW ST_LineString(awkb, 0)

CREATE CONSTRUCTOR METHOD ST_LineString
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_LineString
   FOR ST_LineString
   RETURN ST_LineFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_LineString
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_LineString
   FOR ST_LineString
   RETURN SELF.ST_SRID(0).ST_Points(apointarray)

CREATE CONSTRUCTOR METHOD ST_LineString
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_LineString
   FOR ST_LineString
   RETURN SELF.ST_SRID(ansrid).ST_Points(apointarray)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The method *ST_LineString(CHARACTER LARGE OBJECT)* takes the following input parameter:

   a) a CHARACTER LARGE OBJECT value *awktorgml*.

© ISO/IEC 2011 – All rights reserved

2) The null-call type preserving SQL-invoked constructor method *ST_LineString(CHARACTER LARGE OBJECT)* returns the result of the value expression: *NEW ST_LineString(awktorgml, 0)*.

3) The method *ST_LineString(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a CHARACTER LARGE OBJECT value *awktorgml*,

   b) an INTEGER value *ansrid*.

4) For the null-call type preserving SQL-invoked constructor method *ST_LineString(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If *awktorgml* contains a LineString XML element in the GML representation, then return the result of the value expression: *ST_LineFromGML(awktorgml, ansrid)*.

   b) Otherwise, return the result of the value expression: *ST_LineFromText(awktorgml, ansrid)*.

5) The method *ST_LineString(BINARY LARGE OBJECT)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *awkb*.

6) The null-call type preserving SQL-invoked constructor method *ST_LineString(BINARY LARGE OBJECT)* returns the result of the value expression: *NEW ST_LineString(awkb, 0)*.

7) The method *ST_LineString(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *awkb*,

   b) an INTEGER value *ansrid*.

8) The null-call type preserving SQL-invoked constructor method *ST_LineString(BINARY LARGE OBJECT, INTEGER)* returns the result of the value expression: *ST_LineFromWKB(awktorgml, ansrid)*.

9) The method *ST_LineString(ST_Point ARRAY)* takes the following input parameters:

   a) an *ST_Point* ARRAY value *apointarray*.

10) The null-call type preserving SQL-invoked constructor method *ST_LineString(ST_Point ARRAY)* returns an *ST_LineString* value with:

    a) The spatial reference system identifier set to 0 (zero).

    b) Using the method *ST_Points(ST_Point ARRAY),* the *ST_PrivatePoints* attribute set to *apointarray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

11) The method *ST_LineString(ST_Point ARRAY, INTEGER)* takes the following input parameters:

    a) an *ST_Point* ARRAY value *apointarray*,

    b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_LineString(ST_Point ARRAY, INTEGER)* returns an *ST_LineString* value with:

    a) The spatial reference system identifier set to *ansrid*.

    b) Using the method *ST_Points(ST_Point ARRAY),* the *ST_PrivatePoints* attribute set to *apointarray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.2.3 ST_Points Methods

Purpose

Observe and mutate the ST_PrivatePoints attribute of an ST_LineString value.

Definition

```
CREATE METHOD ST_Points()
   RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_LineString
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_PrivatePoints
      END

CREATE METHOD ST_Points
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_LineString
   FOR ST_LineString
   BEGIN
      -- If apointarray is the null value, contains null elements, or
      -- contains consecutive duplicate points, then raise an exception.
      CALL ST_CheckConsecDups(apointarray);
      -- If SELF is the null value, then return the null value.
      IF SELF IS NULL THEN
         RETURN CAST (NULL AS ST_LineString);
      END IF;
      -- Check that there are no mixed spatial reference
      -- systems between SELF and apointarray.
      IF (CARDINALITY(apointarray) > 0) AND
         (SELF.ST_SRID() <> ST_CheckSRID(apointarray)) THEN
         SIGNAL SQLSTATE '2FF10'
            SET MESSAGE_TEXT = 'mixed spatial reference systems';
      END IF;
      RETURN
         SELF.ST_PrivateDimension(1).
            ST_PrivateCoordinateDimension(ST_GetCoordDim(apointarray)).
            ST_PrivateIs3D(ST_GetIs3D(apointarray)).
            ST_PrivateIsMeasured(ST_GetIsMeasured(apointarray)).
            ST_PrivatePoints(apointarray);
   END
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

Description

1) The method *ST_Points()* has no input parameters.
2) For the null-call method *ST_Points()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the *ST_PrivatePoints* attribute of SELF.
3) The method *ST_Points(ST_Point ARRAY)* takes the following input parameters:

   a) an *ST_Point* ARRAY value *apointarray*.
4) For the type preserving method *ST_Points(ST_Point ARRAY)*:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

a) Call the procedure *ST_CheckConsecDups(ST_Geometry ARRAY)* to check if *apointarray* is the null value, contains null elements, or contains consecutive duplicate points.

b) Case:

   i) If SELF is the null value, then return the null value.

   ii) If the cardinality of *apointarray* is greater than 0 (zero) and the spatial reference system of SELF is not equal to *ST_CheckSRID(apointarray)*, then an exception condition is raised: *SQL/MM Spatial exception – mixed spatial reference systems*.

   iii) Otherwise, return an *ST_LineString* value with:

      1) The dimension set to 1 (one).

      2) The coordinate dimension set to the value expression: *ST_GetCoordDim(apointarray)*.

      3) The *ST_PrivateIs3D* attribute set to the value expression: *ST_GetIs3D(apointarray)*.

      4) The *ST_PrivateIsMeasured* attribute set to the value expression: *ST_GetIsMeasured(apointarray)*.

      5) The *ST_PrivatePoints* attribute set to *apointarray*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.2.4 ST_NumPoints Method

Purpose

Return the cardinality of the ST_PrivatePoints attribute of an ST_LineString value.

Definition

```
CREATE METHOD ST_NumPoints()
   RETURNS INTEGER
   FOR ST_LineString
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CARDINALITY(SELF.ST_PrivatePoints)
      END
```

Description

1) The method *ST_NumPoints()* has no input parameters.
2) For the null-call method *ST_NumPoints()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise, return the cardinality of the *ST_PrivatePoints* attribute.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.2.5 ST_PointN Method

Purpose

Return the specified element in the ST_PrivatePoints attribute of an ST_LineString value.

Definition

```
CREATE METHOD ST_PointN
   (aposition INTEGER)
   RETURNS ST_Point
   FOR ST_LineString
   BEGIN
      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS ST_Point);
      END IF;
      IF aposition < 1 OR
         aposition > SELF.ST_NumPoints() THEN
         BEGIN
            SIGNAL SQLSTATE '01F01'
               SET MESSAGE_TEXT = 'invalid position';
            RETURN CAST (NULL AS ST_Point);
         END;
      END IF;
      RETURN SELF.ST_PrivatePoints[aposition];
   END
```

Description

1) The method *ST_PointN(INTEGER)* takes the following input parameters:
   a) an INTEGER value *aposition*.
2) For the null-call method *ST_PointN(INTEGER)*:

   Case:
   a) If SELF is an empty set, then return the null value.
   b) If *aposition* is less than 1 (one) or greater than the cardinality of the *ST_PrivatePoints* attribute, then:
      i) A completion condition is raised: *SQL/MM Spatial warning – invalid position*.
      ii) Return the null value.
   c) Otherwise, return an *ST_Point* value at element *aposition* in the *ST_PrivatePoints* attribute of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.2.6 ST_StartPoint Method

Purpose

Return the start point of an ST_LineString value.

Definition

```
CREATE METHOD ST_StartPoint()
   RETURNS ST_Point
   FOR ST_LineString
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_Points()[1]
      END
```

Description

1) The method *ST_StartPoint()* has no input parameters.
2) For the null-call method *ST_StartPoint()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise, return the result of the value expression: *SELF.ST_Points()[1]*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.2.7 ST_EndPoint Method

Purpose

Return the end point of an ST_LineString value.

Definition

```
CREATE METHOD ST_EndPoint()
   RETURNS ST_Point
   FOR ST_LineString
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_Points()[SELF.ST_NumPoints()]
      END
```

Description

1) The method *ST_EndPoint()* has no input parameters.
2) For the null-call method *ST_EndPoint()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise, return the result of the value expression: *SELF.ST_Points()[SELF.ST_NumPoints()]*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.2.8 ST_LineFromText Functions

Purpose

Return an ST_LineString value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_LineString value.

Definition

```
CREATE FUNCTION ST_LineFromText
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
    RETURNS ST_LineString
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    RETURN ST_LineFromText(awkt, 0)

CREATE FUNCTION ST_LineFromText
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
     ansrid INTEGER)
    RETURNS ST_LineString
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    BEGIN
       --
       -- See Description
       --
    END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_LineFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_LineFromText(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_LineFromText*(*awkt*, 0).
3) The function *ST_LineFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_LineFromText(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkt* is the well-known text representation of an *ST_LineString* value.

      If *awkt* is not producible in the BNF for <linestring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromText(awkt, ansrid) AS ST_LineString)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.2.9 ST_LineFromWKB Functions

Purpose

Return an ST_LineString value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_LineString value.

Definition

```
CREATE FUNCTION ST_LineFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_LineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_LineFromWKB(awkb,0)

CREATE FUNCTION ST_LineFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_LineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_LineFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_LineFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_LineFromWKB(awkb, 0)*.
3) The function *ST_LineFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_LineFromWKB(BINARY LARGE OBJECT, INTEGER)*:
   a) The parameter *awkb* is the well-known binary representation of an *ST_LineString* value.

      If *awkb* is not producible in the BNF for <linestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_LineString)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.2.10 ST_LineFromGML Functions

Purpose

Return an ST_LineString value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representationof an ST_LineString value.

Definition

```
CREATE FUNCTION ST_LineFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
   RETURNS ST_LineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_LineFromGML(agml, 0)

CREATE FUNCTION ST_LineFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
    ansrid INTEGER)
   RETURNS ST_LineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_LineFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_LineFromGML(CHARACTER LARGE OBJECT)* returns the result of the value expression *ST_LineFromGML(agml, 0)*.
3) The function *ST_LineFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_LineFromGML(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If the parameter *agml* does not contain a LineString XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromGML(agml, ansrid) AS ST_LineString)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.3 ST_CircularString Type and Routines

7.3.1 ST_CircularString Type

Purpose

The ST_CircularString type is a subtype of the ST_Curve type and represents a continuous locus of points from the start point to the end point with a circular interpolation between points.

Definition

```
CREATE TYPE ST_CircularString
   UNDER ST_Curve
   AS (
      ST_PrivatePoints ST_Point
         ARRAY[ST_MaxGeometryArrayElements] DEFAULT ARRAY[]
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_CircularString
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
      RETURNS ST_CircularString
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_CircularString
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
       ansrid INTEGER)
      RETURNS ST_CircularString
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_CircularString
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
      RETURNS ST_CircularString
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_CircularString
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
       ansrid INTEGER)
      RETURNS ST_CircularString
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_CircularString
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_CircularString
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CircularString
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_CircularString
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Points()
   RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements]
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Points
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_CircularString
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_NumPoints()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_PointN
   (aposition INTEGER)
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_MidPointRep()
   RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements]
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_StartPoint()
   RETURNS ST_Point,
OVERRIDING METHOD ST_EndPoint()
   RETURNS ST_Point
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.
4) The attribute *ST_PrivatePoints* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivatePoints*.

Description

1) The *ST_CircularString* type provides for public use:
   a) a method *ST_CircularString(CHARACTER LARGE OBJECT)*,
   b) a method *ST_CircularString(CHARACTER LARGE OBJECT, INTEGER)*,
   c) a method *ST_CircularString(BINARY LARGE OBJECT)*,
   d) a method *ST_CircularString(BINARY LARGE OBJECT, INTEGER)*,
   e) a method *ST_CircularString(ST_Point ARRAY)*,
   f) a method *ST_CircularString(ST_Point ARRAY, INTEGER)*,
   g) a method *ST_Points()*,
   h) a method *ST_Points(ST_Point ARRAY)*,
   i) a method *ST_NumPoints()*,
   j) a method *ST_PointN(INTEGER)*,
   k) a method *ST_MidPointRep()*,
   l) an overriding method *ST_StartPoint()*,
   m) an overriding method *ST_EndPoint()*,
   n) a function *ST_CircularFromTxt(CHARACTER LARGE OBJECT)*,
   o) a function *ST_CircularFromTxt(CHARACTER LARGE OBJECT, INTEGER)*,
   p) a function *ST_CircularFromWKB(BINARY LARGE OBJECT)*,
   q) a function *ST_CircularFromWKB(BINARY LARGE OBJECT, INTEGER)*,
   r) a function *ST_CircularFromGML(CHARACTER LARGE OBJECT)*,
   s) a function *ST_CircularFromGML(CHARACTER LARGE OBJECT, INTEGER)*.
2) The *ST_PrivatePoints* attribute contains the collection of *ST_Point* values.
3) The *ST_PrivatePoints* attribute shall not be the null value. The elements in the *ST_PrivatePoints* attribute shall not be the null value.
4) All the *ST_Point* values in the *ST_PrivatePoints* attribute shall be in the same spatial reference system as the *ST_CircularString* value.
5) The coordinate dimension of an *ST_CircularString* value is the number of coordinate values associated with the position.
6) An *ST_CircularString* value consists of one or more circular arc segments connected end to end. The first segment is defined by three points. The first point is the start point of the arc segment. The second point is any intermediate point on the arc segment other than the start or end point. The third point is the end point of the arc segment. Subsequent segments are defined by their intermediate and end points only, as the start point is implicitly defined as the previous segment's end point. In the special case where a segment is a complete circle, that is, the start and end points are coincident, then the intermediate point shall be the midpoint of the segment.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7) Let *NSEG* be the number of circular arc segments in the *ST_CircularString* value. If *SELF.NumPoints* is equal to 2 * *NSEG* + 1, then the *ST_CircularString* value is well formed.

8) A circular arc segment is the locus of points defined as follows:

   Case:

   a) If the start, intermediate, and end points of an arc segment are not collinear, then the circular arc segment is the locus of points a distance *R* from the center of the arc, beginning at the start point, passing through the intermediate point, and ending at the end point of the circular arc segment. The distance *R* is the radius of the circular arc segment, and is equal to the distance from the center of the circular arc segment and the start, intermediate, or end points. The center of the circular arc segment is defined as follows:

      Case:

      i) If the segment is a complete circle, then the center is located at the midpoint of the line connecting the start point and the intermediate point.

      ii) Otherwise, let *CHORD1* be the line connecting the start point of a circular arc segment and the intermediate point on the segment. Let *CHORD2* be the line connecting the intermediate point with the end point of this arc segment. Then the center of the circular arc segment is located at the intersection of the perpendicular bisectors of *CHORD1* and *CHORD2*.

   b) If the start, intermediate, and end points of an arc segment are collinear, then the resultant arc segment degenerates to a straight line for which center and radius are not defined. In this case, the circular arc segment is the locus of points defined by the straight line connecting the start and end points.

9) If the cardinality of the attribute *ST_PrivatePoints* is three, then the *ST_CircularString* value is considered a *circular arc*.

10) If an *ST_CircularString* value is simple and closed, then it is considered a *circular ring*.

11) An *ST_CircularString* value returned by the constructor function corresponds to the empty set.

12) An *ST_CircularString* value with the cardinality of the *ST_PrivatePoints* attribute equal to 0 (zero) corresponds to the empty set.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.3.2 ST_CircularString Methods

Purpose

Return an ST_CircularString value constructed from either the well-known text representation, the well-known binary representation, the GML representation, or the specified ST_Point values.

Definition

```
CREATE CONSTRUCTOR METHOD ST_CircularString
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_CircularString
   FOR ST_CircularString
   RETURN NEW ST_CircularString(awktorgml, 0)

CREATE CONSTRUCTOR METHOD ST_CircularString
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_CircularString
   FOR ST_CircularString
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_CircularString
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_CircularString
   FOR ST_CircularString
   RETURN NEW ST_CircularString(awkb, 0)

CREATE CONSTRUCTOR METHOD ST_CircularString
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_CircularString
   FOR ST_CircularString
   RETURN ST_CircularFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_CircularString
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_CircularString
   FOR ST_CircularString
   RETURN SELF.ST_SRID(0).ST_Points(apointarray)

CREATE CONSTRUCTOR METHOD ST_CircularString
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_CircularString
   FOR ST_CircularString
   RETURN SELF.ST_SRID(ansrid).ST_Points(apointarray)
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The method *ST_CircularString(CHARACTER LARGE OBJECT)* takes the following input parameter:

   a) a CHARACTER LARGE OBJECT value *awktorgml*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) The null-call type preserving SQL-invoked constructor method *ST_CircularString(CHARACTER LARGE OBJECT)* returns the result of the value expression: *NEW ST_CircularString(awktorgml, 0)*.

3) The method *ST_CircularString(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a CHARACTER LARGE OBJECT value *awktorgml*,

   b) an INTEGER value *ansrid*.

4) For the null-call type preserving SQL-invoked constructor method *ST_CircularString(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If *awktorgml* contains a Circle XML element in the GML representation or ArcString XML element in the GML representation, then return the result of the value expression: *ST_CircularFromGML(awktorgml, ansrid)*.

   b) Otherwise, return the result of the value expression: *ST_CircularFromTxt(awktorgml, ansrid)*.

5) The method *ST_CircularString(BINARY LARGE OBJECT)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *awkb*.

6) The null-call type preserving SQL-invoked constructor method *ST_CircularString(BINARY LARGE OBJECT)* returns the result of the value expression: *NEW ST_CircularString(awkb, 0)*.

7) The method *ST_CircularString(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *awkb*,

   b) an INTEGER value *ansrid*.

8) The null-call type preserving SQL-invoked constructor method *ST_CircularString(BINARY LARGE OBJECT, INTEGER)* returns the result of the value expression: *ST_CircularFromWKB(awkb, ansrid)*.

9) The method *ST_CircularString(ST_Point ARRAY)* takes the following input parameters:

   a) an *ST_Point* ARRAY value *apointarray*

10) The null-call type preserving SQL-invoked constructor method *ST_CircularString(ST_Point ARRAY)* returns an *ST_CircularString* value with:

   a) The spatial reference system identifier set to 0 (zero).

   b) Using the method *ST_Points(ST_Point ARRAY),* the attribute *ST_PrivatePoints* array set to *apointarray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

11) The method *ST_CircularString(ST_Point ARRAY, INTEGER)* takes the following input parameters:

   a) an *ST_Point* ARRAY value *apointarray*,

   b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_CircularString(ST_Point ARRAY, INTEGER)* returns an *ST_CircularString* value with:

   a) The spatial reference system identifier set to *ansrid*.

   b) Using the method *ST_Points(ST_Point ARRAY),* the attribute *ST_PrivatePoints* array set to *apointarray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.3.3 ST_Points Methods

Purpose

Observe and mutate the attribute ST_PrivatePoints of an ST_CircularString value.

Definition

```
CREATE METHOD ST_Points()
   RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_CircularString
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_PrivatePoints
      END

CREATE METHOD ST_Points
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_CircularString
   FOR ST_CircularString
   BEGIN
      -- If apointarray is the null value, contains null elements, or
      -- contains consecutive duplicate points, then raise an exception.
      CALL ST_CheckConsecDups(apointarray);
      -- If SELF is the null value, then return the null value.
      IF SELF IS NULL THEN
         RETURN CAST (NULL AS ST_CircularString);
      END IF;
      -- Check that there are no mixed spatial reference
      -- systems between SELF and apointarray.
      IF (CARDINALITY(apointarray) > 0) AND
         (SELF.ST_SRID() <> ST_CheckSRID(apointarray)) THEN
         SIGNAL SQLSTATE '2FF10'
            SET MESSAGE_TEXT = 'mixed spatial reference systems';
      END IF;
      RETURN
         SELF.ST_PrivateDimension(1).
            ST_PrivateCoordinateDimension(ST_GetCoordDim(apointarray)).
            ST_PrivateIs3D(ST_GetIs3D(apointarray)).
            ST_PrivateIsMeasured(ST_GetIsMeasured(apointarray)).
            ST_PrivatePoints(apointarray);
   END
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

Description

1) The method *ST_Points()* has no input parameters.
2) For the null-call method *ST_Points()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the attribute *ST_PrivatePoints* of SELF.
3) The method *ST_Points(ST_Point ARRAY)* takes the following input parameters:

   a) an *ST_Point ARRAY* value *apointarray*.
4) For the type preserving method *ST_Points(ST_Point ARRAY)*:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

a) Call the procedure *ST_CheckConsecDups(ST_Geometry ARRAY)* to check if *apointarray* is the null value or contains null elements.

b) Case:

   i) If SELF is the null value, then return the null value.

   ii) If the cardinality of *apointarray* is greater than 0 (zero) and the spatial reference system of SELF is not equal to *ST_CheckSRID*(*apointarray*), then an exception condition is raised: *SQL/MM Spatial exception – mixed spatial reference systems*.

   iii) Otherwise, return an *ST_CircularString* value with:

      1) the dimension set to 1 (one).

      2) The coordinate dimension set to the value expression: *ST_GetCoordDim(apointarray)*.

      3) The *ST_PrivateIs3D* attribute set to the value expression: *ST_GetIs3D(apointarray)*.

      4) The *ST_PrivateIsMeasured* attribute set to the value expression: *ST_GetIsMeasured(apointarray)*.

      5) the attribute *ST_PrivatePoints* set to *apointarray*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.3.4 ST_NumPoints Method

Purpose

Return the cardinality of the ST_PrivatePoints attribute of an ST_CircularString value.

Definition

```
CREATE METHOD ST_NumPoints()
   RETURNS INTEGER
   FOR ST_CircularString
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CARDINALITY(SELF.ST_PrivatePoints)
      END
```

Description

1) The method *ST_NumPoints()* has no input parameters.
2) For the null-call method *ST_NumPoints()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise, return the cardinality of the *ST_PrivatePoints* attribute.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.3.5 ST_PointN Method

Purpose

Return the specified element in the ST_PrivatePoints attribute of an ST_CircularString value.

Definition

```
CREATE METHOD ST_PointN
   (aposition INTEGER)
   RETURNS ST_Point
   FOR ST_CircularString
   BEGIN
      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS ST_Point);
      END IF;
      IF aposition < 1 OR
         aposition > SELF.ST_NumPoints() THEN
         BEGIN
            SIGNAL SQLSTATE '01F01'
               SET MESSAGE_TEXT = 'invalid position';
            RETURN CAST (NULL AS ST_Point);
         END;
      END IF;
      RETURN SELF.ST_PrivatePoints[aposition];
   END
```

Description

1) The method *ST_PointN(INTEGER)* takes the following input parameters:
   a) an INTEGER value *aposition*.
2) For the null-call method *ST_PointN(INTEGER)*:

   Case:
   a) If SELF is an empty set, then return the null value.
   b) If *aposition* is less than 1 (one) or greater than the cardinality of the attribute *ST_PrivatePoints*, then:
      i) A completion condition is raised: *SQL/MM Spatial warning – invalid position*.
      ii) Return the null value.
   c) Otherwise, return an *ST_Point* value at element *aposition* in the attribute *ST_PrivatePoints* of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.3.6 ST_MidPointRep Method

Purpose

Return an ST_Point ARRAY which uniquely identifies an ST_CircularString value, including the start, mid, and end points of each curve segment.

Definition

```
CREATE METHOD ST_MidPointRep()
   RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_CircularString
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

Description

1) The method *ST_MidPointRep()* has no input parameters.
2) For the null-call method *ST_MidPointRep()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return an *ST_Points ARRAY* such that:

      i) For the first circular arc segment of the curve, the points returned are the start, mid, and end points of the segment.

      ii) For all subsequent segments, the points returned are the mid and end points of the respective segment.
3) If *SELF.ST_Is3D()* is equal to 1 (one), then:

   a) The z coordinate values are not considered in the calculation.

   b) The *ST_Point* values do not include the z coordinate value.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one), then:

   a) The m coordinate values are not considered in the calculation.

   b) The *ST_Point* values do not include the m coordinate value.
5) The spatial reference system identifier of the returned *ST_Point* values is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.3.7 ST_StartPoint Method

Purpose

Return the start point of an ST_CircularString value.

Definition

```
CREATE METHOD ST_StartPoint()
   RETURNS ST_Point
   FOR ST_CircularString
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_Points()[1]
      END
```

Description

1) The method *ST_StartPoint()* has no input parameters.
2) For the null-call method *ST_StartPoint()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise, return the result of the value expression: *SELF.ST_Points()[1]*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.3.8 ST_EndPoint Method

Purpose

Return the end point of an ST_CircularString value.

Definition

```
CREATE METHOD ST_EndPoint()
   RETURNS ST_Point
   FOR ST_CircularString
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_Points()[SELF.ST_NumPoints()]
      END
```

Description

1) The method *ST_EndPoint()* has no input parameters.
2) For the null-call method *ST_EndPoint()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise, return the result of the value expression: *SELF.ST_Points()[SELF.ST_NumPoints()]*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.3.9 ST_CircularFromTxt Functions

Purpose

Return an ST_CircularString value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_CircularString value.

Definition

```
CREATE FUNCTION ST_CircularFromTxt
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_CircularString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_CircularFromTxt(awkt, 0)

CREATE FUNCTION ST_CircularFromTxt
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_CircularString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_CircularFromTxt(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_CircularFromTxt(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_CircularFromTxt(awkt, 0)*.
3) The function *ST_CircularFromTxt(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_CircularFromTxt(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkt* is the well-known text representation of an *ST_CircularString* value.

      If *awkt* is not producible in the BNF for <circularstring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromText(awkt, ansrid) AS ST_CircularString)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.3.10 ST_CircularFromWKB Functions

Purpose

Return an ST_CircularString value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_CircularString value.

Definition

```
CREATE FUNCTION ST_CircularFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_CircularString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_CircularFromWKB(awkb, 0)

CREATE FUNCTION ST_CircularFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_CircularString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_CircularFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_CircularFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_CircularFromWKB(awkb, 0)*.
3) The function *ST_CircularFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_CircularFromWKB(BINARY LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkb* is the well-known binary representation of an *ST_CircularString* value.

      If *awkb* is not producible in the BNF for <circularstring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_CircularString)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.3.11 ST_CircularFromGML Functions

Purpose

Return an ST_CircularString value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representationof an ST_CircularString value.

Definition

```
CREATE FUNCTION ST_CircularFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
   RETURNS ST_CircularString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_CircularFromGML(agml, 0)

CREATE FUNCTION ST_CircularFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
    ansrid INTEGER)
   RETURNS ST_CircularString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_CircularFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_CircularFromGML(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_CircularFromGML(agml, 0)*.
3) The function *ST_CircularFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_CircularFromGML(CHARACTER LARGE OBJECT, INTEGER)*:
   a) If the parameter *agml* does not contain a Circle XML element in the GML representation or ArcString XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromGML(agml, ansrid) AS ST_CircularString)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.4 ST_CompoundCurve Type and Routines

7.4.1 ST_CompoundCurve Type

Purpose

The general notion of a compound curve is a sequence of contiguous curves such that adjacent curves are joined at their end points. The contributing curve types are limited to ST_LineString and ST_CircularString values. Furthermore, the end point of each curve shall be coincident with the start point of the next curve in the list.

Definition

```
CREATE TYPE ST_CompoundCurve
   UNDER ST_Curve
   AS (
      ST_PrivateCurves ST_Curve
         ARRAY[ST_MaxGeometryArrayElements] DEFAULT ARRAY[]
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_CompoundCurve
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
      RETURNS ST_CompoundCurve
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_CompoundCurve
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
       ansrid INTEGER)
      RETURNS ST_CompoundCurve
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_CompoundCurve
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
      RETURNS ST_CompoundCurve
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_CompoundCurve
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
       ansrid INTEGER)
      RETURNS ST_CompoundCurve
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_CompoundCurve(acurve ST_Curve)
   RETURNS ST_CompoundCurve
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CompoundCurve
   (acurve ST_Curve,
    ansrid INTEGER)
   RETURNS ST_CompoundCurve
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CompoundCurve
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_CompoundCurve
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CompoundCurve
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_CompoundCurve
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Curves()
   RETURNS ST_Curve ARRAY[ST_MaxGeometryArrayElements]
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Curves
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_CompoundCurve
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_NumCurves()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_CurveN
   (aposition INTEGER)
   RETURNS ST_Curve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_StartPoint()
   RETURNS ST_Point,

OVERRIDING METHOD ST_EndPoint()
   RETURNS ST_Point
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.
4) The attribute *ST_PrivateCurves* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivateCurves*.

Description

1) The *ST_CompoundCurve* type provides for public use:
   a) a method *ST_CompoundCurve(CHARACTER LARGE OBJECT)*,
   b) a method *ST_CompoundCurve(CHARACTER LARGE OBJECT, INTEGER)*,
   c) a method *ST_CompoundCurve(BINARY LARGE OBJECT)*,
   d) a method *ST_CompoundCurve(BINARY LARGE OBJECT, INTEGER)*,
   e) a method *ST_CompoundCurve(ST_Curve)*,
   f) a method *ST_CompoundCurve(ST_Curve, INTEGER)*,
   g) a method *ST_CompoundCurve(ST_Curve ARRAY)*,
   h) a method *ST_CompoundCurve(ST_Curve ARRAY, INTEGER)*,
   i) a method *ST_Curves()*,
   j) a method *ST_Curves(ST_Curve ARRAY)*,
   k) a method *ST_NumCurves()*,
   l) a method *ST_CurveN(INTEGER)*,
   m) an overriding method *ST_StartPoint()*,
   n) an overriding method *ST_EndPoint()*,
   o) a function *ST_CompoundFromTxt(CHARACTER LARGE OBJECT)*,
   p) a function *ST_CompoundFromTxt(CHARACTER LARGE OBJECT, INTEGER)*,
   q) a function *ST_CompoundFromWKB(BINARY LARGE OBJECT)*,
   r) a function *ST_CompoundFromWKB(BINARY LARGE OBJECT, INTEGER)*,
   s) a function *ST_CompoundFromGML(CHARACTER LARGE OBJECT)*,
   t) a function *ST_CompoundFromGML(CHARACTER LARGE OBJECT, INTEGER)*.
2) The *ST_PrivateCurves* attribute contains a collection of *ST_Curve* values.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3) If each *ST_Curve* value in the *ST_PrivateCurves* attribute is well formed, then the *ST_CompoundCurve* value is well formed.

4) All the *ST_Curve* values in the *ST_PrivateCurves* attribute are in the same spatial reference system as the *ST_CompoundCurve* value.

5) The *ST_PrivateCurves* attribute shall not be the null value. The elements in the *ST_PrivateCurves* attribute shall not be the null value.

6) The coordinate dimension of an *ST_CompoundCurve* value is the number of coordinate values associated with the position.

7) An *ST_CompoundCurve* value consists of one or more curves connected end to end. The contributing curve types are limited to *ST_LineString* and *ST_CircularString* values. Furthermore, the end point of each curve shall be coincident with the start point of the next curve in the list.

8) If an *ST_CompoundCurve* value is simple and closed, then it is considered a ring.

9) An *ST_CompoundCurve* value returned by the constructor function corresponds to the empty set.

10) An *ST_CompoundCurve* value with the cardinality of the attribute *ST_PrivateCurves* equal to 0 (zero) corresponds to the empty set.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.4.2 ST_CompoundCurve Methods

Purpose

Return an ST_CompoundCurve value constructed from either the well-known text representation, the well-known binary representation, the GML representation, or the specified ST_Curve values.

Definition

```
CREATE CONSTRUCTOR METHOD ST_CompoundCurve
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_CompoundCurve
   FOR ST_CompoundCurve
   RETURN NEW ST_CompoundCurve(awktorgml, 0)

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_CompoundCurve
   FOR ST_CompoundCurve
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_CompoundCurve
   FOR ST_CompoundCurve
   RETURN NEW ST_CompoundCurve(awkb, 0)

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_CompoundCurve
   FOR ST_CompoundCurve
   RETURN ST_CompoundFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
   (acurve ST_Curve)
   RETURNS ST_CompoundCurve
   FOR ST_CompoundCurve
   RETURN SELF.ST_SRID(0).ST_Curves(ARRAY[acurve])

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
   (acurve ST_Curve,
    ansrid INTEGER)
   RETURNS ST_CompoundCurve
   FOR ST_CompoundCurve
   RETURN SELF.ST_SRID(ansrid).ST_Curves(ARRAY[acurve])

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_CompoundCurve
   FOR ST_CompoundCurve
   RETURN SELF.ST_SRID(0).ST_Curves(acurvearray)

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_CompoundCurve
   FOR ST_CompoundCurve
   RETURN SELF.ST_SRID(ansrid).ST_Curves(acurvearray)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The method *ST_CompoundCurve(CHARACTER LARGE OBJECT)* takes the following input parameter:
   a) a CHARACTER LARGE OBJECT value *awktorgml*.
2) The null-call type preserving SQL-invoked constructor method *ST_CompoundCurve(CHARACTER LARGE OBJECT)* returns the result of the value expression: *NEW ST_CompoundCurve(awktorgml, 0)*.
3) The method *ST_CompoundCurve(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awktorgml*,
   b) an INTEGER value *ansrid*.
4) For the null-call type preserving SQL-invoked constructor method *ST_CompoundCurve(CHARACTER LARGE OBJECT, INTEGER)*:

   Case;

   a) If *awktorgml* contains a CompositeCurve XML element in the GML representation, then return the result of the value expression: *ST_CompoundFromGML(awktorgml, ansrid)*.
   b) Otherwise, return the result of the value expression: *ST_CompoundFromTxt(awktorgml, ansrid)*.
5) The method *ST_CompoundCurve(BINARY LARGE OBJECT)* takes the following input parameter:
   a) a BINARY LARGE OBJECT value *awkb*.
6) The null-call type preserving SQL-invoked constructor method *ST_CompoundCurve(BINARY LARGE OBJECT)* returns the result of the value expression: *NEW ST_CompoundCurve(awkb, 0)*.
7) The method *ST_CompoundCurve(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
8) The null-call type preserving SQL-invoked constructor method *ST_CompoundCurve(BINARY LARGE OBJECT, INTEGER)* returns the result of the value expression: *ST_CompoundFromWKB(awkb, ansrid)*.
9) The method *ST_CompoundCurve(ST_Curve)* takes the following input parameters:
   b) an *ST_Curve* value *acurve*.
10) The null-call type preserving SQL-invoked constructor method *ST_CompoundCurve(ST_Curve)* returns an *ST_CompoundCurve* value with:
   a) The spatial reference system identifier set to 0 (zero).
   b) Using the method *ST_Curves(ST_Curve ARRAY),* the *ST_PrivateCurves* attribute set to ARRAY[*acurve*], the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.
11) The method *ST_CompoundCurve(ST_Curve, INTEGER)* takes the following input parameters:
   a) an *ST_Curve* value *acurve*,
   b) an INTEGER value *ansrid*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

12) The null-call type preserving SQL-invoked constructor method *ST_CompoundCurve(ST_Curve, INTEGER)* returns an *ST_CompoundCurve* value with:

    a) The spatial reference system identifier set to *ansrid*.

    b) Using the method *ST_Curves(ST_Curve ARRAY)*, the *ST_PrivateCurves* attribute set to ARRAY[*acurve*], the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

13) The method *ST_CompoundCurve(ST_Curve ARRAY)* takes the following input parameters:

    a) an *ST_Curve* ARRAY value *acurvearray*.

14) The null-call type preserving SQL-invoked constructor method *ST_CompoundCurve(ST_Curve ARRAY)* returns an *ST_CompoundCurve* value with:

    a) The spatial reference system identifier set to 0 (zero).

    b) Using the method *ST_Curves(ST_Curve ARRAY)*, the *ST_PrivateCurves* attribute set to *acurvearray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

15) The method *ST_CompoundCurve(ST_Curve ARRAY, INTEGER)* takes the following input parameters:

    a) an *ST_Curve* ARRAY value *acurvearray*,

    b) an INTEGER value *ansrid*.

16) The null-call type preserving SQL-invoked constructor method *ST_CompoundCurve(ST_Curve ARRAY, INTEGER)* returns an *ST_CompoundCurve* value with:

    a) The spatial reference system identifier set to *ansrid*.

    b) Using the method *ST_Curves(ST_Curve ARRAY)*, the *ST_PrivateCurves* attribute set to *acurvearray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.4.3 ST_Curves Methods

Purpose

Observe and mutate the ST_PrivateCurves attribute of an ST_CompoundCurve value.

Definition

```
CREATE METHOD ST_Curves()
   RETURNS ST_Curve ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_CompoundCurve
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_PrivateCurves
      END

CREATE METHOD ST_Curves
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_CompoundCurve
   FOR ST_CompoundCurve
   BEGIN
      DECLARE counter INTEGER;

      -- If acurvearray is the null value or contains null elements,
      -- then raise an exception.
      CALL ST_CheckNulls(acurvearray);
      -- If SELF is the null value, then return the null value.
      IF SELF IS NULL THEN
         RETURN CAST (NULL AS ST_CompoundCurve);
      END IF;
      -- Check that there are no mixed spatial reference
      -- systems between SELF and acurvearray.
      IF (CARDINALITY(acurvearray) > 0) AND
         (SELF.ST_SRID() <> ST_CheckSRID(acurvearray)) THEN
         SIGNAL SQLSTATE '2FF10'
            SET MESSAGE_TEXT = 'mixed spatial reference systems';
      END IF;
      -- If any ST_Curve value in acurvearray is an ST_CompoundCurve
      -- value, then raise an exception.
      SET counter = 1;
      WHILE counter <= CARDINALITY(acurvearray) DO
            IF acurvearray[acounter] IS OF (ST_CompoundCurve) THEN
            SIGNAL SQLSTATE '2FF02'
               SET MESSAGE_TEXT = 'invalid argument';
         END IF;
         SET counter = counter + 1;
      END WHILE;
      -- If the start point of any curve is not coincident with the end
      -- point of the previous curve, then raise an exception
      SET counter = 2;
      WHILE counter <= CARDINALITY(acurvearray) DO
            IF acurvearray[counter].ST_StartPoint() <>
               acurvearray[counter-1].ST_EndPoint() THEN
            SIGNAL SQLSTATE '2FF11'
               SET MESSAGE_TEXT = 'non-contiguous curves';
         END IF;
         SET counter = counter + 1;
      END WHILE;
      -- If SELF is the null value, then return the null value. Otherwise,
      -- return an ST_CompoundCurve value with the ST_PrivateCurves
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
        -- attribute set to acurvearray.
        RETURN
           SELF.ST_PrivateDimension(1).
              ST_PrivateCoordinateDimension(ST_GetCoordDim(acurvearray)).
              ST_PrivateIs3D(ST_GetIs3D(acurvearray)).
              ST_PrivateIsMeasured(ST_GetIsMeasured(acurvearray)).
              ST_PrivateCurves(acurvearray);
    END
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

Description

1) The method *ST_Curves()* has no input parameters.
2) For the null-call method *ST_Curves()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the *ST_PrivateCurves* attribute of SELF.
3) The method *ST_Curves(ST_Curve ARRAY)* takes the following input parameters:

   a) an *ST_Curve* ARRAY value *acurvearray*.
4) For the type preserving method *ST_Curves(ST_Curve ARRAY)*:

   a) Call the procedure *ST_CheckNulls(ST_Geometry ARRAY)* to check if *acurvearray* is the null value or contains null elements.

   b) Case:

      i) If SELF is the null value, then return the null value.

      ii) If the cardinality of *acurvearray* is greater than 0 (zero) and the spatial reference system of SELF is not equal to *ST_CheckSRID(acurvearray)*, then an exception condition is raised: *SQL/MM Spatial exception – mixed spatial reference systems*.

      iii) If any *ST_Curve* value in *acurvearray* is an *ST_CompoundCurve* value, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

      iv) If the start point of any *ST_Curve* value in *acurvearray* is not equal to the end point of the previous *ST_Curve* value in *acurvearray*, then an exception condition is raised: *SQL/MM Spatial exception – non-contiguous curves*.

      v) Otherwise, return an *ST_CompoundCurve* value with:

         1) The dimension set to 1 (one).

         2) The coordinate dimension set to the value expression: *ST_GetCoordDim(acurvearray)*.

         3) The *ST_PrivateIs3D* attribute set to the value expression: *ST_GetIs3D(acurvearray)*.

         4) The *ST_PrivateIsMeasured* attribute set to the value expression: *ST_GetIsMeasured(acurvearray)*.

         5) The *ST_PrivateCurves* attribute set to *acurvearray*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.4.4 ST_NumCurves Method

Purpose

Return the cardinality of the ST_PrivateCurves attribute of an ST_CompoundCurve value.

Definition

```
CREATE METHOD ST_NumCurves()
   RETURNS INTEGER
   FOR ST_CompoundCurve
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CARDINALITY(SELF.ST_PrivateCurves)
      END
```

Description

1) The method *ST_NumCurves()* has no input parameters.
2) For the null-call method *ST_NumCurves()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise, return the cardinality of the *ST_PrivateCurves* attribute.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.4.5 ST_CurveN Method

Purpose

Return the specified element in the ST_PrivateCurves attribute of an ST_CompoundCurve value.

Definition

```
CREATE METHOD ST_CurveN
   (aposition INTEGER)
   RETURNS ST_Curve
   FOR ST_CompoundCurve
   BEGIN
      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS ST_Curve);
      END IF;
      IF aposition < 1 OR
         aposition > CARDINALITY(SELF.ST_PrivateCurves) THEN
         BEGIN
            SIGNAL SQLSTATE '01F01'
               SET MESSAGE_TEXT = 'invalid position';
            RETURN CAST (NULL AS ST_Curve);
         END;
      END IF;
      RETURN SELF.ST_PrivateCurves[aposition];
   END
```

Description

1) The method *ST_CurveN(INTEGER)* takes the following input parameters:
   a) an INTEGER value *aposition*.
2) For the null-call method *ST_CurveN(INTEGER)*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) If *aposition* is less than 1 (one) or greater than the cardinality of the *ST_PrivateCurves* attribute, then:
      i) A completion condition is raised: *SQL/MM Spatial warning – invalid position*.
      ii) Return the null value.
   c) Otherwise, return an *ST_Curve* value at element *aposition* in the *ST_PrivateCurves* attribute of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.4.6 ST_StartPoint Method

Purpose

Return an ST_Point value that is the start point of an ST_CompoundCurve value.

Definition

```
CREATE METHOD ST_StartPoint()
   RETURNS ST_Point
   FOR ST_CompoundCurve
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_Curves()[1].ST_Points()[1]
      END
```

Description

1) The method *ST_StartPoint()* has no input parameters.
2) For the null-call method *ST_StartPoint()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise, return the result of the value expression: *SELF.ST_Curves()[1].ST_Points()[1]*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.4.7 ST_EndPoint Method

Purpose

Return an ST_Point value that is the end point of an ST_CompoundCurve value.

Definition

```
CREATE METHOD ST_EndPoint()
   RETURNS ST_Point
   FOR ST_CompoundCurve
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_Curves()[SELF.ST_NumCurves()].
               ST_Points[SELF.ST_Curves()[SELF.ST_NumCurves()].
               ST_NumPoints()]
      END
```

Description

1) The method *ST_EndPoint()* has no input parameters.
2) For the null-call method *ST_EndPoint()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the result of the value expression: *SELF.ST_Curves()[SELF.ST_NumCurves()].ST_Points()[SELF.ST_Curves()[SELF.ST_NumCurves()]. ST_NumPoints()]*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.4.8 ST_CompoundFromTxt Functions

Purpose

Return an ST_CompoundCurve value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_CompoundCurve value.

Definition

```
CREATE FUNCTION ST_CompoundFromTxt
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_CompoundCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_CompoundFromTxt(awkt, 0)

CREATE FUNCTION ST_CompoundFromTxt
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_CompoundCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_CompoundFromTxt(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_CompoundFromTxt(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_CompoundFromTxt(awkt, 0)*.
3) The function *ST_CompoundFromTxt(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_CompoundFromTxt(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkt* is the well-known text representation of an *ST_CompoundCurve* value.

      If *awkt* is not producible in the BNF for <compoundcurve text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromText(awkt, ansrid) AS ST_CompoundCurve)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.4.9 ST_CompoundFromWKB Functions

Purpose

Return an ST_CompoundCurve value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_CompoundCurve value.

Definition

```
CREATE FUNCTION ST_CompoundFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_CompoundCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_CompoundFromWKB(awkb, 0)

CREATE FUNCTION ST_CompoundFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_CompoundCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_CompoundFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_CompoundFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_CompoundFromWKB(awkb, 0)*.
3) The function *ST_CompoundFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_CompoundFromWKB(BINARY LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkb* is the well-known binary representation of an *ST_CompoundCurve* value.

      If *awkb* is not producible in the BNF for <compoundcurve binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_CompoundCurve)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

7.4.10 ST_CompoundFromGML Functions

Purpose

Return an ST_CompoundCurve value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representationof an ST_CompoundCurve value.

Definition

```
CREATE FUNCTION ST_CompoundFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
   RETURNS ST_CompoundCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_CompoundFromGML(agml, 0)

CREATE FUNCTION ST_CompoundFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
    ansrid INTEGER)
   RETURNS ST_CompoundCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_CompoundFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_CompoundFromGML(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_CompoundFromGML(agml, 0)*.
3) The function *ST_CompoundFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_CompoundFromGML(CHARACTER LARGE OBJECT, INTEGER)*:
   a) If the parameter *agml* does not contain a CompositeCurve XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromGML(agml, ansrid) AS ST_CompoundCurve)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8 Surface Types

8.1 ST_Surface Type and Routines

8.1.1 ST_Surface Type

Purpose

The ST_Surface type is a supertype for 2-dimensional geometry types.

Definition

```
CREATE TYPE ST_Surface
   UNDER ST_Geometry
   NOT INSTANTIABLE
   NOT FINAL

   METHOD ST_Area()
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_Area
      (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_3DArea()
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_3DArea
      (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_Perimeter()
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_Perimeter
      (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
      RETURNS DOUBLE PRECISION
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_3DPerimeter()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DPerimeter
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Centroid()
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DCentroid()
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_PointOnSurface()
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DPointOnSurf()
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_IsWorld()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

Definitional Rules

1) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The *ST_Surface* type provides for public use:
   a) a method *ST_Area()*,
   b) a method *ST_Area(CHARACTER VARYING)*,
   c) a method *ST_3DArea()*,

© ISO/IEC 2011 – All rights reserved

d) a method *ST_3DArea(CHARACTER VARYING)*,

e) a method *ST_Perimeter()*,

f) a method *ST_Perimeter(CHARACTER VARYING)*,

g) a method *ST_3DPerimeter()*,

h) a method *ST_3DPerimeter(CHARACTER VARYING)*,

i) a method *ST_Centroid()*,

j) a method *ST_3DCentroid()*,

k) a method *ST_PointOnSurface()*,

l) a method *ST_3DPointOnSurf()*,

m) a method *ST_IsWorld()*.

2) An *ST_Surface* value is a 2-dimensional *ST_Geometry* value that consists of a single connected interior that is associated with one exterior ring and zero or more interior rings. *ST_Surface* values in three-dimensional coordinate space are isomorphic to planar *ST_Surface* values. Stitching together simple surfaces along their boundaries forms polyhedral *ST_Surface* values and polyhedral surfaces in three-dimensional coordinate space may not be planar.

3) The dimension of an *ST_Surface* value is 2.

4) The boundary of an *ST_Surface* value is the collection of the exterior ring and interior rings.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.1.2 ST_Area Methods

Purpose

Return the area measurement of an ST_Surface value, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_Area()
   RETURNS DOUBLE PRECISION
   FOR ST_Surface
   BEGIN
      --
      -- See Description
      --
   END


CREATE METHOD ST_Area
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   FOR ST_Surface
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The method *ST_Area()* has no input parameters.
2) For the null-call method *ST_Area()*:
   a) Case:
      i) If SELF is an empty set, then return the null value.
      ii) Otherwise, return the implementation-defined area of SELF, such that z and m coordinate values are not considered in the calculation, as measured in its spatial reference system.
   b) Case:
      i) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_Area()* is in the linear unit of measure identified by <linear unit> squared.
      ii) Otherwise, the value returned by *ST_Area()* is in an implementation-defined unit of measure.
3) The method *ST_Area(CHARACTER VARYING)* takes the following input parameter:
   a) a CHARACTER VARYING value *aunit*.
4) For the null-call method *ST_Area(CHARACTER VARYING)*:
   a) The values for *aunit* shall be a supported <unit name>.
   b) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.
   c) If the unit specified by *aunit* is not supported by the implementation to compute the area of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.
   d) Case:
      i) If SELF is an empty set, then return the null value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) Otherwise, return the implementation-defined area of SELF, such that z and m coordinate values are not considered in the calculation, as measured in its spatial reference system.

e) The returned value is in the units indicated by *aunit*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.1.3 ST_3DArea Methods

Purpose

Return the area measurement of an ST_Surface value, considering z coordinate values in the calculations.

Definition

```
CREATE METHOD ST_3DArea()
   RETURNS DOUBLE PRECISION
   FOR ST_Surface
   BEGIN
      --
      -- See Description
      --
   END


CREATE METHOD ST_3DArea
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   FOR ST_Surface
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The method *ST_3DArea()* has no input parameters.
2) For the null-call method *ST_3DArea()*:
   a) Case:
      i) If SELF is an empty set, then return the null value.
      ii) Otherwise, return the implementation-defined area of SELF, such that z (but not m) coordinate values are considered in the calculation, as measured in its spatial reference system.
   b) Case:
      i) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_Area()* is in the linear unit of measure identified by <linear unit> squared.
      ii) Otherwise, the value returned by *ST_3DArea()* is in an implementation-defined unit of measure.
3) The method *ST_3DArea(CHARACTER VARYING)* takes the following input parameter:
   a) a CHARACTER VARYING value *aunit*.
4) For the null-call method *ST_3DArea(CHARACTER VARYING)*:
   a) The values for *aunit* shall be a supported <unit name>.
   b) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.
   c) If the unit specified by *aunit* is not supported by the implementation to compute the area of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.
   d) Case:
      i) If SELF is an empty set, then return the null value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) Otherwise, return the implementation-defined area of SELF, such that z (but not m) coordinate values are considered in the calculation, as measured in its spatial reference system.

e) The returned value is in the units indicated by *aunit*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.1.4 ST_Perimeter Methods

Purpose

Return the length measurement of the boundary of an ST_Surface value, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_Perimeter()
   RETURNS DOUBLE PRECISION
   FOR ST_Surface
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_Boundary().ST_Length()
      END

CREATE METHOD ST_Perimeter
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   FOR ST_Surface
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_Boundary().ST_Length(aunit)
      END
```

Definitional Rules

1) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The method *ST_Perimeter()* has no input parameters.
2) For the null-call method *ST_Perimeter()*:
   a) Case:
      i) If SELF is an empty set, then return the null value.
      ii) Otherwise, return the implementation-defined length of the boundary of SELF, such that z and m coordinate values are not considered in the calculation, as measured in its spatial reference system.
   b) Case:
      i) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_Perimeter()* is in the linear unit of measure identified by <linear unit> squared.
      ii) Otherwise, the value returned by *ST_Perimeter()* is in an implementation-defined unit of measure.
3) The method *ST_Perimeter(CHARACTER VARYING)* takes the following input parameter:
   a) a CHARACTER VARYING value *aunit*.
4) For the null-call method *ST_Perimeter(CHARACTER VARYING)*:
   a) The values for *aunit* shall be a supported <unit name>.
   b) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

c) If the unit specified by *aunit* is not supported by the implementation to compute the length of the boundary of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

d) Case:

   i) If SELF is an empty set, then return the null value.

   ii) Otherwise, return the implementation-defined length of the boundary of SELF, such that z and m coordinate values are not considered in the calculation, as measured in its spatial reference system.

e) The returned value is in the units indicated by *aunit*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.1.5 ST_3DPerimeter Methods

Purpose

Return the length measurement of the boundary of an ST_Surface value, considering z coordinate values in the calculations.

Definition

```
CREATE METHOD ST_3DPerimeter()
   RETURNS DOUBLE PRECISION
   FOR ST_Surface
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_3DBoundary().ST_3DLength()
      END

CREATE METHOD ST_3DPerimeter
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   FOR ST_Surface
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_3DBoundary().ST_3DLength(aunit)
      END
```

Definitional Rules

1) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The method *ST_3DPerimeter()* has no input parameters.
2) For the null-call method *ST_3DPerimeter()*:
   a) Case:
      i) If SELF is an empty set, then return the null value.
      ii) Otherwise, return the implementation-defined length of the boundary of SELF, such that z (but not m) coordinate values are considered in the calculation, as measured in its spatial reference system.
   b) Case:
      i) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_3DPerimeter()* is in the linear unit of measure identified by <linear unit> squared.
      ii) Otherwise, the value returned by *ST_3DPerimeter()* is in an implementation-defined unit of measure.
3) The method *ST_3DPerimeter(CHARACTER VARYING)* takes the following input parameter:
   a) a CHARACTER VARYING value *aunit*.
4) For the null-call method *ST_3DPerimeter(CHARACTER VARYING)*:
   a) The values for *aunit* shall be a supported <unit name>.
   b) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

c) If the unit specified by *aunit* is not supported by the implementation to compute the length of the boundary of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

d) Case:

   i) If SELF is an empty set, then return the null value.

   ii) Otherwise, return the implementation-defined length of the boundary of SELF, such that z (but not m) coordinate values are considered in the calculation, as measured in its spatial reference system.

e) The returned value is in the units indicated by *aunit*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.1.6 ST_Centroid Method

Purpose

Return the 2D ST_Point value that is the mathematical centroid of the ST_Surface value, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

Definition

```
CREATE METHOD ST_Centroid()
   RETURNS ST_Point
   FOR ST_Surface
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
      -- ELSE
            --
            -- See Description
            --
      END
```

Description

1) The method *ST_Centroid()* has no input parameters.
2) For the null-call method *ST_Centroid()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise:

      i) Return the mathematical centroid of the *ST_Surface* value. The result is not guaranteed to spatially intersect the *ST_Surface* value.

      ii) If *SELF.ST_Is3D()* is equal to 1 (one), then:

         1) The z coordinate values are not considered in the calculation.

         2) The *ST_Point* value does not include the z coordinate value.

      iii) If *SELF.ST_IsMeasured()* is equal to 1 (one), then:

         1) The m coordinate values are not considered in the calculation.

         2) The *ST_Point* value does not include the m coordinate value.

      iv) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.1.7 ST_3DCentroid Method

Purpose

Return the ST_Point value that is the mathematical centroid of the ST_Surface value, considering z coordinate values in the calculations and including them in the resultant geometry.

Definition

```
CREATE METHOD ST_3DCentroid()
   RETURNS ST_Point
   FOR ST_Surface
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
      -- ELSE
            --
            -- See Description
            --
      END
```

Description

1) The method *ST_3DCentroid()* has no input parameters.
2) For the null-call method *ST_3DCentroid()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise:

      i) Return the mathematical centroid of the *ST_Surface* value. The result is not guaranteed to spatially intersect the *ST_Surface* value.

      ii) If *SELF.ST_Is3D()* is equal to 1 (one), then:

         1) The z coordinate values are considered in the calculation.

         2) The *ST_Point* value includes the z coordinate value.

      iii) If *SELF.ST_IsMeasured()* is equal to 1 (one), then:

         1) The m coordinate values are not considered in the calculation.

         2) The *ST_Point* value does not include the m coordinate value.

      iv) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.1.8 ST_PointOnSurface Method

Purpose

Return an ST_Point value guaranteed to spatially intersect the ST_Surface value, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

Definition

```
CREATE METHOD ST_PointOnSurface()
   RETURNS ST_Point
   FOR ST_Surface
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
      -- ELSE
            --
            -- See Description
            --
      END
```

Description

1) The method *ST_PointOnSurface()* has no input parameters.
2) For the null-call method *ST_PointOnSurface()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise:
      i) Return an *ST_Point* value guaranteed to spatially intersect the *ST_Surface* value.
      ii) If *SELF.ST_Is3D()* is equal to 1 (one), then:
         1) The z coordinate values are not considered in the calculation.
         2) The *ST_Point* value does not include the z coordinate value.
      iii) If *SELF.ST_IsMeasured()* is equal to 1 (one), then:
         1) The m coordinate values are not considered in the calculation.
         2) The *ST_Point* value does not include the m coordinate value.
      iv) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.1.9 ST_3DPointOnSurf Method

Purpose

Return an ST_Point value guaranteed to spatially intersect the ST_Surface value, considering z coordinate values in the calculations and including them in the resultant geometry.

Definition

```
CREATE METHOD ST_3DPointOnSurf()
   RETURNS ST_Point
   FOR ST_Surface
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
      -- ELSE
            --
            -- See Description
            --
      END
```

Description

1) The method *ST_3DPointOnSurf()* has no input parameters.
2) For the null-call method *ST_3DPointOnSurf()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise:
      i) Return an *ST_Point* value guaranteed to spatially 3D intersect the *ST_Surface* value.
      ii) If *SELF.ST_Is3D()* is equal to 1 (one), then:
         1) The z coordinate values are considered in the calculation.
         2) The *ST_Point* value includes the z coordinate value.
      iii) If *SELF.ST_IsMeasured()* is equal to 1 (one), then:
         1) The m coordinate values are not considered in the calculation.
         2) The *ST_Point* value does not include the m coordinate value.
      iv) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.1.10 ST_IsWorld Method

Purpose

Test if the exterior of the ST_Surface value is the empty set, ignoring z coordinate values in the calculations.

Definition

```
CREATE METHOD ST_IsWorld()
   RETURNS INTEGER
   FOR ST_Surface
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_IsWorld()* has no input parameters.
2) For the null-call method *ST_IsWorld()*:

   Case:

   a) If the exterior of the *ST_Surface* value corresponds to the empty set, then return 1 (one).

   b) Otherwise, return 0 (zero).
3) If *SELF.ST_Is3D()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.2 ST_CurvePolygon Type and Routines

8.2.1 ST_CurvePolygon Type

Purpose

The ST_CurvePolygon type is a subtype of the ST_Surface type and values represent a planar surface whose boundary is specified by one exterior ring and zero or more interior rings. Each interior ring defines a hole in the curve polygon.

Definition

```
CREATE TYPE ST_CurvePolygon
   UNDER ST_Surface
   AS (
      ST_PrivateExteriorRing ST_Curve,
      ST_PrivateInteriorRings ST_Curve
         ARRAY[ST_MaxGeometryArrayElements] DEFAULT ARRAY[]
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_CurvePolygon
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
      RETURNS ST_CurvePolygon
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_CurvePolygon
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
       ansrid INTEGER)
      RETURNS ST_CurvePolygon
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_CurvePolygon
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
      RETURNS ST_CurvePolygon
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_CurvePolygon
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
       ansrid INTEGER)
      RETURNS ST_CurvePolygon
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_CurvePolygon
   (acurve ST_Curve)
   RETURNS ST_CurvePolygon
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CurvePolygon
   (acurve ST_Curve,
    ansrid INTEGER)
   RETURNS ST_CurvePolygon
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CurvePolygon
   (acurve ST_Curve,
    acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_CurvePolygon
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CurvePolygon
   (acurve ST_Curve,
    acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_CurvePolygon
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_ExteriorRing()
   RETURNS ST_Curve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_ExteriorRing(acurve ST_Curve)
   RETURNS ST_CurvePolygon
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_InteriorRings()
   RETURNS ST_Curve ARRAY[ST_MaxGeometryArrayElements]
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_InteriorRings
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_CurvePolygon
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_NumInteriorRing()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_InteriorRingN
   (aposition INTEGER)
   RETURNS ST_Curve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_CurvePolyToPoly()
   RETURNS ST_Polygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.
4) The attribute *ST_PrivateExteriorRing* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivateExteriorRing*.
5) The attribute *ST_PrivateInteriorRings* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivateInteriorRings*.

Description

1) The *ST_CurvePolygon* type provides for public use:
   a) a method *ST_CurvePolygon(CHARACTER LARGE OBJECT)*,
   b) a method *ST_CurvePolygon(CHARACTER LARGE OBJECT, INTEGER)*,
   c) a method *ST_CurvePolygon(BINARY LARGE OBJECT)*,
   d) a method *ST_CurvePolygon(BINARY LARGE OBJECT, INTEGER)*,
   e) a method *ST_CurvePolygon(ST_Curve)*,
   f) a method *ST_CurvePolygon(ST_Curve, INTEGER)*,
   g) a method *ST_CurvePolygon(ST_Curve, ST_Curve ARRAY)*,
   h) a method *ST_CurvePolygon(ST_Curve, ST_Curve ARRAY, INTEGER)*,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

i) a method *ST_ExteriorRing()*,

j) a method *ST_ExteriorRing(ST_Curve)*,

k) a method *ST_InteriorRings()*,

l) a method *ST_InteriorRings(ST_Curve ARRAY)*,

m) a method *ST_NumInteriorRing()*,

n) a method *ST_InteriorRingN(INTEGER)*,

o) a method *ST_CurvePolyToPoly()*,

p) a function *ST_CPolyFromText(CHARACTER LARGE OBJECT)*,

q) a function *ST_CPolyFromText(CHARACTER LARGE OBJECT, INTEGER)*,

r) a function *ST_CPolyFromWKB(BINARY LARGE OBJECT)*,

s) a function *ST_CPolyFromWKB(BINARY LARGE OBJECT, INTEGER)*,

t) a function *ST_CPolyFromGML(CHARACTER LARGE OBJECT)*,

u) a function *ST_CPolyFromGML(CHARACTER LARGE OBJECT, INTEGER)*.

2) The *ST_PrivateExteriorRing* attribute is an *ST_Curve* value that is a ring.

3) The *ST_PrivateInteriorRings* attribute is a collection of *ST_Curve* values. Each *ST_Curve* value in the collection is a ring.

4) The *ST_PrivateExteriorRing* attribute shall not be the null value.

5) The *ST_PrivateInteriorRings* attribute shall not be the null value. The elements in the *ST_PrivateInteriorRings* attribute shall not be the null value. If the *ST_CurvePolygon* value does not have interior rings, then the *ST_PrivateInteriorRings* attribute is set to an empty *ST_Curve* ARRAY value.

6) All the *ST_Curve* values in the *ST_PrivateExteriorRing* attribute and *ST_PrivateInteriorRings* attribute shall be in the same spatial reference system as the *ST_CurvePolygon* value.

7) The coordinate dimension of an *ST_CurvePolygon* value is the number of coordinate values associated with the position.

8) An *ST_CurvePolygon* value is simple.

9) The ring in the *ST_PrivateExteriorRing* attribute and the rings in the *ST_PrivateInteriorRings* attribute represent the boundary of the *ST_CurvePolygon* value.

10) An *ST_CurvePolygon* value is topologically closed.

11) The rings in the boundary may spatially intersect at most only a single point:

$$\forall\ p \in \textit{ST_CurvePolygon},\ \forall\ c_1, c_2 \in \text{Boundary}(p),\ c_1 \neq c_2,$$
$$\forall\ a_1, a_2 \in \textit{ST_Point},\ a_1, a_2 \in c_1,\ a_1 \neq a_2,\ [\ a_1 \in c_2 \Rightarrow a_2 \notin c_2\ ]$$

12) An *ST_CurvePolygon* value shall not have cut lines, spikes or punctures:

$$\forall\ p \in \textit{ST_CurvePolygon},\ p = \text{Closure}(\text{Interior}(p))$$

13) The interior of every *ST_CurvePolygon* value is a connected point set.

14) The exterior of an *ST_CurvePolygon* with one or more holes is not connected. Each hole defines a disconnected component of the exterior.

15) An *ST_CurvePolygon* is a topologically closed point set.

16) An *ST_CurvePolygon* value returned by the constructor function corresponds to the empty set.

17) An *ST_CurvePolygon* value corresponds to the empty set if the *ST_PrivateExteriorRing* attribute corresponds to the empty set.

18) An *ST_CurvePolygon* value is well formed only if all the *ST_Curve* values in the *ST_PrivateExteriorRing* attribute and *ST_PrivateInteriorRings* attribute are well formed.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.2.2 ST_CurvePolygon Methods

Purpose

Return an ST_CurvePolygon value constructed from either the well-known text representation, the well-known binary representation, the GML representation, or the specified ST_Curve values.

Definition

```
CREATE CONSTRUCTOR METHOD ST_CurvePolygon
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_CurvePolygon
   FOR ST_CurvePolygon
   RETURN NEW ST_CurvePolygon(awktorgml, 0)

CREATE CONSTRUCTOR METHOD ST_CurvePolygon
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_CurvePolygon
   FOR ST_CurvePolygon
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_CurvePolygon
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_CurvePolygon
   FOR ST_CurvePolygon
   RETURN NEW ST_CurvePolygon(awkb, 0)

CREATE CONSTRUCTOR METHOD ST_CurvePolygon
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_CurvePolygon
   FOR ST_CurvePolygon
   RETURN ST_CPolyFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_CurvePolygon
   (acurve ST_Curve)
   RETURNS ST_CurvePolygon
   FOR ST_CurvePolygon
   RETURN SELF.ST_SRID(0).ST_ExteriorRing(acurve).
      ST_InteriorRings(CAST(ARRAY[] AS
         ST_Curve ARRAY[ST_MaxGeometryArrayElements]))

CREATE CONSTRUCTOR METHOD ST_CurvePolygon
   (acurve ST_Curve,
    ansrid INTEGER)
   RETURNS ST_CurvePolygon
   FOR ST_CurvePolygon
   RETURN SELF.ST_SRID(ansrid).ST_ExteriorRing(acurve).
      ST_InteriorRings(CAST(ARRAY[] AS
         ST_Curve ARRAY[ST_MaxGeometryArrayElements]))

CREATE CONSTRUCTOR METHOD ST_CurvePolygon
   (acurve ST_Curve,
    acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_CurvePolygon
   FOR ST_CurvePolygon
   RETURN SELF.ST_SRID(0).ST_ExteriorRing(acurve).
      ST_InteriorRings(acurvearray)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CREATE CONSTRUCTOR METHOD ST_CurvePolygon
   (acurve ST_Curve,
    acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_CurvePolygon
   FOR ST_CurvePolygon
   RETURN SELF.ST_SRID(ansrid).ST_ExteriorRing(acurve).
      ST_InteriorRings(acurvearray)
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The method *ST_CurvePolygon(CHARACTER LARGE OBJECT)* takes the following input parameter:
   a) a CHARACTER LARGE OBJECT value *awktorgml*.
2) The null-call type preserving SQL-invoked constructor method *ST_CurvePolygon(CHARACTER LARGE OBJECT)* returns the result of the value expression: *NEW ST_CurvePolygon(awktorgml, 0)*.
3) The method *ST_CurvePolygon(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awktorgml*,
   b) an INTEGER value *ansrid*.
4) For the null-call type preserving SQL-invoked constructor method *ST_CurvePolygon(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If *awktorgml* contains a Surface XML element in the GML representation containing a PolygonPatch XML element, then return the result of the value expression: *ST_CPolyFromGML(awktorgml, ansrid)*.
   b) Otherwise, return the result of the value expression: *ST_CPolygFromText(awktorgml, ansrid)*.
5) The method *ST_CurvePolygon(BINARY LARGE OBJECT)* takes the following input parameter:
   a) a BINARY LARGE OBJECT value *awkb*.
6) The null-call type preserving SQL-invoked constructor method *ST_CurvePolygon(BINARY LARGE OBJECT)* returns the result of the value expression: *NEW ST_CurvePolygon(awkb, 0)*.
7) The method *ST_CurvePolygon(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
8) The null-call type preserving SQL-invoked constructor method *ST_CurvePolygon(BINARY LARGE OBJECT, INTEGER)* returns the result of the value expression: *ST_CPolygFromWKB(awktorgml, ansrid)*.
9) The method *ST_CurvePolygon(ST_Curve)* takes the following input parameters:
   b) an *ST_Curve* value *acurve*.
10) The null-call type preserving SQL-invoked constructor method *ST_CurvePolygon(ST_Curve)* returns an *ST_CurvePolygon* value with:
   a) The spatial reference system identifier set to 0 (zero).

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

b) Using the method *ST_ExteriorRing(ST_Curve)*, the *ST_PrivateExteriorRing* attribute set to *acurve*, the *ST_PrivateDimension* attribute set to *ST_GetCoordDim(acurve)*, and the *ST_PrivateCoordinateDimension* attribute set to 2.

c) Using the method *ST_InteriorRings(ST_Curve ARRAY)*, the *ST_PrivateInteriorRings* attribute set to an empty *ST_Curve* ARRAY value.

11) The method *ST_CurvePolygon(ST_Curve, INTEGER)* takes the following input parameters:

a) an *ST_Curve* value *acurve*,

b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_CurvePolygon(ST_Curve, INTEGER)* returns an *ST_CurvePolygon* value with:

a) The spatial reference system identifier set to *ansrid*.

b) Using the method *ST_ExteriorRing(ST_Curve)*, the *ST_PrivateExteriorRing* attribute set to *acurve*, the *ST_PrivateDimension* attribute set to *ST_GetCoordDim(acurve)*, and the *ST_PrivateCoordinateDimension* attribute set to 2.

c) Using the method *ST_InteriorRings(ST_Curve ARRAY)*, the *ST_PrivateInteriorRings* attribute set to an empty *ST_Curve* ARRAY value.

13) The method *ST_CurvePolygon(ST_Curve, ST_Curve ARRAY)* takes the following input parameters:

a) an *ST_Curve* value *acurve*,

b) an *ST_Curve* ARRAY value *acurvearray*.

14) The null-call type preserving SQL-invoked constructor method *ST_CurvePolygon(ST_Curve, ST_Curve ARRAY)* returns an *ST_CurvePolygon* value with:

a) The spatial reference system identifier set to 0 (zero).

b) Using the method *ST_ExteriorRing(ST_Curve)*, the *ST_PrivateExteriorRing* attribute set to *acurve*, the *ST_PrivateDimension* attribute set to *ST_GetCoordDim(acurve, acurvearray)*, and the *ST_PrivateCoordinateDimension* attribute set to 2.

c) Using the method *ST_InteriorRings(ST_Curve ARRAY)*, the *ST_PrivateInteriorRings* attribute set to *acurvearray*.

15) The method *ST_CurvePolygon(ST_Curve, ST_Curve ARRAY, INTEGER)* takes the following input parameters:

a) an *ST_Curve* value *acurve*,

b) an *ST_Curve* ARRAY value *acurvearray*,

c) an INTEGER value *ansrid*.

16) The null-call type preserving SQL-invoked constructor method *ST_CurvePolygon(ST_Curve, ST_Curve ARRAY, INTEGER)* returns an *ST_CurvePolygon* value with:

a) The spatial reference system identifier set to *ansrid*.

b) Using the method *ST_ExteriorRing(ST_Curve)*, the *ST_PrivateExteriorRing* attribute set to *acurve*, the *ST_PrivateDimension* attribute set to *ST_GetCoordDim(acurve, acurvearray)*, and the *ST_PrivateCoordinateDimension* attribute set to 2.

c) Using the method *ST_InteriorRings(ST_Curve ARRAY)*, the *ST_PrivateInteriorRings* attribute set to *acurvearray*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.2.3 ST_ExteriorRing Methods

Purpose

Observe and mutate the ST_PrivateExteriorRing attribute of an ST_CurvePolygon value.

Definition

```
CREATE METHOD ST_ExteriorRing()
   RETURNS ST_Curve
   FOR ST_CurvePolygon
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_PrivateExteriorRing
      END

CREATE METHOD ST_ExteriorRing
   (acurve ST_Curve)
   RETURNS ST_CurvePolygon
   FOR ST_CurvePolygon
   BEGIN
      DECLARE acounter INTEGER;

      IF acurve IS NULL THEN
         SIGNAL SQLSTATE '2FF03'
            SET MESSAGE_TEXT = 'null argument';
      END IF;
      -- If SELF is the null value, then return the null value.
      IF SELF IS NULL THEN
         RETURN CAST (NULL AS ST_CurvePolygon);
      END IF;
      -- Check that there are no mixed spatial reference
      -- systems between SELF and acurve.
      IF SELF.ST_SRID() <> acurve.ST_SRID() THEN
         SIGNAL SQLSTATE '2FF10'
            SET MESSAGE_TEXT = 'mixed spatial reference systems';
      END IF;
      -- If acurve is not a ring, then raise an exception.
      IF acurve.ST_IsRing() = 0 THEN
         SIGNAL SQLSTATE '2FF02'
            SET MESSAGE_TEXT = 'invalid argument';
      END IF;
      -- For all interior rings
      SET acounter = 1;
      WHILE acounter <= CARDINALITY(SELF.ST_InteriorRings()) DO
         -- If the current interior ring as a polygon is not within
         -- acurve as a polygon, then raise an exception
         IF SELF.ST_InteriorRings()[acounter].ST_Within(
            SELF.ST_CurvePolygon(acurve, SELF.ST_SRID())) = 0 THEN
            SIGNAL SQLSTATE '2FF02'
               SET MESSAGE_TEXT = 'invalid argument';
         END IF;
         -- If the current interior ring intersects acurve
         -- with a dimension greater than 0 (zero), then
         -- raise an exception.
         IF SELF.ST_InteriorRings()[acounter].ST_Intersection(acurve).
            ST_Dimension() > 0 THEN
            SIGNAL SQLSTATE '2FF02'
               SET MESSAGE_TEXT = 'invalid argument';
         END IF;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
            SET acounter = acounter + 1;
        END WHILE;
        -- Return an ST_CurvePolygon value with the ST_PrivateExteriorRing
        -- attribute set to acurve.
        RETURN
            SELF.ST_PrivateDimension(2).
                ST_PrivateCoordinateDimension(ST_GetCoordDim(acurve,
                    SELF.ST_PrivateInteriorRings)).
                ST_PrivateIs3D(acurve.ST_PrivateIs3D()).
                ST_PrivateIsMeasured(acurve.ST_PrivateIsMeasured()).
                ST_PrivateExteriorRing(acurve);
    END
```

Description

1) The method *ST_ExteriorRing()* has no input parameters.
2) For the null-call method *ST_ExteriorRing()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise, return the *ST_PrivateExteriorRing* attribute of SELF.
3) The method *ST_ExteriorRing(ST_Curve)* takes the following input parameters:

   a) an *ST_Curve* value *acurve*.
4) For the type preserving method *ST_ExteriorRing(ST_Curve)*:

   Case:

   a) If *acurve* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If SELF is the null value, then return the null value.
   c) If the spatial reference system of SELF is not equal to the spatial reference system of *acurve*, then an exception condition is raised: *SQL/MM Spatial exception – mixed spatial reference systems*.
   d) If *acurve* is not a ring, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
   e) If any two rings in *acurve* and the interior rings of SELF spatially intersect with dimension of the result greater than 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
   f) If any ring in *acurvearray* is not spatially within an *ST_CurvePolygon* value formed from the exterior ring of SELF, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
   g) Otherwise, return an *ST_CurvePolygon* value with:

      i) The dimension set to 2.
      ii) The coordinate dimension set to the value expression: *ST_GetCoordDim(acurve, SELF.ST_PrivateInteriorRings)*.
      iii) The *ST_PrivateIs3D* attribute set to the value expression: *acurve.ST_PrivateIs3D()*.
      iv) The *ST_PrivateIsMeasured* attribute set to the value expression: *acurve.ST_PrivateIsMeasured()*.
      v) The *ST_PrivateExteriorRing* attribute set to *acurve*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.2.4 ST_InteriorRings Methods

Purpose

Observe and mutate the ST_PrivateInteriorRings attribute of an ST_CurvePolygon value.

Definition

```
CREATE METHOD ST_InteriorRings()
   RETURNS ST_Curve ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_CurvePolygon
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_PrivateInteriorRings
      END

CREATE METHOD ST_InteriorRings
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_CurvePolygon
   FOR ST_CurvePolygon
   BEGIN
      DECLARE acounter INTEGER;
      DECLARE bcounter INTEGER;

      IF SELF.ST_ExteriorRing() IS NULL THEN
         SIGNAL SQLSTATE '2FF07'
            SET MESSAGE_TEXT = 'null exterior ring';
      END IF;
      -- If acurvearray is the null value or contains null elements,
      -- then raise an exception.
      CALL ST_CheckNulls(acurvearray);
      -- If SELF is the null value, then return the null value.
      IF SELF IS NULL THEN
         RETURN SELF;
      END IF;
      -- Check that there are no mixed spatial reference
      -- systems between SELF and acurvearray.
      IF (CARDINALITY(acurvearray) > 0) AND
         (SELF.ST_SRID() <> ST_CheckSRID(acurvearray)) THEN
         SIGNAL SQLSTATE '2FF10'
            SET MESSAGE_TEXT = 'mixed spatial reference systems';
      END IF;
      -- If any ST_Curve value is not a ring, then
      -- raise an exception.
      SET acounter = 1;
      WHILE acounter <= CARDINALITY(acurvearray) DO
         IF acurvearray[acounter].ST_IsRing() = 0 THEN
            SIGNAL SQLSTATE '2FF02'
               SET MESSAGE_TEXT = 'invalid argument';
         END IF;
         SET acounter = acounter + 1;
      END WHILE;
      -- For all rings in acurvearray
      SET acounter = 1;
      WHILE acounter <= CARDINALITY(acurvearray) DO
         -- If the current interior ring as a polygon not within
         -- the exterior ring as a polygon, then raise an exception
         IF acurvearray[acounter].ST_Within(
            SELF.ST_CurvePolygon(SELF.ST_ExteriorRing(),
            SELF.ST_SRID())) = 0 THEN
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
            SIGNAL SQLSTATE '2FF02'
               SET MESSAGE_TEXT = 'invalid argument';
         END IF;
         -- If the current interior ring intersects the exterior
         -- ring with a dimension greater than zero, then
         -- raise an exception.
         IF acurvearray[acounter].ST_Intersection(
            SELF.ST_ExteriorRing()).ST_Dimension() > 0 THEN
            SIGNAL SQLSTATE '2FF02'
               SET MESSAGE_TEXT = 'invalid argument';
         END IF;
         SET acounter = acounter + 1;
      END WHILE;
      SET acounter = 1;
      -- For each ring pair in acurvearray
      WHILE acounter <= CARDINALITY(acurvearray)-1 DO
         SET bcounter = acounter+1;
         WHILE bcounter <= CARDINALITY(acurvearray) DO
            -- If the current interior ring pair overlap, then
            -- raise an exception.
            IF SELF.ST_CurvePolygon(acurvearray[acounter],
               SELF.ST_SRID()).ST_Overlaps(
               SELF.ST_CurvePolygon(acurvearray[bcounter],
               SELF.ST_SRID())) = 1 THEN
               SIGNAL SQLSTATE '2FF02'
                  SET MESSAGE_TEXT = 'invalid argument';
            END IF;
            -- If the current interior ring pair intersect
            -- with a dimension greater than zero, then
            -- raise an exception.
            IF acurvearray[acounter].ST_Intersection(
               acurvearray[bcounter]).ST_Dimension() > 0 THEN
               SIGNAL SQLSTATE '2FF02'
                  SET MESSAGE_TEXT = 'invalid argument';
            END IF;
            SET bcounter = bcounter + 1;
         END WHILE;
         SET acounter = acounter + 1;
      END WHILE;
      -- Return an ST_CurvePolygon value with the ST_PrivateInteriorRings
      -- attribute set to acurvearray.
      RETURN SELF.ST_PrivateCoordinateDimension(ST_GetCoordDim(
            SELF.ST_PrivateExteriorRing, acurvearray)).
         ST_PrivateInteriorRings(acurvearray);
   END
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

Description

1) The method *ST_InteriorRings()* has no input parameters.
2) For the null-call method *ST_InteriorRings()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the *ST_PrivateInteriorRings* attribute of SELF.
3) The method *ST_InteriorRings(ST_Curve ARRAY)* takes the following input parameters:

© ISO/IEC 2011 – All rights reserved

a) an *ST_Curve* ARRAY value *acurvearray*.

4) For the type preserving method *ST_InteriorRings(ST_Curve ARRAY)*:

Case:

a) If *SELF.ST_ExteriorRing()* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null exterior ring*.

b) Call the procedure *ST_CheckNulls(ST_Geometry ARRAY)* to check if *acurvearray* is the null value or contains null elements.

c) If SELF is the null value, then return the null value.

d) If the cardinality of *acurvearray* is greater than 0 (zero) and the spatial reference system of SELF is not equal to *ST_CheckSRID*(*acurvearray*), then an exception condition is raised: *SQL/MM Spatial exception – mixed spatial reference systems*.

e) If any *ST_Curve* value in *acurvearray* is not a ring, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

f) If any rings in *acurvearray* and the exterior ring of SELF spatially intersect with dimension of the result greater than 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

g) If any ring in *acurvearray* is not spatially within an *ST_CurvePolygon* value formed from the exterior ring of SELF, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

h) If any two rings in *acurvearray*, formed into *ST_CurvePolygon* values with no interior rings spatially overlap, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

i) Otherwise, return an *ST_CurvePolygon* value with

i) The coordinate dimension set to the value expression: *ST_GetCoordDim(SELF.ST_PrivateExteriorRing, acurvearray)*.

ii) The *ST_PrivateInteriorRings* attribute set to *acurvearray*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.2.5 ST_NumInteriorRing Method

Purpose

Return the cardinality of the ST_PrivateInteriorRings attribute of an ST_CurvePolygon value.

Definition

```
CREATE METHOD ST_NumInteriorRing()
   RETURNS INTEGER
   FOR ST_CurvePolygon
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CARDINALITY(SELF.ST_PrivateInteriorRings)
      END
```

Description

1) The method *ST_NumInteriorRing()* has no input parameters.
2) For the null-call method *ST_NumInteriorRing()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise, return the cardinality of the *ST_PrivateInteriorRings* attribute.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.2.6 ST_InteriorRingN Method

Purpose

Return the specified element in the ST_PrivateInteriorRings attribute of an ST_CurvePolygon value.

Definition

```
CREATE METHOD ST_InteriorRingN
   (aposition INTEGER)
   RETURNS ST_Curve
   FOR ST_CurvePolygon
   BEGIN
      IF SELF.ST_IsEmpty() = 1  THEN
         RETURN CAST (NULL AS ST_Curve);
      END IF;
      IF aposition < 1 OR
         aposition > CARDINALITY(SELF.ST_PrivateInteriorRings) THEN
         BEGIN
            SIGNAL SQLSTATE '01F01'
               SET MESSAGE_TEXT = 'invalid position';
            RETURN CAST (NULL AS ST_Curve);
         END;
      END IF;
      RETURN SELF.ST_PrivateInteriorRings[aposition];
   END
```

Description

1) The method *ST_InteriorRingN(INTEGER)* takes the following input parameters:
   a) an INTEGER value *aposition*.
2) For the null-call method *ST_InteriorRingN(INTEGER)*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) If *aposition* is less than one or greater than the cardinality of the *ST_PrivateInteriorRings* attribute, then:
      i) A completion condition is raised: *SQL/MM Spatial warning – invalid position*.
      ii) Return the null value.
   c) Otherwise, return an *ST_Curve* value at element *aposition* in the *ST_PrivateInteriorRings* attribute of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.2.7 ST_CurvePolyToPoly Method

Purpose

Return the ST_Polygon approximation of an ST_CurvePolygon value.

Definition

```
CREATE METHOD ST_CurvePolyToPoly()
   RETURNS ST_Polygon
   FOR ST_CurvePolygon
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_CurvePolyToPoly()* has no input parameters.
2) For the null-call method *ST_CurvePolyToPoly()*:

   Case:

   a) If SELF is an empty set, then return an empty set of type *ST_Polygon*.
   b) Otherwise, return the implementation-defined *ST_Polygon* value approximation of the *ST_CurvePolygon* value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.2.8 ST_CPolyFromText Functions

Purpose

Return an ST_CurvePolygon value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_CurvePolygon value.

Definition

```
CREATE FUNCTION ST_CPolyFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_CurvePolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_CPolyFromText(awkt, 0)

CREATE FUNCTION ST_CPolyFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_CurvePolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_CPolyFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_CPolyFromText(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_CPolyFromText(awkt, 0)*.
3) The function *ST_CPolyFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_CPolyFromText(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkt* is the well-known text representation of an *ST_CurvePolygon* value.

      If *awkt* is not producible in the BNF for <curvepolygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromText(awkt, ansrid) AS ST_CurvePolygon)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.2.9 ST_CPolyFromWKB Functions

Purpose

Return an ST_CurvePolygon value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_CurvePolygon value.

Definition

```
CREATE FUNCTION ST_CPolyFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_CurvePolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_CPolyFromWKB(awkb, 0)

CREATE FUNCTION ST_CPolyFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_CurvePolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_CPolyFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_CPolyFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_CPolyFromWKB(awkb, 0)*.
3) The function *ST_CPolyFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_CPolyFromWKB(BINARY LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkb* is the well-known binary representation of an *ST_CurvePolygon* value.

      If *awkb* is not producible in the BNF for <curvepolygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_CurvePolygon)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.2.10 ST_CPolyFromGML Functions

Purpose

Return an ST_CurvePolygon value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representationof an ST_CurvePolygon value.

Definition

```
CREATE FUNCTION ST_CPolyFromGML
    (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
    RETURNS ST_CurvePolygon
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    RETURN ST_CPolyFromGML(agml, 0)

CREATE FUNCTION ST_CPolyFromGML
    (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
     ansrid INTEGER)
    RETURNS ST_CurvePolygon
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    BEGIN
       --
       -- See Description
       --
    END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_CPolyFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_CPolyFromGML*(CHARACTER LARGE OBJECT) returns the result of the value expression: *ST_CPolyFromGML(agml, 0)*.
3) The function *ST_CPolyFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_CPolyFromGML(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If the parameter *agml* does not contain a Surface XML element in the GML representation containing a PolygonPatch XML element, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromGML(agml, ansrid) AS ST_CurvePolygon)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.3 ST_Polygon Type and Routines

8.3.1 ST_Polygon Type

Purpose

The ST_Polygon type is a subtype of the ST_CurvePolygon type and represents a planar surface whose boundary is defined by linear rings.

Definition

```
CREATE TYPE ST_Polygon
   UNDER ST_CurvePolygon
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_Polygon
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
      RETURNS ST_Polygon
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_Polygon
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
       ansrid INTEGER)
      RETURNS ST_Polygon
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_Polygon
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
      RETURNS ST_Polygon
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_Polygon
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
       ansrid INTEGER)
      RETURNS ST_Polygon
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_Polygon
      (alinestring ST_LineString)
      RETURNS ST_Polygon
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_Polygon
   (alinestring ST_LineString,
    ansrid INTEGER)
   RETURNS ST_Polygon
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Polygon
   (alinestring ST_LineString,
    alinestringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_Polygon
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Polygon
   (alinestring ST_LineString,
    alinestringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_Polygon
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_ExteriorRing()
   RETURNS ST_LineString,

OVERRIDING METHOD ST_ExteriorRing
   (acurve ST_Curve)
   RETURNS ST_Polygon,

OVERRIDING METHOD ST_InteriorRings()
   RETURNS ST_LineString ARRAY[ST_MaxGeometryArrayElements],

OVERRIDING METHOD ST_InteriorRings
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_Polygon,

OVERRIDING METHOD ST_InteriorRingN
   (aposition INTEGER)
   RETURNS ST_LineString
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The *ST_Polygon* type provides for public use:

© ISO/IEC 2011 – All rights reserved

a) a method *ST_Polygon(CHARACTER LARGE OBJECT)*,
b) a method *ST_Polygon(CHARACTER LARGE OBJECT, INTEGER)*,
c) a method *ST_Polygon(BINARY LARGE OBJECT)*,
d) a method *ST_Polygon(BINARY LARGE OBJECT, INTEGER)*,
e) a method *ST_Polygon(ST_LineString)*,
f) a method *ST_Polygon(ST_LineString, INTEGER)*,
g) a method *ST_Polygon(ST_LineString, ST_LineString ARRAY)*,
h) a method *ST_Polygon(ST_LineString, ST_LineString ARRAY, INTEGER)*,
i) an overriding method *ST_ExteriorRing()*,
j) an overriding method *ST_ExteriorRing(ST_Curve)*,
k) an overriding method *ST_InteriorRings()*,
l) an overriding method *ST_InteriorRings(ST_Curve ARRAY)*,
m) an overriding method *ST_InteriorRingN(INTEGER)*,
n) a function *ST_PolyFromText(CHARACTER LARGE OBJECT)*,
o) a function *ST_PolyFromText(CHARACTER LARGE OBJECT, INTEGER)*,
p) a function *ST_PolyFromWKB(BINARY LARGE OBJECT)*,
q) a function *ST_PolyFromWKB(BINARY LARGE OBJECT, INTEGER)*,
r) a function *ST_PolyFromGML(CHARACTER LARGE OBJECT)*,
s) a function *ST_PolyFromGML(CHARACTER LARGE OBJECT, INTEGER)*,
t) a function *ST_BdPolyFromText(CHARACTER LARGE OBJECT)*,
u) a function *ST_BdPolyFromText(CHARACTER LARGE OBJECT, INTEGER)*,
v) a function *ST_BdPolyFromWKB(BINARY LARGE OBJECT)*,
w) a function *ST_BdPolyFromWKB(BINARY LARGE OBJECT, INTEGER)*.

2) The *ST_PrivateExteriorRing* attribute is an *ST_LineString* value that is a linear ring.
3) The *ST_PrivateInteriorRings* attribute is a collection of *ST_LineString* values. Each *ST_LineString* value in the collection is a linear ring.
4) The linear ring in the *ST_PrivateExteriorRing* attribute and the linear rings in the *ST_PrivateInteriorRings* attribute represent the boundary of the *ST_Polygon* value.
5) An *ST_Polygon* value returned by the constructor function corresponds to the empty set.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.3.2 ST_Polygon Methods

Purpose

Return an ST_Polygon value constructed from either the well-known text representation, the well-known binary representation, the GML representation, or the specified ST_LineString values.

Definition

```
CREATE CONSTRUCTOR METHOD ST_Polygon
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_Polygon
   FOR ST_Polygon
   RETURN NEW ST_Polygon(awktorgml, 0)

CREATE CONSTRUCTOR METHOD ST_Polygon
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_Polygon
   FOR ST_Polygon
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_Polygon
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_Polygon
   FOR ST_Polygon
   RETURN NEW ST_Polygon(awkb, 0)

CREATE CONSTRUCTOR METHOD ST_Polygon
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_Polygon
   FOR ST_Polygon
   RETURN ST_PolyFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_Polygon
   (alinestring ST_LineString)
   RETURNS ST_Polygon
   FOR ST_Polygon
   RETURN SELF.ST_SRID(0).ST_ExteriorRing(alinestring).
      ST_InteriorRings(CAST(ARRAY[] AS
         ST_LineString ARRAY[ST_MaxGeometryArrayElements]))

CREATE CONSTRUCTOR METHOD ST_Polygon
   (alinestring ST_LineString,
    ansrid INTEGER)
   RETURNS ST_Polygon
   FOR ST_Polygon
   RETURN SELF.ST_SRID(ansrid).ST_ExteriorRing(alinestring).
      ST_InteriorRings(CAST(ARRAY[] AS
         ST_LineString ARRAY[ST_MaxGeometryArrayElements]))

CREATE CONSTRUCTOR METHOD ST_Polygon
   (alinestring ST_LineString,
    alinestringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_Polygon
   FOR ST_Polygon
   RETURN SELF.ST_SRID(0).ST_ExteriorRing(alinestring).
      ST_InteriorRings(alinestringarray)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CREATE CONSTRUCTOR METHOD ST_Polygon
   (alinestring ST_LineString,
    alinestringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_Polygon
   FOR ST_Polygon
   RETURN SELF.ST_SRID(ansrid).ST_ExteriorRing(alinestring).
      ST_InteriorRings(alinestringarray)
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The method *ST_Polygon(CHARACTER LARGE OBJECT)* takes the following input parameter:
   a) a CHARACTER LARGE OBJECT value *awktorgml*.
2) The null-call type preserving SQL-invoked constructor method *ST_Polygon(CHARACTER LARGE OBJECT)* returns the result of the value expression: *NEW ST_Polygon(awktorgml, 0)*.
3) The method *ST_Polygon(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awktorgml*,
   b) an INTEGER value *ansrid*.
4) For the null-call type preserving SQL-invoked constructor method *ST_Polygon(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If *awktorgml* contains a Polygon XML element in the GML representation, then return the result of the value expression: *ST_PolyFromGML(awktorgml, ansrid)*.
   b) Otherwise, return the result of the value expression: *ST_PolyFromText(awktorgml, ansrid)*.
5) The method *ST_Polygon(BINARY LARGE OBJECT)* takes the following input parameter:
   a) a BINARY LARGE OBJECT value *awkb*.
6) The null-call type preserving SQL-invoked constructor method *ST_Polygon(BINARY LARGE OBJECT)* return the result of the value expression: *NEW ST_Polygon(awkb, 0)*.
7) The method *ST_Polygon(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
8) The null-call type preserving SQL-invoked constructor method *ST_Polygon(BINARY LARGE OBJECT, INTEGER)* returns the result of the value expression: *ST_PolyFromWKB(awkb, ansrid)*.
9) The method *ST_Polygon(ST_LineString)* takes the following input parameters:
   b) an *ST_LineString* value *alinestring*.
10) The null-call type preserving SQL-invoked constructor method *ST_Polygon(ST_LineString)* returns an *ST_Polygon* value with:
   a) The spatial reference system identifier set to 0 (zero).
   b) Using the method *ST_ExteriorRing(ST_Curve)*, the *ST_PrivateExteriorRing* attribute set to *alinestring*, the *ST_PrivateDimension* attribute set to , and the *ST_PrivateCoordinateDimension* attribute set to 2.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

c) Using the method *ST_InteriorRings(ST_Curve ARRAY)*, the *ST_PrivateInteriorRings* attribute set to an empty *ST_LineString* ARRAY value.

11) The method *ST_Polygon(ST_LineString, INTEGER)* takes the following input parameters:

a) an *ST_LineString* value *alinestring*,

b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_Polygon(ST_LineString, INTEGER)* returns an *ST_Polygon* value with:

a) The spatial reference system identifier set to *ansrid*.

b) Using the method *ST_ExteriorRing(ST_Curve)*, the *ST_PrivateExteriorRing* attribute set to *alinestring*, the *ST_PrivateDimension* attribute set to *ST_GetCoordDim(alinestring)*, and the *ST_PrivateCoordinateDimension* attribute set to 2.

c) Using the method *ST_InteriorRings(ST_Curve ARRAY)*, the *ST_PrivateInteriorRings* attribute set to an empty *ST_LineString* ARRAY value.

13) The method *ST_Polygon(ST_LineString, ST_LineString ARRAY)* takes the following input parameters:

a) an *ST_LineString* value *alinestring*,

b) an *ST_LineString* ARRAY value *alinestringarray*.

14) The null-call type preserving SQL-invoked constructor method *ST_Polygon(ST_LineString, ST_LineString ARRAY)* returns an *ST_Polygon* value with:

a) The spatial reference system identifier set to 0 (zero).

b) Using the method *ST_ExteriorRing(ST_Curve)*, the *ST_PrivateExteriorRing* attribute set to *alinestring*, the *ST_PrivateDimension* attribute set to *ST_GetCoordDim(alinestring, alinestringarray)*, and the *ST_PrivateCoordinateDimension* attribute set to 2.

c) Using the method *ST_InteriorRings(ST_Curve ARRAY)*, the *ST_PrivateInteriorRings* attribute set to *alinestringarray*.

15) The method *ST_Polygon(ST_LineString, ST_LineString ARRAY, INTEGER)* takes the following input parameters:

a) an *ST_LineString* value *alinestring*,

b) an *ST_LineString* ARRAY value *alinestringarray*,

c) an INTEGER value *ansrid*.

16) The null-call type preserving SQL-invoked constructor method *ST_Polygon(ST_LineString, ST_LineString ARRAY, INTEGER)* returns an *ST_Polygon* value with:

a) The spatial reference system identifier set to *ansrid*.

b) Using the method *ST_ExteriorRing(ST_Curve)*, the *ST_PrivateExteriorRing* attribute set to *alinestring*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to 2.

c) Using the method *ST_InteriorRings(ST_Curve ARRAY)*, the *ST_PrivateInteriorRings* attribute set to *alinestringarray*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.3.3 ST_ExteriorRing Methods

Purpose

Observe and mutate the ST_PrivateExteriorRing attribute of an ST_Polygon value.

Definition

```
CREATE METHOD ST_ExteriorRing()
   RETURNS ST_LineString
   FOR ST_Polygon
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CAST(SELF.ST_PrivateExteriorRing AS ST_LineString)
      END

CREATE METHOD ST_ExteriorRing
   (acurve ST_Curve)
   RETURNS ST_Polygon
   FOR ST_Polygon
   BEGIN
      -- If acurve is not an ST_LineString, then raise an exception
      IF acurve IS NOT OF (ST_LineString) THEN
         SIGNAL SQLSTATE '2FF12'
            SET MESSAGE_TEXT = 'curve value is not a linestring value';
      END IF;
      RETURN (SELF AS ST_CurvePolygon).ST_ExteriorRing(acurve);
   END
```

Description

1) The method *ST_ExteriorRing()* has no input parameters.
2) For the null-call method *ST_ExteriorRing()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the *ST_PrivateExteriorRing* attribute of SELF.
3) The method *ST_ExteriorRing(ST_Curve)* takes the following input parameters:

   a) an *ST_Curve* value *alinestring*.
4) For the type preserving method *ST_ExteriorRing(ST_Curve)*:

   Case:

   a) If *acurve* is not an *ST_LineString* value, then an exception condition is raised: *SQL/MM Spatial exception – curve value is not a linestring value*.

   b) Otherwise, return an *ST_Polygon* value as a result of the value expression: *(SELF AS ST_CurvePolygon).ST_ExteriorRing(acurve)*

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.3.4 ST_InteriorRings Methods

Purpose

Observe and mutate the ST_PrivateInteriorRings attribute of an ST_Polygon value.

Definition

```
CREATE METHOD ST_InteriorRings()
   RETURNS ST_LineString ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_Polygon
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CAST(SELF.ST_PrivateInteriorRings AS
               ST_LineString ARRAY[ST_MaxGeometryArrayElements])
      END

CREATE METHOD ST_InteriorRings
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_Polygon
   FOR ST_Polygon
   BEGIN
      DECLARE counter INTEGER;

      -- Check if curves are ST_LineString values
      SET counter = 1;
      WHILE counter <= CARDINALITY(acurvearray) DO
         -- If the current element is not an ST_LineString value, then
         -- raise an exception.
         IF acurvearray[counter] IS NOT OF (ST_LineString) THEN
            SIGNAL SQLSTATE '2FF08'
               SET MESSAGE_TEXT = 'element is not an ST_LineString type';
         END IF;
         SET counter = counter + 1;
      END WHILE;
      RETURN (SELF AS ST_CurvePolygon).ST_InteriorRings(acurvearray);
   END
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

Description

1) The method *ST_InteriorRings()* has no input parameters.
2) For the null-call method *ST_InteriorRings()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the *ST_PrivateInteriorRings* attribute of SELF.
3) The method *ST_InteriorRings(ST_Curve ARRAY)* takes the following input parameters:

   a) an *ST_Curve* ARRAY value *acurvearray*.
4) For the type preserving method *ST_InteriorRings(ST_Curve ARRAY)*:

   Case:

   a) If any element in *acurvearray* is not an *ST_LineString* value, then an exception condition is raised: *SQL/MM Spatial exception – element is not an ST_LineString type*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

b) Otherwise, return an *ST_Polygon* value as a result of the value expression: *(SELF AS ST_CurvePolygon).ST_InteriorRings(acurvearray).*

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.3.5 ST_InteriorRingN Method

Purpose

Return the specified element in the ST_PrivateInteriorRings attribute of an ST_Polygon value.

Definition

```
CREATE METHOD ST_InteriorRingN
   (aposition INTEGER)
   RETURNS ST_LineString
   FOR ST_Polygon
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            TREAT((SELF AS ST_CurvePolygon).ST_InteriorRingN(aposition) AS
               ST_LineString)
      END
```

Description

1) The method *ST_InteriorRingN(INTEGER)* takes the following input parameters:
   a) an INTEGER value *aposition*.
2) For the null-call method *ST_InteriorRingN(INTEGER)*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise, return an *ST_LineString* value as a result of the value expression *TREAT((SELF AS ST_CurvePolygon).ST_InteriorRingN(aposition) AS ST_LineString)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.3.6 ST_PolyFromText Functions

Purpose

Return an ST_Polygon value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_Polygon value.

Definition

```
CREATE FUNCTION ST_PolyFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_Polygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_PolyFromText(awkt, 0)

CREATE FUNCTION ST_PolyFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_Polygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_PolyFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_PolyFromText(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_PolyFromText(awkt, 0)*.
3) The function *ST_PolyFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_PolyFromText(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkt* is the well-known text representation of an *ST_Polygon* value.

      If *awkt* is not producible in the BNF for <polygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromText(awkt, ansrid) AS ST_Polygon)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.3.7 ST_PolyFromWKB Functions

Purpose

Return an ST_Polygon value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_Polygon value.

Definition

```
CREATE FUNCTION ST_PolyFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_Polygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_PolyFromWKB(awkb, 0)

CREATE FUNCTION ST_PolyFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_Polygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_PolyFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_PolyFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_PolyFromWKB(awkb, 0)*.
3) The function *ST_PolyFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_PolyFromWKB(BINARY LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkb* is the well-known binary representation of an *ST_Polygon* value.

      If *awkb* is not producible in the BNF for <polygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_Polygon)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.3.8 ST_PolyFromGML Functions

Purpose

Return an ST_Polygon value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representationof an ST_Polygon value.

Definition

```
CREATE FUNCTION ST_PolyFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
   RETURNS ST_Polygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_PolyFromGML(agml, 0)

CREATE FUNCTION ST_PolyFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
    ansrid INTEGER)
   RETURNS ST_Polygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_PolyFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_PolyFromGML(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_PolyFromGML(agml, 0)*.
3) The function *ST_PolyFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_PolyFromGML(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If the parameter *agml* does not contain a Polygon XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromGML(agml, ansrid) AS ST_Polygon)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.3.9 ST_BdPolyFromText Functions

Purpose

Return an ST_Polygon value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_MultiLineString value.

Definition

```
CREATE FUNCTION ST_BdPolyFromText
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
    RETURNS ST_Polygon
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    RETURN ST_BdPolyFromText(awkt, 0)

CREATE FUNCTION ST_BdPolyFromText
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
     ansrid INTEGER)
    RETURNS ST_Polygon
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    BEGIN
       --
       -- See Description
       --
    END
```

Description

1) The function *ST_BdPolyFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) For the null-call function *ST_BdPolyFromText(CHARACTER LARGE OBJECT)* returns an *ST_Polygon* value as the result of the value expression: *ST_BdPolyFromText(awkt, 0)*.
3) The function *ST_BdPolyFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_BdPolyFromText(CHARACTER LARGE OBJECT, INTEGER)*:
   a) The parameter *awkt* is the well-known text representation of an *ST_MultiLineString* value. If *awkt* is not producible in the BNF for <multilinestring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
   b) Use *ST_MLineFromText(CHARACTER LARGE OBJECT)* to transform *awkt* to an *ST_MultiLineString* value, *AMLS*.
   c) If any *ST_LineString* value in *AMLS* is not a linear ring, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
   d) Using an implementation-dependent algorithm, an exterior linear ring, *ELR*, and an array of zero or more interior rings, *AILR*, are determined from the array of linear rings in *AMLS*.
   e) Return an *ST_Polygon* value with:
      i) The spatial reference system identifier set to *ansrid*.
      ii) Using the method *ST_ExteriorRing(ST_LineString)*, the *ST_PrivateExteriorRing* attribute set to *ELR*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

iii) Using the method *ST_InteriorRings(ST_LineString ARRAY)*, the *ST_PrivateInteriorRings* attribute set to *AILR*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.3.10 ST_BdPolyFromWKB Functions

Purpose

Return an ST_MultiPolygon value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_MultiLineString value.

Definition

```
CREATE FUNCTION ST_BdPolyFromWKB
    (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
    RETURNS ST_Polygon
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    RETURN ST_BdPolyFromWKB(awkb, 0)

CREATE FUNCTION ST_BdPolyFromWKB
    (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
     ansrid INTEGER)
    RETURNS ST_Polygon
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    BEGIN
       --
       -- See Description
       --
    END
```

Description

1) The function *ST_BdPolyFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) For the null-call function *ST_BdPolyFromWKB(BINARY LARGE OBJECT)* returns an *ST_Polygon* value as the result of the value expression: *ST_BdPolyFromText(awkt, 0)*.
3) The function *ST_BdPolyFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_BdPolyFromWKB(BINARY LARGE OBJECT, INTEGER)*:
   a) The parameter *awkb* is the well-known binary representation of an *ST_MultiLineString* value. If *awkb* is not producible in the BNF for <multilinestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
   b) Use *ST_MLineFromWKB(BINARY LARGE OBJECT)* to transform *awkb* to an *ST_MultiLineString* value, *AMLS*.
   c) If any *ST_LineString* value in *AMLS* is not a linear ring, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
   d) Using an implementation-dependent algorithm, an exterior linear ring, *ELR*, and an array of zero or more interior rings, *AILR*, are determined from the array of linear rings in *AMLS*.
   e) Return an *ST_Polygon* value with:
      i) The spatial reference system identifier set to *ansrid*.
      ii) Using the method *ST_ExteriorRing(ST_LineString)*, the *ST_PrivateExteriorRing* attribute set to *ELR*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

iii) Using the method *ST_InteriorRings(ST_LineString ARRAY)*, the *ST_PrivateInteriorRings* attribute set to *AILR*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.4 ST_Triangle Type and Routines

8.4.1 ST_Triangle Type

Purpose

The ST_Triangle type is a subtype of ST_Polygon with an exterior boundary having exactly four points and no interior boundaries.

Definition

```
CREATE TYPE ST_Triangle
   UNDER ST_Polygon
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_Triangle
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
      RETURNS ST_Triangle
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_Triangle
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
       ansrid INTEGER)
      RETURNS ST_Triangle
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_Triangle
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
      RETURNS ST_Triangle
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
   CONSTRUCTOR METHOD ST_Triangle
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
       ansrid INTEGER)
      RETURNS ST_Triangle
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_Triangle
      (alinestring ST_LineString)
      RETURNS ST_Triangle
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_Triangle
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
    (alinestring ST_LineString,
     ansrid INTEGER)
    RETURNS ST_Triangle
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Triangle
    (apointarray ST_Point ARRAY[4])
    RETURNS ST_Triangle
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Triangle
    (apointarray ST_Point ARRAY[4],
     ansrid INTEGER)
    RETURNS ST_Triangle
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

METHOD ST_Points()
    RETURNS ST_Point ARRAY[4]
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

METHOD ST_Points
    (apointarray ST_Point ARRAY[4])
    RETURNS ST_Triangle
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    CALLED ON NULL INPUT,

METHOD ST_3DSlope()
    RETURNS DOUBLE PRECISION
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_ExteriorRing()
    RETURNS ST_LineString,

OVERRIDING METHOD ST_ExteriorRing
    (acurve ST_Curve)
    RETURNS ST_Triangle,

OVERRIDING METHOD ST_InteriorRings()
    RETURNS ST_LineString ARRAY[ST_MaxGeometryArrayElements],
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
OVERRIDING METHOD ST_InteriorRings
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_Triangle,

OVERRIDING METHOD ST_InteriorRingN
   (aposition INTEGER)
   RETURNS ST_LineString
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The *ST_Triangle* type provides for public use:
   a) a method *ST_Triangle(CHARACTER LARGE OBJECT)*,
   b) a method *ST_Triangle(CHARACTER LARGE OBJECT, INTEGER)*,
   c) a method *ST_Triangle(BINARY LARGE OBJECT)*,
   d) a method *ST_Triangle(BINARY LARGE OBJECT, INTEGER)*,
   e) a method *ST_Triangle(ST_LineString)*,
   f) a method *ST_Triangle(ST_LineString, INTEGER)*,
   g) a method *ST_Triangle(ST_Point ARRAY)*,
   h) a method *ST_Triangle(ST_Point ARRAY, INTEGER)*,
   i) a method *ST_Points()*,
   j) a method *ST_Points(ST_Point ARRAY)*,
   k) a method *ST_3DSlope()*,
   l) an overriding method *ST_ExteriorRing()*,
   m) an overriding method *ST_ExteriorRing(ST_Curve)*,
   n) an overriding method *ST_InteriorRings()*,
   o) an overriding method *ST_InteriorRings(ST_Curve ARRAY)*,
   p) an overriding method *ST_InteriorRingN(INTEGER)*,
   q) a function *ST_TriFromText(CHARACTER LARGE OBJECT)*,
   r) a function *ST_TriFromText(CHARACTER LARGE OBJECT, INTEGER)*,
   s) a function *ST_TriFromWKB(BINARY LARGE OBJECT)*,
   t) a function *ST_TriFromWKB(BINARY LARGE OBJECT, INTEGER)*,
   u) a function *ST_TriFromGML(CHARACTER LARGE OBJECT)*,
   v) a function *ST_TriFromGML(CHARACTER LARGE OBJECT, INTEGER)*,
2) The *ST_PrivateExteriorRing* attribute is an *ST_LineString* value that is a linear ring with *ST_NumPoints* = 4.
3) The ST_Triangle value has no interior rings so the *ST_PrivateInteriorRings* attribute is set to an empty *ST_LineString* ARRAY value.
4) The linear ring in the *ST_PrivateExteriorRing* attribute represents the boundary of the *ST_Triangle* value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5) An *ST_Triangle* value returned by the constructor function corresponds to the empty set.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.4.2 ST_Triangle Methods

Purpose

Return an ST_Triangle value constructed from either the well-known text representation, the well-known binary representation, the GML representation, the specified ST_LineString value, or the specified ST_Point values.

Definition

```
CREATE CONSTRUCTOR METHOD ST_Triangle
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_Triangle
   FOR ST_Triangle
   RETURN NEW ST_Triangle(awktorgml, 0)

CREATE CONSTRUCTOR METHOD ST_Triangle
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_Triangle
   FOR ST_Triangle
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_Triangle
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_Triangle
   FOR ST_Triangle
   RETURN NEW ST_Triangle(awkb, 0)

CREATE CONSTRUCTOR METHOD ST_Triangle
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_Triangle
   FOR ST_Triangle
   RETURN ST_TriFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_Triangle
   (alinestring ST_LineString)
   RETURNS ST_Triangle
   FOR ST_Triangle
   RETURN SELF.ST_SRID(0).ST_ExteriorRing(alinestring).
      ST_InteriorRings(CAST(ARRAY[] AS
         ST_LineString ARRAY[ST_MaxGeometryArrayElements]))

CREATE CONSTRUCTOR METHOD ST_Triangle
   (alinestring ST_LineString,
    ansrid INTEGER)
   RETURNS ST_Triangle
   FOR ST_Triangle
   RETURN SELF.ST_SRID(ansrid).ST_ExteriorRing(alinestring).
      ST_InteriorRings(CAST(ARRAY[] AS
         ST_LineString ARRAY[ST_MaxGeometryArrayElements]))

CREATE CONSTRUCTOR METHOD ST_Triangle
   (apointarray ST_Point ARRAY)
   RETURNS ST_Triangle
   FOR ST_Triangle
   RETURN SELF.ST_SRID(0).ST_ExteriorRing.ST_LineString(apointarray).
      ST_InteriorRings(CAST(ARRAY[] AS
         ST_LineString ARRAY[ST_MaxGeometryArrayElements]))

CREATE CONSTRUCTOR METHOD ST_Triangle
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
(alinestring ST_LineString,
 ansrid INTEGER)
RETURNS ST_Triangle
FOR ST_Triangle
RETURN SELF.ST_SRID(ansrid).
   ST_ExteriorRing.ST_LineString(apointarray).
   ST_InteriorRings(CAST(ARRAY[] AS
      ST_LineString ARRAY[ST_MaxGeometryArrayElements]))
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The method *ST_Triangle(CHARACTER LARGE OBJECT)* takes the following input parameter:
   a) a CHARACTER LARGE OBJECT value *awktorgml*.
2) The null-call type preserving SQL-invoked constructor method *ST_Triangle(CHARACTER LARGE OBJECT)* returns the result of the value expression: *NEW ST_Triangle(awktorgml, 0)*.
3) The method *ST_Triangle(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awktorgml*,
   b) an INTEGER value *ansrid*.
4) For the null-call type preserving SQL-invoked constructor method *ST_Triangle(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If *awktorgml* contains a Triangle XML element in the GML representation, then return the result of the value expression: *ST_TriFromGML(awktorgml, ansrid)*.
   b) Otherwise, return the result of the value expression: *ST_TriFromText(awktorgml, ansrid)*.
5) The method *ST_Triangle(BINARY LARGE OBJECT)* takes the following input parameter:
   a) a BINARY LARGE OBJECT value *awkb*.
6) The null-call type preserving SQL-invoked constructor method *ST_Triangle(BINARY LARGE OBJECT)* return the result of the value expression: *NEW ST_Triangle(awkb, 0)*.
7) The method *ST_Triangle(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
8) The null-call type preserving SQL-invoked constructor method *ST_Triangle(BINARY LARGE OBJECT, INTEGER)* returns the result of the value expression: *ST_TriFromWKB(awkb, ansrid)*.
9) The method *ST_Triangle(ST_LineString)* takes the following input parameters:
   a) an *ST_LineString* value *alinestring*.
10) The null-call type preserving SQL-invoked constructor method *ST_Triangle(ST_LineString)* returns an *ST_Triangle* value with:
   a) The spatial reference system identifier set to 0 (zero).

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

b) Using the method *ST_ExteriorRing(ST_Curve)*, the *ST_PrivateExteriorRing* attribute set to *alinestring*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to *alinestring.ST_CoordDim()*.

c) Using the method *ST_InteriorRings(ST_Curve ARRAY)*, the *ST_PrivateInteriorRings* attribute set to an empty *ST_LineString* ARRAY value.

11) The method *ST_Triangle(ST_LineString, INTEGER)* takes the following input parameters:

a) an *ST_LineString* value *alinestring*,

b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_Triangle(ST_LineString, INTEGER)* returns an *ST_Triangle* value with:

a) The spatial reference system identifier set to *ansrid*.

b) Using the method *ST_ExteriorRing(ST_Curve)* :

i) the *ST_PrivateDimension* attribute set to 2,

ii) the *ST_PrivateCoordinateDimension* attribute set to the value expression *ST_GetCoordDim(alinestring)*,

iii) the *ST_PrivateIs3D* attribute set to the value expression *alinestring.ST_Is3D()*,

iv) the *ST_PrivateIsMeasured* attribute set to the value expression *alinestring.ST_IsMeasured(),*

v) the *ST_PrivateExteriorRing* attribute set to *alinestring*.

c) Using the method *ST_InteriorRings(ST_Curve ARRAY)*, the *ST_PrivateInteriorRings* attribute set to an empty *ST_LineString* ARRAY value.

13) The method *ST_Triangle(ST_Point ARRAY)* takes the following input parameters:

a) an *ST_Point* ARRAY value *apointarray*.

14) The null-call type preserving SQL-invoked constructor method *ST_Triangle(ST_Point ARRAY)* returns an *ST_Triangle* value with:

a) The spatial reference system identifier set to 0 (zero).

b) Using the method *ST_ExteriorRing(ST_Curve)*:

i) the *ST_PrivateDimension* attribute set to 2,

ii) the *ST_PrivateCoordinateDimension* attribute set to the value expression *ST_GetCoordDim(apointarray)*,

iii) the *ST_PrivateIs3D* attribute set to the value expression *apointarray.ST_Is3D()*,

iv) the *ST_PrivateIsMeasured* attribute set to the value expression *ST_GetIsMeasured(apointarray),*

v) the *ST_PrivateExteriorRingf* attribute set to *ST_Linestring(apointarray)*.

c) Using the method *ST_InteriorRings(ST_Curve ARRAY)*, the *ST_PrivateInteriorRings* attribute set to an empty *ST_LineString* ARRAY value.

15) The method *ST_Triangle(ST_Point ARRAY, INTEGER)* takes the following input parameters:

a) an *ST_Point* ARRAY value *apointarray*,

b) an INTEGER value *ansrid*.

16) The null-call type preserving SQL-invoked constructor method *ST_Triangle(ST_Point ARRAY, INTEGER)* returns an *ST_Triangle* value with:

a) The spatial reference system identifier set to *ansrid*.

b) Using the method *ST_ExteriorRing(ST_Curve)*, the *ST_PrivateExteriorRing* attribute set to *ST_Linestring(apointarray)*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to *ST_Linestring(apointarray).ST_CoordDim()*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

c) Using the method *ST_InteriorRings(ST_Curve ARRAY)*, the *ST_PrivateInteriorRings* attribute set to an empty *ST_LineString* ARRAY value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.4.3 ST_Points Methods

Purpose

Observe and mutate the ST_PrivatePoints attribute of the ST_LineString value of the ST_PrivateExteriorRing attribute of an ST_Triangle value.

Definition

```
CREATE METHOD ST_Points()
   RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_Triangle
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_PrivateExteriorRing.ST_PrivatePoints
      END

CREATE METHOD ST_Points
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_LineString
   FOR ST_LineString
   BEGIN
      IF CARDINALITY(apointarray) <> 4 THEN
         SIGNAL SQLSTATE '2FF02'
            SET MESSAGE_TEXT = 'invalid argument';
       END IF;
       RETURN SELF.ST_ExteriorRing(NEW ST_LineString(apointarray));
   END
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

Description

1) The method *ST_Points()* has no input parameters.
2) For the null-call method *ST_Points()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the *ST_PrivateExteriorRing* attribute of SELF.
3) The method *ST_Points(ST_Point ARRAY)* takes the following input parameters:

   a) an *ST_Point* ARRAY value *apointarray*.
4) For the type preserving method *ST_Points(ST_Point ARRAY)*:

   a) If *apointarray* does not have exactly four elements, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

   b) Otherwise, return an *ST_Triangle* value with the exterior ring set to *ST_LineString*(*apointarray*,SELF.*ST_SRID()*).

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.4.4 ST_3DSlope Method

Purpose

Return the slope of an ST_Triangle value as a ratio.

Definition

```
CREATE METHOD ST_3DSlope()
   RETURNS DOUBLE PRECISION
   FOR ST_Triangle
   BEGIN
   --
   -- See Description
   --
   END
```

Description

1) The method *ST_3DSlope()* has no input parameters.
2) For the null-call method *ST_3DSlope()*:
   a) Case:
      i) If SELF is an empty set, then return the null value.
      ii) If *SELF.ST_Is3D()* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial Exception – geometry needs to be 3D*.
      iii) Otherwise, return the implementation-defined slope of SELF, such that z coordinate values are considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.4.5 ST_ExteriorRing Methods

Purpose

Observe and mutate the ST_PrivateExteriorRing attribute of an ST_Triangle value.

Definition

```
CREATE METHOD ST_ExteriorRing()
   RETURNS ST_LineString
   FOR ST_Triangle
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CAST(SELF.ST_PrivateExteriorRing AS ST_LineString)
      END

CREATE METHOD ST_ExteriorRing
   (acurve ST_Curve)
   RETURNS ST_Triangle
   FOR ST_Triangle
   BEGIN
      -- If acurve is not an ST_LineString, then raise an exception
      IF acurve IS NOT OF (ST_LineString) THEN
         SIGNAL SQLSTATE '2FF12'
            SET MESSAGE_TEXT = 'curve value is not a linestring value';
      END IF;
      -- If acurve is not an ST_LineString having exactly 4 points,
      -- then raise an exception
      IF (TREAT(acurve AS ST_LineString).ST_NumPoints() <> 4) THEN SIGNAL
      SQLSTATE '2FF75'
         SET MESSAGE_TEXT = 'exterior ring must have exactly 4 points';
      END IF;
      RETURN (SELF AS ST_CurvePolygon).ST_ExteriorRing(acurve);
   END
```

Description

1) The method *ST_ExteriorRing()* has no input parameters.
2) For the null-call method *ST_ExteriorRing()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the *ST_PrivateExteriorRing* attribute of SELF.
3) The method *ST_ExteriorRing(ST_Curve)* takes the following input parameters:

   a) an *ST_Curve* value alinestring.
4) For the type preserving method *ST_ExteriorRing(ST_Curve)*:

   Case:

   a) If acurve is not an *ST_LineString* value, then an exception condition is raised: *SQL/MM Spatial exception – curve value is not a linestring value*.

   b) If acurve is not an *ST_LineString* value having exactly four points, then an exception condition is raised: *SQL/MM Spatial exception – exterior ring must have exactly 4 points*.

   c) Otherwise, return an *ST_Triangle* value as a result of the value expression: *(SELF AS ST_CurvePolygon).ST_ExteriorRing(acurve)*

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.4.6 ST_InteriorRings Methods

Purpose

Observe and mutate the ST_PrivateInteriorRings attribute of an ST_Triangle value.

Definition

```
CREATE METHOD ST_InteriorRings()
   RETURNS ST_LineString ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_Triangle
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CAST(SELF.ST_PrivateInteriorRings AS
               ST_LineString ARRAY[ST_MaxGeometryArrayElements])
      END

CREATE METHOD ST_InteriorRings
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_Triangle
   FOR ST_Triangle
   BEGIN
      -- Check if acurvearray is empty
      IF CARDINALITY(acurvearray) > 0 THEN
         SIGNAL SQLSTATE '2FF66'
            SET MESSAGE_TEXT = 'triangles cannot have holes';
      END IF;
      RETURN (SELF AS ST_CurvePolygon).ST_InteriorRings(acurvearray);
   END
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

Description

1) The method *ST_InteriorRings()* has no input parameters.
2) For the null-call method *ST_InteriorRings()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the *ST_PrivateInteriorRings* attribute of SELF.
3) The method *ST_InteriorRings(ST_Curve ARRAY)* takes the following input parameters:

   a) an *ST_Curve* ARRAY value *acurvearray*.
4) For the type preserving method *ST_InteriorRings(ST_Curve ARRAY)*:

   Case:

   a) If *acurvearray* contains any elements, then an exception condition is raised: *SQL/MM Spatial exception – triangles cannot have holes*.

   b) Otherwise, return an *ST_Triangle* value as a result of the value expression: *(SELF AS ST_CurvePolygon).ST_InteriorRings(acurvearray).*

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.4.7 ST_InteriorRingN Method

Purpose

Return the specified element in the ST_PrivateInteriorRings attribute of an ST_Triangle value.

Definition

```
CREATE METHOD ST_InteriorRingN
   (aposition INTEGER)
   RETURNS ST_LineString
   FOR ST_Triangle
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SIGNAL SQLSTATE '2FF66'
               SET MESSAGE_TEXT = 'triangles cannot have holes'
      END
```

Description

1) The method *ST_InteriorRingN(INTEGER)* takes the following input parameters:
   a) an INTEGER value *aposition*.
2) For the null-call method *ST_InteriorRingN(INTEGER)*:

   Case:
   a) If SELF is an empty set, then return the null value.
   b) Otherwise, an exception condition is raised: *SQL/MM Spatial exception – triangles cannot have holes*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.4.8 ST_TriFromText Functions

Purpose

Return an ST_Triangle value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_Triangle value.

Definition

```
CREATE FUNCTION ST_TriFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_Triangle
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_TriFromText(awkt, 0)

CREATE FUNCTION ST_TriFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_Triangle
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_TriFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_TriFromText(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_TriFromText(awkt, 0)*.
3) The function *ST_TriFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_TriFromText(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkt* is the well-known text representation of an *ST_Triangle* value.

      If *awkt* is not producible in the BNF for <triangle text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromText(awkt, ansrid) AS ST_Triangle)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.4.9 ST_TriFromWKB Functions

Purpose

Return an ST_Triangle value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_Triangle value.

Definition

```
CREATE FUNCTION ST_TriFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_Triangle
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_TriFromWKB(awkb, 0)

CREATE FUNCTION ST_TriFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_Triangle
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_TriFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_TriFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_TriFromWKB(awkb, 0)*.
3) The function *ST_TriFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_TriFromWKB(BINARY LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkb* is the well-known binary representation of an *ST_Triangle* value.

      If *awkb* is not producible in the BNF for <triangle binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_Triangle)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.4.10 ST_TriFromGML Functions

Purpose

Return an ST_Triangle value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_Triangle value.

Definition

```
CREATE FUNCTION ST_TriFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
   RETURNS ST_Triangle
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_TriFromGML(agml, 0)

CREATE FUNCTION ST_TriFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
    ansrid INTEGER)
   RETURNS ST_Triangle
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_TriFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_TriFromGML(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_TriFromGML(agml, 0)*.
3) The function *ST_TriFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_TriFromGML(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If the parameter *agml* does not contain a Triangle XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromGML(agml, ansrid) AS ST_Triangle)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.5 ST_PolyhdrlSurface Type and Routines

8.5.1 ST_PolyhdrlSurface Type

Purpose

The ST_PolyhdrlSurface type is a subtype of ST_Surface composed of contiguous polygon surfaces (ST_Polygon) connected along their common boundary curves. This differs from ST_Surface only in the restriction on the types of surface patches acceptable.

Definition

```
CREATE TYPE ST_PolyhdrlSurface
   UNDER ST_Surface
   AS (
      ST_PrivatePatches ST_Polygon
         ARRAY[ST_MaxGeometryArrayElements] DEFAULT ARRAY[]
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_PolyhdrlSurface method
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
      RETURNS ST_PolyhdrlSurface
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_PolyhdrlSurface method
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
       ansrid INTEGER)
      RETURNS ST_PolyhdrlSurface
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_PolyhdrlSurface method
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
      RETURNS ST_PolyhdrlSurface
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_PolyhdrlSurface method
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
       ansrid INTEGER)
      RETURNS ST_PolyhdrlSurface
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_PolyhdrlSurface method
   (apolygonarray ST_Polygon ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_PolyhdrlSurface
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_PolyhdrlSurface method
   (apolygonarray ST_Polygon ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_PolyhdrlSurface
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Patchesatches method()
   RETURNS ST_Polygon ARRAY[ST_MaxGeometryArrayElements]
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_PatchesPatches method
   (apolygonarray ST_Polygon ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_PolyhdrlSurface
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_NumPatchesNumPatches method()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_PatchN
   atchN method(aposition INTEGER)
   RETURNS ST_Polygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

4) The attribute *ST_PrivatePatches* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivatePatches*.

© ISO/IEC 2011 – All rights reserved

Description

1) The *ST_PolyhdrlSurface* type provides for public use:
   a) a method *ST_PolyhdrlSurface* method*(CHARACTER LARGE OBJECT)*,
   b) a method *ST_PolyhdrlSurface* method*(CHARACTER LARGE OBJECT, INTEGER)*,
   c) a method *ST_PolyhdrlSurface* method*(BINARY LARGE OBJECT)*,
   d) a method *ST_PolyhdrlSurface* method*(BINARY LARGE OBJECT, INTEGER)*,
   e) a method *ST_PolyhdrlSurface* method*(ST_Polygon ARRAY)*,
   f) a method *ST_PolyhdrlSurface* method*(ST_Polygon ARRAY, INTEGER)*,
   g) a method *ST_Patches* method*()*,
   h) a method *ST_Patches* method*(ST_Polygon ARRAY)*,
   i) a method *ST_NumPatches*NumPatches method*()*,
   j) a method *ST_PatchN* method*(INTEGER)*,
   k) a function *ST_PhSFromText* method*(CHARACTER LARGE OBJECT)*,
   l) a function *ST_PhSFromText(CHARACTER LARGE OBJECT, INTEGER)*,
   m) a function *ST_PhSFromWKB(BINARY LARGE OBJECT)*,
   n) a function *ST_PhSFromWKB(BINARY LARGE OBJECT, INTEGER)*,
   o) a function *ST_PhSFromGML(CHARACTER LARGE OBJECT)*,
   p) a function *ST_PhSFromGML(CHARACTER LARGE OBJECT, INTEGER)*.
2) The *ST_PrivatePatches* attribute is a collection of contiguous *ST_Polygon* values connected along their common boundary curves.
3) The *ST_PrivatePatches* method attribute shall not be the null value. The elements in the *ST_PrivatePatches* attribute shall not be the null value.
4) An *ST_PolyhdrlSurface* is contiguous.
5) An *ST_PolyhdrlSurface* is a topologically closed point set.
6) An *ST_PolyhdrlSurface* is a simple, closed polyhedron and is topologically isomorphic to the surface of a sphere.
7) If an *ST_PolyhdrlSurface* value is closed, then it bounds a solid.
8) All of the *ST_Polygon* values in the *ST_PrivatePatches* attribute attribute shall be in the same spatial reference system as the *ST_PolyhdrlSurface* value.
9) The coordinate dimension of an *ST_PolyhdrlSurface* value is the number of coordinate values associated with the *ST_Polygon* values in the *ST_PrivatePatches* attribute having the lowest coordinate dimension.
10) The rings in the boundary may spatially intersect at most only a single point:

$$\forall\ p \in \textit{ST_PolyhdrlSurface},\ \forall\ c_1, c_2 \in \text{Boundary}(p),\ c_1 \neq c_2,$$
$$\forall\ a_1, a_2 \in \textit{ST_Point},\ a_1, a_2 \in c_1,\ a_1 \neq a_2,\ [\ a_1 \in c_2 \Rightarrow a_2 \notin c_2\ ]$$

11) An *ST_PolyhdrlSurface* value shall not have cut lines, spikes or punctures:

$$\forall\ p \in \textit{ST_PolyhdrlSurface},\ p = \text{Closure}(\text{Interior}(p))$$

12) The interior of every *ST_PolyhdrlSurface* value is a connected point set.
13) The exterior of an *ST_PolyhdrlSurface* with one or more holes is not connected. Each hole defines a disconnected component of the exterior.
14) An *ST_PolyhdrlSurface* value returned by the constructor function corresponds to the empty set.
15) An *ST_PolyhdrlSurface* value corresponds to the empty set if the *ST_PrivatePatches* attribute attribute corresponds to the empty set.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

16) An *ST_PolyhdrlSurface* value is well formed only if all of the *ST_Polygon* values in the *ST_PrivatePatches* attribute attribute are well formed.

17) For all *ST_Polygon* values in the *ST_PrivatePatches* attribute attribute, *ST_Polygon.Is3D* = 1 (one).

18) For each pair of *ST_Polygon* values in the *ST_PrivatePatches* attribute attribute that "touch", the common boundary shall be expressible as a finite collection of *ST_LineStrings*. Each such *ST_LineString* shall be part of the boundary of at most 2 *ST_Polygon* values in the *ST_PrivatePatches* attribute attribute.

19) For any two *ST_Polygon* values in the *ST_PrivatePatches* attribute attribute that share a common boundary, the "top" of the *ST_Polygon* values shall be consistent. This means that when two linear rings from these two *ST_Polygon* values traverse the common boundary segment, they do so in opposite directions.

20) If all of the *ST_Polygon* values in the *ST_PrivatePatches* attribute attribute are in alignment (that is, if their normals are parallel), then the whole stitched *ST_PolyhdrlSurface* value is co-planar and can be represented as a single patch (*ST_Polygon* value).

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.5.2 ST_PolyhdrlSurface Methods

Purpose

Return an ST_PolyhdrlSurface value constructed from either the well-known text representation, the well-known binary representation, the GML representation, or the specified ST_Polygon values

Definition

```
CREATE CONSTRUCTOR METHOD ST_PolyhdrlSurface
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_PolyhdrlSurface
   FOR ST_PolyhdrlSurface
   RETURN NEW ST_PolyhdrlSurface(awktorgml, 0)

CREATE CONSTRUCTOR METHOD ST_PolyhdrlSurface
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_PolyhdrlSurface
   FOR ST_PolyhdrlSurface
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_PolyhdrlSurface
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_PolyhdrlSurface
   FOR ST_PolyhdrlSurface
   RETURN NEW ST_PolyhdrlSurface(awkb, 0)

CREATE CONSTRUCTOR METHOD ST_PolyhdrlSurface
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_PolyhdrlSurface
   FOR ST_PolyhdrlSurface
   RETURN ST_PhSFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_PolyhdrlSurface
   (apolygonarray ST_Polygon ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_PolyhdrlSurface
   FOR ST_PolyhdrlSurface
   RETURN SELF.ST_SRID(0).ST_Patches(apolygonarray)

CREATE CONSTRUCTOR METHOD ST_PolyhdrlSurface
   (apolygonarray ST_Polygon  ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_PolyhdrlSurface
   FOR ST_PolyhdrlSurface
    RETURN SELF.ST_SRID(ansrid). ST_Patches(apolygonarray)
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The method *ST_PolyhdrlSurface(CHARACTER LARGE OBJECT)* takes the following input parameter:

   a) a CHARACTER LARGE OBJECT value *awktorgml*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) The null-call type preserving SQL-invoked constructor method *ST_PolyhdrlSurface(CHARACTER LARGE OBJECT)* returns the result of the value expression: *NEW ST_PolyhdrlSurface(awktorgml, 0)*.

3) The method *ST_PolyhdrlSurface(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a CHARACTER LARGE OBJECT value *awktorgml*,

   b) an INTEGER value *ansrid*.

4) For the null-call type preserving SQL-invoked constructor method *ST_PolyhdrlSurface(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If *awktorgml* contains a PolyhedralSurface XML element in the GML representation, then return the result of the value expression: *ST_PhSFromGML(awktorgml, ansrid)*.

   b) Otherwise, return the result of the value expression: *ST_PhSFromText(awktorgml, ansrid)*.

5) The method *ST_PolyhdrlSurface(BINARY LARGE OBJECT)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *awkb*.

6) The null-call type preserving SQL-invoked constructor method *ST_PolyhdrlSurface(BINARY LARGE OBJECT)* returns the result of the value expression: *NEW ST_PolyhdrlSurface(awkb, 0)*.

7) The method *ST_PolyhdrlSurface(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *awkb*,

   b) an INTEGER value *ansrid*.

8) The null-call type preserving SQL-invoked constructor method *ST_PolyhdrlSurface(BINARY LARGE OBJECT, INTEGER)* returns the result of the value expression: *ST_PhSFromWKB(awktorgml, ansrid)*.

9) The method *ST_PolyhdrlSurface(ST_Polygon ARRAY)* takes the following input parameters:

   a) an *ST_Polygon* ARRAY value *apolygonarray*.

10) The null-call type preserving SQL-invoked constructor method *ST_PolyhdrlSurface(ST_Polygon ARRAY)* returns an *ST_PolyhdrlSurface* value with:

   a) The spatial reference system identifier set to 0 (zero).

   b) Using the method *ST_Patches(ST_Polygon ARRAY)*:

      i) the *ST_PrivateDimension* attribute set to 2,

      ii) the *ST_PrivateCoordinateDimension* attribute set to the value expression *ST_GetCoordDim(apolygonarray)*,

      iii) the *ST_PrivateIs3D* attribute set to 1 (one),

      iv) the *ST_PrivateIsMeasured* attribute set to the value expression *ST_GetIsMeasured(apolygonarray)*,

      v) the *ST_PrivatePatches* attribute attribute set to *apolygonarray*.

11) The method *ST_PolyhdrlSurface(ST_Polygon ARRAY, INTEGER)* takes the following input parameters:

   a) an *ST_Polygon* ARRAY value *apolygonarray*,

   b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_PolyhdrlSurface(ST_Polygon ARRAY, INTEGER)* returns an *ST_PolyhdrlSurface* value with:

   a) The spatial reference system identifier set to *ansrid*.

   b) Using the method *ST_Patches(ST_Polygon ARRAY)*:

      i) the *ST_PrivateDimension* attribute set to 2,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) the *ST_PrivateCoordinateDimension* attribute set to the value expression *ST_GetCoordDim(apolygonarray)*,

iii) the *ST_PrivateIs3D* attribute set to 1 (one),

iv) the *ST_PrivateIsMeasured* attribute set to the value expression *ST_GetIsMeasured(apolygonarray),*

v) the *ST_PrivatePatches* attribute set to *apolygonarray*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.5.3 ST_Patches Methods

Purpose

Observe and mutate the ST_PrivatePatches attribute of an ST_PolyhdrlSurface value.

Definition

```
CREATE METHOD ST_Patches()
   RETURNS ST_Polygon ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_PolyhdrlSurface
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_PrivatePatches
      END
CREATE METHOD ST_Patches
   (apolygonarray ST_Polygon ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_PolyhdrlSurface
   FOR ST_PolyhdrlSurface
   BEGIN
      DECLARE acounter INTEGER;
      DECLARE anothercounter INTEGER;
      DECLARE yetanothercounter INTEGER;
      DECLARE test INTEGER;
      DECLARE intersection ST_Geometry;

      -- If apolygonarray is the null value or contains null elements,
      -- then raise an exception.
      CALL ST_CheckNulls(apolygonarray);
      -- If apolygonarray is not 3D, then raise an exception.
      IF ST_GetIs3D(apolygonarray) = 0 THEN
         SIGNAL SQLSTATE '2FF02'
            SET MESSAGE_TEXT = 'invalid argument'
      END IF;
      -- If SELF is the null value, then return the null value.
      IF SELF IS NULL THEN
         RETURN SELF;
      END IF;
      -- Check that there are no mixed spatial reference
      -- systems between SELF and apolygonarray.
      IF (CARDINALITY(apolygonarray) > 0) AND
         (SELF.ST_SRID() <> ST_CheckSRID(apolygonarray)) THEN
         SIGNAL SQLSTATE '2FF10'
            SET MESSAGE_TEXT = 'mixed spatial reference systems';
      END IF;
      -- For all patches, check that they are contiguous with another
      -- patch.  If so, check that the intersecting boundary does not
      -- overlap the boundary of another patch
      SET acounter = 1;
      WHILE acounter <= CARDINALITY(apolygonarray) DO
         -- Every polygon must intersect the boundary of at least one
         -- other polygon in the input array such that the intersection
         -- is a (multi)linestring.  If not, raise an exception
         SET test = 0;
         SET anothercounter = 1;
         WHILE anothercounter <= CARDINALITY(apolygonarray) AND test = 0
            DO
            IF anothercounter <> acounter THEN
               IF apolygonarray[acounter].ST_3DIntersects
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
            (apolygonarray[anothercounter]) = 1 THEN
            IF apolygonarray[acounter].ST_3DIntersection
               (apolygonarray[anothercounter]) IS OF (ST_Curve)
               OR apolygonarray[acounter].ST_3DIntersection
               (apolygonarray[anothercounter]) IS OF (ST_MultiCurve)
               THEN
                  SET test = 1;
            END IF;
         END IF;
      END IF;
      SET anothercounter = anothercounter + 1;
   END WHILE;
   IF test = 0 THEN
      SIGNAL SQLSTATE '2FF02'
         SET MESSAGE_TEXT = 'invalid argument';
   END IF;
   SET acounter = acounter + 1;
END WHILE;
SET acounter = 1;
WHILE acounter <= CARDINALITY(apolygonarray) DO
   -- The interior of the linestring of intersection between any two
   -- polygons in the input array must not intersect with the
   -- boundary of any other polygon in the input array. If not,
   -- raise an exception
   SET test = 0;
   SET anothercounter = 1;
   WHILE anothercounter <= CARDINALITY(apolygonarray) AND test = 0
      DO
      IF anothercounter <> acounter THEN
         IF apolygonarray[acounter].ST_3DIntersects
            (apolygonarray[anothercounter]) = 1 THEN
            SET intersection =
               apolygonarray[acounter].ST_3DIntersection
                  (apolygonarray[anothercounter]);
            SET yetanothercounter = 1;
            WHILE yetanothercounter <= CARDINALITY(apolygonarray) DO
               IF yetanothercounter <> acounter AND
               yetanothercounter <> anothercounter THEN
               IF intersection.ST_3DIntersects
                  (apolygonarray[yetanothercounter]) = 1 AND
                  intersection.ST_3DIntersection
                  (apolygonarray[yetanothercounter]).ST_Dimension
                  > 0 THEN
                  SET test = 1;
               END IF;
            END IF;
         END WHILE;
      END IF;
   END IF;
   SET anothercounter = anothercounter + 1;
END WHILE;
IF test = 1;
   SIGNAL SQLSTATE '2FF02'
      SET MESSAGE_TEXT = 'invalid argument';
END IF;
SET acounter = acounter + 1;
END WHILE;
-- Return an ST_PolyhdrlSurface value with the ST_PrivatePatches
-- attribute set to apolygonarray.
RETURN
   SELF.ST_PrivateDimension(2).
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
            ST_PrivateCoordinateDimension(ST_GetCoordDim(apolygonarray)).
            ST_PrivateIs3D(1).
            ST_PrivateIsMeasured(ST_GetIsMeasured(apolygonarray)).
            ST_PrivatePatches(apolygonarray);
    END
```

Description

1) The method *ST_Patches()* has no input parameters.
2) For the null-call method *ST_Patches()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the *ST_PrivatePatches* attribute of SELF.
3) The method *ST_Patches(ST_Polygon ARRAY)* takes the following input parameters:

   a) an *ST_Polygon* ARRAY value *apolygonarray*.
4) For the type preserving method *ST_Patches(ST_Polygon ARRAY)*:

   Case:

   a) Call the procedure *ST_CheckNulls(ST_Geometry ARRAY)* to check if *apolygonarray* is the null value or contains null elements.

   b) If *apolygonarray* is not 3D, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

   c) If SELF is the null value, then return the null value.

   d) If the cardinality of *apolygonarray* is greater than 0 (zero) and the spatial reference system of SELF is not equal to *ST_CheckSRID*(*apolygonarray*), then an exception condition is raised: *SQL/MM Spatial exception – mixed spatial reference systems*.

   e) Let P1 and P2 be two different *ST_Polygon* values in *apolygonarray*. If, for every P1 in apolygonarray, there does not exist a P2 such that the intersection of their boundaries is a line, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

   f) Let P3 be an *ST_Polygon* value in *apolygonarray* different from P1 and P2. Let L1 be the (multi)line intersection of the boundaries of P1 and P2. If L1 intersects the boundary of any P3 with dimension of the result greater than 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

   g) Otherwise, return an *ST_PolyhdrlSurface* value with:

      i) The dimension set to 2.

      ii) The coordinate dimension set to the value expression: *ST_GetCoordDim(apolygonarray)*.

      iii) The *ST_PrivateIs3D* attribute set to 1 (one).

      iv) The *ST_PrivateIsMeasured* attribute set to the value expression: *ST_GetIsMeasured(apolygonarray)*.

      v) The *ST_PrivatePatches* attribute set to *apolygonarray*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.5.4 ST_NumPatches Method

Purpose

Return the cardinality of the ST_PrivatePatches attribute of an ST_PolyhdrlSurface value.

Definition

```
CREATE METHOD ST_NumPatches()
   RETURNS INTEGER
   FOR ST_PolyhdrlSurface
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CARDINALITY(SELF.ST_PrivatePatches)
      END
```

Description

1) The method *ST_NumPatches()* has no input parameters.
2) For the null-call method *ST_NumPatches()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise, return the cardinality of the *ST_PrivatePatches* attribute.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.5.5 ST_PatchN Method

Purpose

Return the specified element in the ST_PrivatePatches attribute of an ST_PolyhdrlSurface value.

Definition

```
CREATE METHOD ST_PatchN
   (aposition INTEGER)
   RETURNS ST_Polygon
   FOR ST_PolyhdrlSurface
   BEGIN
      IF SELF.ST_IsEmpty() = 1  THEN
         RETURN CAST (NULL AS ST_Polygon);
      END IF;
      IF aposition < 1 OR
         aposition > CARDINALITY(SELF.ST_PrivatePatches) THEN
         BEGIN
            SIGNAL SQLSTATE '01F01'
               SET MESSAGE_TEXT = 'invalid position';
            RETURN CAST (NULL AS ST_Polygon);
         END;
      END IF;
      RETURN SELF.ST_PrivatePatches[aposition];
   END
```

Description

1) The method *ST_PatchN(INTEGER)* takes the following input parameters:
   a) an INTEGER value *aposition*.
2) For the null-call method *ST_PatchN(INTEGER)*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) If *aposition* is less than one or greater than the cardinality of the *ST_PrivatePatches* attribute, then:
      i) A completion condition is raised: *SQL/MM Spatial warning – invalid position*.
      ii) Return the null value.
   c) Otherwise, return an *ST_Polygon* value at element *aposition* in the *ST_PrivatePatches* attribute of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.5.6 ST_PhSFromText Functions

Purpose

Return an ST_PolyhdrlSurface value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_PolyhdrlSurface value.

Definition

```
CREATE FUNCTION ST_PhSFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_PolyhdrlSurface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_PhSFromText(awkt, 0)

CREATE FUNCTION ST_PhSFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_PolyhdrlSurface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_PhSFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_PhSFromText(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_PhSFromText(awkt, 0)*.
3) The function *ST_PhSFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_PhSFromText(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkt* is the well-known text representation of an *ST_PolyhdrlSurface* value.

      If *awkt* is not producible in the BNF for <polyhedralsurface text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromText(awkt, ansrid) AS ST_PolyhdrlSurface)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.5.7 ST_PhSFromWKB Functions

Purpose

Return an ST_PolyhdrlSurface value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_PolyhdrlSurface value.

Definition

```
CREATE FUNCTION ST_PhSFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_PolyhdrlSurface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_PhSFromWKB(awkb, 0)

CREATE FUNCTION ST_PhSFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_PolyhdrlSurface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_PhSFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_PhSFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_PhSFromWKB(awkb, 0)*.
3) The function *ST_PhSFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_PhSFromWKB(BINARY LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkb* is the well-known binary representation of an *ST_PolyhdrlSurface* value.

      If *awkb* is not producible in the BNF for <polyhedralsurface binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_PolyhdrlSurface)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.5.8 ST_PhSFromGML Functions

Purpose

Return an ST_PolyhdrlSurface value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_PolyhdrlSurface value.

Definition

```
CREATE FUNCTION ST_PhSFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
   RETURNS ST_PolyhdrlSurface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_PhSFromGML(agml, 0)

CREATE FUNCTION ST_PhSFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
    ansrid INTEGER)
   RETURNS ST_PolyhdrlSurface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_PhSFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_PhSFromGML*(CHARACTER LARGE OBJECT) returns the result of the value expression: *ST_PhSFromGML(agml, 0)*.
3) The function *ST_PhSFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_PhSFromGML(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If the parameter *agml* does not contain a PolyhedralSurface XML element in the GML representation , then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromGML(agml, ansrid) AS ST_PolyhdrlSurface)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.6 ST_TIN Type and Routines

8.6.1 ST_TIN Type

Purpose

The ST_TIN type is a subtype of ST_Surface composed of contiguous triangle surfaces (ST_Triangle) connected along their common boundary linestrings. This differs from ST_PolyhdrlSurface in the restriction on the types of surface patches acceptable and the addition of TIN elements which constrain the triangulation.

Definition

```
CREATE TYPE ST_TIN
   UNDER ST_PolyhdrlSurface
   AS (
      ST_PrivateElements ST_TINElement
         ARRAY[ST_MaxGeometryArrayElements] DEFAULT ARRAY[],
      ST_PrivateMaxSideLength DOUBLE PRECISION DEFAULT NULL
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_TIN
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
      RETURNS ST_TIN
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_TIN
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
       ansrid INTEGER)
      RETURNS ST_TIN
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_TIN
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
      RETURNS ST_TIN
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_TIN
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_TIN
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_TIN
   (triangles ST_Triangle ARRAY[ST_MaxGeometryArrayElements],
    elements ST_TINElement ARRAY[ST_MaxGeometryArrayElements],
    maxsidelength DOUBLE PRECISION)
   RETURNS ST_TIN
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_TIN
   (triangles ST_Triangle ARRAY[ST_MaxGeometryArrayElements],
    elements ST_TINElement ARRAY[ST_MaxGeometryArrayElements],
    maxsidelength DOUBLE PRECISION,
    ansrid INTEGER)
   RETURNS ST_TIN
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_TIN
   (elements ST_TINElement ARRAY[ST_MaxGeometryArrayElements],
    maxsidelength DOUBLE PRECISION)
   RETURNS ST_TIN
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_TIN
   (elements ST_TINElement ARRAY[ST_MaxGeometryArrayElements],
    maxsidelength DOUBLE PRECISION,
    ansrid INTEGER)
   RETURNS ST_TIN
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_TINElements()
   RETURNS ST_TINElement ARRAY[ST_MaxGeometryArrayElements]
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_TINElements
   (elements ST_TINElement ARRAY[ST_MaxGeometryArrayElements],
    triangulate INTEGER)
   RETURNS ST_TIN
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_MaxSideLength()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_MaxSideLength
   (maxsidelength DOUBLE PRECISION,
    triangulate INTEGER)
   RETURNS ST_TIN
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_TINTable()
   RETURNS TABLE
   (item CHARACTER VARYING(30),
    ordernumber INTEGER,
    xcoord DOUBLE PRECISION,
    ycoord DOUBLE PRECISION,
    zcoord DOUBLE PRECISION,
    triangle INTEGER ARRAY[3],
    visibility INTEGER,
    elementID INTEGER,
    elementtag CHARACTER VARYING(64),
    element INTEGER ARRAY[ST_MaxIntegerArrayElements])
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_TINTable
   (tin_table_name DT1,
    item_column DT2,
    ordernumber_column DT2,
    xcoord_column DT2,
    ycoord_column DT2,
    zcoord_column DT2,
    triangle_column DT2,
    visibility_column DT2,
    elementID_column DT2,
    elementtag_column DT2,
    element_column DT2,
    maxsidelength DOUBLE PRECISION)
   RETURNS ST_TIN
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_Clip
   (clipboundary ST_Polygon)
   RETURNS ST_TIN
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

OVERRIDING METHOD ST_Patches()
   RETURNS ST_Triangle ARRAY[ST_MaxGeometryArrayElements],

OVERRIDING METHOD ST_Patches
   (triangles ST_Triangle ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_TIN
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.
4) *ST_MaxIntegerArrayElements* is the implementation-defined maximum cardinality of an array of *INTEGER* elements.
5) *DT1* is data type of variable-length character string with character set SQL_IDENTIFIER and implementation-defined maximum length.
6) *DT2* is data type of variable-length character string with character set SQL_IDENTIFIER and maximum length not less than 128 characters.
7) The attribute *ST_PrivatePatches* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivatePatches*.
8) The attribute *ST_PrivateElements* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivateElements*.
9) The attribute *ST_PrivateMaxSideLength* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivateMaxSideLength*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Description

1) The *ST_TIN* type provides for public use:
   a) a method *ST_TIN(CHARACTER LARGE OBJECT)*,
   b) a method *ST_TIN(CHARACTER LARGE OBJECT, INTEGER)*,
   c) a method *ST_TIN(BINARY LARGE OBJECT)*,
   d) a method *ST_TIN(BINARY LARGE OBJECT, INTEGER)*,
   e) a method *ST_TIN(ST_Triangle ARRAY, ST_TINElement ARRAY, DOUBLE PRECISION)*,
   f) a method *ST_TIN(ST_Triangle ARRAY, ST_TINElement ARRAY, DOUBLE PRECISION, INTEGER)*,
   g) a method *ST_TIN(ST_TINElement ARRAY, DOUBLE PRECISION)*,
   h) a method *ST_TIN(ST_TINElement ARRAY, DOUBLE PRECISION, INTEGER)*,
   i) a method *ST_TINElements()*,
   j) a method *ST_TINElements(ST_TINElement ARRAY, INTEGER)*,
   k) a method *ST_MaxSideLength()*,
   l) a method *ST_MaxSideLength(DOUBLE PRECISION, INTEGER)*,
   m) a method *ST_TINTable()*,
   n) a method *ST_TINTable(DT1, DT2, DT2, DT2, DT2, DT2, DT2, DT2, DT2, DT2, DT2, DOUBLE PRECISION)*,
   o) a method *ST_Clip(ST_Polygon)*,
   p) an overriding method *ST_Patches()*,
   q) an overriding method *ST_Patches(ST_Triangle ARRAY)*,
   r) a function *ST_TINFromText(CHARACTER LARGE OBJECT)*,
   s) a function *ST_TINFromText(CHARACTER LARGE OBJECT, INTEGER)*,
   t) a function *ST_TINFromWKB(BINARY LARGE OBJECT)*,
   u) a function *ST_TINFromWKB(BINARY LARGE OBJECT, INTEGER)*,
   v) a function *ST_TINFromGML(CHARACTER LARGE OBJECT)*,
   w) a function *ST_TINFromGML(CHARACTER LARGE OBJECT, INTEGER)*.
2) The *ST_PrivatePatches* attribute is a collection of contiguous *ST_Triangle* values connected along their common boundary curves.
3) An *ST_TIN* value returned by the constructor function corresponds to the empty set.
4) The *ST_PrivateElements* attribute is a collection of *ST_TINElement* values representing constraints from which the TIN surface can be generated.
5) The *ST_PrivateMaxSideLength* attribute restricts the maximum length of each side of an *ST_Triangle* value that is an element in the *ST_PrivatePatches* attribute.
6) It is implementation-defined whether the restriction imposed by the *ST_PrivateMaxSideLength* attribute applies to all of the *ST_Triangle* values in the *ST_PrivatePatches* attribute or just those which lie along the boundary of the TIN surface.
7) The spatial reference systems of the *ST_Triangle* values in the *ST_PrivatePatches* attribute and of all the *ST_Geometry* values contained in the *ST_TINElements* values in the *ST_PrivateElements* attribute shall be equal to the spatial reference system of their *ST_TIN* value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.6.2 ST_TIN Methods

Purpose

Return an ST_TIN value constructed from either the well-known text representation; the well-known binary representation; the GML representation; the specified ST_Triangle triangle values, ST_TINElement TIN element values and the DOUBLE PRECISION maximum side length value; or the specified ST_TINElement TIN element values and the DOUBLE PRECISION maximum side length value.

Definition

```
CREATE CONSTRUCTOR METHOD ST_TIN
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_TIN
   FOR ST_TIN
   RETURN NEW ST_TIN(awktorgml, 0)

CREATE CONSTRUCTOR METHOD ST_TIN
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_TIN
   FOR ST_TIN
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_TIN
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_TIN
   FOR ST_TIN
   RETURN NEW ST_TIN(awkb, 0)

CREATE CONSTRUCTOR METHOD ST_TIN
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_TIN
   FOR ST_TIN
   RETURN ST_TINFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_TIN
   (triangles ST_Triangle ARRAY[ST_MaxGeometryArrayElements],
    elements ST_TINElement ARRAY[ST_MaxGeometryArrayElements],
    maxsidelength DOUBLE PRECISION)
   RETURNS ST_TIN
   FOR ST_TIN
   RETURN SELF.ST_SRID(0).
      ST_Patches(triangles).
      ST_TINElements(elements,0).
      ST_MaxSideLength(maxsidelength,0)

CREATE CONSTRUCTOR METHOD ST_TIN
   (triangles ST_Triangle ARRAY[ST_MaxGeometryArrayElements],
    elements ST_TINElement ARRAY[ST_MaxGeometryArrayElements],
    maxsidelength DOUBLE PRECISION,
    ansrid INTEGER)
   RETURNS ST_TIN
   FOR ST_TIN
   RETURN SELF.ST_SRID(ansrid).
      ST_Patches(triangles).
      ST_TINElements(elements,0).
      ST_MaxSideLength(maxsidelength,0)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CREATE CONSTRUCTOR METHOD ST_TIN
   (elements ST_TINElement ARRAY[ST_MaxGeometryArrayElements],
    maxsidelength DOUBLE PRECISION)
   RETURNS ST_TIN
   FOR ST_TIN
   BEGIN
      --
      -- Create ST_Triangle ARRAY triangles from TIN elements
      -- and maxsidelength
      --
      -- See Description
      --
   RETURN SELF.ST_SRID(0).
      ST_Patches(triangles).
      ST_TINElements(elements,0).
      ST_MaxSideLength(maxsidelength,0)
   END

CREATE CONSTRUCTOR METHOD ST_TIN
   (elements ST_TINElement ARRAY[ST_MaxGeometryArrayElements],
    maxsidelength DOUBLE PRECISION,
    ansrid INTEGER)
   RETURNS ST_TIN
   FOR ST_TIN
   BEGIN
      --
      -- Create ST_Triangle ARRAY triangles from TIN elements
      -- and maxsidelength
      --
      -- See Description
      --
   RETURN SELF.ST_SRID(ansrid).
      ST_Patches(triangles).
      ST_TINElements(elements,0).
      ST_MaxSideLength(maxsidelength,0)
   END
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The method *ST_TIN(CHARACTER LARGE OBJECT)* takes the following input parameter:
   a) a CHARACTER LARGE OBJECT value *awktorgml*.
2) The null-call type preserving SQL-invoked constructor method *ST_TIN(CHARACTER LARGE OBJECT)* returns the result of the value expression: *NEW ST_TIN(awktorgml, 0)*.
3) The method *ST_TIN(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awktorgml*,
   b) an INTEGER value *ansrid*.
4) For the null-call type preserving SQL-invoked constructor method *ST_TIN(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

a) If *awktorgml* contains a TIN XML element in the GML representation, then return the result of the value expression: *ST_TINFromGML(awktorgml, ansrid)*.

b) Otherwise, return the result of the value expression: *ST_TINFromText(awktorgml, ansrid)*.

5) The method *ST_TIN(BINARY LARGE OBJECT)* takes the following input parameter:

a) a BINARY LARGE OBJECT value *awkb*.

6) The null-call type preserving SQL-invoked constructor method *ST_TIN(BINARY LARGE OBJECT)* returns the result of the value expression: *NEW ST_TIN(awkb, 0)*.

7) The method *ST_TIN(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:

a) a BINARY LARGE OBJECT value *awkb*,

b) an INTEGER value *ansrid*.

8) The null-call type preserving SQL-invoked constructor method *ST_TIN(BINARY LARGE OBJECT, INTEGER)* returns the result of the value expression: *ST_TINFromWKB(awktorgml, ansrid)*.

9) The method *ST_TIN(ST_Triangle ARRAY, ST_TINElement ARRAY, DOUBLE PRECISION)* takes the following input parameters:

a) an *ST_Triangle* ARRAY value *triangles*,

b) an *ST_TINElement* ARRAY value *elements*,

c) a DOUBLE PRECISION value *maxsidelength*.

10) The null-call type preserving SQL-invoked constructor method *ST_TIN(ST_Triangle ARRAY, ST_TINElement ARRAY, DOUBLE PRECISION)* returns an *ST_TIN* value with:

a) The spatial reference system identifier set to 0 (zero).

b) Using the method *ST_Patches(ST_Triangle ARRAY)*, the *ST_PrivatePatches* attribute set to *triangles*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to *ST_GetCoordDim(triangles)*.

c) Using the method *ST_TINElements(ST_TINElement ARRAY, INTEGER)*, the *ST_PrivateElements* attribute set to *elements*.

d) Using the method *ST_MaxSideLength(DOUBLE PRECISION, INTEGER)*, the *ST_PrivateMaxSideLength* attribute set to *maxsidelength*.

11) The method *ST_TIN(ST_Triangle ARRAY, ST_TINElement ARRAY, DOUBLE PRECISION, INTEGER)* takes the following input parameters:

a) an *ST_Triangle* ARRAY value *triangles*,

b) an *ST_TINElement* ARRAY value *elements*,

c) a DOUBLE PRECISION value *maxsidelength*,

d) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_TIN(ST_Triangle ARRAY, ST_TINElement ARRAY, DOUBLE PRECISION, INTEGER)* returns an *ST_TIN* value with:

a) The spatial reference system identifier set to *ansrid*.

b) Using the method *ST_Patches(ST_Triangle ARRAY)*, the *ST_PrivatePatches* attribute set to *triangles*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to *ST_GetCoordDim(triangles)*.

c) Using the method *ST_TINElements(ST_TINElement ARRAY, INTEGER)*, the *ST_PrivateElements* attribute set to *elements*.

d) Using the method *ST_MaxSideLength(DOUBLE PRECISION, INTEGER)*, the *ST_PrivateMaxSideLength* attribute set to *maxsidelength*.

13) The method *ST_TIN(ST_TINElement ARRAY, DOUBLE PRECISION)* takes the following input parameters:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

a) an *ST_TINElement* ARRAY value *elements*,

b) a DOUBLE PRECISION value *maxsidelength*.

14) The null-call type preserving SQL-invoked constructor method *ST_TIN(ST_TINElement ARRAY, DOUBLE PRECISION)* returns an *ST_TIN* value with:

a) The spatial reference system identifier set to 0 (zero).

b) Using the method *ST_Patches(ST_Triangle ARRAY)*, the *ST_PrivatePatches* attribute set to *triangles*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to *ST_GetCoordDim(triangles)* where *triangles* is an *ST_Triangle* ARRAY obtained by applying the implementation-defined triangulation algorithm to the *ST_TINElement* ARRAY *elements* and constrained or modified by *maxsidelength*.

c) Using the method *ST_MaxSideLength(DOUBLE PRECISION, INTEGER)*, the *ST_PrivateMaxSideLength* attribute set to *maxsidelength*.

15) The method *ST_TIN(ST_TINElement ARRAY, DOUBLE PRECISION, INTEGER)* takes the following input parameters:

a) an *ST_TINElement* ARRAY value *elements*,

b) a DOUBLE PRECISION value *maxsidelength*,

c) an INTEGER value *ansrid*.

16) The null-call type preserving SQL-invoked constructor method *ST_TIN(ST_TINElement ARRAY, DOUBLE PRECISION, INTEGER)* returns an *ST_TIN* value with:

a) The spatial reference system identifier set to *ansrid*.

b) Using the method *ST_Patches(ST_Triangle ARRAY)*, the *ST_PrivatePatches* attribute set to *triangles*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to *ST_GetCoordDim(triangles)* where *triangles* is an *ST_Triangle* ARRAY obtained by applying the implementation-defined triangulation algorithm to the *ST_TINElement* ARRAY *elements* and constrained or modified by *maxsidelength*.

c) Using the method *ST_MaxSideLength(DOUBLE PRECISION, INTEGER)*, the *ST_PrivateMaxSideLength* attribute set to *maxsidelength*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.6.3 ST_TINElements Methods

Purpose

Observe and mutate the ST_PrivateElements attribute of an ST_TIN value.

Definition

```
CREATE METHOD ST_TINElements()
   RETURNS ST_TINElement ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_TIN
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_PrivateElements
      END

CREATE METHOD ST_TINElements
   (elements ST_TINElement ARRAY[ST_MaxGeometryArrayElements],
    triangulate INTEGER)
   RETURNS ST_TIN
   FOR ST_TIN
   BEGIN
      DECLARE triangles ST_Triangle ARRAY[ST_MaxGeometryArrayElements];
      DECLARE acounter INTEGER;

      -- If SELF is the null value, then return the null value.
      IF SELF IS NULL THEN
         RETURN SELF;
      END IF;
      -- Check that there are no mixed spatial reference
      -- systems between SELF and elements.
      SET acounter = 1;
      WHILE acounter <= CARDINALITY(SELF.ST_TINElements()) DO
         IF SELF.ST_SRID() <>
         SELF.ST_TINElements()[acounter].ST_ElementGeometry().ST_SRID THEN
            SIGNAL SQLSTATE '2FF10'
               SET MESSAGE_TEXT = 'mixed spatial reference systems';
         END IF;
         SET acounter = acounter + 1;
      END WHILE;
      -- If triangulate = 1, (re)triangulate the TIN surface
      triangles = SELF.ST_Patches();
      --
      -- Update triangles by triangulating - See Description
      --
      -- Return an ST_TIN value with the ST_PrivateControlPoints
      -- attribute set to controlpoints and, if triangulate = 1,
      -- with the ST_PrivatePatches attribute set to triangles.
      RETURN
         CASE
            WHEN triangulate = 1 THEN
               SELF.ST_PrivatePatches(triangles).
                  ST_PrivateElements(elements)
            ELSE
               SELF.ST_PrivateElements(elements);
   END
```

Description

1) The method *ST_TINElements()* has no input parameters.
2) For the null-call method *ST_TINElements()*:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Case:

a) If SELF is an empty set, then return the null value.

b) Otherwise, return the *ST_PrivateElements* attribute of SELF.

3) The method *ST_TINElements(ST_TINElement ARRAY, INTEGER)* takes the following input parameters:

a) an *ST_TINElement* ARRAY value *elements*.

b) an *INTEGER* value *triangulate*.

4) For the type preserving method *ST_TINElements(ST_TINElement ARRAY, INTEGER)*:

Case:

a) If SELF is the null value, then return the null value.

b) If the spatial reference system of SELF is not equal to the spatial reference system of all of the element geometries, then an exception condition is raised: *SQL/MM Spatial exception – mixed spatial reference systems*.

c) Case:

i) If *triangulate* = 1, then:

1) let *triangles* = *SELF.Patches()*.

2) let *elements* = *SELF.TINElements()*.

3) let *maxsidelength* = *SELF.MaxSideLength()*.

4) update *triangles* by applying the implementation-defined triangulation algorithm to the *ST_TINElements* ARRAY *elements* and constrained or modified by *maxsidelength*

5) return an *ST_TIN* value with the *ST_PrivatePatches* attribute set to *triangles* and the *ST_PrivateElements* attribute set to *elements*.

ii) Otherwise, return an *ST_TIN* value with the *ST_PrivateElements* attribute set to *elements*.

5) It is implementation-defined which of the predefined TIN element types are supported, which additional TIN element types are supported, what type of *ST_Geometry* each requires, what behavior is to be expected during triangulation and what exceptions might be raised.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.6.4 ST_MaxSideLength Methods

Purpose

Observe and mutate the ST_PrivateMaxSideLength attribute of an ST_TIN value.

Definition

```
CREATE METHOD ST_MaxSideLength()
   RETURNS DOUBLE PRECISION
   FOR ST_TIN
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_PrivateMaxSideLength
      END

CREATE METHOD ST_MaxSideLength
   (maxsidelength DOUBLE PRECISION,
    triangulate INTEGER)
   RETURNS ST_TIN
   FOR ST_TIN
   BEGIN
      DECLARE triangles ST_Triangle ARRAY[ST_MaxGeometryArrayElements];

      IF maxsidelength IS NULL THEN
         SIGNAL SQLSTATE '2FF03'
            SET MESSAGE_TEXT = 'null argument';
      IF maxsidelength <= 0
         SIGNAL SQLSTATE '2FF02'
            SET MESSAGE_TEXT = 'invalid argument';
      -- If SELF is the null value, then return the null value.
      IF SELF IS NULL THEN
         RETURN SELF;
      END IF;
      -- If triangulate = 1, (re)triangulate the TIN surface
      triangles = SELF.ST_Patches();
      --
      -- Update triangles by triangulating - See Description
      --
      -- Return an ST_TIN value with the ST_PrivateMaxSideLength
      -- attribute set to maxsidelength and, if triangulate = 1,
      -- with the ST_PrivatePatches attribute set to triangles.
      RETURN
         CASE
            WHEN triangulate = 1 THEN
               SELF.ST_PrivatePatches(triangles).
                  ST_PrivateMaxSideLength(maxsidelength)
            ELSE
               SELF. ST_PrivateMaxSideLength(maxsidelength);
   END
```

Description

1) The method *ST_MaxSideLength()* has no input parameters.
2) For the null-call method *ST_MaxSideLength()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the *ST_PrivateMaxSideLength* attribute of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3) The method *ST_MaxSideLength(DOUBLE PRECISION, INTEGER)* takes the following input parameters:
   a) a DOUBLE PRECISION value *maxsidelength*.
   b) an INTEGER value *triangulate*.
4) For the type preserving method *ST_MaxSideLength(DOUBLE PRECISION, INTEGER)*:

   Case:
   a) If *maxsidelength* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *maxsidelength* is not greater than 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
   c) If SELF is the null value, then return the null value.
   d) Case:
      i) If *triangulate* = 1, then:
         1) let *triangles* = *SELF.Patches()*.
         2) let *elements* = *SELF.TINElements()*.
         3) update *triangles* by applying the implementation-defined triangulation algorithm to the *ST_TINElement* ARRAY *elements* and constrained or modified by *maxsidelength*
         4) return an *ST_TIN* value with the *ST_PrivatePatches* attribute set to *triangles* and the *ST_PrivateMaxSideLength* attribute set to *maxsidelength*.
      ii) Otherwise, return an *ST_TIN* value with the *ST_PrivateMaxSideLength* attribute set to *maxsidelength*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.6.5 ST_TINTable Methods

Purpose

Observe and mutate the ST_TIN value in table format with point references.

Definition

```
CREATE METHOD ST_TINTable()
   RETURNS TABLE
   (item CHARACTER VARYING(30),
    ordernumber INTEGER,
    xcoord DOUBLE PRECISION,
    ycoord DOUBLE PRECISION,
    zcoord DOUBLE PRECISION,
    triangle INTEGER ARRAY[3],
    visibility INTEGER,
    elementID INTEGER,
    elementtag CHARACTER VARYING(64),
    element INTEGER ARRAY[ST_MaxIntegerArrayElements])
   FOR ST_TIN
   BEGIN
      DECLARE triangles ST_Triangle ARRAY[ST_MaxGeometryArrayElements];
      DECLARE elements ST_TINElement ARRAY[ST_MaxGeometryArrayElements];

      -- Get ST_TIN ARRAY values
      triangles = SELF.ST_Patches();
      elements = SELF.ST_TINElements();
      --
      -- See Description
      --
   END

CREATE METHOD ST_TINTable
   (tin_table_name DT1,
    item_column DT2,
    ordernumber_column DT2,
    xcoord_column DT2,
    ycoord_column DT2,
    zcoord_column DT2,
    triangle_column DT2,
    visibility_column DT2,
    elementID_column DT2,
    elementtag_column DT2,
    element_column DT2,
    maxsidelength DOUBLE PRECISION)
   RETURNS ST_TIN
   FOR ST_TIN
   BEGIN
      DECLARE triangles ST_Triangle ARRAY[ST_MaxGeometryArrayElements];
      DECLARE elements ST_TINElement ARRAY[ST_MaxGeometryArrayElements];

      --
      -- See Description
      --
      RETURN SELF.ST_SRID(0).
         ST_Patches(triangles).
         ST_TINElements(elements,0).
         ST_MaxSideLength(maxsidelength,0)
   END
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Definitional Rules

1) *ST_MaxIntegerArrayElements* is the implementation-defined maximum cardinality of an array of *INTEGER* elements.
2) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* elements.
3) *DT1* is data type of variable-length character string with character set SQL_IDENTIFIER and implementation-defined maximum length.
4) *DT2* is data type of variable-length character string with character set SQL_IDENTIFIER and maximum length not less than 128 characters.

Description

1) The method *ST_TINTable()* has no input parameters.
2) The method *TINTable()* returns the following value:
   a) a table value consisting of ten columns:
      i) a column item of type CHARACTER VARYING(30),
      ii) a column ordernumber of type INTEGER,
      iii) a column xcoord of type DOUBLE PRECISION,
      iv) a column *ycoord* of type DOUBLE PRECISION,
      v) a column *zcoord* of type DOUBLE PRECISION,
      vi) a column *triangle* of type INTEGER ARRAY[3],
      vii) a column *visibility* of type INTEGER, having values of -1 (hole), 0 (void), and 1 (visible),
      viii) a column *elementID* of type INTEGER,
      ix) a column *elementtag* of type CHARACTER VARYING(64),
      x) a column *element* of type INTEGER ARRAY[*ST_MaxIntegerArrayElements*].
3) For the null-call method *ST_TINTable()*:
   a) Let *triangles* be the *ST_Triangle* ARRAY returned by *SELF.PrivatePatches()*.
   b) Let *elements* be the *ST_TINElement* ARRAY returned by *SELF.TINElements()*.
   c) Let *points* be the *ST_Point* ARRAY created as follows
      i) For each *ST_Point* value in the *ST_MultiPoint* geometry of a TIN element in *elements* having *ST_ElementType()* = 'random points', add the *ST_Point* value to *points*,
      ii) For each *ST_Point* value in the *ST_MultiPoint* geometry of a TIN element in *elements* having *ST_ElementType()* = 'group spot', add the *ST_Point* value to *points*,
      iii) For each *ST_Point* value in the *ST_Polygon* geometry of a TIN element in *elements* having *ST_ElementType()* = 'boundary', add the *ST_Point* value to *points*,
      iv) For each *ST_Point* value in the *ST_LineString* geometry of a TIN element in *elements* having *ST_ElementType()* = 'breakline', add the *ST_Point* value to *points*,
      v) For each *ST_Point* value in the *ST_LineString* geometry of a TIN element in *elements* having *ST_ElementType()* = 'soft break', add the *ST_Point* value to *points*,
      vi) For each *ST_Point* value in the *ST_LineString* geometry of a TIN element in *elements* having *ST_ElementType()* = 'control contour', add the *ST_Point* value to *points*,
      vii) For each *ST_Point* value in the *ST_Polygon* geometry of a TIN element in *elements* having *ST_ElementType()* = 'break void', add the *ST_Point* value to *points*,
      viii) For each *ST_Point* value in the *ST_Polygon* geometry of a TIN element in *elements* having *ST_ElementType()* = 'drape void', add the *ST_Point* value to *points*,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ix) For each *ST_Point* value in the *ST_Polygon* geometry of a TIN element in *elements* having *ST_ElementType()* = 'void', add the *ST_Point* value to *points*,

x) For each *ST_Point* value in the *ST_Polygon* geometry of a TIN element in *elements* having *ST_ElementType()* = 'hole', add the *ST_Point* value to *points*,

xi) For each *ST_Point* value in the *ST_LineString* geometry of a TIN element in *elements* having *ST_ElementType()* = 'stop line', add the *ST_Point* value to *points*,

xii) Remove duplicate *ST_Point* values from points,

xiii) Reconstruct *points* by sorting the *ST_Point* values first by their *ST_Point.ST_X* value and then by their *ST_Point.ST_Y* value, both in increasing order.

d) Let *CP*, *T, V*, and E be *INTEGER* values.

e) For *CP* = 1 to *CARDINALITY(points)*:

i) Add a row to the TIN table with:

1) *item* = "point",

2) *ordernumber* = CP,

3) *xcoord* = *points[CP].ST_X*,

4) *ycoord* = *points[CP].ST_Y*,

5) *zcoord* = *points[CP].ST_Z*.

f) Output *triangles* as follows:

i) For *T* = 1 to *CARDINALITY(triangles)*:

1) Let *CP1* be an *INTEGER* value such that *points[CP1]* = *triangles[T].ST_Points[1]*.

2) Let *CP2* be an *INTEGER* value such that *points[CP2]* = *triangles[T].ST_Points[2]*.

3) Let *CP3* be an *INTEGER* value such that *points[CP3]* = *triangles[T].ST_Points[3]*.

4) Let *CPA* be an *INTEGER* value that is the smallest of *CP1*, *CP2* and *CP3*.

5) Let *CPB* be an *INTEGER* value that is the median of *CP1*, *CP2* and *CP3*.

6) Let *CPC* be an *INTEGER* value that is the largest of *CP1*, *CP2* and *CP3*.

7) For *E* = 1 to *CARDINALITY(elements)*:

Case

A) if (*elements[E].ElementType()* = 'break void' or *elements[E].ElementType()* = 'drape void' or *elements[E].ElementType()* = 'void') and *triangles[T].ST_Within(elements[E].ST_ElementGeometry())* = 1, then *V* = 0 (zero).

B) if *elements[E].ElementType()* = 'stop line' and *elements[E].ST_ElementGeometry()).ST_Crosses(triangles[T])* = 1, then *V* = 0 (zero).

C) if *elements[E].ElementType()* = 'hole' and *triangles[T].ST_Within(elements[E].ST_ElementGeometry())* = 1, then *V* = -1.

D) otherwise, *V* = 1 (one).

8) Add a row to the TIN table with:

A) *item* = "triangle",

B) *ordernumber* = *T,*

C) *visibility* = *V,*

D) *triangle* = *ARRAY[CPA CPB CPC].*

ii) Sort the triangles in the TIN Table by their *CPA*, then *CPB* and then *CPC* values in increasing order

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

iii) Update the *ordernumber* of the sorted triangles in the TIN Table according to their sorted order position, with the triangle having the lowest *CPA, CPB, CPC* values getting an *ordernumber* equal to 1 (one) up to the triangle having the highest *CPA, CPB, CPC* values getting an *ordernumber* equal to *CARDINALITY(triangles)*.

g) Output *elements* as follows:

i) For *E* = 1 to *CARDINALITY(elements)*:

Case:

1) If *elements[E].ST_ElementGeometry().ST_GeometryType()* = 'ST_LineString':

A) Let *numepoints* be an *INTEGER* value equal to *TREAT(elements[E].ST_ElementGeometry() AS ST_LineString).ST_NumPoints*.

B) Let *CP4* be an *INTEGER* value such that *points[CP4]* = *TREAT(elements[E].ST_ElementGeometry() AS ST_LineString).ST_PointN[1]*.

C) Add a row to the TIN table with:

i) *item* = *elements[E].ElementType()*,

ii) *ordernumber* = *E*,

iii) *elementID* = *elements[E].ElementID()*.

iv) *elementtag* = *elements[E].ElementTag()*.

v) *element* = *ARRAY[CP4]*.

D) Let *epointcounter* be an *INTEGER* value.

E) For *epointcounter* = 2 to *numepoints:*

i) Let *CP5* be an INTEGER value such that *controlpoints[CP5]* = *TREAT(elements[E].ST_ElementGeometry() AS ST_LineString).ST_PointN[epointcounter].*

ii) SET *element* = *element* || *CP5*.

2) If *elements[E].ST_ElementGeometry().ST_GeometryType()* = 'ST_Polygon':

A) Let *numepoints* be an *INTEGER* value equal to *TREAT(elements[E].ST_ElementGeometry() AS ST_Polygon).ST_ExteriorRing().ST_NumPoints*.

B) Let *CP4* be an *INTEGER* value such that *points[CP4]* = *TREAT(elements[E].ST_ElementGeometry() AS ST_Polygon).ST_ExteriorRing().ST_PointN[1].*

C) Add a row to the TIN table with:

i) *item* = *elements[E].ElementType()*,

ii) *ordernumber* = *E*,

iii) *elementID* = *elements[E].ElementID()*.

iv) *elementtag* = *elements[E].ElementTag()*.

v) *element* = *ARRAY[CP4]*.

D) Let *epointcounter* be an *INTEGER* value.

E) For *epointcounter* = 2 to *numepoints:*

i) Let *CP5* be an INTEGER value such that *controlpoints[CP5]* = *TREAT(elements[E].ST_ElementGeometry() AS ST_Polygon).ST_ExteriorRing().ST_PointN[epointcounter].*

ii) SET *element* = *element* || *CP5.*

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

F) If *TREAT(elements[E].ST_ElementGeometry() AS ST_Polygon).ST_NumInteriorRing() > 0*, then

i) Let *numholes* be an INTEGER value equal to *TREAT(elements[E].ST_ElementGeometry() AS ST_Polygon).ST_NumInteriorRing()*.

ii) Let *holecounter* be an INTEGER value.

iii) For *holecounter* = 1 to *numholes*:

1) Let *numipoints* be an INTEGER value equal to *elements[E]. TREAT(ST_ElementGeometry() AS ST_Polygon).ST_InteriorRingN(holecounter).ST_NumPoints()*.

2) Let *CP6* be an INTEGER value such that *points[CP6] = elements[E]. TREAT(ST_ElementGeometry() AS ST_Polygon).ST_InteriorRingN(holecounter).ST_PointN[1]*.

3) Add a row to the TIN table with:

a) *item = elements[E].ElementType()* || ' hole',

b) *ordernumber* = E,

c) *elementID = elements[E].ElementID()*.

d) *elementtag = elements[E].ElementTag()*.

e) *element* = ARRAY[*CP6*].

4) Let *ipointcounter* be an INTEGER value.

5) For *ipointcounter* = 2 to *numipoints*:

a) Let *CP7* be an INTEGER value such that *controlpoints[CP7] = TREAT(elements[E].ST_ElementGeometry() AS ST_Polygon).ST_InteriorRingN(holecounter). ST_PointN[ipointcounter]*.

b) SET *element = element* || *CP7*.

ii) Sort the elements in the TIN Table by their *item* value, then by the *INTEGER* values in their *element* ARRAY, from *element[1]* to *element[CARDINALITY(element)]*, all in increasing order

iii) Update the *ordernumber* of the sorted elements in the TIN Table according to their sorted order position, with the element having the lowest *item* and *INTEGER* values getting an *ordernumber* equal to 1 (one) up to the element having the highest *item* and *INTEGER* values getting an *ordernumber* equal to *CARDINALITY(elements)*.

4) The method *ST_TINTable(DT1, DT2, DT2, DT2, DT2, DT2, DT2, DT2, DT2, DT2, DT2, DOUBLE PRECISION)* takes the following input parameters:

a) a *DT1* value *tin_table_name*, which has the name of a referenced table.

i) The table consists of at least ten columns:

1) an item column of type CHARACTER VARYING(30) which identifies the type of TIN item contained in the row,

2) an ordernumber column of type INTEGER which identifies the position of the item value in sort order,

3) an xcoord column of type DOUBLE PRECISION which contains:

Case:

A) if *item* = 'point', then the x coordinate of the point,

B) otherwise, the NULL value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

4) a *ycoord* column of type DOUBLE PRECISION which contains:

   Case:

   A) if *item* = 'point', then the y coordinate of the point,

   B) otherwise, the NULL value.

5) a *zcoord* column of type DOUBLE PRECISION which contains:

   Case:

   A) if *item* = 'point', then the z coordinate of the point,

   B) otherwise, the NULL value.

6) a *triangle* column of type INTEGER ARRAY[3] which contains:

   Case:

   A) if *item* = 'triangle', then an array of the ordernumbers of the 3 points that define the triangle (the fourth point is always the same as the first so it is not included),

   B) otherwise, the NULL value.

7) a *visibility* column of type INTEGER which contains:

   Case:

   A) if *item* = 'triangle', then the *visibility* value of the triangle,

   B) otherwise, the NULL value.

8) an *elementID* column of type INTEGER which contains:

   Case:

   A) if *item* = 'element', then the *elementID* value of the element,

   B) otherwise, the NULL value.

9) an *elementtag* column of type CHARACTER VARYING(64) which contains:

   Case:

   A) if *item* = 'element', then the *elementtag* value of the element,

   B) otherwise, the NULL value.

10) an *element* column of type INTEGER ARRAY[*ST_MaxIntegerArrayElements*] which contains:

    Case:

    A) if *item* = 'element' and the element geometry is of type 'ST_LineString', then an array of the ordernumbers of the points that define the linestring,

    B) if *item* = 'element' and the element geometry is of type 'ST_Polygon', then an array of the ordernumbers of the points that define the exterior ring of the *ST_Polygon*,

    C) otherwise, the NULL value.

ii) Let *S* be the tin_table_name value.

iii) Let *V* be the character string that is the value of TRIM(BOTH ' ' FROM S).

iv) If *V* value does not conform to the Format and Syntax Rules of <table name> specified in ISO/IEC 9075-2, or the table specified by *paths_table_name* value does not exist in the system, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

b) a *DT2* value *item_column*, which has the name of the item column in the table specified by *tin_table_name*. If the column specified by *item_column* does not exist in the table specified by *tin_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

c) a *DT2* value *ordernumber_column*, which has the name of the ordernumber column in the table specified by *tin_table_name*. If the column specified by *ordernumber_column* does not exist in the table specified by *tin_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

d) a *DT2* value *xcoord_column*, which has the name of the xcoord column in the table specified by *tin_table_name*. If the column specified by *xcoord_column* does not exist in the table specified by *tin_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

e) a *DT2* value *ycoord_column*, which has the name of the *ycoord* column in the table specified by *tin_table_name*. If the column specified by *ycoord_column* does not exist in the table specified by *tin_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

f) a *DT2* value *zcoord_column*, which has the name of the *zcoord* column in the table specified by *tin_table_name*. If the column specified by *zcoord_column* does not exist in the table specified by *tin_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

g) a *DT2* value *triangle_column*, which has the name of the column for triangle in the table specified by *tin_table_name*. If the column specified by *triangle_column* does not exist in the table specified by *tin_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

h) a *DT2* value *visibility_column*, which has the name of the column for visibility in the table specified by *tin_table_name*. If the column specified by *visibility_column* does not exist in the table specified by *tin_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

i) a *DT2* value *elementID_column*, which has the name of the column for *elementID* in the table specified by *tin_table_name*. If the column specified by *elementID_column* does not exist in the table specified by *tin_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

j) a *DT2* value *elementtag_column*, which has the name of the column for *elementtag* in the table specified by *tin_table_name*. If the column specified by *elementtag_column* does not exist in the table specified by *tin_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

k) a *DT2* value *element_column*, which has the name of the column for element in the table specified by *tin_table_name*. If the column specified by *element_column* does not exist in the table specified by *tin_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

l) a DOUBLE PRECISION value *maxsidelength*, which specifies the maximum allowable triangle side length.

5) For the type preserving method *ST_TINTable(DT1, DT2, DT2, DT2, DT2, DT2, DT2, DT2, DT2, DT2, DT2, DOUBLE PRECISION)*:

a) Create a point ARRAY:

i) If there are not at least three UNIQUE rows in the table specified by *tin_table_name* having an *item_column* value of 'point', then an exception condition is raised: *SQL/MM Spatial exception – at least 3 points are required*.

ii) Let *points be an* empty *ST_Point* ARRAY,

iii) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'point':

1) Let *X* be the DOUBLE PRECISION *xcoord_column* value,

2) Let *Y* be the DOUBLE PRECISION *ycoord_column* value,

3) Let *Z* be the DOUBLE PRECISION *zcoord_column* value,

4) Create an *ST_Point* value *CP* as *ST_Point(X, Y, Z)*,

5) SET *points* = *points* || *CP*.

b) Create a triangle ARRAY:

i) Let *triangles be an* empty *ST_Triangle* ARRAY,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'triangle':

1) Let *Tarray* be the INTEGER ARRAY *triangle_column* value,

2) Let *TP1* be an *ST_Point* value equal to the *CP* control point *ST_Point* value created above for the row having an *item_column* value of "controlpoint" and an *index_column* value equal to *Tarray[1]*,

3) Let *TP2* be an *ST_Point* value equal to the *CP* control point *ST_Point* value created above for the row having an *item_column* value of "controlpoint" and an *index_column* value equal to *Tarray[2]*,

4) Let *TP3* be an *ST_Point* value equal to the *CP* control point *ST_Point* value created above for the row having an *item_column* value of "controlpoint" and an *index_column* value equal to *Tarray[3]*,

5) Let *TP4* be an *ST_Point* value equal to *TP1*,

6) Let *Tpointarray[TP1 TP2 TP3 TP4]* be an *ST_Point* ARRAY,

7) Create an *ST_Triangle* value *T* as *ST_Triangle(Tpointarray)*,

8) SET *triangles* = *triangles* || *T*.

c) Create an element ARRAY:

i) Let *elements be an* empty *ST_TINElement* ARRAY.

ii) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'random points':

1) Let *elementtype* be the CHARACTER VARYING *item_column* value.

2) Let *elementID* be the INTEGER *elementID_column* value.

3) Let *elementtag* be the CHARACTER VARYING *elementtag_column* value.

4) Let *integerarray* be the INTEGER ARRAY *element_column* value.

5) Let *numpoints* be an INTEGER value equal to *CARDINALITY(integerarray)*.

6) Let *P1* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *ordernumber_column* value equal to *integerarray[1]*.

7) Let *pointarray[P1]* be an *ST_Point* ARRAY.

8) Let *pointcounter* be an INTEGER value.

9) For *pointcounter* = 2 to *numpoints:*

A) Let *P* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *index_column* value equal to *integerarray[pointcounter]*.

B) SET *pointarray* = *pointarray* || *P.*

10) Create an *ST_MultiPoint* value *elementgeometry* as *ST_MultiPoint(pointarray)*,

11) Create an *element* value *element* as *ST_TINElement(elementtype, elementID, elementtag, elementgeometry)*,

12) SET *elements* = *elements* || *element*.

iii) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'group spot':

1) Let *elementtype* be the CHARACTER VARYING *item_column* value.

2) Let *elementID* be the INTEGER *elementID_column* value.

3) Let *elementtag* be the CHARACTER VARYING *elementtag_column* value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

4) Let *integerarray* be the INTEGER ARRAY *element_column* value.

5) Let *numpoints* be an INTEGER value equal to *CARDINALITY(integerarray)*.

6) Let *P1* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *ordernumber_column* value equal to *integerarray[1]*.

7) Let *pointarray[P1]* be an *ST_Point* ARRAY.

8) Let *pointcounter* be an INTEGER value.

9) For *pointcounter* = 2 to *numpoints:*

   A) Let *P* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *index_column* value equal to *integerarray[pointcounter]*.

   B) SET *pointarray* = *pointarray* || *P.*

10) Create an *ST_MultiPoint* value *elementgeometry* as *ST_MultiPoint(pointarray)*,

11) Create an *element* value *element* as *ST_TINElement(elementtype, elementID, elementtag, elementgeometry)*,

12) SET *elements* = *elements* || *element*.

iv) For the UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'boundary':

1) Let *elementtype* be the CHARACTER VARYING *item_column* value.

2) Let *elementID* be the INTEGER *elementID_column* value.

3) Let *elementtag* be the CHARACTER VARYING *elementtag_column* value.

4) Let *integerarray* be the INTEGER ARRAY *element_column* value.

5) Let *numpoints* be an INTEGER value equal to *CARDINALITY(integerarray)*.

6) Let *P1* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *ordernumber_column* value equal to *integerarray[1]*.

7) Let *pointarray[P1]* be an *ST_Point* ARRAY.

8) Let *pointcounter* be an INTEGER value.

9) For *pointcounter* = 2 to *numpoints:*

   A) Let *P* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *index_column* value equal to *integerarray[pointcounter]*.

   B) SET *pointarray* = *pointarray* || *P.*

10) Create an *ST_Polygon* value *elementgeometry* as *ST_Polygon(ST_LineString(pointarray))*,

11) Create an *element* value *element* as *ST_TINElement(elementtype, elementID, elementtag, elementgeometry)*,

12) SET *elements* = *elements* || *element*.

v) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'breakline':

1) Let *elementtype* be the CHARACTER VARYING *item_column* value.

2) Let *elementID* be the INTEGER *elementID_column* value.

3) Let *elementtag* be the CHARACTER VARYING *elementtag_column* value.

4) Let *integerarray* be the INTEGER ARRAY *element_column* value.

5) Let *numpoints* be an INTEGER value equal to *CARDINALITY(integerarray)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6) Let *P1* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *ordernumber_column* value equal to *integerarray[1]*.

7) Let *pointarray[P1]* be an *ST_Point* ARRAY.

8) Let *pointcounter* be an INTEGER value.

9) For *pointcounter* = 2 to *numpoints:*

   A) Let *P* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *index_column* value equal to *integerarray[pointcounter]*.

   B) SET *pointarray* = *pointarray* || *P.*

10) Create an *ST_Linestring* value *elementgeometry* as *ST_LineString(pointarray)*,

11) Create an *element* value *element* as *ST_TINElement(elementtype, elementID, elementtag, elementgeometry)*,

12) SET *elements* = *elements* || *element*.

vi) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'soft break':

1) Let *elementtype* be the CHARACTER VARYING *item_column* value.

2) Let *elementID* be the INTEGER *elementID_column* value.

3) Let *elementtag* be the CHARACTER VARYING *elementtag_column* value.

4) Let *integerarray* be the INTEGER ARRAY *element_column* value.

5) Let *numpoints* be an INTEGER value equal to *CARDINALITY(integerarray)*.

6) Let *P1* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *ordernumber_column* value equal to *integerarray[1]*.

7) Let *pointarray[P1]* be an *ST_Point* ARRAY.

8) Let *pointcounter* be an INTEGER value.

9) For *pointcounter* = 2 to *numpoints:*

   A) Let *P* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *index_column* value equal to *integerarray[pointcounter]*.

   B) SET *pointarray* = *pointarray* || *P.*

10) Create an *ST_Linestring* value *elementgeometry* as *ST_LineString(pointarray)*,

11) Create an *element* value *element* as *ST_TINElement(elementtype, elementID, elementtag, elementgeometry)*,

12) SET *elements* = *elements* || *element*.

vii) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'control contour':

1) Let *elementtype* be the CHARACTER VARYING *item_column* value.

2) Let *elementID* be the INTEGER *elementID_column* value.

3) Let *elementtag* be the CHARACTER VARYING *elementtag_column* value.

4) Let *integerarray* be the INTEGER ARRAY *element_column* value.

5) Let *numpoints* be an INTEGER value equal to *CARDINALITY(integerarray)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6) Let *P1* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *ordernumber_column* value equal to *integerarray[1]*.

7) Let *pointarray[P1]* be an *ST_Point* ARRAY.

8) Let *pointcounter* be an INTEGER value.

9) For *pointcounter* = 2 to *numpoints:*

   A) Let *P* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *index_column* value equal to *integerarray[pointcounter]*.

   B) SET *pointarray* = *pointarray* || *P.*

10) Create an *ST_Linestring* value *elementgeometry* as *ST_LineString(pointarray)*,

11) Create an *element* value *element* as *ST_TINElement(elementtype, elementID, elementtag, elementgeometry)*,

12) SET *elements* = *elements* || *element*.

viii) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'break void':

1) Let *elementtype* be the CHARACTER VARYING *item_column* value.

2) Let *elementID* be the INTEGER *elementID_column* value.

3) Let *elementtag* be the CHARACTER VARYING *elementtag_column* value.

4) Let *integerarray* be the INTEGER ARRAY *element_column* value.

5) Let *numpoints* be an INTEGER value equal to *CARDINALITY(integerarray)*.

6) Let *P1* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *ordernumber_column* value equal to *integerarray[1]*.

7) Let *pointarray[P1]* be an *ST_Point* ARRAY.

8) Let *pointcounter* be an INTEGER value.

9) For *pointcounter* = 2 to *numpoints:*

   A) Let *P* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *index_column* value equal to *integerarray[pointcounter]*.

   B) SET *pointarray* = *pointarray* || *P.*

10) Create an *ST_Polygon* value *elementgeometry* as *ST_Polygon(ST_LineString(pointarray))*,

11) Create an *element* value *element* as *ST_TINElement(elementtype, elementID, elementtag, elementgeometry)*,

12) SET *elements* = *elements* || *element*.

ix) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'break void hole':

1) Let *holeintegerarray* be the INTEGER ARRAY *element_column* value.

2) Let *numholepoints* be an INTEGER value equal to *CARDINALITY(holeintegerarray)*.

3) Let *HP1* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *ordernumber_column* value equal to *holeintegerarray[1]*.

4) Let *holepointarray[HP1]* be an *ST_Point* ARRAY.

5) Let *holepointcounter* be an INTEGER value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6) For *holepointcounter* = 2 to *numholepoints*:

   A) Let *HP* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *index_column* value equal to *holeintegerarray[holepointcounter]*.

   B) SET *holepointarray* = *holepointarray* || *HP*.

7) Create an *ST_LineString* value *holering* as *ST_LineString(holepointarray)*.

8) Let *HE* be an INTEGER value such that *elements[HE].ST_ElementType* = 'break void' and *holering.ST_Within(elements[HE].ST_ElementGeometry().ST_ExteriorRing()) = 1* and *ST_Area(elements[HE].ST_ElementGeometry().ST_ExteriorRing().ST_Envelope)* is the smallest area of all break void elements created above.

9) Let *holesarray* be the *ST_Curve* ARRAY equal to *elements[HE].ST_ElementGeometry().ST_InteriorRings()*.

10) SET *holesarray* = *holesarray* || *holering*.

11) Add the holes found so far to the break void element using *elements[HE].ST_ElementGeometry.ST_InteriorRings(holesarray)*.

x) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'drape void':

1) Let *elementtype* be the CHARACTER VARYING *item_column* value.

2) Let *elementID* be the INTEGER *elementID_column* value.

3) Let *elementtag* be the CHARACTER VARYING *elementtag_column* value.

4) Let *integerarray* be the INTEGER ARRAY *element_column* value.

5) Let *numpoints* be an INTEGER value equal to *CARDINALITY(integerarray)*.

6) Let *P1* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *ordernumber_column* value equal to *integerarray[1]*.

7) Let *pointarray[P1]* be an *ST_Point* ARRAY.

8) Let *pointcounter* be an INTEGER value.

9) For *pointcounter* = 2 to *numpoints:*

   A) Let *P* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *index_column* value equal to *integerarray[pointcounter]*.

   B) SET *pointarray* = *pointarray* || *P.*

10) Create an *ST_Polygon* value *elementgeometry* as *ST_Polygon(ST_LineString(pointarray))*,

11) Create an *element* value *element* as *ST_TINElement(elementtype, elementID, elementtag, elementgeometry)*,

12) SET *elements* = *elements* || *element*.

xi) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'drape void hole':

1) Let *holeintegerarray* be the INTEGER ARRAY *element_column* value.

2) Let *numholepoints* be an INTEGER value equal to *CARDINALITY(holeintegerarray)*.

3) Let *HP1* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *ordernumber_column* value equal to *holeintegerarray[1]*.

4) Let *holepointarray[HP1]* be an *ST_Point* ARRAY.

5) Let *holepointcounter* be an INTEGER value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6) For *holepointcounter* = *2* to *numholepoints*:

A) Let *HP* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *index_column* value equal to *holeintegerarray[holepointcounter]*.

B) SET *holepointarray* = *holepointarray* || *HP*.

7) Create an *ST_LineString* value *holering* as *ST_LineString(holepointarray)*.

8) Let *HE* be an INTEGER value such that *elements[HE].ST_ElementType* = 'drape void' and *holering.ST_Within(elements[HE].ST_ElementGeometry().ST_ExteriorRing()) = 1* and *ST_Area(elements[HE].ST_ElementGeometry().ST_ExteriorRing().ST_Envelope)* is the smallest area of all drape void elements created above.

9) Let *holesarray* be the *ST_Curve* ARRAY equal to *elements[HE].ST_ElementGeometry().ST_InteriorRings()*.

10) SET *holesarray* = *holesarray* || *holering*.

11) Add the holes found so far to the drape void element using *elements[HE].ST_ElementGeometry.ST_InteriorRings(holesarray)*.

xii) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'void':

1) Let *elementtype* be the CHARACTER VARYING *item_column* value.

2) Let *elementID* be the INTEGER *elementID_column* value.

3) Let *elementtag* be the CHARACTER VARYING *elementtag_column* value.

4) Let *integerarray* be the INTEGER ARRAY *element_column* value.

5) Let *numpoints* be an INTEGER value equal to *CARDINALITY(integerarray)*.

6) Let *P1* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *ordernumber_column* value equal to *integerarray[1]*.

7) Let *pointarray[P1]* be an *ST_Point* ARRAY.

8) Let *pointcounter* be an INTEGER value.

9) For *pointcounter* = 2 to *numpoints:*

A) Let *P* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *index_column* value equal to *integerarray[pointcounter]*.

B) SET *pointarray* = *pointarray* || *P.*

10) Create an *ST_Polygon* value *elementgeometry* as *ST_Polygon(ST_LineString(pointarray))*,

11) Create an *element* value *element* as *ST_TINElement(elementtype, elementID, elementtag, elementgeometry)*,

12) SET *elements* = *elements* || *element*.

xiii) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'void hole':

1) Let *holeintegerarray* be the INTEGER ARRAY *element_column* value.

2) Let *numholepoints* be an INTEGER value equal to *CARDINALITY(holeintegerarray)*.

3) Let *HP1* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *ordernumber_column* value equal to *holeintegerarray[1]*.

4) Let *holepointarray[HP1]* be an *ST_Point* ARRAY.

5) Let *holepointcounter* be an INTEGER value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6) For *holepointcounter = 2* to *numholepoints*:

   A) Let *HP* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *index_column* value equal to *holeintegerarray[holepointcounter]*.

   B) SET *holepointarray = holepointarray || HP*.

7) Create an *ST_LineString* value *holering* as *ST_LineString(holepointarray)*.

8) Let *HE* be an INTEGER value such that *elements[HE].ST_ElementType* = 'void' and *holering.ST_Within(elements[HE].ST_ElementGeometry().ST_ExteriorRing()) = 1* and *ST_Area(elements[HE].ST_ElementGeometry().ST_ExteriorRing().ST_Envelope)* is the smallest area of all void elements created above.

9) Let holesarray be the *ST_Curve* ARRAY equal to *elements[HE].ST_ElementGeometry().ST_InteriorRings()*.

10) SET *holesarray = holesarray || holering*.

11) Add the holes found so far to the void element using *elements[HE].ST_ElementGeometry.ST_InteriorRings(holesarray)*.

xiv) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'hole':

1) Let *elementtype* be the CHARACTER VARYING *item_column* value.

2) Let *elementID* be the INTEGER *elementID_column* value.

3) Let *elementtag* be the CHARACTER VARYING *elementtag_column* value.

4) Let *integerarray* be the INTEGER ARRAY *element_column* value.

5) Let *numpoints* be an INTEGER value equal to *CARDINALITY(integerarray)*.

6) Let *P1* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *ordernumber_column* value equal to *integerarray[1]*.

7) Let *pointarray[P1]* be an *ST_Point* ARRAY.

8) Let *pointcounter* be an INTEGER value.

9) For *pointcounter* = 2 to *numpoints:*

   A) Let *P* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *index_column* value equal to *integerarray[pointcounter]*.

   B) SET *pointarray = pointarray || P.*

10) Create an *ST_Polygon* value *elementgeometry* as *ST_Polygon(ST_LineString(pointarray))*,

11) Create an *element* value *element* as *ST_TINElement(elementtype, elementID, elementtag, elementgeometry)*,

12) SET *elements = elements || element*.

xv) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'hole hole':

1) Let *holeintegerarray* be the INTEGER ARRAY *element_column value*.

2) Let *numholepoints* be an INTEGER value equal to *CARDINALITY(holeintegerarray)*.

3) Let *HP1* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *ordernumber_column* value equal to *holeintegerarray[1]*.

4) Let *holepointarray[HP1]* be an *ST_Point* ARRAY.

5) Let *holepointcounter* be an INTEGER value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6) For *holepointcounter* = 2 to *numholepoints*:

A) Let *HP* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *index_column* value equal to *holeintegerarray[holepointcounter]*.

B) SET *holepointarray* = *holepointarray* || *HP*.

7) Create an *ST_LineString* value *holering* as *ST_LineString(holepointarray)*.

8) Let *HE* be an INTEGER value such that *elements[HE].ST_ElementType* = 'hole' and *holering.ST_Within(elements[HE].ST_ElementGeometry().ST_ExteriorRing()) = 1* and *ST_Area(elements[HE].ST_ElementGeometry().ST_ExteriorRing().ST_Envelope)* is the smallest area of all hole elements created above.

9) Let *holesarray* be the *ST_Curve* ARRAY equal to *elements[HE].ST_ElementGeometry().ST_InteriorRings()*.

10) SET *holesarray* = *holesarray* || *holering*.

11) Add the holes found so far to the hole element using *elements[HE].ST_ElementGeometry.ST_InteriorRings(holesarray)*.

xvi) For each UNIQUE row in the table specified by *tin_table_name* having an *item_column* value of 'stop line':

1) Let *elementtype* be the CHARACTER VARYING *item_column* value.

2) Let *elementID* be the INTEGER *elementID_column* value.

3) Let *elementtag* be the CHARACTER VARYING *elementtag_column* value.

4) Let *integerarray* be the INTEGER ARRAY *element_column* value.

5) Let *numpoints* be an INTEGER value equal to *CARDINALITY(integerarray)*.

6) Let *P1* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *ordernumber_column* value equal to *integerarray[1]*.

7) Let *pointarray[P1]* be an *ST_Point* ARRAY.

8) Let *pointcounter* be an INTEGER value.

9) For *pointcounter* = 2 to *numpoints:*

A) Let *P* be an *ST_Point* value equal to the *CP* point *ST_Point* value created above for the row having an *item_column* value of 'point' and an *index_column* value equal to *integerarray[pointcounter]*.

B) SET *pointarray* = *pointarray* || *P.*

10) Create an *ST_Linestring* value *elementgeometry* as *ST_LineString(pointarray)*,

11) Create an *element* value *element* as *ST_TINElement(elementtype, elementID, elementtag, elementgeometry)*,

12) SET *elements* = *elements* || *element*.

d) If *CARDINALITY(triangles)* = 0, create *ST_Triangles* by applying the implementation-defined triangulation algorithm to the *ST_Point* ARRAY *points* and constrained or modified by *elements* and *maxsidelength*.

e) Return an *ST_TIN* value with:

i) The spatial reference system identifier set to 0 (zero).

ii) Using the method *ST_Patches(ST_Triangle* ARRAY*)*:

1) the *ST_PrivateDimension* attribute set to 2,

2) the *ST_PrivateCoordinateDimension* attribute set to the value expression *ST_GetCoordDim(triangles)*,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3) the *ST_PrivateIs3D* attribute set to 1 (one),

4) the *ST_PrivateIsMeasured* attribute set to the value expression *ST_GetIsMeasured(triangles),*

5) the *ST_PrivatePatches* attribute set to *triangles*.

iii) Using the method *ST_TINElements(ST_TINElement* ARRAY*, INTEGER)*, the *ST_PrivateElements* attribute set to *elements*.

iv) Using the method *ST_MaxSideLength(DOUBLE PRECISION, INTEGER)*, the *ST_PrivateMaxSideLength* attribute set to *maxsidelength*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.6.6 ST_Clip Method

Purpose

Returns that part of an ST_TIN value that is within the clipping boundary.

Definition

```
CREATE METHOD ST_Clip
   (clippolygon ST_Polygon)
   RETURNS ST_TIN
   FOR ST_TIN
   BEGIN
      --
      --  See Description
      --
   END
```

Description

1) The method *ST_Clip(ST_Polygon)* takes the following input parameters:
   a) an *ST_Polygon* value *clippolygon*.
2) For the type preserving method *ST_Clip(ST_Polygon)*:
   a) The ST_Polygon value *clippolygon* becomes the new boundary for *SELF*.
   b) That part of *SELF* which is outside of this boundary is removed.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.6.7 ST_Patches Methods

Purpose

Observe and mutate the ST_PrivatePatches attribute of an ST_TIN value.

Definition

```
CREATE METHOD ST_Patches()
   RETURNS ST_Triangle ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_TIN
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_PrivatePatches
      END

CREATE METHOD ST_Patches
   (triangles ST_Triangle ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_TIN
   FOR ST_TIN
   BEGIN
      -- If triangles is not an ST_Triangle ARRAY, then raise an exception
      IF triangles IS NOT OF (ST_Triangle ARRAY) THEN
         SIGNAL SQLSTATE '2FF67'
            SET MESSAGE_TEXT = 'polygon value is not a triangle value';
      END IF;
      RETURN (SELF AS ST_PolyhdrlSurface).ST_Patches(triangles);
   END
```

Description

1) The method *ST_Patches()* has no input parameters.
2) For the null-call method *ST_Patches()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the *ST_PrivatePatches* attribute of SELF.
3) The method *ST_Patches(ST_Triangle ARRAY)* takes the following input parameters:

   a) an *ST_Triangle* value *apolygonarray*.
4) For the type preserving method *ST_Patches(ST_Triangle ARRAY)*:

   Case:

   a) If *triangles* is not an ST_Triangle ARRAY value, then an exception condition is raised: *SQL/MM Spatial exception – polygon value is not a triangle value*.

   b) Otherwise, return an *ST_TIN* value as a result of the value expression: *(SELF AS ST_PolyhdrlSurface).ST_Patches(triangles).*

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.6.8 ST_TINFromText Functions

Purpose

Return an ST_TIN value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_TIN value.

Definition

```
CREATE FUNCTION ST_TINFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_TIN
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_TINFromText(awkt, 0)

CREATE FUNCTION ST_TINFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_TIN
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_TINFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_TINFromText(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_TINFromText(awkt, 0)*.
3) The function *ST_TINFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_TINFromText(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkt* is the well-known text representation of an *ST_TIN* value.

      If *awkt* is not producible in the BNF for <tin text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromText(awkt, ansrid) AS ST_TIN)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.6.9 ST_TINFromWKB Functions

Purpose

Return an ST_TIN value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_TIN value.

Definition

```
CREATE FUNCTION ST_TINFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_TIN
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_TINFromWKB(awkb, 0)

CREATE FUNCTION ST_TINFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_TIN
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_TINFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_TINFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_TINFromWKB(awkb, 0)*.
3) The function *ST_TINFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_TINFromWKB(BINARY LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkb* is the well-known binary representation of an *ST_TIN* value.

      If *awkb* is not producible in the BNF for <tin binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_TIN)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8.6.10 ST_TINFromGML Functions

Purpose

Return an ST_TIN value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representation of an ST_TIN value

Definition

```
CREATE FUNCTION ST_TINFromGML
    (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
    RETURNS ST_TIN
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    RETURN ST_TINFromGML(agml, 0)

CREATE FUNCTION ST_TINFromGML
    (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
     ansrid INTEGER)
    RETURNS ST_TIN
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    BEGIN
       --
       -- See Description
       --
    END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_TINFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_TINFromGML*(CHARACTER LARGE OBJECT) returns the result of the value expression: *ST_TINFromGML(agml, 0)*.
3) The function *ST_TINFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_TINFromGML(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If the parameter *agml* does not contain a TIN XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromGML(agml, ansrid) AS ST_TIN)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9 Geometry Collection Types

9.1 ST_GeomCollection Type and Routines

9.1.1 ST_GeomCollection Type

Purpose

The ST_GeomCollection type is a subtype of ST_Geometry and represents a collection of zero or more ST_Geometry values.

Definition

```
CREATE TYPE ST_GeomCollection
   UNDER ST_Geometry
   AS (
      ST_PrivateGeometries ST_Geometry
         ARRAY[ST_MaxGeometryArrayElements] DEFAULT ARRAY[]
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_GeomCollection
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
      RETURNS ST_GeomCollection
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_GeomCollection
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
       ansrid INTEGER)
      RETURNS ST_GeomCollection
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_GeomCollection
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
      RETURNS ST_GeomCollection
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_GeomCollection
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
       ansrid INTEGER)
      RETURNS ST_GeomCollection
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_GeomCollection
   (ageometry ST_Geometry)
   RETURNS ST_GeomCollection
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_GeomCollection
   (ageometry ST_Geometry,
    ansrid INTEGER)
   RETURNS ST_GeomCollection
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_GeomCollection
   (ageometryarray ST_Geometry
      ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_GeomCollection
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_GeomCollection
   (ageometryarray ST_Geometry
      ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_GeomCollection
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Geometries()
   RETURNS ST_Geometry ARRAY[ST_MaxGeometryArrayElements]
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Geometries
   (ageometryarray ST_Geometry
      ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_GeomCollection
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_NumGeometries()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_GeometryN
   (aposition INTEGER)
   RETURNS ST_Geometry
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.
4) The attribute *ST_PrivateGeometries* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST_PrivateGeometries*.

Description

1) The *ST_GeomCollection* type provides for public use:
   a) a method *ST_GeomCollection(CHARACTER LARGE OBJECT)*,
   b) a method *ST_GeomCollection(CHARACTER LARGE OBJECT, INTEGER)*,
   c) a method *ST_GeomCollection(BINARY LARGE OBJECT)*,
   d) a method *ST_GeomCollection(BINARY LARGE OBJECT, INTEGER)*,
   e) a method *ST_GeomCollection(ST_Geometry)*,
   f) a method *ST_GeomCollection(ST_Geometry, INTEGER)*,
   g) a method *ST_GeomCollection(ST_Geometry ARRAY)*,
   h) a method *ST_GeomCollection(ST_Geometry ARRAY, INTEGER)*,
   i) a method *ST_Geometries()*,
   j) a method *ST_Geometries(ST_Geometry ARRAY)*,
   k) a method *ST_NumGeometries()*,
   l) a method *ST_GeometryN(INTEGER)*,
   m) a function *ST_GeomCollFromTxt(CHARACTER LARGE OBJECT)*,
   n) a function *ST_GeomCollFromTxt(CHARACTER LARGE OBJECT, INTEGER)*,
   o) a function *ST_GeomCollFromWKB(BINARY LARGE OBJECT)*,
   p) a function *ST_GeomCollFromWKB(BINARY LARGE OBJECT, INTEGER)*,
   q) a function *ST_GeomCollFromGML(CHARACTER LARGE OBJECT)*,
   r) a function *ST_GeomCollFromGML(CHARACTER LARGE OBJECT, INTEGER)*.
2) The *ST_PrivateGeometries* attribute contains the collection of *ST_Geometry* values.
3) The *ST_PrivateGeometries* attribute shall not be the null value. The elements in the *ST_PrivateGeometries* attribute shall not be the null value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

4) The coordinate dimension of an *ST_GeomCollection* value is the number of coordinate values associated with the position.

5) The dimension of an *ST_GeomCollection* value is the maximum dimension value of all the *ST_Geometry* values in the *ST_PrivateGeometries* attribute.

6) An *ST_GeomCollection* value returned by the constructor function corresponds to the empty set.

7) An *ST_GeomCollection* value with no elements in the *ST_PrivateGeometries* attribute corresponds to the empty set.

8) Subtypes of *ST_GeomCollection* may restrict membership based on dimension and may place other constraints such as the degree that the elements spatially intersect between *ST_Geometry* values.

9) A value with the most specific type of *ST_GeomCollection* is simple if:

   a) all the elements in the *ST_PrivateGeometries* attribute are simple.

   b) the interior of any element in the *ST_PrivateGeometries* attribute does not intersect the interior of any other element in the *ST_PrivateGeometries* attribute.

10) An *ST_GeomCollection* value is well formed only if all of the *ST_Geometry* values in *ST_PrivateGeometries* attribute are well formed.

11) All the *ST_Geometry* values in the *ST_PrivateGeometries* attribute shall be in the same spatial reference system as the *ST_GeomCollection* value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.1.2 ST_GeomCollection Methods

Purpose

Return an ST_GeomCollection value constructed from either the well-known text representation, the well-known binary representation, the GML representation, or the specified ST_Geometry values.

Definition

```
CREATE CONSTRUCTOR METHOD ST_GeomCollection
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_GeomCollection
   FOR ST_GeomCollection
   RETURN NEW ST_GeomCollection(awktorgml, 0)

CREATE CONSTRUCTOR METHOD ST_GeomCollection
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_GeomCollection
   FOR ST_GeomCollection
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_GeomCollection
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_GeomCollection
   FOR ST_GeomCollection
   RETURN NEW ST_GeomCollection(awkb, 0)

CREATE CONSTRUCTOR METHOD ST_GeomCollection
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_GeomCollection
   FOR ST_GeomCollection
   RETURN ST_GeomCollFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_GeomCollection
   (ageometry ST_Geometry)
   RETURNS ST_GeomCollection
   FOR ST_GeomCollection
   RETURN SELF.ST_SRID(ageometry.ST_SRID()).
      ST_Geometries(ARRAY[ageometry])

CREATE CONSTRUCTOR METHOD ST_GeomCollection
   (ageometry ST_Geometry,
    ansrid INTEGER)
   RETURNS ST_GeomCollection
   FOR ST_GeomCollection
   RETURN SELF.ST_SRID(ansrid).ST_Geometries(ARRAY[ageometry])

CREATE CONSTRUCTOR METHOD ST_GeomCollection
   (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_GeomCollection
   FOR ST_GeomCollection
   RETURN SELF.ST_SRID(ST_CheckSRID(ageometryarray)).
      ST_Geometries(ageometryarray)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CREATE CONSTRUCTOR METHOD ST_GeomCollection
   (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_GeomCollection
   FOR ST_GeomCollection
   RETURN SELF.ST_SRID(ansrid).ST_Geometries(ageometryarray)
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The method *ST_GeomCollection(CHARACTER LARGE OBJECT)* takes the following input parameter:
   a) a CHARACTER LARGE OBJECT value *awktorgml*.
2) The null-call type preserving SQL-invoked constructor method *ST_GeomCollection(CHARACTER LARGE OBJECT)* returns the result of the value expression: *NEW ST_GeomCollection(awktorgml, 0)*.
3) The method *ST_GeomCollection(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awktorgml*,
   b) an INTEGER value *ansrid*.
4) For the null-call type preserving SQL-invoked constructor method *ST_GeomCollection(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If *awktorgml* contains a MultiGeometry XML element in the GML representation, then return the result of the value expression: *ST_GeomCollFromGML(awktorgml, ansrid)*.
   b) Otherwise, return the result of the value expression: *ST_GeomCollFromTxt(awktorgml, ansrid)*.
5) The method *ST_GeomCollection(BINARY LARGE OBJECT)* takes the following input parameter:
   a) a BINARY LARGE OBJECT value *awkb*.
6) The null-call type preserving SQL-invoked constructor method *ST_GeomCollection(BINARY LARGE OBJECT)* returns the result of the value expression: *NEW ST_GeomCollection(awkb, 0)*.
7) The method *ST_GeomCollection(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
8) The null-call type preserving SQL-invoked constructor method *ST_GeomCollection(BINARY LARGE OBJECT, INTEGER)* returns the result of the value expression: *ST_GeomCollFromWKB(awkb, ansrid)*.
9) The method *ST_GeomCollection(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
10) The null-call type preserving SQL-invoked constructor method *ST_GeomCollection(ST_Geometry)* returns an *ST_GeomCollection* value with:
    a) The spatial reference system identifier set to the spatial reference system identifier of *ageometry*.
    b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the *ST_PrivateGeometries* attribute set to ARRAY[*ageometry*], the *ST_PrivateDimension* attribute set to *ST_MaxDimension(ARRAY[ageometry])*, and the *ST_PrivateCoordinateDimension* attribute set to 2.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11) The method *ST_GeomCollection(ST_Geometry, INTEGER)* takes the following input parameters:

    a) an *ST_Geometry* value *ageometry*,

    b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_GeomCollection(ST_Geometry, INTEGER)* returns an *ST_GeomCollection* value with:

    a) The spatial reference system identifier set to *ansrid*.

    b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the *ST_PrivateGeometries* attribute set to *ARRAY[ageometry]*, the *ST_PrivateDimension* attribute set to *ST_MaxDimension(ARRAY[ageometry])*, and the *ST_PrivateCoordinateDimension* attribute set to 2.

13) The method *ST_GeomCollection(ST_Geometry ARRAY)* takes the following input parameters:

    a) an *ST_Geometry* ARRAY value *ageometryarray*.

14) The null-call type preserving SQL-invoked constructor method *ST_GeomCollection(ST_Geometry ARRAY)* returns an *ST_GeomCollection* value with:

    a) The spatial reference system identifier set to the value expression: *ST_CheckSRID(ageometryarray)*.

    b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the *ST_PrivateGeometries* attribute set to *ageometryarray*, the *ST_PrivateDimension* attribute set to *ST_MaxDimension(ARRAY[ageometry])*, and the *ST_PrivateCoordinateDimension* attribute set to 2.

15) The method *ST_GeomCollection(ST_Geometry ARRAY, INTEGER)* takes the following input parameters:

    a) an *ST_Geometry* ARRAY value *ageometryarray*,

    b) an INTEGER value *ansrid*.

16) The null-call type preserving SQL-invoked constructor method *ST_GeomCollection(ST_Geometry ARRAY, INTEGER)* returns an *ST_GeomCollection* value with:

    a) The spatial reference system identifier set to *ansrid*.

    b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the *ST_PrivateGeometries* attribute set to *ageometryarray*, the *ST_PrivateDimension* attribute set to *ST_MaxDimension(ARRAY[ageometry])*, and the *ST_PrivateCoordinateDimension* attribute set to 2.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.1.3 ST_Geometries Methods

Purpose

Observe and mutate the ST_PrivateGeometries attribute of an ST_GeomCollection value.

Definition

```
CREATE METHOD ST_Geometries()
   RETURNS ST_Geometry ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_GeomCollection
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            SELF.ST_PrivateGeometries
      END

CREATE METHOD ST_Geometries
   (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_GeomCollection
   FOR ST_GeomCollection
   BEGIN
      -- If ageometryarray is the null value or contains null elements,
      -- then raise an exception.
      CALL ST_CheckNulls(ageometryarray);
      -- If SELF is the null value, then return the null value.
      IF SELF IS NULL THEN
         RETURN CAST (NULL AS ST_GeomCollection);
      END IF;
      -- Check that there are no mixed spatial reference
      -- systems between SELF and ageometryarray.
      IF (CARDINALITY(ageometryarray) > 0) AND
         (SELF.ST_SRID() <> ST_CheckSRID(ageometryarray)) THEN
         SIGNAL SQLSTATE '2FF10'
            SET MESSAGE_TEXT = 'mixed spatial reference systems';
      END IF;
      RETURN
         SELF.ST_PrivateDimension(ST_MaxDimension(ageometryarray)).
            ST_PrivateCoordinateDimension(ST_GetCoordDim(ageometryarray)).
            ST_PrivateIs3D(ST_GetIs3D(ageometryarray)).
            ST_PrivateIsMeasured(ST_GetIsMeasured(ageometryarray)).
            ST_PrivateGeometries(ageometryarray);
   END
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

Description

1) The method *ST_Geometries()* has no input parameters.
2) For the null-call method *ST_Geometries()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the value of the *ST_PrivateGeometries* attribute.
3) The method *ST_Geometries(ST_Geometry ARRAY)* takes the following input parameters:

   a) an *ST_Geometry* ARRAY value *ageometryarray*.
4) For the type preserving method *ST_Geometries(ST_Geometry ARRAY)*:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

a) Call the procedure *ST_CheckNulls(ST_Geometry ARRAY)* to check if *ageometryarray* is the null value or contains null elements.

b) Case:

   i) If SELF is the null value, then return the null value.

   ii) If the cardinality of *ageometryarray* is greater than 0 (zero) and the spatial reference system of SELF is not equal to *ST_CheckSRID(ageometryarray)*, then an exception condition is raised: *SQL/MM Spatial exception – mixed spatial reference systems*.

   iii) Otherwise, return an *ST_GeomCollection* value with:

      1) The dimension set to *ST_MaxDimension(ageometryarray)*.

      2) The coordinate dimension set to the value expression: *ST_GetCoordDim(ageometryarray)*.

      3) The *ST_PrivateIs3D* attribute set to the value expression: *ST_GetIs3D(ageometryarray)*.

      4) The *ST_PrivateIsMeasured* attribute set to the value expression: *ST_GetIsMeasured(ageometryarray)*.

      5) The *ST_PrivateGeometries* attribute set to *ageometryarray*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.1.4 ST_NumGeometries Method

Purpose

Return the cardinality of the ST_PrivateGeometries attribute of an ST_GeomCollection value.

Definition

```
CREATE METHOD ST_NumGeometries()
   RETURNS INTEGER
   FOR ST_GeomCollection
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CARDINALITY(SELF.ST_PrivateGeometries)
      END
```

Description

1) The method *ST_NumGeometries()* has no input parameters.
2) For the null-call method *ST_NumGeometries()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise, return the cardinality of the *ST_PrivateGeometries* attribute.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.1.5 ST_GeometryN Method

Purpose

Return the specified ST_Geometry value in the ST_PrivateGeometries attribute of an ST_GeomCollection value.

Definition

```
CREATE METHOD ST_GeometryN
   (aposition INTEGER)
   RETURNS ST_Geometry
   FOR ST_GeomCollection
   BEGIN
      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS ST_Geometry);
      END IF;
      IF aposition < 1 OR
         aposition > CARDINALITY(SELF.ST_PrivateGeometries) THEN
         BEGIN
            SIGNAL SQLSTATE '01F01'
               SET MESSAGE_TEXT = 'invalid position';
            RETURN CAST (NULL AS ST_Geometry);
         END;
      END IF;
      RETURN SELF.ST_PrivateGeometries[aposition];
   END
```

Description

1) The method *ST_GeometryN(INTEGER)* takes the following input parameters:
   a) an INTEGER value *aposition*.
2) For the null-call method *ST_GeometryN(INTEGER)*:

   Case:
   a) If SELF is an empty set, then return the null value.
   b) If aposition is less than one or greater than the cardinality of the *ST_PrivateGeometries* attribute, then:
      i) A completion condition is raised: *SQL/MM Spatial warning – invalid position*.
      ii) Return the null value.
   c) Otherwise, return the element of the *ST_PrivateGeometries* attribute at position aposition.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.1.6 ST_GeomCollFromTxt Functions

Purpose

Return an ST_GeomCollection value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_GeomCollection value.

Definition

```
CREATE FUNCTION ST_GeomCollFromTxt
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
    RETURNS ST_GeomCollection
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    RETURN ST_GeomCollFromText(awkt, 0)

CREATE FUNCTION ST_GeomCollFromTxt
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
     ansrid INTEGER)
    RETURNS ST_GeomCollection
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    BEGIN
       --
       -- See Description
       --
    END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_GeomCollFromTxt(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_GeomCollFromTxt(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_GeomCollFromTxt(awkt, 0)*.
3) The function *ST_GeomCollFromTxt(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_GeomCollFromTxt(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkt* is the well-known text representation of an *ST_GeomCollection* value.

      If *awkt* is not producible in the BNF for <geometrycollection text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromText(awkt, ansrid) AS ST_GeomCollection)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.1.7 ST_GeomCollFromWKB Functions

Purpose

Return an ST_GeomCollection value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_GeomCollection value.

Definition

```
CREATE FUNCTION ST_GeomCollFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_GeomCollection
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_GeomCollFromWKB(awkb, 0)

CREATE FUNCTION ST_GeomCollFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_GeomCollection
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_GeomCollFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_GeomCollFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_GeomCollFromWKB(awkb, 0)*.
3) The function *ST_GeomCollFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_GeomCollFromWKB(BINARY LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkb* is the well-known binary representation of an *ST_GeomCollection* value.

      If *awkb* is not producible in the BNF for <geometrycollection binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_GeomCollection)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.1.8 ST_GeomCollFromGML Functions

Purpose

Return an ST_GeomCollection value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representationof an ST_GeomCollection value.

Definition

```
CREATE FUNCTION ST_GeomCollFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
   RETURNS ST_GeomCollection
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_GeomCollFromGML(agml, 0)

CREATE FUNCTION ST_GeomCollFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
    ansrid INTEGER)
   RETURNS ST_GeomCollection
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_GeomCollFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_GeomCollFromGML(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_GeomCollFromGML(agml, 0)*.
3) The function *ST_GeomCollFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_GeomCollFromGML(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If the parameter *agml* does not contain a MultiGeometry XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromGML(agml, ansrid) AS ST_GeomCollection)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.2 ST_MultiPoint Type and Routines

9.2.1 ST_MultiPoint Type

Purpose

The ST_MultiPoint type is a 0-dimensional geometry and represents a collection of ST_Point values.

Definition

```
CREATE TYPE ST_MultiPoint
   UNDER ST_GeomCollection
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_MultiPoint
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
      RETURNS ST_MultiPoint
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiPoint
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
       ansrid INTEGER)
      RETURNS ST_MultiPoint
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiPoint
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
      RETURNS ST_MultiPoint
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiPoint
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
       ansrid INTEGER)
      RETURNS ST_MultiPoint
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiPoint
      (apointarray ST_Point
         ARRAY[ST_MaxGeometryArrayElements])
      RETURNS ST_MultiPoint
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_MultiPoint
   (apointarray ST_Point
      ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_MultiPoint
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_Geometries()
   RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements],

OVERRIDING METHOD ST_Geometries
   (ageometryarray ST_Geometry
      ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiPoint
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The *ST_MultiPoint* type provides for public use:
   a) a method *ST_MultiPoint(CHARACTER LARGE OBJECT)*,
   b) a method *ST_MultiPoint(CHARACTER LARGE OBJECT, INTEGER)*,
   c) a method *ST_MultiPoint(BINARY LARGE OBJECT)*,
   d) a method *ST_MultiPoint(BINARY LARGE OBJECT, INTEGER)*,
   e) a method *ST_MultiPoint(ST_Point ARRAY)*,
   f) a method *ST_MultiPoint(ST_Point ARRAY, INTEGER)*,
   g) an overriding method *ST_Geometries()*,
   h) an overriding method *ST_Geometries(ST_Geometry ARRAY)*,
   i) a function *ST_MPointFromText(CHARACTER LARGE OBJECT)*,
   j) a function *ST_MPointFromText(CHARACTER LARGE OBJECT, INTEGER)*,
   k) a function *ST_MPointFromWKB(BINARY LARGE OBJECT)*,
   l) a function *ST_MPointFromWKB(BINARY LARGE OBJECT, INTEGER)*,
   m) a function *ST_MPointFromGML(CHARACTER LARGE OBJECT)*,
   n) a function *ST_MPointFromGML(CHARACTER LARGE OBJECT, INTEGER)*.
2) The dimension of an *ST_MultiPoint* value is 0 (zero).
3) The elements of the *ST_PrivateGeometries* attribute are restricted to *ST_Point* values.
4) The *ST_Point* values in the *ST_PrivateGeometries* attribute are not connected or ordered.
5) If no two *ST_Point* values in the *ST_MultiPoint* value are equal, then the *ST_MultiPoint* value is simple.
   a) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

b) If *SELF.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

6) The boundary of an *ST_MultiPoint* value is the empty set.

7) An *ST_MultiPoint* value is well formed only if and only if all of the *ST_Point* values in the *ST_PrivateGeometries* attribute are well formed.

8) An *ST_MultiPoint* value returned by the constructor function corresponds to the empty set.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.2.2 ST_MultiPoint Methods

Purpose

Return an ST_MultiPoint value constructed from either the well-known text representation, the well-known binary representation, the GML representation, or the specified ST_Point values.

Definition

```
CREATE CONSTRUCTOR METHOD ST_MultiPoint
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_MultiPoint
   FOR ST_MultiPoint
   RETURN NEW ST_MultiPoint(awktorgml, 0)

CREATE CONSTRUCTOR METHOD ST_MultiPoint
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_MultiPoint
   FOR ST_MultiPoint
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_MultiPoint
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_MultiPoint
   FOR ST_MultiPoint
   RETURN NEW ST_MultiPoint(awkb, 0)

CREATE CONSTRUCTOR METHOD ST_MultiPoint
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_MultiPoint
   FOR ST_MultiPoint
   RETURN ST_MPointFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_MultiPoint
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiPoint
   FOR ST_MultiPoint
   RETURN SELF.ST_SRID(0).ST_Geometries(apointarray)

CREATE CONSTRUCTOR METHOD ST_MultiPoint
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_MultiPoint
   FOR ST_MultiPoint
   RETURN SELF.ST_SRID(ansrid).ST_Geometries(apointarray)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The method *ST_MultiPoint(CHARACTER LARGE OBJECT)* takes the following input parameter:
   a) a CHARACTER LARGE OBJECT value *awktorgml*.

© ISO/IEC 2011 – All rights reserved

2) The null-call type preserving SQL-invoked constructor method *ST_MultiPoint(CHARACTER LARGE OBJECT)* returns the result of the value expression: *NEW ST_MultiPoint(awktorgml, 0)*.

3) The method *ST_MultiPoint(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a CHARACTER LARGE OBJECT value *awktorgml*,

   b) an INTEGER value *ansrid*.

4) For the null-call type preserving SQL-invoked constructor method *ST_MultiPoint(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If *awktorgml* contains a MultiPoint XML element in the GML representation, then return the result of the value expression: *ST_MPointFromGML*(*awktorgml*, *ansrid*).

   b) Otherwise, return the result of the value expression: *ST_MPointFromText*(*awktorgml*, *ansrid*).

5) The method *ST_MultiPoint(BINARY LARGE OBJECT)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *awkb*.

6) The null-call type preserving SQL-invoked constructor method *ST_MultiPoint(BINARY LARGE OBJECT)* returns the result of the value expression: *NEW ST_MultiPoint(awktorgml, 0)*.

7) The method *ST_MultiPoint(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *awkb*,

   b) an INTEGER value *ansrid*.

8) The null-call type preserving SQL-invoked constructor method *ST_MultiPoint(BINARY LARGE OBJECT, INTEGER)* returns the result of the value expression: *ST_MPointFromWKB(awkb, ansrid)*.

9) The method *ST_MultiPoint(ST_Point ARRAY)* takes the following input parameters:

   a) an *ST_Point* ARRAY value *apointarray*.

10) The null-call type preserving SQL-invoked constructor method *ST_MultiPoint(ST_Point ARRAY)* returns an *ST_MultiPoint* value with:

   a) The spatial reference system identifier set to 0 (zero).

   b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the *ST_PrivateGeometries* attribute set to *apointarray*, the *ST_PrivateDimension* attribute set to 0 (zero), and the *ST_PrivateCoordinateDimension* attribute set to 2.

11) The method *ST_MultiPoint(ST_Point ARRAY, INTEGER)* takes the following input parameters:

   a) an *ST_Point* ARRAY value *apointarray*,

   b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_MultiPoint(ST_Point ARRAY, INTEGER)* returns an *ST_MultiPoint* value with:

   a) The spatial reference system identifier set to *ansrid*.

   b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the *ST_PrivateGeometries* attribute set to *apointarray*, the *ST_PrivateDimension* attribute set to 0 (zero), and the *ST_PrivateCoordinateDimension* attribute set to 2.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.2.3 ST_Geometries Methods

Purpose

Observe and mutate the ST_PrivateGeometries attribute of an ST_MultiPoint value.

Definition

```
CREATE METHOD ST_Geometries()
   RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_MultiPoint
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CAST(SELF.ST_PrivateGeometries AS
               ST_Point ARRAY[ST_MaxGeometryArrayElements])
      END

CREATE METHOD ST_Geometries
   (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiPoint
   FOR ST_MultiPoint
   BEGIN
      DECLARE apointarray ST_Point
         ARRAY[ST_MaxGeometryArrayElements];

      -- Cast ageometryarray to an ST_Point ARRAY
      SET apointarray = CAST(ageometryarray AS
         ST_Point ARRAY[ST_MaxGeometryArrayElements]);
      -- If SELF is the null value, then return the null value.
      -- Otherwise, return an ST_MultiPoint value containing
      -- apointarray.
      RETURN
         CASE
            WHEN SELF IS NULL THEN
               NULL
            ELSE
               (SELF AS ST_GeomCollection).
                  ST_Geometries(apointarray)
         END;
   END
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

Description

1) The method *ST_Geometries()* has no input parameters.
2) For the null-call method *ST_Geometries()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the value of the *ST_PrivateGeometries* attribute as an *ST_Point* ARRAY.
3) The method *ST_Geometries(ST_Geometry ARRAY)* takes the following input parameters:

   a) an *ST_Geometry ARRAY* value *ageometryarray*.
4) For the type preserving method *ST_Geometries(ST_Geometry ARRAY)*:

   a) Let *APOINTARRAY* be the result of casting *ageometryarray* to an *ST_Point* ARRAY value (implicitly using *ST_ToPointAry(ST_Geometry ARRAY)*).

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

b) Case:

i) If SELF is the null value, then return the null value.

ii) Otherwise, using the method *ST_Geometries(ST_Geometry ARRAY)* for type *ST_GeomCollection*, return an *ST_MultiPoint* value with:

1) The dimension set to 0 (zero).

2) The coordinate dimension set to 2.

3) The *ST_PrivateGeometries* attribute set to *APOINTARRAY*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.2.4 ST_MPointFromText Functions

Purpose

Return an ST_MultiPoint value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_MultiPoint value.

Definition

```
CREATE FUNCTION ST_MPointFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_MultiPoint
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_MPointFromText(awkt, 0)

CREATE FUNCTION ST_MPointFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_MultiPoint
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_MPointFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_MPointFromText(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_MPointFromText(awkt, 0)*.
3) The function *ST_MPointFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_MPointFromText(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkt* is the well-known text representation of an *ST_MultiPoint* value.

      If *awkt* is not producible in the BNF for <multipoint text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromText(awkt, ansrid) AS ST_MultiPoint)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.2.5 ST_MPointFromWKB Functions

Purpose

Return an ST_MultiPoint value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_MultiPoint value.

Definition

```
CREATE FUNCTION ST_MPointFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_MultiPoint
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_MPointFromWKB(awkb, 0)

CREATE FUNCTION ST_MPointFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_MultiPoint
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_MPointFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_MPointFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_MPointFromWKB(awkb, 0)*.
3) The function *ST_MPointFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_MPointFromWKB(BINARY LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkb* is the well-known binary representation of an *ST_MultiPoint* value.

      If *awkb* is not producible in the BNF for <multipoint binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_MultiPoint)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.2.6 ST_MPointFromGML Functions

Purpose

Return an ST_MultiPoint value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representationof an ST_MultiPoint value.

Definition

```
CREATE FUNCTION ST_MPointFromGML
    (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
    RETURNS ST_MultiPoint
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    RETURN ST_MPointFromGML(agml, 0)

CREATE FUNCTION ST_MPointFromGML
    (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
     ansrid INTEGER)
    RETURNS ST_MultiPoint
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    BEGIN
       --
       -- See Description
       --
    END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_MPointFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_MPointFromGML(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_MPointFromGML(agml, 0)*.
3) The function *ST_MPointFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_MPointFromGML(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If the parameter *agml* does not contain a MultiPoint XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromGML(agml, ansrid) AS ST_MultiPoint)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.3 ST_MultiCurve Type and Routines

9.3.1 ST_MultiCurve Type

Purpose

The ST_MultiCurve type is a 1-dimensional geometry and represents a collection of ST_Curve.

Definition

```
CREATE TYPE ST_MultiCurve
   UNDER ST_GeomCollection
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_MultiCurve
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
      RETURNS ST_MultiCurve
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiCurve
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
       ansrid INTEGER)
      RETURNS ST_MultiCurve
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiCurve
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
      RETURNS ST_MultiCurve
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiCurve
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
       ansrid INTEGER)
      RETURNS ST_MultiCurve
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiCurve
      (acurvearray ST_Curve
         ARRAY[ST_MaxGeometryArrayElements])
      RETURNS ST_MultiCurve
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_MultiCurve
   (acurvearray ST_Curve
      ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_MultiCurve
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_IsClosed()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DIsClosed()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Length()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Length
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DLength()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DLength
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_Geometries()
   RETURNS ST_Curve ARRAY[ST_MaxGeometryArrayElements],
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
OVERRIDING METHOD ST_Geometries
   (ageometryarray ST_Geometry
      ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiCurve
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The *ST_MultiCurve* type provides for public use:
   a) a method *ST_MultiCurve(CHARACTER LARGE OBJECT)*,
   b) a method *ST_MultiCurve(CHARACTER LARGE OBJECT, INTEGER)*,
   c) a method *ST_MultiCurve(BINARY LARGE OBJECT)*,
   d) a method *ST_MultiCurve(BINARY LARGE OBJECT, INTEGER)*,
   e) a method *ST_MultiCurve(ST_Curve ARRAY)*,
   f) a method *ST_MultiCurve(ST_Curve ARRAY, INTEGER)*,
   g) a method *ST_IsClosed()*,
   h) a method *ST_3DIsClosed()*,
   i) a method *ST_Length()*,
   j) a method *ST_Length(CHARACTER VARYING)*,
   k) a method *ST_3DLength()*,
   l) a method *ST_3DLength(CHARACTER VARYING)*,
   m) an overriding method *ST_Geometries()*,
   n) an overriding method *ST_Geometries(ST_Geometry ARRAY)*,
   o) a function *ST_MCurveFromText(CHARACTER LARGE OBJECT)*,
   p) a function *ST_MCurveFromText(CHARACTER LARGE OBJECT, INTEGER)*,
   q) a function *ST_MCurveFromWKB(BINARY LARGE OBJECT)*,
   r) a function *ST_MCurveFromWKB(BINARY LARGE OBJECT, INTEGER)*,
   s) a function *ST_MCurveFromGML(CHARACTER LARGE OBJECT)*,
   t) a function *ST_MCurveFromGML(CHARACTER LARGE OBJECT, INTEGER)*.
2) The dimension of an *ST_MultiCurve* value is 1 (one).
3) The elements of an *ST_MultiCurve* value are *ST_Curve* values.
4) If all of the elements in the *ST_PrivateGeometries* attribute are simple and any two elements only spatially intersect at the boundaries of both elements, then an *ST_MultiCurve* is simple.
   a) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation of *ST_IsSimple*.
   b) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are considered in the calculation of *ST_3DIsSimple*.
   c) If *SELF.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the .

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

5) The boundary of an *ST_MultiCurve* value is obtained by applying the mod 2 union rule: an *ST_Point* value is in the boundary of an *ST_MultiCurve* if it is in the boundaries of an odd number of elements of the *ST_MultiCurve*.

6) An *ST_MultiCurve* value is closed if all of its elements are closed. The boundary of a closed *ST_MultiCurve* is the empty set.

   a) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation of *ST_IsClosed*.

   b) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are considered in the calculation of *ST_3DIsClosed*.

   c) If *SELF.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

7) An *ST_MultiCurve* value is defined as topologically closed.

8) An *ST_MultiCurve* value is well formed only if all of the *ST_Curve* values in the *ST_PrivateGeometries* attribute are well formed.

9) An *ST_MultiCurve* value returned by the constructor function corresponds to the empty set.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.3.2 ST_MultiCurve Methods

Return an ST_MultiCurve value constructed from either the well-known text representation, the well-known binary representation, the GML representation, or the specified ST_Curve values.

Definition

```
CREATE CONSTRUCTOR METHOD ST_MultiCurve
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_MultiCurve
   FOR ST_MultiCurve
   RETURN NEW ST_MultiCurve(awktorgml, 0)

CREATE CONSTRUCTOR METHOD ST_MultiCurve
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_MultiCurve
   FOR ST_MultiCurve
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_MultiCurve
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_MultiCurve
   FOR ST_MultiCurve
   RETURN NEW ST_MultiCurve(awkb, 0)

CREATE CONSTRUCTOR METHOD ST_MultiCurve
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_MultiCurve
   FOR ST_MultiCurve
   RETURN ST_MCurveFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_MultiCurve
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiCurve
   FOR ST_MultiCurve
   RETURN SELF.ST_SRID(0).ST_Geometries(acurvearray)

CREATE CONSTRUCTOR METHOD ST_MultiCurve
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_MultiCurve
   FOR ST_MultiCurve
   RETURN SELF.ST_SRID(ansrid).ST_Geometries(acurvearray)
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The method *ST_MultiCurve(CHARACTER LARGE OBJECT)* takes the following input parameter:
   a) a CHARACTER LARGE OBJECT value *awktorgml*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) The null-call type preserving SQL-invoked constructor method *ST_MultiCurve(CHARACTER LARGE OBJECT)* returns the result of the value expression: *NEW ST_MultiCurve(awktorgml, 0)*.

3) The method *ST_MultiCurve(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a CHARACTER LARGE OBJECT value *awktorgml*,

   b) an INTEGER value *ansrid*.

4) For the null-call type preserving SQL-invoked constructor method *ST_MultiCurve(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If *awktorgml* contains a MultiCurve XML element in the GML representation, then return the result of the value expression: *ST_MCurveFromGML(awktorgml, ansrid)*.

   b) Otherwise, return the result of the value expression: *ST_MCurveFromText(awktorgml, ansrid)*.

5) The method *ST_MultiCurve(BINARY LARGE OBJECT)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *awkb*.

6) The null-call type preserving SQL-invoked constructor method *ST_MultiCurve(BINARY LARGE OBJECT)* returns the result of the value expression: *NEW ST_MultiCurve(awkb, 0)*.

7) The method *ST_MultiCurve(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *awkb*,

   b) an INTEGER value *ansrid*.

8) The null-call type preserving SQL-invoked constructor method *ST_MultiCurve(BINARY LARGE OBJECT, INTEGER)* returns the result of the value expression: *ST_MCurveFromWKB(awkb, ansrid)*.

9) The method *ST_MultiCurve(ST_Curve ARRAY)* takes the following input parameters:

   a) an *ST_Curve* ARRAY value *acurvearray*.

10) The null-call type preserving SQL-invoked constructor method *ST_MultiCurve(ST_Curve ARRAY)* returns an *ST_MultiCurve* value with:

    a) The spatial reference system identification set to 0 (zero).

    b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the attribute *ST_PrivateGeometries* set to *acurvearray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

11) The method *ST_MultiCurve(ST_Curve ARRAY, INTEGER)* takes the following input parameters:

    a) an *ST_Curve* ARRAY value *acurvearray*,

    b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_MultiCurve(ST_Curve ARRAY, INTEGER)* returns an *ST_MultiCurve* value with:

    a) The spatial reference system identification set to *ansrid*.

    b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the attribute *ST_PrivateGeometries* set to *acurvearray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.3.3 ST_IsClosed Method

Purpose

Test if an ST_MultiCurve value is closed, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_IsClosed()
   RETURNS INTEGER
   FOR ST_MultiCurve
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty = 1 THEN
            0
         ELSE
            SELF.ST_Boundary().ST_IsEmpty()
      END
```

Description

1) The method *ST_IsClosed()* has no input parameters.
2) The null-call method *ST_IsClosed()* returns:

   Case:

   a) If SELF is the empty set, then 0 (zero).
   b) If the boundary of the *ST_MultiCurve* value is the empty set, then 1 (one).
   c) Otherwise, 0 (zero).
3) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.3.4 ST_3DIsClosed Method

Purpose

Test if an ST_MultiCurve value is closed, considering z coordinate values in the calculations.

Definition

```
CREATE METHOD ST_3DIsClosed()
   RETURNS INTEGER
   FOR ST_MultiCurve
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty = 1 THEN
            0
         ELSE
            SELF.ST_3DBoundary().ST_IsEmpty()
      END
```

Description

1) The method *ST_3DIsClosed()* has no input parameters.
2) The null-call method *ST_3DIsClosed()* returns:

   Case:

   a) If SELF is the empty set, then 0 (zero).
   b) If the boundary of the *ST_MultiCurve* value is the empty set, then 1 (one).
   c) Otherwise, 0 (zero).
3) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are considered in the calculation.
4) If *SELF.ST_IsMeasured()* is equal to 1 (one), then the m coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.3.5 ST_Length Methods

Purpose

Return the length measurement of an ST_MultiCurve value, ignoring z and m coordinate valuess in the calculations.

Definition

```
CREATE METHOD ST_Length()
   RETURNS DOUBLE PRECISION
   FOR ST_MultiCurve
   BEGIN
      DECLARE length DOUBLE PRECISION;
      DECLARE counter INTEGER;

      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS DOUBLE PRECISION);
      END IF;
      SET length = 0.0;
      SET counter = 1;
      WHILE counter <= SELF.ST_NumGeometries() DO
         SET length = length + SELF.ST_GeometryN(counter).ST_Length();
         SET counter = counter + 1;
      END WHILE;
      RETURN length;
   END

CREATE METHOD ST_Length
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   FOR ST_MultiCurve
   BEGIN
      DECLARE length DOUBLE PRECISION;
      DECLARE counter INTEGER;

      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS DOUBLE PRECISION);
      END IF;
      SET length = 0.0;
      SET counter = 1;
      WHILE counter <= SELF.ST_NumGeometries() DO
         SET length = length + SELF.ST_GeometryN(counter).ST_Length(aunit);
         SET counter = counter + 1;
      END WHILE;
      RETURN length;
   END
```

Definitional Rules

1) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The method *ST_Length()* has no input parameters.
2) For the null-call method *ST_Length()*:
   a) Case:
      i) If SELF is an empty set, then return the null value.
      ii) Otherwise, return the sum of the *ST_Length()* values of each element in the *ST_PrivateGeometries* attribute of SELF.
   b) Case:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

i) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_Length()* is in the linear unit of measure identified by <linear unit>.

ii) Otherwise, the value returned by *ST_Length()* is in an implementation-defined unit of measure.

3) The method *ST_Length(CHARACTER VARYING)* takes the following input parameter:

a) a CHARACTER VARYING value *aunit*.

4) For the null-call method *ST_Length(CHARACTER VARYING)*:

a) The values for *aunit* shall be a supported <unit name>.

b) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.

c) If the unit specified by *aunit* is not supported by the implementation to compute the sum of the *ST_Length(aunit)* values of each element in *ST_PrivateGeometries* attribute of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

d) Case:

i) If SELF is an empty set, then return the null value.

ii) Otherwise, return the sum of the *ST_Length(aunit)* values of each element in the *ST_PrivateGeometries* attribute of SELF.

e) Te returned value is in the units indicated by *aunit*.

5) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.3.6 ST_3DLength Methods

Purpose

Return the length measurement of an ST_MultiCurve value, considering z coordinate values in the calculations.

Definition

```
CREATE METHOD ST_3DLength()
   RETURNS DOUBLE PRECISION
   FOR ST_MultiCurve
   BEGIN
      DECLARE length DOUBLE PRECISION;
      DECLARE counter INTEGER;

      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS DOUBLE PRECISION);
      END IF;
      SET length = 0.0;
      SET counter = 1;
      WHILE counter <= SELF.ST_NumGeometries() DO
         SET length = length + SELF.ST_GeometryN(counter).ST_3DLength();
         SET counter = counter + 1;
      END WHILE;
      RETURN length;
   END

CREATE METHOD ST_3DLength
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   FOR ST_MultiCurve
   BEGIN
      DECLARE length DOUBLE PRECISION;
      DECLARE counter INTEGER;

      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS DOUBLE PRECISION);
      END IF;
      SET length = 0.0;
      SET counter = 1;
      WHILE counter <= SELF.ST_NumGeometries() DO
         SET length = length + SELF.ST_GeometryN(counter).ST_3DLength(aunit);
         SET counter = counter + 1;
      END WHILE;
      RETURN length;
   END
```

Definitional Rules

1) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The method *ST_3DLength()* has no input parameters.
2) For the null-call method *ST_3DLength()*:
   a) Case:
      i) If SELF is an empty set, then return the null value.
      ii) Otherwise, return the sum of the *ST_3DLength()* values of each element in the *ST_PrivateGeometries* attribute of SELF.
   b) Case:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

i) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_3DLength()* is in the linear unit of measure identified by <linear unit>.

ii) Otherwise, the value returned by *ST_3DLength()* is in an implementation-defined unit of measure.

3) The method *ST_3DLength(CHARACTER VARYING)* takes the following input parameter:

a) a CHARACTER VARYING value *aunit*.

4) For the null-call method *ST_3DLength(CHARACTER VARYING)*:

a) The values for *aunit* shall be a supported <unit name>.

b) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.

c) If the unit specified by *aunit* is not supported by the implementation to compute the sum of the *ST_3DLength(aunit)* values of each element in *ST_PrivateGeometries* attribute of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

d) Case:

i) If SELF is an empty set, then return the null value.

ii) Otherwise, return the sum of the *ST_3DLength(aunit)* values of each element in the *ST_PrivateGeometries* attribute of SELF.

e) The returned value is in the units indicated by *aunit*.

5) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.3.7 ST_Geometries Methods

Purpose

Observe and mutate the ST_PrivateGeometries attribute of an ST_MultiCurve value.

Definition

```
CREATE METHOD ST_Geometries()
   RETURNS ST_Curve ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_MultiCurve
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CAST(SELF.ST_PrivateGeometries AS ST_Curve
               ARRAY[ST_MaxGeometryArrayElements])
      END

CREATE METHOD ST_Geometries
   (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiCurve
   FOR ST_MultiCurve
   BEGIN
      DECLARE acurvearray ST_Curve
         ARRAY[ST_MaxGeometryArrayElements];

      -- Cast ageometryarray to an ST_Curve ARRAY
      SET acurvearray = CAST(ageometryarray AS
         ST_Curve ARRAY[ST_MaxGeometryArrayElements]);
      -- If SELF is the null value, then return the null value. Otherwise,
      -- return an ST_MultiCurve value containing acurvearray.
      RETURN
         CASE
            WHEN SELF IS NULL THEN
               NULL
            ELSE
               (SELF AS ST_GeomCollection).
                  ST_Geometries(acurvearray)
         END;
   END
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

Description

1) The method *ST_Geometries()* has no input parameters.
2) For the null-call method *ST_Geometries()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the value of the *ST_PrivateGeometries* attribute as an *ST_Curve ARRAY*.
3) The method *ST_Geometries(ST_Geometry ARRAY)* takes the following input parameters:

   a) an *ST_Geometry ARRAY* value *ageometryarray*.
4) For the type preserving method *ST_Geometries(ST_Geometry ARRAY)*:

   a) Let *ACURVEARRAY* be the result of casting *ageometryarray* to an *ST_Curve* ARRAY value (implicitly using *ST_ToCurveAry(ST_Geometry ARRAY)*).

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

b) Case:

i) If SELF is the null value, then return the null value.

ii) Otherwise, return an *ST_MultiCurve* value with:

1) The dimension set to 1 (one).

2) Using the method *ST_Geometries(ST_Geometry ARRAY)* for type *ST_GeomCollection*, the *ST_PrivateGeometries* attribute set to *ACURVEARRAY*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.3.8 ST_MCurveFromText Functions

Purpose

Return an ST_MultiCurve value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_MultiCurve value.

Definition

```
CREATE FUNCTION ST_MCurveFromText
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
    RETURNS ST_MultiCurve
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    RETURN ST_MCurveFromText(awkt, 0)

CREATE FUNCTION ST_MCurveFromText
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
     ansrid INTEGER)
    RETURNS ST_MultiCurve
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
    BEGIN
       --
       -- See Description
       --
    END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_MCurveFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_MCurveFromText(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_MCurveFromText(awkt, 0)*.
3) The function *ST_MCurveFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_MCurveFromText(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkt* is the well-known text representation of an *ST_MultiCurve* value.

      If *awkt* is not producible in the BNF for <multicurve text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromText(awkt, ansrid) AS ST_MultiCurve)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.3.9 ST_MCurveFromWKB Functions

Purpose

Return an ST_MultiCurve value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_MultiCurve value.

Definition

```
CREATE FUNCTION ST_MCurveFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_MultiCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_MCurveFromWKB(awkb, 0)

CREATE FUNCTION ST_MCurveFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_MultiCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_MCurveFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_MCurveFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_MCurveFromWKB(awkb, 0)*.
3) The function *ST_MCurveFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_MCurveFromWKB(BINARY LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkb* is the well-known binary representation of an *ST_MultiCurve* value.

      If *awkb* is not producible in the BNF for <multicurve binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_MultiCurve)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.3.10 ST_MCurveFromGML Functions

Purpose

Return an ST_MultiCurve value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representationof an ST_MultiCurve value.

Definition

```
CREATE FUNCTION ST_MCurveFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
   RETURNS ST_MultiCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_MCurveFromGML(agml, 0)

CREATE FUNCTION ST_MCurveFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
    ansrid INTEGER)
   RETURNS ST_MultiCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_MCurveFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_MCurveFromGML*(CHARACTER LARGE OBJECT) returns the result of the value expression: *ST_MCurveFromGML(agml, 0)*.
3) The function *ST_MCurveFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_MCurveFromGML(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If the parameter *agml* does not contain a MultiCurve XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromGML(agml, ansrid) AS ST_MultiCurve)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.4 ST_MultiLineString Type and Routines

9.4.1 ST_MultiLineString Type

Purpose

The ST_MultiLineString type is a subtype of the ST_MultiCurve and represents a collection of ST_LineString values.

Definition

```
CREATE TYPE ST_MultiLineString
   UNDER ST_MultiCurve
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_MultiLineString
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
      RETURNS ST_MultiLineString
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiLineString
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
       ansrid INTEGER)
      RETURNS ST_MultiLineString
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiLineString
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
      RETURNS ST_MultiLineString
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiLineString
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
       ansrid INTEGER)
      RETURNS ST_MultiLineString
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_MultiLineString
   (alinestringarray ST_LineString
      ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiLineString
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiLineString
   (alinestringarray ST_LineString
      ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_MultiLineString
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_Geometries()
   RETURNS ST_LineString ARRAY[ST_MaxGeometryArrayElements],

OVERRIDING METHOD ST_Geometries
   (ageometryarray ST_Geometry
      ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiLineString
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The *ST_MultiLineString* type provides for public use:
   a) a method *ST_MultiLineString(CHARACTER LARGE OBJECT)*,
   b) a method *ST_MultiLineString(CHARACTER LARGE OBJECT, INTEGER)*,
   c) a method *ST_MultiLineString(BINARY LARGE OBJECT)*,
   d) a method *ST_MultiLineString(BINARY LARGE OBJECT, INTEGER)*,
   e) a method *ST_MultiLineString(ST_LineString ARRAY)*,
   f) a method *ST_MultiLineString(ST_LineString ARRAY, INTEGER)*,
   g) an overriding method *ST_Geometries()*,
   h) an overriding method *ST_Geometries(ST_Geometry ARRAY)*,
   i) a function *ST_MLineFromText(CHARACTER LARGE OBJECT)*,
   j) a function *ST_MLineFromText(CHARACTER LARGE OBJECT, INTEGER)*,
   k) a function *ST_MLineFromWKB(BINARY LARGE OBJECT)*,
   l) a function *ST_MLineFromWKB(BINARY LARGE OBJECT, INTEGER)*,
   m) a function *ST_MLineFromGML(CHARACTER LARGE OBJECT)*,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

n) a function *ST_MLineFromGML(CHARACTER LARGE OBJECT, INTEGER)*.

2) The elements of the *ST_PrivateGeometries* attribute are restricted to *ST_LineString* values.

3) An *ST_MultiLineString* value is well formed only if and only if all of the *ST_LineString* values in the *ST_PrivateGeometries* attribute are well formed.

4) An *ST_MultiLineString* value returned by the constructor function corresponds to the empty set.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.4.2 ST_MultiLineString Methods

Purpose

Return an ST_MultiLineString value constructed from either the well-known text representation, the well-known binary representation, the GML representation, or the specified ST_LineString values.

Definition

```
CREATE CONSTRUCTOR METHOD ST_MultiLineString
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_MultiLineString
   FOR ST_MultiLineString
   RETURN NEW ST_MultiLineString(awktorgml, 0)

CREATE CONSTRUCTOR METHOD ST_MultiLineString
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_MultiLineString
   FOR ST_MultiLineString
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_MultiLineString
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_MultiLineString
   FOR ST_MultiLineString
   RETURN NEW ST_MultiLineString(awkb, 0)

CREATE CONSTRUCTOR METHOD ST_MultiLineString
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_MultiLineString
   FOR ST_MultiLineString
   RETURN ST_MLineFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_MultiLineString
   (alinestringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiLineString
   FOR ST_MultiLineString
   RETURN SELF.ST_SRID(0).ST_Geometries(alinestringarray)

CREATE CONSTRUCTOR METHOD ST_MultiLineString
   (alinestringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_MultiLineString
   FOR ST_MultiLineString
   RETURN SELF.ST_SRID(ansrid).ST_Geometries(alinestringarray)
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The method *ST_MultiLineString(CHARACTER LARGE OBJECT)* takes the following input parameter:

   a) a CHARACTER LARGE OBJECT value *awktorgml*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) The null-call type preserving SQL-invoked constructor method *ST_MultiLineString(CHARACTER LARGE OBJECT)* returns the result of the value expression: *NEW ST_MultiLineString(awktorgml, 0)*.

3) The method *ST_MultiLineString(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a CHARACTER LARGE OBJECT value *awktorgml*,

   b) an INTEGER value *ansrid*.

4) For the null-call type preserving SQL-invoked constructor method *ST_MultiLineString(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If *awktorgml* contains a MultiLineString XML element in the GML representation, then return the result of the value expression: *ST_MLineStringFromGML(awktorgml, ansrid)*.

   b) Otherwise, return the result of the value expression: *ST_MLineFromText(awktorgml, ansrid)*.

5) The method *ST_MultiLineString(BINARY LARGE OBJECT)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *awkb*.

6) The null-call type preserving SQL-invoked constructor method *ST_MultiLineString(BINARY LARGE OBJECT)* returns the result of the value expression: *NEW ST_MultiLineString(awkb, 0)*.

7) The method *ST_MultiLineString(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *awkb*,

   b) an INTEGER value *ansrid*.

8) The null-call type preserving SQL-invoked constructor method *ST_MultiLineString(BINARY LARGE OBJECT, INTEGER)* returns the result of the value expression: *ST_MLineFromWKB(awkb, ansrid)*.

9) The method *ST_MultiLineString(ST_LineString ARRAY)* takes the following input parameters:

   a) an *ST_LineString* ARRAY value *alinestringarray*.

10) The null-call type preserving SQL-invoked constructor method *ST_MultiLineString(ST_LineString ARRAY)* returns an *ST_MultiLineString* value with:

    a) The spatial reference system identifier set to 0 (zero).

    b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the *ST_PrivateGeometries* attribute set to *alinestringarray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

11) The method *ST_MultiLineString(ST_LineString ARRAY, INTEGER)* takes the following input parameters:

    a) an *ST_LineString* ARRAY value *alinestringarray*,

    b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_MultiLineString(ST_LineString ARRAY, INTEGER)* returns an *ST_MultiLineString* value with:

    a) The spatial reference system identifier set to *ansrid*.

    b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the *ST_PrivateGeometries* attribute set to *alinestringarray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.4.3 ST_Geometries Methods

Purpose

Observe and mutate the ST_PrivateGeometries attribute of an ST_MultiLineString value.

Definition

```
CREATE METHOD ST_Geometries()
   RETURNS ST_LineString ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_MultiLineString
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CAST(SELF.ST_PrivateGeometries AS
               ST_LineString ARRAY[ST_MaxGeometryArrayElements])
      END

CREATE METHOD ST_Geometries
   (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiLineString
   FOR ST_MultiLineString
   BEGIN
      DECLARE alinestringarray ST_LineString
         ARRAY[ST_MaxGeometryArrayElements];

      -- Cast ageometryarray to an ST_LineString ARRAY
      SET alinestringarray = CAST(ageometryarray AS
         ST_LineString ARRAY[ST_MaxGeometryArrayElements]);
      -- If SELF is the null value, then return the null value. Otherwise,
      -- return an ST_MultiLineString value containing alinestringarray.
      RETURN
         CASE
            WHEN SELF IS NULL THEN
               NULL
            ELSE
               (SELF AS ST_MultiCurve).
                  ST_Geometries(alinestringarray)
         END;
   END
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

Description

1) The method *ST_Geometries()* has no input parameters.
2) For the null-call method *ST_Geometries()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the value of the *ST_PrivateGeometries* attribute as an *ST_LineString ARRAY*.
3) The method *ST_Geometries(ST_Geometry ARRAY)* takes the following input parameters:

   a) an *ST_Geometry ARRAY* value *ageometryarray*.
4) For the type preserving method *ST_Geometries(ST_Geometry ARRAY)*:

   a) Let *ALINESTRINGARRAY* be the result of casting *ageometryarray* to an *ST_LineString* ARRAY value (implicitly using *ST_ToLineStringAry(ST_Geometry ARRAY)*).

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

b) Case:

i) If SELF is the null value, then return the null value.

ii) Otherwise, return an *ST_MultiLineString* value with:

1) The dimension set to 1 (one).

2) Using the method *ST_Geometries(ST_Geometry ARRAY)* for type *ST_MultiCurve*, the *ST_PrivateGeometries* attribute set to *ALINESTRINGARRAY*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.4.4 ST_MLineFromText Functions

Purpose

Return an ST_MultiLineString value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_MultiLineString value.

Definition

```
CREATE FUNCTION ST_MLineFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_MultiLineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_MLineFromText(awkt, 0)

CREATE FUNCTION ST_MLineFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_MultiLineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_MLineFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_MLineFromText(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_MLineFromText(awkt, 0)*.
3) The function *ST_MLineFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_MLineFromText(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkt* is the well-known text representation of an *ST_MultiLineString* value.

      If *awkt* is not producible in the BNF for <multilinestring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromText(awkt, ansrid) AS ST_MultiLineString)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.4.5 ST_MLineFromWKB Functions

Purpose

Return an ST_MultiLineString value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_MultiLineString value.

Definition

```
CREATE FUNCTION ST_MLineFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_MultiLineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_MLineFromWKB(awkb, 0)

CREATE FUNCTION ST_MLineFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_MultiLineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_MLineFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_MLineFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_MLineFromWKB(awkb, 0)*.
3) The function *ST_MLineFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_MLineFromWKB(BINARY LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkb* is the well-known binary representation of an *ST_MultiLineString* value.

      If *awkb* is not producible in the BNF for <multilinestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_MultiLineString)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.4.6 ST_MLineFromGML Functions

Purpose

Return an ST_MultiLineString value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representationof an ST_MultiLineString value.

Definition

```
CREATE FUNCTION ST_MLineFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
   RETURNS ST_MultiLineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_MLineFromGML(agml, 0)

CREATE FUNCTION ST_MLineFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
    ansrid INTEGER)
   RETURNS ST_MultiLineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_MLineFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_MLineFromGML(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_MLineFromGML(agml, 0)*.
3) The function *ST_MLineFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_MLineFromGML(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If the parameter *agml* does not contain a MultiLineString XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromGML(agml, ansrid) AS ST_MultiLineString)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.5 ST_MultiSurface Type and Routines

9.5.1 ST_MultiSurface Type

Purpose

The ST_MultiSurface type is a 2-dimensional geometry and represents a collection of ST_Surface values.

Definition

```
CREATE TYPE ST_MultiSurface
   UNDER ST_GeomCollection
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_MultiSurface
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
      RETURNS ST_MultiSurface
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiSurface
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
       ansrid INTEGER)
      RETURNS ST_MultiSurface
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiSurface
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
      RETURNS ST_MultiSurface
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiSurface
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
       ansrid INTEGER)
      RETURNS ST_MultiSurface
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiSurface
      (asurfacearray ST_Surface
         ARRAY[ST_MaxGeometryArrayElements])
      RETURNS ST_MultiSurface
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_MultiSurface
   (asurfacearray ST_Surface
      ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_MultiSurface
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Area()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Area
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DArea()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DArea
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Perimeter()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Perimeter
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DPerimeter()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_3DPerimeter
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Centroid()
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DCentroid()
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_PointOnSurface()
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_3DPointOnSurf()
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_Geometries()
   RETURNS ST_Surface ARRAY[ST_MaxGeometryArrayElements],

OVERRIDING METHOD ST_Geometries
   (ageometryarray ST_Geometry
      ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiSurface
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.
4) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The *ST_MultiSurface* type provides for public use:
   a) a method *ST_MultiSurface(CHARACTER LARGE OBJECT)*,
   b) a method *ST_MultiSurface(CHARACTER LARGE OBJECT, INTEGER)*,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

c) a method *ST_MultiSurface(BINARY LARGE OBJECT)*,

d) a method *ST_MultiSurface(BINARY LARGE OBJECT, INTEGER)*,

e) a method *ST_MultiSurface(ST_Surface ARRAY)*,

f) a method *ST_MultiSurface(ST_Surface ARRAY, INTEGER)*,

g) a method *ST_Area()*,

h) a method *ST_Area(CHARACTER VARYING)*,

i) a method *ST_3DArea()*,

j) a method *ST_3DArea(CHARACTER VARYING)*,

k) a method *ST_Perimeter()*,

l) a method *ST_Perimeter(CHARACTER VARYING)*,

m) a method *ST_3DPerimeter()*,

n) a method *ST_3DPerimeter(CHARACTER VARYING)*,

o) a method *ST_Centroid()*,

p) a method *ST_3DCentroid()*,

l) a method *ST_PointOnSurface()*,

q) a method *ST_3DPointOnSurf()*,

r) an overriding method *ST_Geometries()*,

s) an overriding method *ST_Geometries(ST_Geometry ARRAY)*,

t) a function *ST_MSurfaceFromTxt(CHARACTER LARGE OBJECT)*,

u) a function *ST_MSurfaceFromTxt(CHARACTER LARGE OBJECT, INTEGER)*,

v) a function *ST_MSurfaceFromWKB(BINARY LARGE OBJECT)*,

w) a function *ST_MSurfaceFromWKB(BINARY LARGE OBJECT, INTEGER)*,

x) a function *ST_MSurfaceFromGML(CHARACTER LARGE OBJECT)*,

y) a function *ST_MSurfaceFromGML(CHARACTER LARGE OBJECT, INTEGER)*.

2) The dimension of an *ST_MultiSurface* value is 2.

3) The interiors of any two *ST_Surface* values in an *ST_MultiSurface* shall not spatially intersect. The boundaries of any two coplanar elements in the *ST_MultiSurface* shall, at most, intersect at a finite number of points.

NOTE If they were to meet along a curve, they could be merged into a single surface.

4) An *ST_MultiSurface* value is simple.

5) An *ST_MultiSurface* value is well formed only if all of the *ST_Surface* values in the *ST_PrivateGeometries* attribute are well formed.

6) An *ST_MultiSurface* value returned by the constructor function corresponds to the empty set.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.5.2 ST_MultiSurface Methods

Return an ST_MultiSurface value constructed from either the well-known text representation, the well-known binary representation, the GML representation, or the specified ST_Surface values.

Definition

```
CREATE CONSTRUCTOR METHOD ST_MultiSurface
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_MultiSurface
   FOR ST_MultiSurface
   RETURN NEW ST_MultiSurface(awktorgml, 0)

CREATE CONSTRUCTOR METHOD ST_MultiSurface
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_MultiSurface
   FOR ST_MultiSurface
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_MultiSurface
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_MultiSurface
   FOR ST_MultiSurface
   RETURN NEW ST_MultiSurface(awkb, 0)

CREATE CONSTRUCTOR METHOD ST_MultiSurface
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_MultiSurface
   FOR ST_MultiSurface
   RETURN ST_MSurfaceFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_MultiSurface
   (asurfacearray ST_Surface ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiSurface
   FOR ST_MultiSurface
   RETURN SELF.ST_SRID(0).ST_Geometries(asurfacearray)

CREATE CONSTRUCTOR METHOD ST_MultiSurface
   (asurfacearray ST_Surface ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_MultiSurface
   FOR ST_MultiSurface
   RETURN SELF.ST_SRID(ansrid).ST_Geometries(asurfacearray)
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The method *ST_MultiSurface(CHARACTER LARGE OBJECT)* takes the following input parameter:
   a) a CHARACTER LARGE OBJECT value *awktorgml*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) The null-call type preserving SQL-invoked constructor method *ST_MultiSurface(CHARACTER LARGE OBJECT)* returns the result of the value expression: *NEW ST_MultiSurface(awktorgml, 0)*.

3) The method *ST_MultiSurface(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a CHARACTER LARGE OBJECT value *awktorgml*,

   b) an INTEGER value *ansrid*.

4) For the null-call type preserving SQL-invoked constructor method *ST_MultiSurface(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If *awktorgml* contains a MultiSurface XML element in the GML representation, then return the result of the value expression: *ST_MSurfaceFromGML(awktorgml, ansrid)*.

   b) Otherwise, return the result of the value expression: *ST_MSurfaceFromTxt(awktorgml, ansrid)*.

5) The method *ST_MultiSurface(BINARY LARGE OBJECT)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *awkb*.

6) The null-call type preserving SQL-invoked constructor method *ST_MultiSurface(BINARY LARGE OBJECT)* returns the result of the value expression: *NEW ST_MultiSurface(awkb, 0)*.

7) The method *ST_MultiSurface(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *awkb*,

   b) an INTEGER value *ansrid*.

8) The null-call type preserving SQL-invoked constructor method *ST_MultiSurface(BINARY LARGE OBJECT, INTEGER)* returns the result of the value expression: *ST_MSurfaceFromWKB(awkb, ansrid)*.

9) The method *ST_MultiSurface(ST_Surface ARRAY)* takes the following input parameters:

   a) an *ST_Surface* ARRAY value *asurfacearray*.

10) The null-call type preserving SQL-invoked constructor method *ST_MultiSurface(ST_Surface ARRAY)* returns an *ST_MultiSurface* value with:

    a) The spatial reference system identification set to 0 (zero),

    b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the attribute *ST_PrivateGeometries* set to *asurfacearray*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to 2.

11) The method *ST_MultiSurface(ST_Surface ARRAY, INTEGER)* takes the following input parameters:

    a) an *ST_Surface* ARRAY value *asurfacearray*,

    b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_MultiSurface(ST_Surface ARRAY, INTEGER)* returns an *ST_MultiSurface* value with:

    a) The spatial reference system identification set to *ansrid*.

    b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the attribute *ST_PrivateGeometries* set to *asurfacearray*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to 2.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.5.3 ST_Area Methods

Purpose

Return the area measurement of an ST_MultiSurface value, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_Area()
   RETURNS DOUBLE PRECISION
   FOR ST_MultiSurface
   BEGIN
      DECLARE area DOUBLE PRECISION;
      DECLARE counter INTEGER;

      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS DOUBLE PRECISION);
      END IF;
      SET area = 0.0;
      SET counter = 1;
      WHILE counter <= SELF.ST_NumGeometries() DO
         SET area = area + SELF.ST_GeometryN(counter).ST_Area();
         SET counter = counter + 1;
      END WHILE;
      RETURN area;
   END

CREATE METHOD ST_Area
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   FOR ST_MultiSurface
   BEGIN
      DECLARE area DOUBLE PRECISION;
      DECLARE counter INTEGER;

      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS DOUBLE PRECISION);
      END IF;
      SET area = 0.0;
      SET counter = 1;
      WHILE counter <= SELF.ST_NumGeometries() DO
         SET area = area + SELF.ST_GeometryN(counter).ST_Area(aunit);
         SET counter = counter + 1;
      END WHILE;
      RETURN area;
   END
```

Definitional Rules

1) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The method *ST_Area()* has no input parameters.
2) For the null-call method *ST_Area()*:
   a) Case:
      i) If SELF is an empty set, then return the null value.
      ii) Otherwise, return the sum of the *ST_Area()* values of the elements in the *ST_PrivateGeometries* attribute of SELF.
   b) Case:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

i) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_Area()* is in the linear unit of measure identified by <linear unit> squared.

ii) Otherwise, the value returned by *ST_Area()* is in an implementation-defined unit of measure.

3) The method *ST_Area(CHARACTER VARYING)* takes the following input parameter:

a) a CHARACTER VARYING value *aunit*.

4) For the null-call method *ST_Area(CHARACTER VARYING)*:

a) The values for *aunit* shall be a supported <unit name>.

b) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.

c) If the unit specified by *aunit* is not supported by the implementation to compute sum of the *ST_Area(aunit)* values of each element in the *ST_PrivateGeometries* attribute of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

d) Case:

i) If SELF is an empty set, then return the null value.

ii) Otherwise, return the sum of the *ST_Area(aunit)* values of each element in the *ST_PrivateGeometries* attribute of SELF.

e) The returned value is in the units indicated by *aunit*.

5) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.5.4 ST_3DArea Methods

Purpose

Return the area measurement of an ST_MultiSurface value, considering z coordinate values in the calculations.

Definition

```
CREATE METHOD ST_3DArea()
   RETURNS DOUBLE PRECISION
   FOR ST_MultiSurface
   BEGIN
      DECLARE area DOUBLE PRECISION;
      DECLARE counter INTEGER;

      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS DOUBLE PRECISION);
      END IF;
      SET area = 0.0;
      SET counter = 1;
      WHILE counter <= SELF.ST_NumGeometries() DO
         SET area = area + SELF.ST_GeometryN(counter).ST_3DArea();
         SET counter = counter + 1;
      END WHILE;
      RETURN area;
   END

CREATE METHOD ST_3DArea
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   FOR ST_MultiSurface
   BEGIN
      DECLARE area DOUBLE PRECISION;
      DECLARE counter INTEGER;

      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS DOUBLE PRECISION);
      END IF;
      SET area = 0.0;
      SET counter = 1;
      WHILE counter <= SELF.ST_NumGeometries() DO
         SET area = area + SELF.ST_GeometryN(counter).ST_3DArea(aunit);
         SET counter = counter + 1;
      END WHILE;
      RETURN area;
   END
```

Definitional Rules

1) *ST_MaxUnitNameLength* is the implementation-defined maximum length used for the character representation of a unit indication.

Description

1) The method *ST_3DArea()* has no input parameters.
2) For the null-call method *ST_3DArea()*:
   a) Case:
      i) If SELF is an empty set, then return the null value.
      ii) Otherwise, return the sum of the *ST_3DArea()* values of the elements in the *ST_PrivateGeometries* attribute of SELF.
   b) Case:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

i) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_3DArea()* is in the linear unit of measure identified by <linear unit> squared.

ii) Otherwise, the value returned by *ST_3DArea()* is in an implementation-defined unit of measure.

3) The method *ST_3DArea(CHARACTER VARYING)* takes the following input parameter:

a) a CHARACTER VARYING value *aunit*.

4) For the null-call method *ST_3DArea(CHARACTER VARYING)*:

a) The values for *aunit* shall be a supported <unit name>.

b) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.

c) If the unit specified by *aunit* is not supported by the implementation to compute sum of the *ST_3DArea(aunit)* values of each element in the *ST_PrivateGeometries* attribute of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

d) Case:

i) If SELF is an empty set, then return the null value.

ii) Otherwise, return the sum of the *ST_3DArea(aunit)* values of each element in the *ST_PrivateGeometries* attribute of SELF.

e) The returned value is in the units indicated by *aunit*.

5) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.5.5 ST_Perimeter Methods

Purpose

Return the length measurement of the boundary of an ST_MultiSurface value, ignoring z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_Perimeter()
   RETURNS DOUBLE PRECISION
   FOR ST_MultiSurface
   BEGIN
      DECLARE perimeter DOUBLE PRECISION;
      DECLARE counter INTEGER;

      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS DOUBLE PRECISION);
      END IF;
      SET perimeter = 0.0;
      SET counter = 1;
      WHILE counter <= SELF.ST_NumGeometries() DO
         SET perimeter = perimeter +
            SELF.ST_GeometryN(counter).ST_Perimeter();
         SET counter = counter + 1;
      END WHILE;
      RETURN perimeter;
   END

CREATE METHOD ST_Perimeter
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   FOR ST_MultiSurface
   BEGIN
      DECLARE perimeter DOUBLE PRECISION;
      DECLARE counter INTEGER;

      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS DOUBLE PRECISION);
      END IF;
      SET perimeter = 0.0;
      SET counter = 1;
      WHILE counter <= SELF.ST_NumGeometries() DO
         SET perimeter = perimeter +
            SELF.ST_GeometryN(counter).ST_Perimeter(aunit);
         SET counter = counter + 1;
      END WHILE;
      RETURN perimeter;
   END
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

Description

1) The method *ST_Perimeter()* has no input parameters.
2) For the null-call method *ST_Perimeter()*:
   a) Case:
      i) If SELF is an empty set, then return the null value.
      ii) Otherwise, return the sum of the *ST_Perimeter* value of the elements in the *ST_PrivateGeometries* attribute of SELF.

© ISO/IEC 2011 – All rights reserved

b) Case:

i) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_Perimeter()* is in the linear unit of measure identified by <linear unit> squared.

ii) Otherwise, the value returned by *ST_Perimeter()* is in an implementation-defined unit of measure.

3) The method *ST_Perimeter(CHARACTER VARYING)* takes the following input parameter:

a) a CHARACTER VARYING value *aunit*.

4) For the null-call method *ST_Perimeter(CHARACTER VARYING)*:

a) The values for *aunit* shall be a supported <unit name>.

b) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.

c) If the unit specified by *aunit* is not supported by the implementation to compute sum of the *ST_Perimeter(aunit)* values of each element in the *ST_PrivateGeometries* attribute of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

d) Case:

i) If SELF is an empty set, then return the null value.

ii) Otherwise, return the sum of the *ST_Perimeter(aunit)* values of each element in the *ST_PrivateGeometries* attribute of SELF.

e) The returned value is in the units indicated by *aunit*.

5) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are not considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.5.6 ST_3DPerimeter Methods

Purpose

Return the length measurement of the boundary of an ST_MultiSurface value, considering z and m coordinate values in the calculations.

Definition

```
CREATE METHOD ST_3DPerimeter()
   RETURNS DOUBLE PRECISION
   FOR ST_MultiSurface
   BEGIN
      DECLARE perimeter DOUBLE PRECISION;
      DECLARE counter INTEGER;

      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS DOUBLE PRECISION);
      END IF;
      SET perimeter = 0.0;
      SET counter = 1;
      WHILE counter <= SELF.ST_NumGeometries() DO
         SET perimeter = perimeter +
            SELF.ST_GeometryN(counter).ST_3DPerimeter();
         SET counter = counter + 1;
      END WHILE;
      RETURN perimeter;
   END

CREATE METHOD ST_3DPerimeter
   (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
   RETURNS DOUBLE PRECISION
   FOR ST_MultiSurface
   BEGIN
      DECLARE perimeter DOUBLE PRECISION;
      DECLARE counter INTEGER;

      IF SELF.ST_IsEmpty() = 1 THEN
         RETURN CAST (NULL AS DOUBLE PRECISION);
      END IF;
      SET perimeter = 0.0;
      SET counter = 1;
      WHILE counter <= SELF.ST_NumGeometries() DO
         SET perimeter = perimeter +
            SELF.ST_GeometryN(counter).ST_3DPerimeter(aunit);
         SET counter = counter + 1;
      END WHILE;
      RETURN perimeter;
   END
```

Description

1) The method *ST_3DPerimeter()* has no input parameters.
2) For the null-call method *ST_3DPerimeter()*:
   a) Case:
      i) If SELF is an empty set, then return the null value.
      ii) Otherwise, return the sum of the *ST_3DPerimeter* value of the elements in the *ST_PrivateGeometries* attribute of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

b) Case:

i) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_3DPerimeter()* is in the linear unit of measure identified by <linear unit> squared.

ii) Otherwise, the value returned by *ST_3DPerimeter()* is in an implementation-defined unit of measure.

3) The method *ST_3DPerimeter(CHARACTER VARYING)* takes the following input parameter:

a) a CHARACTER VARYING value *aunit*.

4) For the null-call method *ST_3DPerimeter(CHARACTER VARYING)*:

a) The values for *aunit* shall be a supported <unit name>.

b) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.

c) If the unit specified by *aunit* is not supported by the implementation to compute sum of the *ST_3DPerimeter(aunit)* values of each element in the *ST_PrivateGeometries* attribute of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

d) Case:

i) If SELF is an empty set, then return the null value.

ii) Otherwise, return the sum of the *ST_3DPerimeter(aunit)* values of each element in the *ST_PrivateGeometries* attribute of SELF.

e) The returned value is in the units indicated by *aunit*.

5) If *SELF.ST_Is3D()* is equal to 1 (one), then the z coordinate values are considered in the calculation.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.5.7 ST_Centroid Method

Purpose

Return the ST_Point value that is the mathematical centroid of the ST_MultiSurface value, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

Definition

```
CREATE METHOD ST_Centroid()
   RETURNS ST_Point
   FOR ST_MultiSurface
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_Centroid()* has no input parameters.
2) For the null-call method *ST_Centroid()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise:

      i) Rreturn the mathematical centroid of the *ST_MultiSurface* value. The result is not guaranteed to spatially intersect an *ST_Surface* value in the *ST_PrivateGeometries* attribute of an *ST_MultiSurface* value.

      ii) If *SELF.ST_Is3D()* is equal to 1 (one), then:

         1) The z coordinate values are not considered in the calculation.

         2) The *ST_Point* value does not include the z coordinate value.

      iii) If *SELF.ST_IsMeasured()* is equal to 1 (one), then:

         1) The m coordinate values are not considered in the calculation.

         2) The *ST_Point* value does not include the m coordinate value.

      iv) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.5.8 ST_3DCentroid Method

Purpose

Return the ST_Point value that is the mathematical centroid of the ST_MultiSurface value, considering z coordinate values in the calculations and including them in the resultant geometry.

Definition

```
CREATE METHOD ST_3DCentroid()
   RETURNS ST_Point
   FOR ST_MultiSurface
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_3DCentroid()* has no input parameters.
2) For the null-call method *ST_3DCentroid()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise:

      i) Return the mathematical centroid of the *ST_MultiSurface* value. The result is not guaranteed to spatially intersect an *ST_Surface* value in the *ST_PrivateGeometries* attribute of an *ST_MultiSurface* value.

      ii) If *SELF.ST_Is3D()* is equal to 1 (one), then:

         1) The z coordinate values are considered in the calculation.

         2) The *ST_Point* value includes the z coordinate value.

      iii) If *SELF.ST_IsMeasured()* is equal to 1 (one), then:

         1) The m coordinate values are not considered in the calculation.

         2) The *ST_Point* value does not include the m coordinate value.

      iv) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.5.9 ST_PointOnSurface Method

Purpose

Return an ST_Point value guaranteed to spatially intersect an ST_Surface value in the ST_PrivateGeometries attribute of an ST_MultiSurface value, ignoring z and m coordinate values in the calculations and not including them in the resultant geometry.

Definition

```
CREATE METHOD ST_PointOnSurface()
   RETURNS ST_Point
   FOR ST_MultiSurface
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_PointOnSurface()* has no input parameters.
2) For the null-call method *ST_PointOnSurface()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise:
      i) Return an *ST_Point* value guaranteed to spatially intersect an element in the collection of the *ST_MultiSurface* value.
      ii) If *SELF.ST_Is3D()* is equal to 1 (one), then:
         1) The z coordinate values are not considered in the calculation.
         2) The *ST_Point* value does not include the z coordinate value.
      iii) If *SELF.ST_IsMeasured()* is equal to 1 (one), then:
         1) The m coordinate values are not considered in the calculation.
         2) The *ST_Point* value does not include the m coordinate value.
      iv) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.5.10 ST_3DPointOnSurf Method

Purpose

Return an ST_Point value guaranteed to spatially intersect an ST_Surface value in the ST_PrivateGeometries attribute of an ST_MultiSurface value, considering z coordinate values in the calculations and including them in the resultant geometry.

Definition

```
CREATE METHOD ST_3DPointOnSurf()
   RETURNS ST_Point
   FOR ST_MultiSurface
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_3DPointOnSurf()* has no input parameters.
2) For the null-call method *ST_PointOnSurf()*:

   Case:

   a) If SELF is an empty set, then return the null value.
   b) Otherwise:
      i) Return an *ST_Point* value guaranteed to spatially 3D intersect an element in the collection of the *ST_MultiSurface* value.
      ii) If *SELF.ST_Is3D()* is equal to 1 (one), then:
         1) The z coordinate values are considered in the calculation.
         2) The *ST_Point* value includes the z coordinate value.
      iii) If *SELF.ST_IsMeasured()* is equal to 1 (one), then:
         1) The m coordinate values are not considered in the calculation.
         2) The *ST_Point* value does not include the m coordinate value.
      iv) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.5.11 ST_Geometries Methods

Purpose

Observe and mutate the ST_PrivateGeometries attribute of an ST_MultiSurface value.

Definition

```
CREATE METHOD ST_Geometries()
   RETURNS ST_Surface ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_MultiSurface
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CAST(SELF.ST_PrivateGeometries AS
               ST_Surface ARRAY[ST_MaxGeometryArrayElements])
      END

CREATE METHOD ST_Geometries
   (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiSurface
   FOR ST_MultiSurface
   BEGIN
      DECLARE acounter INTEGER;
      DECLARE bcounter INTEGER;
      DECLARE asurfacearray ST_Surface ARRAY[ST_MaxGeometryArrayElements];

      -- Cast ageometryarray to an ST_Surface ARRAY
      SET asurfacearray = CAST(ageometryarray AS
         ST_Surface ARRAY[ST_MaxGeometryArrayElements]);
      -- If any two surfaces intersect with the dimension of the result
      -- greater than 0 (zero), then raise an exception.
      SET acounter = 1;
      WHILE acounter <= CARDINALITY(asurfacearray)-1 DO
         SET bcounter = acounter+1;
         WHILE bcounter <= CARDINALITY(asurfacearray) DO
            IF asurfacearray[acounter].ST_Intersection(
               asurfacearray[bcounter]).ST_Dimension() > 0 THEN
               SIGNAL SQLSTATE '2FF02'
                  SET MESSAGE_TEXT = 'invalid argument';
            END IF;
            SET bcounter = bcounter + 1;
         END WHILE;
         SET acounter = acounter + 1;
      END WHILE;
      -- If SELF is the null value, then return the null value. Otherwise,
      -- return an ST_MultiSurface value containing asurfacearray.
      RETURN
         CASE
            WHEN SELF IS NULL THEN
               NULL
            ELSE
               (SELF AS ST_GeomCollection).
                  ST_Geometries(asurfacearray)
         END;
   END
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Description

1) The method *ST_Geometries()* has no input parameters.
2) For the null-call method *ST_Geometries()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the value of the *ST_PrivateGeometries* attribute as an *ST_Surface ARRAY*.
3) The method *ST_Geometries(ST_Geometry ARRAY)* takes the following input parameters:

   a) an *ST_Geometry ARRAY* value *ageometryarray*.
4) For the type preserving method *ST_Geometries(ST_Geometry ARRAY)*:

   a) Let *ASURFACEARRAY* be the result of casting *ageometryarray* to an *ST_Surface* ARRAY value (implicitly using *ST_ToSurfaceAry(ST_Geometry ARRAY)*).

   b) Case:

      i) If any two elements of *ASURFACEARRAY* intersect with more than a finite number of points, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

      ii) If SELF is the null value, then return the null value.

      iii) Otherwise, return an *ST_MultiSurface* value with:

         1) The dimension set to 2.

         2) Using the method *ST_Geometries(ST_Geometry ARRAY)* for type *ST_GeomCollection*, the *ST_PrivateGeometries* attribute set to *ASURFACEARRAY*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.5.12 ST_MSurfaceFromTxt Functions

Purpose

Return an ST_MultiSurface value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_MultiSurface value.

Definition

```
CREATE FUNCTION ST_MSurfaceFromTxt
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_MultiSurface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_MSurfaceFromText(awkt, 0)

CREATE FUNCTION ST_MSurfaceFromTxt
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_MultiSurface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_MSurfaceFromTxt(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_MSurfaceFromTxt(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_MSurfaceFromTxt(awkt, 0)*.
3) The function *ST_MSurfaceFromTxt(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_MSurfaceFromTxt(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkt* is the well-known text representation of an *ST_MultiSurface* value.

      If *awkt* is not producible in the BNF for <multisurface text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromText(awkt, ansrid) AS ST_MultiSurface)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.5.13 ST_MSurfaceFromWKB Functions

Purpose

Return an ST_MultiSurface value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_MultiSurface value.

Definition

```
CREATE FUNCTION ST_MSurfaceFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_MultiSurface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_MSurfaceFromWKB(awkb, 0)

CREATE FUNCTION ST_MSurfaceFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_MultiSurface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_MSurfaceFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_MSurfaceFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_MSurfaceFromWKB(awkb, 0)*.
3) The function *ST_MSurfaceFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_MSurfaceFromWKB(BINARY LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkb* is the well-known binary representation of an *ST_MultiSurface* value.

      If *awkb* is not producible in the BNF for <multisurface binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_MultiSurface)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.5.14 ST_MSurfaceFromGML Functions

Purpose

Return an ST_MultiSurface value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representationof an ST_MultiSurface value.

Definition

```
CREATE FUNCTION ST_MSurfaceFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
   RETURNS ST_MultiSurface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_MSurfaceFromGML(agml, 0)

CREATE FUNCTION ST_MSurfaceFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
    ansrid INTEGER)
   RETURNS ST_MultiSurface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_MSurfaceFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_MSurfaceFromGML*(CHARACTER LARGE OBJECT) returns the result of the value expression: *ST_MSurfaceFromGML(agml, 0)*.
3) The function *ST_MSurfaceFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_MSurfaceFromGML(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If the parameter *agml* does not contain a MultiSurface XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromGML(agml, ansrid) AS ST_MultiSurface)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.6 ST_MultiPolygon Type and Routines

9.6.1 ST_MultiPolygon Type

Purpose

The ST_MultiPolygon type is a subtype of the ST_MultiSurface and represents a collection of ST_Polygon values.

Definition

```
CREATE TYPE ST_MultiPolygon
   UNDER ST_MultiSurface
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_MultiPolygon
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
      RETURNS ST_MultiPolygon
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiPolygon
      (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
       ansrid INTEGER)
      RETURNS ST_MultiPolygon
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiPolygon
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
      RETURNS ST_MultiPolygon
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_MultiPolygon
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
       ansrid INTEGER)
      RETURNS ST_MultiPolygon
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_MultiPolygon
   (apolygonarray ST_Polygon
      ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiPolygon
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiPolygon
   (apolygonarray ST_Polygon
      ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_MultiPolygon
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_Geometries()
   RETURNS ST_Polygon ARRAY[ST_MaxGeometryArrayElements],

OVERRIDING METHOD ST_Geometries
   (ageometryarray ST_Geometry
      ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiPolygon
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.
2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The *ST_MultiPolygon* type provides for public use:
   a) a method *ST_MultiPolygon(CHARACTER LARGE OBJECT)*,
   b) a method *ST_MultiPolygon(CHARACTER LARGE OBJECT, INTEGER)*,
   c) a method *ST_MultiPolygon(BINARY LARGE OBJECT)*,
   d) a method *ST_MultiPolygon(BINARY LARGE OBJECT, INTEGER)*,
   e) a method *ST_MultiPolygon(ST_Polygon ARRAY)*,
   f) a method *ST_MultiPolygon(ST_Polygon ARRAY, INTEGER)*,
   g) an overriding method *ST_Geometries()*,
   h) an overriding method *ST_Geometries(ST_Geometry ARRAY)*,
   i) a function *ST_MPolyFromText(CHARACTER LARGE OBJECT)*,
   j) a function *ST_MPolyFromText(CHARACTER LARGE OBJECT, INTEGER)*,
   k) a function *ST_MPolyFromWKB(BINARY LARGE OBJECT)*,
   l) a function *ST_MPolyFromWKB(BINARY LARGE OBJECT, INTEGER)*,
   m) a function *ST_MPolyFromGML(CHARACTER LARGE OBJECT)*,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

n) a function *ST_MPolyFromGML(CHARACTER LARGE OBJECT, INTEGER)*,

o) a function *ST_BdMPolyFromText(CHARACTER LARGE OBJECT)*,

p) a function ST_*BdMPolyFromText(CHARACTER LARGE OBJECT, INTEGER)*,

q) a function *ST_BdMPolyFromWKB(BINARY LARGE OBJECT)*,

r) a function *ST_BdMPolyFromWKB(BINARY LARGE OBJECT, INTEGER)*.

2) The elements of the *ST_PrivateGeometries* attribute are restricted to *ST_Polygon* values.

3) The interiors of any two *ST_Polygon* values that are elements of the *ST_PrivateGeometries* attribute shall not spatially intersect.

$\forall$ m $\in$ *ST_MultiPolygon*, $\forall$ p_i, $p_j \in$ m.*ST_Geometries()*, $i \neq j$, Interior(p_i) $\cap$ Interior(p_j) = $\varnothing$

4) The boundaries of any two *ST_Polygon* values that are coplanar elements of the *ST_PrivateGeometries* attribute may only intersect at a finite number of points.

$\forall$ m $\in$ *ST_MultiPolygon*, $\forall$ p_i, $p_j \in$ m.*ST_Geometries()* p_i, p_j coplanar,
$\forall$ $c_i \in$ Boundary(p_i), $c_j \in$ Boundary(p_j) $c_i \cap c_j$ = { p_1, ..., p_k | $p_i \in$ *ST_Point*, $1 \leq i \leq k$ }

5) An *ST_MultiPolygon* is a topologically closed point set.

6) An *ST_MultiPolygon* shall not have cut lines, spikes or punctures.

$\forall$ m $\in$ *ST_MultiPolygon*, m = Closure(Interior(m))

7) The interior of an *ST_MultiPolygon* with more than one *ST_Polygon* value is not a connected point set. The number of connected components of the interior of an *ST_MultiPolygon* is equal to the cardinality of the *ST_PrivateGeometries* attribute.

8) The boundary of an *ST_MultiPolygon* is a set of linear rings corresponding to the boundaries of the *ST_Polygon* values of the *ST_PrivateGeometries*. Each linear ring in the boundary of the *ST_MultiPolygon* is in the boundary of exactly one *ST_Polygon* in the *ST_PrivateGeometries* attribute. Every linear ring in the boundary of an *ST_Polygon* in the *ST_PrivateGeometries* attribute is in the boundary of the *ST_MultiPolygon*.

9) An *ST_MultiPolygon* value is well formed only if and only if all of the *ST_Polygon* values in the *ST_PrivateGeometries* attribute are well formed.

10) An *ST_MultiPolygon* value returned by the constructor function corresponds to the empty set.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.6.2 ST_MultiPolygon Methods

Purpose

Return an ST_MultiPolygon value constructed from either the well-known text representation, the well-known binary representation, the GML representation, or the specified ST_Polygon values.

Definition

```
CREATE CONSTRUCTOR METHOD ST_MultiPolygon
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_MultiPolygon
   FOR ST_MultiPolygon
   RETURN NEW ST_MultiPolygon(awktorgml, 0)

CREATE CONSTRUCTOR METHOD ST_MultiPolygon
   (awktorgml CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_MultiPolygon
   FOR ST_MultiPolygon
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_MultiPolygon
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_MultiPolygon
   FOR ST_MultiPolygon
   RETURN NEW ST_MultiPolygon(awkb, 0)

CREATE CONSTRUCTOR METHOD ST_MultiPolygon
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_MultiPolygon
   FOR ST_MultiPolygon
   RETURN ST_MPolyFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_MultiPolygon
   (apolygonarray ST_Polygon ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiPolygon
   FOR ST_MultiPolygon
   RETURN SELF.ST_SRID(0).ST_Geometries(apolygonarray)

CREATE CONSTRUCTOR METHOD ST_MultiPolygon
   (apolygonarray ST_Polygon ARRAY[ST_MaxGeometryArrayElements],
    ansrid INTEGER)
   RETURNS ST_MultiPolygon
   FOR ST_MultiPolygon
   RETURN SELF.ST_SRID(ansrid).ST_Geometries(apolygonarray)
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

3) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text and GML representations of an *ST_Geometry* value.

Description

1) The method *ST_MultiPolygon(CHARACTER LARGE OBJECT)* takes the following input parameter:

   a) a CHARACTER LARGE OBJECT value *awktorgml*.

© ISO/IEC 2011 – All rights reserved

2) The null-call type preserving SQL-invoked constructor method *ST_MultiPolygon(CHARACTER LARGE OBJECT)* returns the result of the value expression: *NEW ST_MultiPolygon(awktorgml, 0)*.

3) The method *ST_MultiPolygon(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a CHARACTER LARGE OBJECT value *awktorgml*,

   b) an INTEGER value *ansrid*.

4) For the null-call type preserving SQL-invoked constructor method *ST_MultiPolygon(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If *awktorgml* contains a MultiPolygon XML element in the GML representation, then return the result of the value expression: *ST_MPolyFromGML(awktorgml, ansrid)*.

   b) Otherwise, return the result of the value expression: *ST_MPolyFromText(awktorgml, ansrid)*.

5) The method *ST_MultiPolygon(BINARY LARGE OBJECT)* takes the following input parameter:

   a) a BINARY LARGE OBJECT value *awkb*.

6) The null-call type preserving SQL-invoked constructor method *ST_MultiPolygon(BINARY LARGE OBJECT)* returns the result of the value expression: *NEW ST_MultiPolygon(awkb, 0)*.

7) The method *ST_MultiPolygon(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *awkb*,

   b) an INTEGER value *ansrid*.

8) The null-call type preserving SQL-invoked constructor method *ST_MultiPolygon(BINARY LARGE OBJECT, INTEGER)* returns the result of the value expression: *ST_MPolygonFromWKB(awkb, ansrid)*.

9) The method *ST_MultiPolygon(ST_Polygon ARRAY)* takes the following input parameters:

   a) an *ST_Polygon* value *apolygonarray*.

10) The null-call type preserving SQL-invoked constructor method *ST_MultiPolygon(ST_Polygon ARRAY)* returns an *ST_MultiPolygon* value with:

    a) The spatial reference system identifier set to 0 (zero).

    b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the *ST_PrivateGeometries* attribute set to *apolygonarray*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to 2.

11) The method *ST_MultiPolygon(ST_Polygon ARRAY, INTEGER)* takes the following input parameters:

    a) an *ST_Polygon* ARRAY value *apolygonarray*,

    b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_MultiPolygon(ST_Polygon ARRAY, INTEGER)* returns an *ST_MultiPolygon* value with:

    a) The spatial reference system identifier set to *ansrid*.

    b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the *ST_PrivateGeometries* attribute set to *apolygonarray*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to 2.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.6.3 ST_Geometries Methods

Purpose

Observe and mutate the ST_PrivateGeometries attribute of an ST_MultiPolygon value.

Definition

```
CREATE METHOD ST_Geometries()
   RETURNS ST_Polygon ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_MultiPolygon
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CAST(SELF.ST_PrivateGeometries AS
               ST_Polygon ARRAY[ST_MaxGeometryArrayElements])
      END

CREATE METHOD ST_Geometries
   (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiPolygon
   FOR ST_MultiPolygon
   BEGIN
      DECLARE acounter INTEGER;
      DECLARE bcounter INTEGER;
      DECLARE apolygonarray ST_Polygon ARRAY[ST_MaxGeometryArrayElements];

      -- Cast ageometryarray to an ST_Polygon ARRAY
      SET apolygonarray = CAST(ageometryarray AS
         ST_Polygon ARRAY[ST_MaxGeometryArrayElements]);
      -- If any two polygons intersect with the dimension of the result
      -- greater than 0 (zero), then raise an exception.
      SET acounter = 1;
      WHILE acounter <= CARDINALITY(apolygonarray)-1 DO
         SET bcounter = acounter+1;
         WHILE bcounter <= CARDINALITY(apolygonarray) DO
            IF apolygonarray[acounter].ST_Intersection(
               apolygonarray[bcounter]).ST_Dimension() > 0 THEN
               SIGNAL SQLSTATE '2FF02'
                  SET MESSAGE_TEXT = 'invalid argument';
            END IF;
            SET bcounter = bcounter + 1;
         END WHILE;
         SET acounter = acounter + 1;
      END WHILE;
      -- If SELF is the null value, then return the null value. Otherwise,
      -- return an ST_MultiPolygon value containing apolygonarray.
      RETURN
         CASE
            WHEN SELF IS NULL THEN
               NULL
            ELSE
               (SELF AS ST_MultiSurface).
                  ST_Geometries(apolygonarray)
         END;
   END
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Description

1) The method *ST_Geometries()* has no input parameters.
2) For the null-call method *ST_Geometries()*:

   Case:

   a) If SELF is an empty set, then return the null value.

   b) Otherwise, return the value of the *ST_PrivateGeometries* attribute as an *ST_Polygon ARRAY*.
3) The method *ST_Geometries(ST_Geometry ARRAY)* takes the following input parameters:

   a) an *ST_Geometry ARRAY* value *ageometryarray*.
4) For the type preserving method *ST_Geometries(ST_Geometry ARRAY)*:

   a) Let *APOLYGONARRAY* be the result of casting *ageometryarray* to an *ST_Polygon* ARRAY value (implicitly using *ST_ToPolygonAry(ST_Geometry ARRAY*).

   b) Case:

      i) If any two elements of *APOLYGONARRAY* intersect with more than a finite number of points, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

      ii) If SELF is the null value, then return the null value.

      iii) Otherwise, return an *ST_MultiPolygon* value with:

         1) The dimension set to 2.

         2) Using the method *ST_Geometries(ST_Geometry ARRAY)* for type *ST_MultiSurface*, the *ST_PrivateGeometries* attribute set to *APOLYGONARRAY*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.6.4 ST_MPolyFromText Functions

Purpose

Return an ST_MultiPolygon value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_MultiPolygon value.

Definition

```
CREATE FUNCTION ST_MPolyFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_MultiPolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_MPolyFromText(awkt, 0)

CREATE FUNCTION ST_MPolyFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    ansrid INTEGER)
   RETURNS ST_MultiPolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

1) The function *ST_MPolyFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The null-call function *ST_MPolyFromText(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_MPolyFromText(awkt, 0)*.
3) The function *ST_MPolyFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_MPolyFromText(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkt* is the well-known text representation of an *ST_MultiPolygon* value.

      If *awkt* is not producible in the BNF for <multipolygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromText(awkt, ansrid) AS ST_MultiPolygon)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.6.5 ST_MPolyFromWKB Functions

Purpose

Return an ST_MultiPolygon value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_MultiPolygon value.

Definition

```
CREATE FUNCTION ST_MPolyFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_MultiPolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_MPolyFromWKB(awkb, 0)

CREATE FUNCTION ST_MPolyFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_MultiPolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum length of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The function *ST_MPolyFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) The null-call function *ST_MPolyFromWKB(BINARY LARGE OBJECT)* returns the result of the value expression: *ST_MPolyFromWKB(awkb, 0)*.
3) The function *ST_MPolyFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_MPolyFromWKB(BINARY LARGE OBJECT, INTEGER)*:

   Case:

   a) The parameter *awkb* is the well-known binary representation of an *ST_MultiPolygon* value.

      If *awkb* is not producible in the BNF for <multipolygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_MultiPolygon)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.6.6 ST_MPolyFromGML Functions

Purpose

Return an ST_MultiPolygon value which is transformed from a CHARACTER LARGE OBJECT value that represents the GML representationof an ST_MultiPolygon value.

Definition

```
CREATE FUNCTION ST_MPolyFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
   RETURNS ST_MultiPolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_MPolyFromGML(agml, 0)

CREATE FUNCTION ST_MPolyFromGML
   (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
    ansrid INTEGER)
   RETURNS ST_MultiPolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxGeometryAsGML* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the GML representation of an *ST_Geometry* value.

Description

1) The function *ST_MPolyFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*.
2) The null-call function *ST_MPolyFromGML(CHARACTER LARGE OBJECT)* returns the result of the value expression: *ST_MPolyFromGML(agml, 0)*.
3) The function *ST_MPolyFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *agml*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_MPolyFromGML(CHARACTER LARGE OBJECT, INTEGER)*:

   Case:

   a) If the parameter *agml* does not contain a MultiPolygon XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
   b) Otherwise, return the result of the value expression: *TREAT(ST_GeomFromGML(agml, ansrid) AS ST_MultiPolygon)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.6.7 ST_BdMPolyFromText Functions

Purpose

Return an ST_MultiPolygon value which is transformed from a CHARACTER LARGE OBJECT value that represents the well-known text representation of an ST_MultiLineString value.

Definition

```
CREATE FUNCTION ST_BdMPolyFromText
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
   RETURNS ST_MultiPolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_BdMPolyFromText(awkt, 0)

CREATE FUNCTION ST_BdMPolyFromText
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
     ansrid INTEGER)
   RETURNS ST_MultiPolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The function *ST_BdMPolyFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) For the null-call function *ST_BdMPolyFromText(CHARACTER LARGE OBJECT)* returns an *ST_MultiPolygon* value as the result of the value expression: *ST_BdMPolyFromText(awkt, 0)*.
3) The function *ST_BdMPolyFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_BdMPolyFromText(CHARACTER LARGE OBJECT, INTEGER)*:
   a) The parameter *awkt* is the well-known text representation of an *ST_MultiLineString* value. If *awkt* is not producible in the BNF for <multilinestring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
   b) Use *ST_MLineFromText(CHARACTER LARGE OBJECT)* to transform *awkt* to an *ST_MultiLineString* value, *AMLS*.
   c) If any *ST_LineString* value in *AMLS* is not a linear ring, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
   d) Using an implementation-dependent algorithm, an array of *ST_Polygon* values, *APA*, is determined from the array of linear rings in *AMLS*.
   e) Return an *ST_MultiPolygon* value with:
      i) The spatial reference system identifier set to *ansrid*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the *ST_PrivateGeometries* attribute set to *APA*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

9.6.8 ST_BdMPolyFromWKB Functions

Purpose

Return an ST_MultiPolygon value which is transformed from a BINARY LARGE OBJECT value that represents the well-known binary representation of an ST_MultiLineString value.

Definition

```
CREATE FUNCTION ST_BdMPolyFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
   RETURNS ST_MultiPolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   RETURN ST_BdMPolyFromWKB(awkb, 0)

CREATE FUNCTION ST_BdMPolyFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    ansrid INTEGER)
   RETURNS ST_MultiPolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The function *ST_BdMPolyFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*.
2) For the null-call function *ST_BdMPolyFromWKB(BINARY LARGE OBJECT)* returns an *ST_MultiPolygon* value as the result of the value expression: *ST_BdMPolyFromWKB(awkt, 0)*.
3) The function *ST_BdMPolyFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) an BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_BdMPolyFromWKB(BINARY LARGE OBJECT, INTEGER)*:
   a) The parameter *awkb* is the well-known binary representation of an *ST_MultiLineString* value. If *awkb* is not producible in the BNF for <multilinestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
   b) Use *ST_MLineFromWKB(BINARY LARGE OBJECT)* to transform *awkb* to an *ST_MultiLineString* value, *AMLS*.
   c) If any *ST_LineString* value in *AMLS* is not a linear ring, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
   d) Using an implementation-dependent algorithm, an array of *ST_Polygon* values, *APA*, is determined from the array of linear rings in *AMLS*.
   e) Return an *ST_MultiPolygon* value with:
      i) The spatial reference system identifier set to *ansrid*.
      ii) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the *ST_PrivateGeometries* attribute set to *APA*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10 Topology-Geometry

10.1 Topo-Geo Topology Schema

10.1.1 Introduction

The Topo-Geo Topology Schema views are defined as being in a schema named <topology-name> enabling these views to be accessed in the same way as any other tables in any other schema. SELECT privilege on all of these views may be granted to individual users so that they can be queried. To update a Topo-Geo, a user shall be granted SELECT privlilege on the views for the Topo-Geo and EXECUTE privilege on the Topo-Geo routines provided. Roles can be used to enable update on a selective set of Topo-Geos. These views are updated only by the topology functions provided.

An implementation may define objects that are associated with <topology-name> that are not defined in this Clause. An implementation may also add columns to tables that are defined in this Clause.

All of the topological primitives contained in the views owned by the <topology-name> schema constitute a single, topologically consistent, topology representing a planar graph. Other topologies (e.g. covering a different spatial extent, existing at a different level of abstraction, or associated with a different set of features) can exist in another, independent <topology-name> schema.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.1.2 ST_NODE view

Purpose

Contains the node type of topological primitives (ST_Node) contained in the <topology-name> Topo-Geo.

Definition

```
CREATE VIEW ST_NODE AS
   SELECT NODE_ID, GEOMETRY, CONTAINING_FACE
      FROM ST_TOPO_GEO.ST_NODE
      WHERE TOPOLOGY = '<topology-name>'
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.1.3 ST_EDGE view

Purpose

Contains the edge type of topological primitives (ST_Edge) contained in the <topology-name> Topo-Geo.

Definition

```
CREATE VIEW ST_EDGE AS
   SELECT EDGE_ID, START_NODE, END_NODE,
          NEXT_LEFT_EDGE, NEXT_RIGHT_EDGE,
          LEFT_FACE, RIGHT_FACE, GEOMETRY
      FROM ST_TOPO_GEO.ST_EDGE
      WHERE TOPOLOGY = '<topology-name>'
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.1.4 ST_FACE view

Purpose

Contains the face type of topological primitives (ST_Face) contained in the <topology-name> Topo-Geo.

Definition

```
CREATE VIEW ST_FACE AS
   SELECT FACE_ID, MBR
      FROM ST_TOPO_GEO.ST_FACE
      WHERE TOPOLOGY = '<topology-name>'
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.2 Topo-Geo Definition Schema

10.2.1 Introduction

The only purpose of this Topo-Geo Definition Schema is to provide a data model to support Topo-Geo views and to assist understanding. The base tables of this Topo-Geo Definition Schema are defined as being in a schema named ST_TOPO_GEO. The table definitions are as complete as the definitional power of ISO/IEC 9075 allows. The table definitions are supplemented with assertions where appropriate. Each description comprises three parts:

1) The function of the definition is stated.
2) The SQL definition of the object is presented as a <table definition>.
3) An explanation of the object.

The specification provides only a model of the base tables that are required, and does not imply that an implementation shall provide the functionality in the manner described in this clause.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.2.2 ST_NODE base table

Purpose

Contains the node type of topological primitives (ST_Node) contained in topology-geometries.

Definition

```
CREATE TABLE ST_NODE
   (
   TOPOLOGY CHARACTER VARYING(ST_MaxTopologyName),
   NODE_ID INTEGER NOT NULL,
   GEOMETRY ST_Point NOT NULL,
   CONTAINING_FACE INTEGER,

   CONSTRAINT ST_NODE_PRIMARY_KEY PRIMARY KEY(TOPOLOGY, NODE_ID),
   CONSTRAINT FACE_EXISTS FOREIGN KEY(TOPOLOGY, CONTAINING_FACE)
      REFERENCES ST_FACE(TOPOLOGY, FACE_ID)
   )
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The value of *TOPOLOGY* is the topology name which distinguishes which nodes are in the topology coverage.
2) Let *T* be the value of *TOPOLOGY* for a given node.
3) The value of *NODE_ID* is the identifier of the node unique among all nodes with a *TOPOLOGY* value equal to *T*.
4) The value of *GEOMETRY* is the spatial location of the node.
5) The value of *CONTAINING_FACE* is the unique identifier of the face containing the node if the node is an isolated node; the face shall have a value of *TOPOLOGY* equal to *T*. Otherwise *CONTAINING_FACE* is the null value.
6) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.2.3 ST_EDGE base table

Purpose

Contains the edge type of topological primitives (ST_Edge) contained in topology-geometries.

Definition

```
CREATE TABLE ST_EDGE
   (
   TOPOLOGY CHARACTER VARYING(ST_MaxTopologyName),
   EDGE_ID INTEGER NOT NULL,
   START_NODE INTEGER NOT NULL,
   END_NODE INTEGER NOT NULL,
   NEXT_LEFT_EDGE INTEGER NOT NULL,
   NEXT_RIGHT_EDGE INTEGER NOT NULL,
   LEFT_FACE INTEGER NOT NULL,
   RIGHT_FACE INTEGER NOT NULL,
   GEOMETRY ST_Curve NOT NULL,

   CONSTRAINT ST_EDGE_PRIMARY_KEY PRIMARY KEY(TOPOLOGY, EDGE_ID),
   CONSTRAINT START_NODE_EXISTS FOREIGN KEY(TOPOLOGY, START_NODE)
      REFERENCES ST_NODE(TOPOLOGY, NODE_ID),
   CONSTRAINT END_NODE_EXISTS FOREIGN KEY(TOPOLOGY, END_NODE)
      REFERENCES ST_NODE(TOPOLOGY, NODE_ID),
   CONSTRAINT NEXT_LEFT_EDGE_EXISTS
      CHECK (TOPOLOGY, ABS(NEXT_LEFT_EDGE)) IN
         (SELECT TOPOLOGY, EDGE_ID FROM ST_EDGE),
   CONSTRAINT NEXT_RIGHT_EDGE_EXISTS
      CHECK (TOPOLOGY, ABS(NEXT_RIGHT_EDGE)) IN
         (SELECT TOPOLOGY, EDGE_ID FROM ST_EDGE)),
   CONSTRAINT LEFT_FACE_EXISTS FOREIGN KEY(TOPOLOGY, LEFT_FACE)
      REFERENCES ST_FACE(TOPOLOGY, FACE_ID),
   CONSTRAINT RIGHT_FACE_EXISTS FOREIGN KEY(TOPOLOGY, RIGHT_FACE)
      REFERENCES ST_FACE(TOPOLOGY, FACE_ID)
   )
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The value of *TOPOLOGY* is the topology name which distinguishes which edges are in the topology coverage.
2) Let *T* be the value of *TOPOLOGY* for a given edge.
3) The value of *EDGE_ID* is the identifier of the edge unique among all edges with a *TOPOLOGY* value equal to *T*.
4) The value of *START_NODE* is the unique identifier of the node at the start of the edge; the start node shall have a value of *TOPOLOGY* equal to *T*.
5) The value of *END_NODE* is the unique identifier of the node at the end of the edge; the end node shall have a value of *TOPOLOGY* equal to *T*.
6) The value of *NEXT_LEFT_EDGE* is the unique identifier of the next edge of the face on the left (when looking in the direction from *START_NODE* to *END_NODE*), moving counterclockwise around the face boundary; the next left edge node shall have a value of *TOPOLOGY* equal to *T*.
7) The value of *NEXT_RIGHT_EDGE* is the unique identifier of the next edge of the face on the right (when looking in the direction from *START_NODE* to *END_NODE*), moving counterclockwise around the face boundary; the next right edge shall have a value of *TOPOLOGY* equal to *T*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

8) The value of *LEFT_FACE* is the unique identifier of the face on the left side of the edge when looking in the direction from *START_NODE* to *END_NODE*; the left face shall have a value of *TOPOLOGY* equal to *T*.

9) The value of *RIGHT_FACE* is the unique identifier of the face on the right side of the edge when looking in the direction from *START_NODE* to *END_NODE*; the right face shall have a value of *TOPOLOGY* equal to *T*.

10) The value of *GEOMETRY* is the geometry of the edge.

11) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.2.4 ST_FACE base table

Purpose

Contains the face type of topological primitives (ST_Face) contained in topology-geometries.

Definition

```
CREATE TABLE ST_FACE
   (
   TOPOLOGY CHARACTER VARYING(ST_MaxTopologyName),
   FACE_ID INTEGER NOT NULL,
   MBR ST_Polygon,

   CONSTRAINT ST_FACE_PRIMARY_KEY PRIMARY KEY(TOPOLOGY, FACE_ID)
   )
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The value of *TOPOLOGY* is the topology name which distinguishes which faces are in the topology coverage.
2) Let *T* be the value of *TOPOLOGY* for a given face.
3) The value of *FACE_ID* is the identifier of the face unique among all faces with a *TOPOLOGY* value equal to *T*.
4) The value of *MBR* is the geometry of the minimum bounding rectangle of the face.
5) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3 Topo-Geo Routines

10.3.1 ST_AddIsoNode Function

Purpose

Insert a row into the <topology-name>.*ST_NODE* view for an isolated node.

Definition

```
CREATE FUNCTION ST_AddIsoNode
   (atopology CHARACTER VARYING(ST_MaxTopologyName),
    aface INTEGER,
    apoint ST_Point)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The function *ST_AddIsoNode(CHARACTER VARYING, INTEGER, ST_Point)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *aface*,
   c) an *ST_Point* value *apoint*.
2) For the function *ST_AddIsoNode(CHARACTER VARYING, INTEGER, ST_Point)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *apoint* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

      Case:

      i) If any *ST_TOPO_GEO.ST_NODE.GEOMETRY* value is equal to *apoint*, then an exception condition is raised: *SQL/MM Spatial exception – coincident node*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) Let *E1* be an INTEGER value equal to *ST_TOPO_GEO.ST_EDGE.EDGE_ID* for an edge in *atopology*. Let *G1* be an *ST_Curve* value equal to *ST_TOPO_GEO.ST_EDGE.GEOMETRY* where *EDGE_ID* is equal to *E1*. If the value returned by *G1.ST_Crosses(apoint)* is equal to 1 (one) for any value of *E1*, then an exception condition is raised: *SQL/MM Spatial exception – edge crosses node.*

iii) If *aface* is not the null value and *apoint.ST_Within(ST_GetFaceGeometry(atopology, aface))* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – not within face*.

vi) Otherwise:

1) generate a unique node id INTEGER value *anodeid*,

2) insert into *ST_TOPO_GEO.ST_NODE* values (*atopology*, *anodeid*, *aface*, *apoint*),

3) return *anodeid*.

3) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.2 ST_MoveIsoNode Procedure

Purpose

Update the <topology-name>.*ST_NODE.GEOMETRY* value of an isolated node.

Definition

```
CREATE PROCEDURE ST_MoveIsoNode
   (atopology CHARACTER VARYING(ST_MaxTopologyName),
    anode INTEGER,
    apoint ST_Point)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The procedure *ST_MoveIsoNode(CHARACTER VARYING, INTEGER, ST_Point)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *anode*,
   c) an *ST_Point* value *apoint*.
2) For the procedure *ST_MoveIsoNode(CHARACTER VARYING, INTEGER, ST_Point)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *anode* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *apoint* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   f) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

      Case:

      i) If any *ST_TOPO_GEO.ST_NODE.GEOMETRY* value is equal to *apoint*, then an exception condition is raised: *SQL/MM Spatial exception – coincident node*.
      ii) If any *ST_TOPO_GEO.ST_EDGE.START_NODE* or *ST_TOPO_GEO.ST_EDGE.END_NODE* value is equal to *anode*, then an exception condition is raised: *SQL/MM Spatial exception – not isolated node*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

iii) Let *E1* be an INTEGER value equal to *ST_TOPO_GEO.ST_EDGE.EDGE_ID* for an edge in *atopology*. Let *G1* be an *ST_Curve* value equal to *ST_TOPO_GEO.ST_EDGE.GEOMETRY* where *EDGE_ID* is equal to *E1*. If the value returned by *G1.ST_Crosses(apoint)* is equal to 1 (one) for any value of *E1*, then an exception condition is raised: *SQL/MM Spatial exception – edge crosses node.*

iv) Otherwise, update *ST_TOPO_GEO.ST_NODE*, set the *GEOMETRY* value equal to *apoint* where *TOPOLOGY* is equal to *atopology* and *NODE_ID* is equal to *anode*.

3) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.3 ST_RemIsoNode Procedure

Purpose

Delete a row in the <topology-name>.*ST_NODE* view corresponding to an isolated node.

Definition

```
CREATE PROCEDURE ST_RemIsoNode
   (atopology CHARACTER VARYING(ST_MaxTopologyName),
    anode INTEGER)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The procedure *ST_RemIsoNode(CHARACTER VARYING, INTEGER)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *anode*.
2) For the procedure *ST_RemIsoNode(CHARACTER VARYING, INTEGER)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *anode* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

      Case:

      i) If any *ST_TOPO_GEO.ST_EDGE.START_NODE* or *ST_TOPO_GEO.ST_EDGE.END_NODE* value is equal to *anode*, then an exception condition is raised: *SQL/MM Spatial exception – not isolated node*.
      ii) Otherwise, delete from *ST_TOPO_GEO.ST_NODE* where *TOPOLOGY* is equal to *atopology* and *NODE_ID* is equal to *anode*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.4 ST_AddIsoEdge Function

Purpose

Insert a row for an isolated edge into the <topology-name>.*ST_EDGE* view connecting two existing isolated nodes.

Definition

```
CREATE FUNCTION ST_AddIsoEdge
   (atopology CHARACTER VARYING(ST_MaxTopologyName),
    anode INTEGER,
    anothernode INTEGER,
    acurve ST_Curve)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The function *ST_AddIsoEdge(CHARACTER VARYING, INTEGER, INTEGER, ST_Curve)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *anode*,
   c) an INTEGER value *anothernode*,
   d) an ST_Curve value *acurve*.
2) For the function *ST_AddIsoEdge(CHARACTER VARYING, INTEGER, INTEGER, ST_Curve)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology.ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *anode* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *anothernode* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   f) If *acurve* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   g) If the value returned by *acurve.ST_IsSimple()* is not equal to 1 (one), then an exception condition is raised: *SQL/MM Spatial exception – curve not simple*.
   h) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Case:

i) If SELECT COUNT(*) FROM *ST_TOPO_GEO.ST_NODE* WHERE *NODE_ID* IS EQUAL TO *anode* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent node.*

ii) If SELECT COUNT(*) FROM *ST_TOPO_GEO.ST_NODE* WHERE *NODE_ID* IS EQUAL TO *anothernode* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent node.*

iii) If any *ST_TOPO_GEO.ST_EDGE.START_NODE* or *ST_TOPO_GEO.ST_EDGE.END_NODE* value is equal to *anode*, then an exception condition is raised: *SQL/MM Spatial exception – not isolated node*.

iv) If any *ST_TOPO_GEO.ST_EDGE.START_NODE* or *ST_TOPO_GEO.ST_EDGE.END_NODE* value is equal to *anothernode*, then an exception condition is raised: *SQL/MM Spatial exception – not isolated node*.

v) Let *F1* be the INTEGER value equal to *ST_TOPO_GEO.ST_NODE.CONTAINING_FACE* where *NODE_ID* is equal to *anode*. Let *F2* be the INTEGER value equal to *ST_TOPO_GEO.ST_NODE.CONTAINING_FACE* where *NODE_ID* is equal to *anothernode*. If *F1* is not equal to *F2*, then an exception condition is raised: *SQL/MM Spatial exception – nodes in different faces*.

vi) If *acurve.ST_Within(ST_GetFaceGeometry(atopology, F1))* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – geometry not within face*.

vii) Let *P1* be the *ST_Point* value equal to *ST_TOPO_GEO.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *anode*. If *P1* is not equal to *acurve.ST_StartPoint()*, then an exception condition is raised: *SQL/MM Spatial exception – start node not geometry start point*.

vii) Let *P2* be the ST_Point value equal to *ST_TOPO_GEO.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *anothernode*. If *P2* is not equal to *acurve.ST_EndPoint()*, then an exception condition is raised: *SQL/MM Spatial exception – end node not geometry end point*.

viii) Let *N1* be a *NODE_ID* value in *ST_TOPO_GEO.ST_NODE* such that N1 does not exist as either an *ST_TOPO_GEO.ST_EDGE.START_NODE* or *ST_TOPO_GEO.ST_EDGE.END_NODE* value. Let G1 be the *ST_TOPO_GEO.ST_NODE.GEOMETRY* value where *NODE_ID* is equal to *N1*. If *acurve.ST_Crosses(G1)* is not equal to 0 (zero) for all values of *N1*, then an exception condition is raised: *SQL/MM Spatial exception – geometry crosses a node*.

ix) Let *G* be an *ST_TOPO_GEO.ST_EDGE.GEOMETRY* value. If *acurve.ST_Intersects(G)* is not equal to 0 (zero) for all values of *G*, then an exception condition is raised: *SQL/MM Spatial exception – geometry intersects an edge*.

x) Otherwise:

1) generate a unique edge id INTEGER value *anedgeid*,

2) insert into TOPO_GEO.*ST_EDGE* values (*atopology*, *anedgeid*, *anode*, *anothernode*, -(*anedgeid*), *anedgeid*, *F1*, *F1*, *acurve*),

3) return *anedgeid*.

3) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.5 ST_GetFaceEdges Function

Purpose

Return a table containing signed edge IDs for the edges that bound a face, in counterclockwise order.

Definition

```
CREATE FUNCTION ST_GetFaceEdges
   (atopology CHARACTER VARYING(ST_MaxTopologyName),
    aface INTEGER)
   RETURNS TABLE
      (sequence INTEGER,
       edge INTEGER)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The function *ST_GetFaceEdges(CHARACTER VARYING, INTEGER)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *aface*.
2) For the function *ST_GetFaceEdges(CHARACTER VARYING, INTEGER)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *aface* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:
      i) Return a table value consisting of the following two columns:
         1) a column *edge* of type INTEGER, which contains the signed edge IDs of the bounding edges of the face having an *ST_TOPO_GEO.ST_FACE.FACE_ID* value equal to *aface*, such that edge IDs for edges having the specified face on the right side of the edge when looking in the direction of the edge from start node to end node are negated
         2) a column *sequence* containing consecutive values of type INTEGER, which represents the ordering of the edges in column *edge* consistent with a counterclockwise traversal around the face. The row with the lowest value for *edge* shall have a *sequence* value of 1 (one).

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.6 ST_ChangeEdgeGeom Procedure

Purpose

Update the <topology-name>.*ST_EDGE.GEOMETRY* value.

Definition

```
CREATE PROCEDURE ST_ChangeEdgeGeom
   (atopology CHARACTER VARYING(ST_MaxTopologyName),
    anedge INTEGER,
    acurve ST_Curve)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The procedure *ST_ChangeEdgeGeom(CHARACTER VARYING, INTEGER, ST_Curve)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *anedge*,
   c) an *ST_Curve* value *acurve*.
2) For the procedure *ST_ChangeEdgeGeom(CHARACTER VARYING, INTEGER, ST_Curve)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *anedge* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *acurve* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   f) If the value returned by *acurve.ST_IsSimple()* is not equal to 1 (one), then an exception condition is raised: *SQL/MM Spatial exception – curve not simple*.
   g) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

      Case:

      i) Let *P1* be the *ST_Point* value equal to *ST_TOPO_GEO.ST_NODE.GEOMETRY* where *NODE_ID* is equal to the value of *ST_TOPO_GEO.ST_EDGE.START_NODE* where *EDGE_ID* is equal to *anedge*. If *P1* is not equal to *acurve.ST_StartPoint()*, then an exception condition is raised: *SQL/MM Spatial exception – start node not geometry start point*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) Let *P2* be the *ST_Point* value equal to *ST_TOPO_GEO.ST_NODE.GEOMETRY* where *NODE_ID* is equal to the value of *ST_TOPO_GEO.ST_EDGE.END_NODE* where *EDGE_ID* is equal to *anothernode*. If *P2* is not equal to *acurve.ST_EndPoint()*, then an exception condition is raised: *SQL/MM Spatial exception – end node not geometry end point*.

iii) Let *N1* be a *NODE_ID* value in *ST_TOPO_GEO.ST_NODE* such that *N1* does not exist as either an *ST_TOPO_GEO.ST_EDGE.START_NODE* or *ST_TOPO_GEO.ST_EDGE.END_NODE* value. Let *G1* be the *ST_TOPO_GEO.ST_NODE.GEOMETRY* value where *NODE_ID* is equal to *N1*. If *acurve.ST_Crosses(G1)* is not equal to 0 (zero) for all values of *N1*, then an exception condition is raised: *SQL/MM Spatial exception – geometry crosses a node*.

iv) Let *G* be an *ST_TOPO_GEO.ST_EDGE.GEOMETRY* value. If *acurve.ST_Crosses(G)* is not equal to 0 (zero) for all values of *G*, then an exception condition is raised: *SQL/MM Spatial exception – geometry intersects an edge*.

v) Otherwise, update *ST_TOPO_GEO.ST_EDGE*, set the *GEOMETRY* value equal to *acurve* where *TOPOLOGY* is equal to atopology and *EDGE_ID* is equal to *anedge*.

3) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.7 ST_RemIsoEdge Procedure

Purpose

Delete a row in the <topology-name>.*ST_EDGE* view corresponding to an isolated edge.

Definition

```
CREATE PROCEDURE ST_RemIsoEdge
   (atopology CHARACTER VARYING(ST_MaxTopologyName),
    anedge INTEGER)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The procedure *ST_RemIsoEdge(CHARACTER VARYING, INTEGER)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *anedge*.
2) For the procedure *ST_RemIsoEdge(CHARACTER VARYING, INTEGER)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *anedge* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

      Case:

      i) If *ST_TOPO_GEO.ST_EDGE.LEFT_FACE* is not equal to *ST_TOPO_GEO.ST_EDGE.RIGHT_FACE*, then an exception condition is raised: *SQL/MM Spatial exception – not isolated edge*.
      ii) Let *N1* be the *ST_TOPO_GEO.ST_EDGE.START_NODE* of the edge whose *EDGE_ID* is equal to *anedge*. If any *ST_TOPO_GEO.ST_EDGE.START_NODE* or *ST_TOPO_GEO.ST_EDGE.END_NODE* value is equal to *N1*, then an exception condition is raised: *SQL/MM Spatial exception – not isolated edge*.
      iii) Let *N2* be the *ST_TOPO_GEO.ST_EDGE.END_NODE* of the edge whose *EDGE_ID* is equal to *anedge*. If any *ST_TOPO_GEO.ST_EDGE.START_NODE* or *ST_TOPO_GEO.ST_EDGE.END_NODE* value is equal to *N2*, then an exception condition is raised: *SQL/MM Spatial exception – not isolated edge*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

iv) Otherwise, delete from *ST_TOPO_GEO.ST_EDGE* where *TOPOLOGY* is equal to *atopology* and *EDGE_ID* is equal to *anedge*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.8 ST_NewEdgesSplit Function

Purpose

Split an edge by creating a new node along an existing edge, deleting the original edge and replacing it with two new edges.

Definition

```
CREATE FUNCTION ST_NewEdgesSplit
    (atopology CHARACTER VARYING(ST_MaxTopologyName),
     anedge INTEGER,
     apoint ST_Point)
    RETURNS INTEGER
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    CALLED ON NULL INPUT
    STATIC DISPATCH
    BEGIN
       --
       -- See Description
       --
    END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The function *ST_NewEdgesSplit(CHARACTER VARYING, INTEGER, ST_Point)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *anedge*,
   c) an *ST_Point* value *apoint*.
2) For the function *ST_NewEdgesSplit(CHARACTER VARYING, INTEGER, ST_Point)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *anedge* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *apoint* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   f) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

      Case:

      i) If there is no row in *ST_TOPO_GEO.ST_EDGE* with an *EDGE_ID* value equal to *anedge*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent edge*.
      ii) Let *G* be equal to the *ST_Curve* value of *ST_TOPO_GEO.ST_EDGE.GEOMETRY* where *EDGE_ID* is equal to *anedge*. If *apoint.ST_Within(G)* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – point not on edge*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

iii) If any *ST_TOPO_GEO.ST_NODE.GEOMETRY* value is equal to *apoint*, then an exception condition is raised: *SQL/MM Spatial exception – coincident node*.

iv) Otherwise:

1) generate a unique node id INTEGER value *newnode*,

2) insert into *ST_TOPO_GEO.ST_NODE* values (*atopology*, *newnode*, *apoint*, NULL),

3) select from *ST_TOPO_GEO.ST_EDGE START_NODE*, *END_NODE*, *NEXT_LEFT_EDGE*, *NEXT_RIGHT_EDGE*, *LEFT_FACE*, *RIGHT_FACE*, and *GEOMETRY* into *oldstart*, *oldend*, *oldnextleft*, *oldnextright*, *oldleft*, *oldright*, and *oldgeom* where *EDGE_ID* is equal to *anedge*,

4) delete from *ST_TOPO_GEO.ST_EDGE* where *TOPOLOGY* is equal to *atopology* and *EDGE_ID* is equal to *anedge*,

5) generate two unique edge id INTEGER values *newedge1* and *newedge2,*

6) create two new *ST_Curve* values, *newgeom1* and *newgeom2* from *oldgeom*, breaking *oldgeom* at the location defined by *apoint*,

7) insert into *ST_TOPO_GEO.ST_EDGE* values (*atopology*, *newedge1*, *oldstart*, *newnode*, *newedge2*, *oldnextright*, *oldleft*, *oldright*, *newgeom1*),

8) insert into *ST_TOPO_GEO.ST_EDGE* values (*atopology*, *newedge2*, *newnode*, *oldend*, *oldnextleft*, -(*newedge1*), *oldleft*, *oldright*, *newgeom2*),

9) make any necessary updates to the next left and right face edge IDs for any edges incident on the start and end nodes of the edge being split.

10) return *newnode*,

v) Both new edges have the same direction as the edge being split.

vi) To determine the two new edge ID values, select *EDGE_ID* from *ST_TOPO_GEO.ST_EDGE* where *START_NODE* or *END_NODE* is equal to *newnode*.

3) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.9 ST_ModEdgeSplit Function

Purpose

Split an edge by creating a new node along an existing edge, modifying the original edge and adding a new edge.

Definition

```
CREATE FUNCTION ST_ModEdgeSplit
    (atopology CHARACTER VARYING(ST_MaxTopologyName),
     anedge INTEGER,
     apoint ST_Point)
    RETURNS INTEGER
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    CALLED ON NULL INPUT
    STATIC DISPATCH
    BEGIN
       --
       -- See Description
       --
    END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The function *ST_ModEdgeSplit(CHARACTER VARYING, INTEGER, ST_Point)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *anedge*,
   c) an *ST_Point* value *apoint*.
2) For the function *ST_ModEdgeSplit(CHARACTER VARYING, INTEGER, ST_Point)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *anedge* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *apoint* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   f) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

      Case:

      i) If there is no row in *ST_TOPO_GEO.ST_EDGE* with an *EDGE_ID* value equal to *anedge*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent edge*.
      ii) Let *G* be equal to the *ST_Curve* value of *ST_TOPO_GEO.ST_EDGE.GEOMETRY* where *EDGE_ID* is equal to *anedge*. If *apoint.ST_Within(G)* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – point not on edge*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

iii) If any *ST_TOPO_GEO.ST_NODE.GEOMETRY* value is equal to *apoint*, then an exception condition is raised: *SQL/MM Spatial exception – coincident node*.

iv) Otherwise:

1) generate a unique node id INTEGER value *newnode*,

2) insert into *ST_TOPO_GEO.ST_NODE* values (*atopology*, *newnode*, *apoint*, NULL),

3) select from *ST_TOPO_GEO.ST_EDGE END_NODE*, *NEXT_LEFT_EDGE*, *LEFT_FACE*, *RIGHT_FACE*, and *GEOMETRY* into *oldend*, *oldnextleft*, *oldleft*, *oldright*, and *oldgeom* where *EDGE_ID* is equal to *anedge*,

4) generate a unique edge id INTEGER value *newedge*,

5) create two new *ST_Curve* values, *newgeom1* and *newgeom2* from *oldgeom*, breaking *oldgeom* at the location defined by *apoint*,

6) update *ST_TOPO_GEO.ST_EDGE* where *TOPOLOGY* is equal to atopology and *EDGE_ID* is equal to *anedge*:

A) set the *END_NODE* value equal to *newnode*,

B) set the *NEXT_LEFT_EDGE* value equal to *newedge*,

C) set the *GEOMETRY* value equal to *newgeom1*,

7) insert into *ST_TOPO_GEO.ST_EDGE* values (*atopology*, *newedge*, *newnode*, *oldend*, *oldnextleft*, -(*anedge*), *oldleft*, *oldright*, *newgeom2*),

8) make any necessary updates to the next left and right face edge IDs for any edges incident on the start and end nodes of the edge being split.

9) return *newnode*,

v) The new and modified edges have the same direction as the edge being split.

vi) To determine the new edge ID value, select *EDGE_ID* from *ST_TOPO_GEO.ST_EDGE* where *START_NODE* is equal to *newnode*.

3) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.10 ST_NewEdgeHeal Function

Purpose

Heal two edges by deleting the node connecting them, deleting both edges, and replacing them with a new edge whose direction is the same as the first edge provided.

Definition

```
CREATE FUNCTION ST_NewEdgeHeal
   (atopology CHARACTER VARYING(ST_MaxTopologyName),
    anedge INTEGER,
    anotheredge INTEGER)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The function *ST_NewEdgeHeal(CHARACTER VARYING, INTEGER, INTEGER)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *anedge*,
   c) an INTEGER value *anotheredge*.
2) For the function *ST_NewEdgeHeal(CHARACTER VARYING, INTEGER, INTEGER)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *anedge* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *anotheredge* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   f) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

      Case:

      i) If there is no row in *ST_TOPO_GEO.ST_EDGE* with an *EDGE_ID* value equal to *anedge*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent edge*.
      ii) If there is no row in *ST_TOPO_GEO.ST_EDGE* with an *EDGE_ID* value equal to *anotheredge*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent edge*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

iii) Check if edges are connected:

1) let *COMMONNODE* be an INTEGER value,

2) let *CASE* be an INTEGER value,

3) let *S1* and *E1* be INTEGER values equal to the *ST_TOPO_GEO.ST_EDGE.START_NODE* and *ST_TOPO_GEO.ST_EDGE.END_NODE* values, respectively, where *EDGE_ID* is equal to *anedge*,

4) let *S2* and *E2* be INTEGER values equal to the *ST_TOPO_GEO.ST_EDGE.START_NODE* and *ST_TOPO_GEO.ST_EDGE.END_NODE* values, respectively, where *EDGE_ID* is equal to *anotheredge*,

5) case:

A) if *E1* is equal to *S2*, then set *CASE* equal to 1 (one) and *COMMONNODE* equal to *E1*,

B) if *E1* is equal to *E2*, then set *CASE* equal to 2 and *COMMONNODE* equal to *E1*,

C) if *S1* is equal to *S2*, then set *CASE* equal to 3 and *COMMONNODE* equal to *S1*,

D) if *S1* is equal to *E2*, then set *CASE* equal to 4 and *COMMONNODE* equal to *S1*,

E) otherwise, an exception condition is raised: *SQL/MM Spatial exception – non-connected edges*,

iv) If *COMMONNODE* is equal to the *ST_TOPO_GEO.ST_EDGE.START_NODE* or *ST_TOPO_GEO.ST_EDGE.END_NODE* value for any edge other than the edge whose *EDGE_ID* is equal to either *anedge* or *anotheredge*, then an exception condition is raised: *SQL/MM Spatial exception – other edges connected*,

v) Otherwise:

1) delete from *ST_TOPO_GEO.ST_NODE* where *TOPOLOGY* is equal to *atopology* and *NODE_ID* is equal to *COMMONNODE*,

2) select from *ST_TOPO_GEO.ST_EDGE START_NODE*, *END_NODE*, *NEXT_LEFT_EDGE*, *NEXT_RIGHT_EDGE*, *LEFT_FACE RIGHT_FACE*, and *GEOMETRY* into *oldstart1*, *oldend1*, *oldnextleft1*, *oldnextright1*, *oldleft*, *oldright*, and *oldgeom1* where *EDGE_ID* is equal to *anedge*,

3) select from *ST_TOPO_GEO.ST_EDGE START_NODE*, *END_NODE*, *NEXT_LEFT_EDGE*, *NEXT_RIGHT_EDGE*, and *GEOMETRY* into *oldstart2*, *oldend2*, *oldnextleft2*, *oldnextright2*, and *oldgeom2* where *EDGE_ID* is equal to *anotheredge*,

4) delete from *ST_TOPO_GEO.ST_EDGE* where *TOPOLOGY* is equal to *atopology* and *EDGE_ID* is equal to *anedge*,

5) delete from *ST_TOPO_GEO.ST_EDGE* where *TOPOLOGY* is equal to *atopology* and *EDGE_ID* is equal to *anotheredge*,

6) generate a unique edge id INTEGER value *newedge*,

7) case:

A) if *CASE* is equal to 1 (one) then:

I) create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by connecting the end of *oldgeom1* to the start of *oldgeom2*,

II) insert into *ST_TOPO_GEO.ST_EDGE* values (*atopology*, *newedge*, *oldstart1*, *oldend2*, *oldnextleft2*, *oldnextright1*, *oldleft, oldright*, *newgeom*),

B) if *CASE* is equal to 2 then:

I) create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by reversing the direction of *oldgeom2* and connecting it to the end of *oldgeom1*,

II) insert into *ST_TOPO_GEO.ST_EDGE* values (*atopology*, *newedge*, *oldstart1*, *oldstart2*, *oldnextright2*, *oldnextright1*, *oldleft*, *oldright*, *newgeom*),

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

C) if *CASE* is equal to 3 then:

I) create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by reversing the direction of *oldgeom2* and connecting *oldgeom1* to the new end,

II) insert into *ST_TOPO_GEO.ST_EDGE* values (*atopology*, *newedge*, *oldend2*, *oldend1*, *oldnextleft1*, *oldnextleft2*, *oldleft*, *oldright*, *newgeom*),

D) if *CASE* is equal to 4 then:

I) create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by connecting *oldgeom1* to the end of *oldgeom2*,

II) insert into *ST_TOPO_GEO.ST_EDGE* values (*atopology*, *newedge*, *oldstart2*, *oldend1*, *oldnextleft1*, *oldnextright2*, *oldleft*, *oldright*, *newgeom*),

8) make any necessary updates to the next left and right face edge IDs for any edges incident on the start and end nodes of the new edge,

9) return *newedge*.

vi) The direction of the new edge shall be the same as the direction of the first edge supplied.

3) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.11 ST_ModEdgeHeal Procedure

Purpose

Heal two edges by deleting the node connecting them, modifying the first edge provided, and deleting the second edge.

Definition

```
CREATE PROCEDURE ST_ModEdgeHeal
   (atopology CHARACTER VARYING(ST_MaxTopologyName),
    anedge INTEGER,
    anotheredge INTEGER)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The procedure *ST_ModEdgeHeal(CHARACTER VARYING, INTEGER, INTEGER)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *anedge*,
   c) an INTEGER value *anotheredge*.
2) For the procedure *ST_ModEdgeHeal(CHARACTER VARYING, INTEGER, INTEGER)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *anedge* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *anotheredge* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   f) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

      Case:

      i) If there is no row in *ST_TOPO_GEO.ST_EDGE* with an *EDGE_ID* value equal to *anedge*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent edge*.
      ii) If there is no row in *ST_TOPO_GEO.ST_EDGE* with an *EDGE_ID* value equal to *anotheredge*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent edge*.
      iii) Check if edges are connected:

         1) let *COMMONNODE* be an INTEGER value,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) let *CASE* be an INTEGER value,

3) let *S1* and *E1* be INTEGER values equal to the *ST_TOPO_GEO.ST_EDGE.START_NODE* and *ST_TOPO_GEO.ST_EDGE.END_NODE* values, respectively, where *EDGE_ID* is equal to *anedge*,

4) let *S2* and *E2* be INTEGER values equal to the *ST_TOPO_GEO.ST_EDGE.START_NODE* and *ST_TOPO_GEO.ST_EDGE.END_NODE* values, respectively, where *EDGE_ID* is equal to *anotheredge*,

5) case:

A) if *E1* is equal to *S2*, then set *CASE* equal to 1 (one) and *COMMONNODE* equal to *E1*,

B) if *E1* is equal to *E2*, then set *CASE* equal to 2 and *COMMONNODE* equal to *E1*,

C) if *S1* is equal to *S2*, then set *CASE* equal to 3 and *COMMONNODE* equal to *S1*,

D) if *S1* is equal to *E2*, then set *CASE* equal to 4 and *COMMONNODE* equal to *S1*,

E) otherwise, an exception condition is raised: *SQL/MM Spatial exception – non-connected edges*,

iv) If *COMMONNODE* is equal to the *ST_TOPO_GEO.ST_EDGE.START_NODE* or *ST_TOPO_GEO.ST_EDGE.END_NODE* value for any edge other than the edge whose *EDGE_ID* is equal to either *anedge* or *anotheredge*, then an exception condition is raised: *SQL/MM Spatial exception – other edges connected*,

v) Otherwise:

1) delete from *ST_TOPO_GEO.ST_NODE* where *TOPOLOGY* is equal to *atopology* and *NODE_ID* is equal to *COMMONNODE*,

2) select from *ST_TOPO_GEO.ST_EDGE GEOMETRY* into *oldgeom1* where *EDGE_ID* is equal to *anedge*,

3) select from *ST_TOPO_GEO.ST_EDGE START_NODE*, *END_NODE*, *NEXT_LEFT_EDGE*, *NEXT_RIGHT_EDGE*, and *GEOMETRY* into *oldstart2*, *oldend2*, *oldnextleft2*, *oldnextright2*, and *oldgeom2* where *EDGE_ID* is equal to *anotheredge*,

4) delete from *ST_TOPO_GEO.ST_EDGE* where *TOPOLOGY* is equal to *atopology* and *EDGE_ID* is equal to *anotheredge*,

5) case:

A) if *CASE* is equal to 1 (one) then:

I) create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by connecting the end of *oldgeom1* to the start of *oldgeom2*,

II) update *ST_TOPO_GEO.ST_EDGE* where *TOPOLOGY* is equal to *atopology* and *EDGE_ID* is equal to *anedge*:

1) set the *END_NODE* value equal to *oldend2*,

2) set the *NEXT_LEFT_EDGE* value equal to *oldnextleft2*,

3) set the *GEOMETRY* value equal to *newgeom*,

B) if *CASE* is equal to 2 then:

I) create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by reversing the direction of *oldgeom2* and connecting it to the end of *oldgeom1*,

II) update *ST_TOPO_GEO.ST_EDGE* where *TOPOLOGY* is equal to *atopology* and *EDGE_ID* is equal to *anedge*:

1) set the *END_NODE* value equal to *oldstart2*,

2) set the *NEXT_LEFT_EDGE* value equal to *oldnextright2*,

3) set the *GEOMETRY* value equal to *newgeom*,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

C) if *CASE* is equal to 3 then:

I) create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by reversing the direction of *oldgeom2* and connecting *oldgeom1* to the new end,

II) update *ST_TOPO_GEO.ST_EDGE* where *TOPOLOGY* is equal to *atopology* and *EDGE_ID* is equal to *anedge*:

1) set the *START_NODE* value equal to *oldend2*,

2) set the *NEXT_RIGHT_EDGE* value equal to *oldnextleft2*,

3) set the *GEOMETRY* value equal to *newgeom*,

D) if *CASE* is equal to 4 then:

I) create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by connecting *oldgeom1* to the end of *oldgeom2*,

II) update *ST_TOPO_GEO.ST_EDGE* where *TOPOLOGY* is equal to *atopology* and *EDGE_ID* is equal to *anedge*:

1) set the *START_NODE* value equal to *oldstart2*,

2) set the *NEXT_RIGHT_EDGE* value equal to *oldnextright2*,

3) set the *GEOMETRY* value equal to *newgeom*,

6) make any necessary updates to the next left and right face edge IDs for any edges incident on the start and end nodes of the new edge.

3) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.12 ST_AddEdgeNewFaces Function

Purpose

Add a new edge and, if in doing so it splits a face, delete the original face and replace it with two new faces.

Definition

```
CREATE FUNCTION ST_AddEdgeNewFaces
   (atopology CHARACTER VARYING(ST_MaxTopologyName),
    anode INTEGER,
    anothernode INTEGER,
    acurve ST_Curve)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The function *ST_AddEdgeNewFaces(CHARACTER VARYING, INTEGER, INTEGER, ST_Curve)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *anode*,
   c) an INTEGER value *anothernode*,
   d) an *ST_Curve* value *acurve*.
2) For the function *ST_AddEdgeNewFaces(CHARACTER VARYING, INTEGER, INTEGER, ST_Curve)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology.ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *anode* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *anothernode* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   f) If *acurve* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   g) If the value returned by *acurve.ST_IsSimple()* is not equal to 1 (one), then an exception condition is raised: *SQL/MM Spatial exception – curve not simple*.
   h) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Case:

i) If there is no row in *ST_TOPO_GEO.ST_NODE* with a *NODE_ID* value equal to *anode*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent node*.

ii) If there is no row in *ST_TOPO_GEO.ST_NODE* with a *NODE_ID* value equal to *anothernode*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent node*.

iii) Let *P1* be the *ST_Point* value equal to *ST_TOPO_GEO.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *anode*. If *P1* is not equal to *acurve.ST_StartPoint()*, then an exception condition is raised: *SQL/MM Spatial exception – start node not geometry start point*.

iv) Let *P2* be the *ST_Point* value equal to *ST_TOPO_GEO.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *anothernode*. If *P2* is not equal to *acurve.ST_EndPoint()*, then an exception condition is raised: *SQL/MM Spatial exception – end node not geometry end point*.

v) Let *N1* be a *NODE_ID* value in *ST_TOPO_GEO.ST_NODE* such that *N1* does not exist as either an *ST_TOPO_GEO.ST_EDGE.START_NODE* or *ST_TOPO_GEO.ST_EDGE.END_NODE* value. Let *G1* be the *ST_TOPO_GEO.ST_NODE.GEOMETRY* value where *NODE_ID* is equal to *N1*. If *acurve.ST_Crosses(G1)* is not equal to 0 (zero) for all values of *N1*, then an exception condition is raised: *SQL/MM Spatial exception – geometry crosses a node*.

vi) Let *G* be an *ST_TOPO_GEO.ST_EDGE.GEOMETRY* value. If *acurve.ST_Crosses(G)* is not equal to 0 (zero) for all values of *G*, then an exception condition is raised: *SQL/MM Spatial exception – geometry crosses an edge*.

vii) If there exists a row in *ST_TOPO_GEO.ST_EDGE* with a *START_NODE* value equal to *anode* and an *END_NODE* value equal to *anothernode* and a *GEOMETRY* value equal to *acurve*, then an exception condition is raised: *SQL/MM Spatial exception – coincident edge*.

viii) If there exists a row in *ST_TOPO_GEO.ST_EDGE* with a *START_NODE* value equal to *anothernode* and an *END_NODE* value equal to *anode* and a *GEOMETRY* value equal to *acurve*, then an exception condition is raised: *SQL/MM Spatial exception – coincident edge*.

ix) Otherwise:

1) generate a unique edge id INTEGER value *anedgeid*,

2) insert into *ST_TOPO_GEO.ST_EDGE* a new row with:

A) a *TOPOLOGY* value equal to *atopology*,

B) an *EDGE_ID* value equal to *anedgeid*,

C) a *START_NODE* value equal to *anode*,

D) an *END_NODE* value equal to *anothernode*,

E) *NEXT_LEFT_EDGE*, *NEXT_RIGHT_EDGE*, *LEFT_FACE*, and *RIGHT_FACE* values as appropriate, and

F) a *GEOMETRY* value equal to *acurve*,

3) if the new edge splits a face, then:

A) let *F* be an INTEGER value equal to the *FACE_ID* of the face being split,

B) delete from *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology* and *FACE_ID* is equal to *F*,

C) generate two unique face id INTEGER values *newface1* and *newface2*,

D) update the appropriate next left and right face edge IDs and left and right face IDs for edges bounding the face being split,

E) let *MBR1* be the *ST_Polygon* value equal to the value of *ST_GetFaceGeometry(atopology,newface1).ST_Envelope()*,

F) let *MBR2* be the *ST_Polygon* value equal to the value of *ST_GetFaceGeometry(atopology,newface2).ST_Envelope()*,

G) insert into *ST_TOPO_GEO.ST_FACE* values (*atopology*, *newface1*, *MBR1*),

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

H) insert into *ST_TOPO_GEO.ST_FACE* values (*atopology*, *newface2*, *MBR2*),

4) return *anedgeid*.

x) To determine the two new face ID values, select *LEFT_FACE* and *RIGHT_FACE* from *ST_TOPO_GEO.ST_EDGE* where *EDGE_ID* is equal to *anedgeid*.

3) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.13 ST_AddEdgeModFace Function

Purpose

Add a new edge and, if in doing so it splits a face, modify the original face and add a new face.

Definition

```
CREATE FUNCTION ST_AddEdgeModFace
   (atopology CHARACTER VARYING(ST_MaxTopologyName),
    anode INTEGER,
    anothernode INTEGER,
    acurve ST_Curve)
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT
  STATIC DISPATCH
  BEGIN
     --
     -- See Description
     --
  END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The function *ST_AddEdgeModFace(CHARACTER VARYING, INTEGER, INTEGER, ST_Curve)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *anode*,
   c) an INTEGER value *anothernode*,
   d) an *ST_Curve* value *acurve*.
2) For the function *ST_AddEdgeModFace(CHARACTER VARYING, INTEGER, INTEGER, ST_Curve)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology.ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *anode* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *anothernode* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   f) If *acurve* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   g) If the value returned by *acurve.ST_IsSimple()* is not equal to 1 (one), then an exception condition is raised: *SQL/MM Spatial exception – curve not simple*.
   h) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

Case:

i) If there is no row in *ST_TOPO_GEO.ST_NODE* with a *NODE_ID* value equal to *anode*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent node*.

ii) If there is no row in *ST_TOPO_GEO.ST_NODE* with a *NODE_ID* value equal to *anothernode*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent node*.

iii) Let *P1* be the *ST_Point* value equal to *ST_TOPO_GEO.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *anode*. If *P1* is not equal to *acurve.ST_StartPoint()*, then an exception condition is raised: *SQL/MM Spatial exception – start node not geometry start point*.

iv) Let *P2* be the *ST_Point* value equal to *ST_TOPO_GEO.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *anothernode*. If *P2* is not equal to *acurve.ST_EndPoint()*, then an exception condition is raised: *SQL/MM Spatial exception – end node not geometry end point*.

v) Let *N1* be a *NODE_ID* value in *ST_TOPO_GEO.ST_NODE* such that *N1* does not exist as either an *ST_TOPO_GEO.ST_EDGE.START_NODE* or *ST_TOPO_GEO.ST_EDGE.END_NODE* value. Let *G1* be the *ST_TOPO_GEO.ST_NODE.GEOMETRY* value where *NODE_ID* is equal to *N1*. If *acurve.ST_Crosses(G1)* is not equal to 0 (zero) for all values of *N1*, then an exception condition is raised: *SQL/MM Spatial exception – geometry crosses a node*.

vi) Let *G* be an *ST_TOPO_GEO.ST_EDGE.GEOMETRY* value. If *acurve.ST_Crosses(G)* is not equal to 0 (zero) for all values of *G*, then an exception condition is raised: *SQL/MM Spatial exception – geometry crosses an edge*.

vii) If there exists a row in *ST_TOPO_GEO.ST_EDGE* with a *START_NODE* value equal to *anode* and an *END_NODE* value equal to *anothernode* and a *GEOMETRY* value equal to *acurve*, then an exception condition is raised: *SQL/MM Spatial exception – coincident edge*.

viii) If there exists a row in *ST_TOPO_GEO.ST_EDGE* with a *START_NODE* value equal to *anothernode* and an *END_NODE* value equal to *anode* and a *GEOMETRY* value equal to *acurve*, then an exception condition is raised: *SQL/MM Spatial exception – coincident edge*.

ix) Otherwise:

1) generate a unique edge id INTEGER value *anedgeid*,

2) insert into *ST_TOPO_GEO.ST_EDGE* a new row with:

A) a *TOPOLOGY* value equal to *atopology*,

B) an *EDGE_ID* value equal to *anedgeid*,

C) a *START_NODE* value equal to *anode*,

D) an *END_NODE* value equal to *anothernode*,

E) *NEXT_LEFT_EDGE*, *NEXT_RIGHT_EDGE*, *LEFT_FACE*, and *RIGHT_FACE* values as appropriate, and

F) a *GEOMETRY* value equal to *acurve*,

3) if the new edge splits a face, then:

A) let *F* be an INTEGER value equal to the *FACE_ID* of the face being split,

B) update *ST_TOPO_GEO.ST_FACE* set MBR equal to the value of *ST_GetFaceGeometry(atopology,F1).ST_Envelope()* where *TOPOLOGY* is equal to *atopology* and *FACE_ID* is equal to *F*,

C) generate a unique face id INTEGER value *newface*,

D) update the appropriate next left and right face edge IDs and left and right face IDs for edges bounding the face being split,

E) let *MBR* be the ST_Polygon value equal to the value of *ST_GetFaceGeometry(atopology,newface).ST_Envelope()*,

F) insert into *ST_TOPO_GEO.ST_FACE* values (*atopology*, *newface*, *MBR*),

4) return *anedgeid*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

x) To determine the ID values for the new face and the modified face, select *LEFT_FACE* and *RIGHT_FACE* from *ST_TOPO_GEO.ST_EDGE* where *EDGE_ID* is equal to *anedgeid*.

3) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.14 ST_RemEdgeNewFace Function

Purpose

Remove an edge and, if the removed edge separated two faces, delete the original faces and replace them with one new face.

Definition

```
CREATE FUNCTION ST_RemEdgeNewFace
   (atopology CHARACTER VARYING(ST_MaxTopologyName),
    anedge INTEGER)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The function *ST_RemEdgeNewFace(CHARACTER VARYING, INTEGER)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *anedge*.
2) For the function *ST_RemEdgeNewFace(CHARACTER VARYING, INTEGER)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *anedge* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

      Case:

      i) If there is no row in *ST_TOPO_GEO.ST_EDGE* with an *EDGE_ID* value equal to *anedge*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent edge*.
      ii) Otherwise:
         1) delete from *ST_TOPO_GEO.ST_EDGE* where *TOPOLOGY* is equal to *atopology* and *EDGE_ID* is equal to *anedge*,
         2) if the edge removal results in the healing of two faces, then:
            A) let *F1* and *F2* be the INTEGER values equal to the *FACE_ID*s of the faces to be healed,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

B) delete from *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology* and *FACE_ID* is equal to *F1*,

C) delete from *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology* and *FACE_ID* is equal to *F2*,

D) generate a unique face id INTEGER value *newface*,

E) update the appropriate next left and right face edge IDs and left and right face IDs for edges bounding the faces being healed,

F) let *MBR* be the *ST_Polygon* value equal to the value of *ST_GetFaceGeometry(atopology,newface).ST_Envelope()*,

G) insert into *ST_TOPO_GEO.ST_FACE* values (*atoploogy*, *newface*, *MBR*),

H) return *newface*.

iii) The nodes in ST_TOPO_GEO.ST_NODE having a NODE_ID value equal to ST_TOPO_GEO.ST_EDGE.START_NODE and ST_TOPO_GEO.ST_EDGE.END_NODE where EDGE_ID is equal to anedge are not deleted.

3) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.15 ST_RemEdgeModFace Procedure

Purpose

Remove an edge and, if the removed edge separated two faces, heal the two faces by modifying one of the faces and delete the other.

Definition

```
CREATE PROCEDURE ST_RemEdgeModFace
    (atopology CHARACTER VARYING(ST_MaxTopologyName),
     anedge INTEGER)
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    CALLED ON NULL INPUT
    BEGIN
       --
       -- See Description
       --
    END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The procedure *ST_RemEdgeModFace(CHARACTER VARYING, INTEGER)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *anedge*.
2) For the procedure *ST_RemEdgeModFace(CHARACTER VARYING, INTEGER)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *anedge* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

      Case:

      i) If there is no row in *ST_TOPO_GEO.ST_EDGE* with an *EDGE_ID* value equal to *anedge*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent edge*.
      ii) Otherwise:
         1) delete from *ST_TOPO_GEO.ST_EDGE* where *TOPOLOGY* is equal to *atopology* and *EDGE_ID* is equal to *anedge*,
         2) if the edge removal results in the healing of two faces, then:
            A) let *F1* and *F2* be the INTEGER values equal to the *FACE_ID*s of the two faces to be healed,
            B) delete from *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology* and *FACE_ID* is equal to *F2*,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

C) update the appropriate next left and right face edge IDs and left and right face IDs for edges bounding the faces being healed,

D) update *ST_TOPO_GEO.ST_FACE* set MBR equal to the value of *ST_GetFaceGeometry(atopology,F1).ST_Envelope()* where *TOPOLOGY* is equal to *atopology* and *FACE_ID* is equal to *F1*.

iii) The nodes in *ST_TOPO_GEO.ST_NODE* having a *NODE_ID* value equal to *ST_TOPO_GEO.ST_EDGE.START_NODE* and *ST_TOPO_GEO.ST_EDGE.END_NODE* where *TOPOLOGY* is equal to *atopology* and *EDGE_ID* is equal to *anedge* are not deleted.

iv) The choice of which face to modify and which to delete is implementation-dependent.

3) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.16 ST_GetFaceGeometry Function

Purpose

Return the exact geometry of a face.

Definition

```
CREATE FUNCTION ST_GetFaceGeometry
   (atopology CHARACTER VARYING(ST_MaxTopologyName),
    aface INTEGER)
   RETURNS ST_Surface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The function *ST_GetFaceGeometry(CHARACTER VARYING, INTEGER)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an INTEGER value *aface*.
2) For the function *ST_GetFaceGeometry(CHARACTER VARYING, INTEGER)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *aface* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

      Case:

      i) If *aface* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – universal face has no geometry*.
      ii) If there is no row in *ST_TOPO_GEO.ST_FACE* with a *FACE_ID* value equal to *aface*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent face*.
      iii) Otherwise:
         1) let *T* be the table returned by *ST_GetFaceEdges(atopology, aface)*,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) for each row in *T*, let *G* be the *ST_Curve* value equal to *ST_TOPO_GEO.ST_EDGE.GEOMETRY* where *EDGE_ID* is equal to the value of *edge* from that row. The *ST_Curve* value shall have the same spatial reference system identifier as the *ST_TOPO_GEO.ST_EDGE.GEOMETRY* value.

3) let *G'* be the *ST_Curve* value derived from *G*, corrected for the direction of *edge*; for each value of *G*:

   Case:

   A) if *edge* is greater than 0 (zero), then *G'* is equal to *G*,

   B) otherwise, construct a value for *G'* by reversing the direction of *G*.

4) case:

   A) if all *G' ST_Curve* values have the same spatial reference system identifier and a valid *ST_Surface* geometry value *geometry* can be constructed from the values of *G'*, connecting them in the order specified by the *sequence* values in *T*, then

      I) create the valid *geometry* value accordingly, with the common spatial reference system identifier, and

      II) return *geometry*.

   B) otherwise, return an empty set of type *ST_Surface*.

5) return *geometry*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.17 ST_InitTopoGeo Procedure

Purpose

Create schema and views for a Topology-geometry.

Definition

```
CREATE PROCEDURE ST_InitTopoGeo
   (atopology CHARACTER VARYING(ST_MaxTopologyName))
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The procedure *ST_InitTopoGeo(CHARACTER VARYING)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*.
2) For the procedure *ST_InitTopoGeo(CHARACTER VARYING)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.SCHEMATA WHERE SCHEMA_NAME = '*atopology*' returns a value equal to 1 (one), then an exception condition is raised: *SQL/MM Spatial exception – schema already exists*.
   c) Otherwise:
      i) create a schema with a name equal to *atopology*,
      ii) for schema *atopology*, create ST_NODE, ST_EDGE, and ST_FACE views in accordance with Subclause 10.1.2, "ST_NODE view", Subclause 10.1.3, "ST_EDGE view", and Subclause 10.1.4, "ST_FACE view",
      iii) insert into *ST_TOPO_GEO.ST_FACE* values (*atopology*, 0, NULL).

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.18 ST_CreateTopoGeo Procedure

Purpose

Create a topologically consistent Topology-geometry from a collection of geometry values.

Definition

```
CREATE PROCEDURE ST_CreateTopoGeo
   (atopology CHARACTER VARYING(ST_MaxTopologyName),
    ageomcollection ST_GeomCollection)
   LANGUAGE SQL
   NOT DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The procedure *ST_CreateTopoGeo(CHARACTER VARYING, ST_GeomCollection)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*,
   b) an *ST_GeomCollection* value *ageomcollection*.
2) For the procedure *ST_CreateTopoGeo(CHARACTER VARYING, ST_GeomCollection)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology.ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) If *ageomcollection* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *aface* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – invalid universal face*.
   f) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.SCHEMATA WHERE SCHEMA_NAME = '*atopology*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent schema*.
   g) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA = '*atopology*' AND TABLE_NAME = '*ST_NODE*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent view*.
   h) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA = '*atopology*' AND TABLE_NAME = '*ST_EDGE*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent view*.
   i) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA = '*atopology*' AND TABLE_NAME = '*ST_FACE*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent view*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

j) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

Case:

i) If SELECT COUNT(*) FROM *ST_TOPO_GEO.ST_NODE* returns a value greater than 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-empty view*.

ii) If SELECT COUNT(*) FROM *ST_TOPO_GEO.ST_EDGE* returns a value greater than 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-empty view*.

iii) If SELECT COUNT(*) FROM *ST_TOPO_GEO.ST_FACE* returns a value greater than 1 (one), then an exception condition is raised: *SQL/MM Spatial exception – non-empty view*.

iv) Otherwise:

1) using the geometry values in *ageomcollection*, create the corresponding consistent topology,

2) for each face:

A) let *faceid* be the generated unique face id positive INTEGER value,

B) let *mbr* be the *ST_Polygon* value which represents the minimum bounding rectangle of the face,

C) insert into *ST_TOPO_GEO.ST_FACE* values (*atopology*, *faceid*, *mbr*),

3) for each node:

A) let *nodeid* be the generated unique node id positive INTEGER value,

B) let *geometry* be the *ST_Point* value which specifies the node location,

C) case:

I) if the node is an isolated node, then let *face* be the INTEGER value equal to the face id of the containing face,

II) otherwise, let *face* be the null value,

D) insert into *ST_TOPO_GEO.ST_NODE* values (*atopology*, *nodeid*, *geometry*, *face*),

4) for each edge:

A) let *edgeid* be the generated unique available edge id positive INTEGER value,

B) let *startnode* be the INTEGER value equal to the node id of the node at the start of the edge,

C) let *endnode* be the INTEGER value equal to the node id of the node at the end of the edge,

D) let *nextleft* be the INTEGER value equal to the edge id of the next edge of the face on the left (when looking in the direction from its start node to its end node), moving counterclockwise around the face boundary,

E) let *ENL* be the INTEGER value equal to the node id of the end node of the edge with edge id equal to *nextleft*. If *endnode* is equal to *ENL* then set *nextleft* = - (*nextleft*).

F) let *nextright* be the INTEGER value equal to the edge id of the next edge of the face on the right (when looking in the direction from the start node edge to its end node), moving counterclockwise around the face boundary,

G) let *leftface* be the INTEGER value equal to the face id of the face on the left side of the edge (when looking in the direction from the start node of the edge to its end node),

H) let *rightface* be the INTEGER value equal to the face id of the face on the right side of the edge (when looking in the direction from the start node of the edge to its end node),

I) let *geometry* be the *ST_Curve* value which represents the geometry of the edge, in the same direction as the edge,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

J) insert into *ST_TOPO_GEO.ST_EDGE* values (*atopology*, *edgeid*, *startnode*, *endnode*, *nextleft*, *nextright*, *leftface*, *rightface*, *geometry*).

3) All geometry values in the *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* base tables in rows which have the same *TOPOLOGY* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

10.3.19 ST_ValidateTopoGeo Function

Purpose

Return a table containing possible topological inconsistencies for a Topology-geometry.

Definition

```
CREATE FUNCTION ST_ValidateTopoGeo
   (atopology CHARACTER VARYING(ST_MaxTopologyName))
   RETURNS TABLE
      (error CHARACTER VARYING(30),
       primitive1 INTEGER,
       primitive2 INTEGER)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The function *ST_ValidateTopoGeo(CHARACTER VARYING)* takes the following input parameters:
   a) a CHARACTER VARYING value *atopology*.
2) For the function *ST_ValidateTopoGeo(CHARACTER VARYING)*:

   Case:

   a) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   c) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   d) Otherwise, for rows in *ST_TOPO_GEO.ST_NODE*, *ST_TOPO_GEO.ST_EDGE*, and *ST_TOPO_GEO.ST_FACE* where *TOPOLOGY* is equal to *atopology*:

      Case:

      i) If SELECT COUNT(*) FROM *ST_TOPO_GEO.ST_NODE* returns a value equal to 0 (zero) and SELECT COUNT(*) FROM *ST_TOPO_GEO.ST_EDGE* returns a value equal to 0 (zero) and SELECT COUNT(*) FROM *ST_TOPO_GEO.ST_FACE* returns a value less than or equal to 1 (one), then an exception condition is raised: *SQL/MM Spatial exception – empty topology*.

      ii) Otherwise,

         1) let *T* be a table value consisting of the following three columns:

            A) a column *error* of type CHARACTER VARYING, which identifies the type of inconsistency found in *atopology*,

            B) a column *primitive1* of type INTEGER, which contains the node, edge, or face id of the first offending topology primitive,

            C) a column *primitive2* of type INTEGER, which contains the node, edge, or face id of the second offending topology primitive,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) for each pair of coincident nodes:

   A) let *E* be the CHARACTER VARYING value equal to 'coincident nodes',

   B) let *N1* be the INTEGER value equal to *ST_TOPO_GEO.ST_NODE.NODE_ID* for a node in *atopology*,

   C) let *G1* be the *ST_Point* value equal to *ST_TOPO_GEO.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *N1*,

   D) let *N2* be the INTEGER value equal to *ST_TOPO_GEO.ST_NODE.NODE_ID* for another node in *atopology*,

   E) let *G2* be the *ST_Point* value equal to *ST_TOPO_GEO.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *N2*,

   F) for all values of *N1* and *N2* such that *G1* is equal to *G2*, insert into *T* values (*E*, *N1*, *N2*),

3) for each node crossed by an edge:

   A) let *E* be the CHARACTER VARYING value equal to 'edge crossed node',

   B) let *N1* be the INTEGER value equal to *ST_TOPO_GEO.ST_NODE.NODE_ID* for a node in *atopology*,

   C) let *G1* be the *ST_Point* value equal to *ST_TOPO_GEO.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *N1*,

   D) let *E2* be the INTEGER value equal to *ST_TOPO_GEO.ST_EDGE.EDGE_ID* for an edge in *atopology*,

   E) let *G2* be the *ST_Curve* value equal to *ST_TOPO_GEO.ST_EDGE.GEOMETRY* where *EDGE_ID* is equal to *E2*,

   F) for all values of *N1* and *E2* such that the value returned by *G2.ST_Crosses(G1)* is equal to 1 (one), insert into *T* values (*E*, *N1*, *E2*),

4) for each non-simple edge:

   A) let *E* be the CHARACTER VARYING value equal to 'edge not simple',

   B) let *E1* be the INTEGER value equal to *ST_TOPO_GEO.ST_EDGE.EDGE_ID* for an edge in *atopology*,

   C) let *G1* be the *ST_Curve* value equal to *ST_TOPO_GEO.ST_EDGE.GEOMETRY* where *EDGE_ID* is equal to *E1*,

   D) let *E2* be equal to the null value,

   E) for all values of *E1* such that the value returned by *G1.ST_IsSimple()* is equal to 0 (zero), insert into *T* values (*E*, *E1*, *E2*),

   F) let *ENR* be the INTEGER value equal to the node id of the end node of the edge with edge id equal to *nextright*. If *startnode* is equal to *ENR* then set *nextright* = - (*nextright*).

5) for each edge crossing another edge:

   A) let *E* be the CHARACTER VARYING value equal to 'edge crosses edge',

   B) let *E1* be the INTEGER value equal to *ST_TOPO_GEO.ST_EDGE.EDGE_ID* for an edge in *atopology*,

   C) let *G1* be the *ST_Curve* value equal to *ST_TOPO_GEO.ST_EDGE.GEOMETRY* where *EDGE_ID* is equal to *E1*,

   D) let *E2* be the INTEGER value equal to *ST_TOPO_GEO.ST_EDGE.EDGE_ID* for another edge in *atopology*,

   E) let *G2* be the *ST_Curve* value equal to *ST_TOPO_GEO.ST_EDGE.GEOMETRY* where *EDGE_ID* is equal to *E2*,

   F) for all values of *E1* and *E2* such that the value returned by *G1.ST_Crosses(G2)* is equal to 1 (one), insert into *T* values (*E*, *E1*, *E2*),

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

6) for each edge having a geometry with a start point not equal to the geometry of its start node:
   A) let *E* be the CHARACTER VARYING value equal to 'geometry mismatch',
   B) let *E1* be the INTEGER value equal to *ST_TOPO_GEO.ST_EDGE.EDGE_ID* for an edge in *atopology*,
   C) let *G1* be the *ST_Curve* value equal to *ST_TOPO_GEO.ST_EDGE.GEOMETRY* where *EDGE_ID* is equal to *E1*,
   D) let *N2* be the INTEGER value equal to *ST_TOPO_GEO.ST_EDGE.START_NODE* where *EDGE_ID* is equal to *E1*,
   E) let *G2* be the *ST_Point* value equal to *ST_TOPO_GEO.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *N2*,
   F) for all values of *E1* and *N2* such that the value returned by *G2.ST_Equals(G1.ST_StartPoint())* is not equal to 1 (one), insert into *T* values (*E*, *E1*, *N2*),
7) for each edge having a geometry with an end point not equal to the geometry of its end node:
   A) let *E* be the CHARACTER VARYING value equal to 'geometry mismatch',
   B) let *E1* be the INTEGER value equal to *ST_TOPO_GEO.ST_EDGE.EDGE_ID* for an edge in *atopology*,
   C) let *G1* be the *ST_Curve* value equal to *ST_TOPO_GEO.ST_EDGE.GEOMETRY* where *EDGE_ID* is equal to *E1*,
   D) let *N2* be the INTEGER value equal to *ST_TOPO_GEO.ST_EDGE.END_NODE* where *EDGE_ID* is equal to *E1*,
   E) let *G2* be the *ST_Point* value equal to *ST_TOPO_GEO.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *N2*,
   F) for all values of *E1* and *N2* such that the value returned by *G2.ST_Equals(G1.ST_EndPoint())* is not equal to 1 (one), insert into *T* values (*E*, *E1*, *N2*),
8) for each face with no edges:
   A) let *E* be the CHARACTER VARYING value equal to 'face without edges',
   B) let *F1* be the INTEGER value equal to *ST_TOPO_GEO.ST_FACE.FACE_ID* for a face in *atopology*,
   C) let *F2* be equal to the null value,
   D) for all values of *F1* such that the value returned by SELECT COUNT(*) FROM *ST_TOPO_GEO.ST_EDGE* WHERE *LEFT_FACE* = *F1* is equal to 0 (zero) and SELECT COUNT(*) FROM *ST_TOPO_GEO.ST_EDGE* WHERE *RIGHT_FACE* = *F1* is also equal to 0 (zero), insert into *T* values (*E*, *F1*, *F2*),
9) for each face overlapping another face:
   A) let *E* be the CHARACTER VARYING value equal to 'face overlaps face',
   B) let *F1* be the INTEGER value equal to *ST_TOPO_GEO.ST_FACE.FACE_ID* for a face in *atopology*,
   C) let G1 be the *ST_Surface* value returned by *ST_GetFaceGeometry(atopology,F1)*,
   D) let *F2* be the INTEGER value equal to *ST_TOPO_GEO.ST_FACE.FACE_ID* for another face in *atopology*,
   E) let G2 be the *ST_Surface* value returned by *ST_GetFaceGeometry(atopology,F2)*,
   F) for all values of *F1* and *F2* such that the value returned by *G1.ST_Overlaps(G2)* is equal to 1 (one), insert into *T* values (*E*, *F1*, *F2*),
10) for each face contained within another face:
   A) let *E* be the CHARACTER VARYING value equal to 'face within face',

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

B) let *F1* be the INTEGER value equal to *ST_TOPO_GEO.ST_FACE.FACE_ID* for a face in *atopology*,

C) let G1 be the *ST_Surface* value returned by *ST_GetFaceGeometry(atopology,F1)*,

D) let *F2* be the INTEGER value equal to *ST_TOPO_GEO.ST_FACE.FACE_ID* for another face in *atopology*,

E) let G2 be the *ST_Surface* value returned by *ST_GetFaceGeometry(atopology,F2)*,

F) for all values of *F1* and *F2* such that the value returned by *G1.ST_Within(G2)* is equal to 1 (one), insert into *T* values (*E*, *F1*, *F2*),

11) missing universal face:

A) let *E* be the CHARACTER VARYING value equal to 'no universal face',

B) if SELECT COUNT(*) FROM *ST_TOPO_GEO.ST_FACE* WHERE *FACE_ID* = 0 (zero) is equal to 0 (zero), insert into *T* values (*E*, NULL, NULL).

12) for each non-universal face for which a valid geometry cannot be created:

A) let *E* be the CHARACTER VARYING value equal to 'invalid face geometry',

B) let *F1* be the non-zero INTEGER value equal to *ST_TOPO_GEO.ST_FACE.FACE_ID* for a non-universal face in *atopology*,

C) let *G1* be the *ST_Surface* value returned by *ST_GetFaceGeometry(atopology, F1)*,

D) for all values of *F1* such that the value returned by *G1.ST_IsEmpty()* is equal to 1 (one), insert into *T* values (*E*, *F1*, NULL).

13) if all geometries do not have the same spatial reference system identifier:

A) let *E* be the CHARACTER VARYING value equal to 'mixed SRIDs',

B) let *N1* be the INTEGER value equal to *ST_TOPO_GEO.ST_NODE.NODE_ID* for a node in *atopology*,

C) let *G1* be the *ST_Point* value equal to *ST_TOPO_GEO.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *N1*,

D) let *S1* be the INTEGER value returned by *G1.ST_SRID()*,

E) let *N2* be the INTEGER value equal to *ST_TOPO_GEO.ST_NODE.NODE_ID* for another node in *atopology*,

F) let *G2* be the *ST_Point* value equal to *ST_TOPO_GEO.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *N2*,

G) let *S2* be the INTEGER value returned by *G2.ST_SRID()*,

H) let *E3* be the INTEGER value equal to *ST_TOPO_GEO.ST_EDGE.EDGE_ID* for an edge in *atopology*,

I) let *G3* be the *ST_Curve* value equal to *ST_TOPO_GEO.ST_EDGE.GEOMETRY* where *EDGE_ID* is equal to *E3*,

J) let *S3* be the INTEGER value returned by *G3.ST_SRID()*,

K) let *F4* be the INTEGER value equal to *ST_TOPO_GEO.ST_FACE.FACE_ID* for a face in *atopology*,

L) let *G4* be the *ST_Surface* value equal to *ST_TOPO_GEO.ST_FACE*.MBR where *FACE_ID* is equal to *F4*,

M) let *S4* be the INTEGER value returned by *G4.ST_SRID()*,

N) if any value of *N2*, any value of *E3* or any value of *F4* having a corresponding non-null value for G4 has a corresponding value of *S2*, *S3* or *S4*, respectively, such that *S2* is not equal to *S1* or *S3* is not equal to *S1* or *S4* is not equal to *S1*, insert into *T* values (*E*, NULL, NULL).

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11 Topology-Network

11.1 Topo-Net Network Schema

11.1.1 Introduction

The Topo-Net Network Schema views are defined as being in a schema named <network-name> enabling these views to be accessed in the same way as any other tables in any other schema. SELECT privilege on all of these views may be granted to individual users so that they can be queried. To update a Topo-Net, a user shall be granted SELECT privlilege on the views for the Topo-Net and EXECUTE privilege on the Topo-Net routines provided. Roles can be used to enable update on a selective set of Topo-Nets. These views are updated only by the topology functions provided.

An implementation may define objects that are associated with <network-name> that are not defined in this Clause. An implementation may also add columns to tables that are defined in this Clause.

All of the topological primitives contained in the views owned by the <network-name> schema constitute a single, topologically consistent, topology. Other topologies (e.g., covering a different spatial extent, existing at a different level of abstraction, or associated with a different set of features) can exist in another, independent <network-name> schema.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.1.2 ST_NODE view

Purpose

Contains the node type of topological primitives (ST_Node) contained in the <network-name> Topo-Net.

Definition

```
CREATE VIEW ST_NODE AS
   SELECT NODE_ID, GEOMETRY
      FROM ST_TOPO_NET.ST_NODE
      WHERE NETWORK = '<network-name>'
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.1.3 ST_LINK view

Purpose

Contains the link type of topological primitives (ST_Link) contained in the <network-name> Topo-Net.

Definition

```
CREATE VIEW ST_LINK AS
   SELECT LINK_ID, START_NODE, END_NODE, GEOMETRY
      FROM ST_TOPO_NET.ST_LINK
      WHERE NETWORK = '<network-name>'
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.2 Topo-Net Definition Schema

11.2.1 Introduction

The only purpose of this Topo-Net Definition Schema is to provide a data model to support Topo-Net views and to assist understanding. The base tables of this Topo-Net Definition Schema are defined as being in a schema named *ST_TOPO_NET*. The table definitions are as complete as the definitional power of ISO/IEC 9075 allows. The table definitions are supplemented with assertions where appropriate. Each description comprises three parts:

1) The function of the definition is stated.
2) The SQL definition of the object is presented as a <table definition>.
3) An explanation of the object.

The specification provides only a model of the base tables that are required, and does not imply that an implementation shall provide the functionality in the manner described in this clause.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.2.2 ST_NODE base table

Purpose

Contains the node type of topological primitives (ST_Node) contained in topology-networks.

Definition

```
CREATE TABLE ST_NODE
    (
    NETWORK CHARACTER VARYING(ST_MaxNetworkName),
    NODE_ID INTEGER NOT NULL,
    GEOMETRY ST_Point,

    CONSTRAINT ST_NODE_PRIMARY_KEY PRIMARY KEY(NETWORK, NODE_ID)
    )
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The value of *NETWORK* is the network name which distinguishes which nodes are in the network.
2) Let *N* be the value of *NETWORK* for a given node.
3) The value of *NODE_ID* is the identifier of the node unique among all nodes with a *NETWORK* value equal to *N*.
4) The value of *GEOMETRY* is the spatial location of the node. For logical networks, *GEOMETRY* is the null value.
3) All non-null geometry values in the *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* base tables in rows which have the same *NETWORK* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.2.3 ST_LINK base table

Purpose

Contains the link type of topological primitives (ST_Link) contained in topology-networks.

Definition

```
CREATE TABLE ST_LINK
   (
   NETWORK CHARACTER VARYING(ST_MaxNetworkName),
   LINK_ID INTEGER NOT NULL,
   START_NODE INTEGER NOT NULL,
   END_NODE INTEGER NOT NULL,
   GEOMETRY ST_Curve,

   CONSTRAINT ST_LINK_PRIMARY_KEY PRIMARY KEY(NETWORK, LINK_ID),
   CONSTRAINT START_NODE_EXISTS FOREIGN KEY(NETWORK, START_NODE)
         REFERENCES ST_NODE(NETWORK, NODE_ID),
   CONSTRAINT END_NODE_EXISTS FOREIGN KEY(NETWORK, END_NODE)
         REFERENCES ST_NODE(NETWORK, NODE_ID)
   )
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The value of *NETWORK* is the network name which distinguishes which nodes are in the network.
2) Let *N* be the value of *NETWORK* for a given link.
3) The value of *LINK_ID* is the identifier of the link unique among all links with a *NETWORK* value equal to *N*.
4) The value of *START_NODE* is the unique identifier of the node at the start of the link; the start node shall have a value of *NETWORK* equal to *N*.
5) The value of *END_NODE* is the unique identifier of the node at the end of the link; the end node shall have a value of *NETWORK* equal to *N*.
6) The value of *GEOMETRY* is the geometry of the link. For logical networks, *GEOMETRY* is the null value.
7) All non-null geometry values in the *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* base tables in rows which have the same *NETWORK* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3 Topo-Net Routines

11.3.1 ST_AddIsoNetNode Function

Purpose

Insert a row into the <network-name>.*ST_NODE* view for an isolated node.

Definition

```
CREATE FUNCTION ST_AddIsoNetNode
   (anetwork CHARACTER VARYING(ST_MaxNetworkName),
    apoint ST_Point)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The function *ST_AddIsoNetNode(CHARACTER VARYING, ST_Point)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*,
   b) an *ST_Point* value *apoint*.
2) For the function *ST_AddIsoNetNode(CHARACTER VARYING, ST_Point)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork.ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) Otherwise:
      i) generate a unique node id INTEGER value *anodeid*,
      ii) insert into *ST_TOPO_NET.ST_NODE* values (*anetwork*, *anodeid*, *apoint*),
      iii) return *anodeid*.
3) All non-null geometry values in the *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* base tables in rows which have the same *NETWORK* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.2 ST_MoveIsoNetNode Procedure

Purpose

Update the <network-name>.*ST_NODE.GEOMETRY* value of an isolated node.

Definition

```
CREATE PROCEDURE ST_MoveIsoNetNode
   (anetwork CHARACTER VARYING(ST_MaxNetworkName),
    anode INTEGER,
    apoint ST_Point)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The procedure *ST_MoveIsoNetNode(CHARACTER VARYING, INTEGER, ST_Point)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*,
   b) an INTEGER value *anode*,
   c) an *ST_Point* value *apoint*.
2) For the procedure *ST_MoveIsoNetNode(CHARACTER VARYING, INTEGER, ST_Point)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork.ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) If *anode* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) Otherwise, for rows in *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork*:

      Case:

      i) If any *ST_TOPO_NET.ST_LINK.START_NODE* or *ST_TOPO_NET.ST_LINK.END_NODE* value is equal to *anode*, then an exception condition is raised: *SQL/MM Spatial exception – not isolated node*.

      ii) Otherwise, update *ST_TOPO_NET.ST_NODE*, set the *GEOMETRY* value equal to *apoint* where *NETWORK* is equal to *anetwork* and *NODE_ID* is equal to *anode*.
3) All non-null geometry values in the *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* base tables in rows which have the same *NETWORK* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.3 ST_RemIsoNetNode Procedure

Purpose

Delete a row in the <network-name>.*ST_NODE* view corresponding to an isolated node.

Definition

```
CREATE PROCEDURE ST_RemIsoNetNode
   (anetwork CHARACTER VARYING(ST_MaxNetworkName),
    anode INTEGER)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The procedure *ST_RemIsoNetNode(CHARACTER VARYING, INTEGER)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*,
   b) an INTEGER value *anode*.
2) For the procedure *ST_RemIsoNetNode(CHARACTER VARYING, INTEGER)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork.ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) If *anode* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) Otherwise, for rows in *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork*:

      Case:

      i) If any *ST_TOPO_NET.ST_LINK.START_NODE* or *ST_TOPO_NET.ST_LINK.END_NODE* value is equal to *anode*, then an exception condition is raised: *SQL/MM Spatial exception – not isolated node*.
      ii) Otherwise, delete from *ST_TOPO_NET.ST_NODE* where *NETWORK* is equal to *anetwork* and *NODE_ID* is equal to *anode*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.4 ST_AddLink Function

Purpose

Insert a row for a link into the <network-name>.*ST_LINK* view connecting two existing nodes.

Definition

```
CREATE FUNCTION ST_AddLink
   (anetwork CHARACTER VARYING(ST_MaxNetworkName),
    anode INTEGER,
    anothernode INTEGER,
    acurve ST_Curve)
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT
  STATIC DISPATCH
  BEGIN
     --
     -- See Description
     --
  END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The function *ST_AddLink(CHARACTER VARYING, INTEGER, INTEGER, ST_Curve)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*,
   b) an INTEGER value *anode*,
   c) an INTEGER value *anothernode*,
   d) an *ST_Curve* value *acurve*.
2) For the function *ST_AddLink(CHARACTER VARYING, INTEGER, INTEGER, ST_Curve)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) If *anode* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *anothernode* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   f) Otherwise, for rows in *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork*:

      Case:

      i) If SELECT COUNT(*) FROM *ST_TOPO_NET.ST_NODE* WHERE *NODE_ID* is equal to *anode* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent node,*

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) If SELECT COUNT(*) FROM *ST_TOPO_NET.ST_NODE* WHERE *NODE_ID* is equal to *anothernode* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent node,*

iii) If *acurve* is not the null value, then:

1) let *P1* be the *ST_Point* value equal to *ST_TOPO_NET.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *anode*. If *P1* is not equal to the null value and *P1* is not equal to *acurve.ST_StartPoint()*, then an exception condition is raised: *SQL/MM Spatial exception – start node not geometry start point*,

2) let *P2* be the *ST_Point* value equal to *ST_TOPO_NET.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *anothernode*. If *P2* is not equal to the null value and *P2* is not equal to *acurve.ST_EndPoint()*, then an exception condition is raised: *SQL/MM Spatial exception – end node not geometry end point*.

iii) Otherwise:

1) generate a unique link id INTEGER value *alinkid*,

2) insert into *ST_TOPO_NET.ST_LINK* values (*anetwork*, *alinkid*, *anode*, *anothernode*, *acurve*),

3) return *alinkid*.

3) All non-null geometry values in the *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* base tables in rows which have the same *NETWORK* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.5 ST_ChangeLinkGeom Procedure

Purpose

Update the <network-name>.*ST_LINK.GEOMETRY* value.

Definition

```
CREATE PROCEDURE ST_ChangeLinkGeom
   (anetwork CHARACTER VARYING(ST_MaxNetworkName),
    alink INTEGER,
    acurve ST_Curve)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The procedure *ST_ChangeLinkGeom(CHARACTER VARYING, INTEGER, ST_Curve)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*,
   b) an INTEGER value *alink*,
   c) an *ST_Curve* value *acurve*.
2) For the procedure *ST_ChangeLinkGeom(CHARACTER VARYING, INTEGER, ST_Curve)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork.ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) If *alink* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) Otherwise, for rows in *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork*:

      Case:

      i) If *acurve* is not the null value, then:

         1) let *P1* be the *ST_Point* value equal to *ST_TOPO_NET.ST_NODE.GEOMETRY* where *NODE_ID* is equal to the value of *ST_TOPO_NET.ST_LINK.START_NODE* where *LINK_ID* is equal to *alink*.

            Case;

            A) If *P1* is equal to the null value, then an exception condition is raised: *SQL/MM Spatial exception – null node geometry*,
            B) Otherwise, if *P1* is not equal to *acurve.ST_StartPoint()*, then an exception condition is raised: *SQL/MM Spatial exception – start node not geometry start point*,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) let *P2* be the *ST_Point* value equal to *ST_TOPO_NET.ST_NODE.GEOMETRY* where *NODE_ID* is equal to the value of *ST_TOPO_NET.ST_LINK.END_NODE* where *LINK_ID* is equal to *anothernode*.

   Case:

   A) if *P2* is equal to the null value, then an exception condition is raised: *SQL/MM Spatial exception – null node geometry*,

   B) otherwise, if *P2* is not equal to *acurve.ST_EndPoint()*, then an exception condition is raised: *SQL/MM Spatial exception – end node not geometry end point*.

ii) Otherwise, update *ST_TOPO_NET.ST_LINK*, set the *GEOMETRY* value equal to *acurve where NETWORK* is equal to *anetwork* and *LINK_ID* is equal to *alink*.

3) All non-null geometry values in the *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* base tables in rows which have the same *NETWORK* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.6 ST_RemoveLink Procedure

Purpose

Delete a row in the <network-name>.*ST_LINK* view corresponding to a link.

Definition

```
CREATE PROCEDURE ST_RemoveLink
   (anetwork CHARACTER VARYING(ST_MaxNetworkName),
    alink INTEGER)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The procedure *ST_RemoveLink(CHARACTER VARYING, INTEGER)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*,
   b) an INTEGER value *alink*.
2) For the procedure *ST_RemoveLink(CHARACTER VARYING, INTEGER)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) If *alink* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) Otherwise, for rows in *ST_TOPO_NET*.*ST_NODE* and *ST_TOPO_NET*.*ST_LINK* where *NETWORK* is equal to *anetwork*:
      i) delete from *ST_TOPO_NET*.*ST_LINK* where *NETWORK* is equal to *anetwork* and *LINK_ID* is equal to *alink*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.7 ST_InitTopoNet Procedure

Purpose

Create schema and views for a topology-network.

Definition

```
CREATE PROCEDURE ST_InitTopoNet
   (anetwork CHARACTER VARYING(ST_MaxNetworkName))
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The procedure *ST_InitTopoNet(CHARACTER VARYING)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*.
2) For the procedure *ST_InitTopoNet(CHARACTER VARYING)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.SCHEMATA WHERE SCHEMA_NAME = '*anetwork*' returns a value equal to 1 (one), then an exception condition is raised: *SQL/MM Spatial exception – schema already exists*.
   c) Otherwise:
      i) create a schema with a name equal to *anetwork*,
      ii) for schema *anetwork*, create ST_NODE and ST_LINK views in accordance with Subclause 11.1.2, "ST_NODE view" and Subclause 11.1.3 "ST_LINK view".

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.8 ST_NewLogLinkSplit Function

Purpose

Split a link in a logical network by creating a new node along an existing link, deleting the original link and replacing it with two new links.

Definition

```
CREATE FUNCTION ST_NewLogLinkSplit
   (anetwork CHARACTER VARYING(ST_MaxNetworkName),
    alink INTEGER)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The function ST_NewLogLinkSplit(CHARACTER VARYING, INTEGER) takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*,
   b) an INTEGER value *alink*.
2) For the function *ST_NewLogLinkSplit(CHARACTER VARYING, INTEGER)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) If *alink* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) Otherwise, for rows in *ST_TOPO_NET*.*ST_NODE* and *ST_TOPO_NET*.*ST_LINK* where *NETWORK* is equal to *anetwork*:

      Case:

      i) If there is no row in *ST_TOPO_NET*.*ST_LINK* with an *LINK_ID* value equal to *alink*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent link*.
      ii) If the value of *ST_TOPO_NET*.*ST_LINK*.*GEOMETRY* where *LINK_ID* is equal to *alink* is not equal to the null value, then an exception condition is raised: *SQL/MM Spatial exception – not a logical link*.
      iii) Otherwise:
         1) generate a unique node id INTEGER value *newnode*,
         2) insert into *ST_TOPO_NET*.*ST_NODE* values (*anetwork*, *newnode*, NULL),

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3) select from *ST_TOPO_NET.ST_LINK START_NODE* and *ST_TOPO_NET.END_NODE* into *oldstart* and *oldend* where *LINK_ID* is equal to *alink*,

4) delete from *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork* and *LINK_ID* is equal to *alink*,

5) generate two unique link id INTEGER values *newlink1* and *newlink2*,

6) insert into *ST_TOPO_NET.ST_LINK* values (*anetwork*, *newlink1*, *oldstart*, *newnode*, NULL),

7) insert into *ST_TOPO_NET.ST_LINK* values (*anetwork*, *newlink2*, *newnode*, *oldend*, NULL),

8) return *newnode*,

iv) Both new links have the same direction as the link being split.

v) To determine the two new link ID values, select *LINK_ID* from *ST_TOPO_NET.ST_LINK* where *START_NODE* or *END_NODE* is equal to *newnode*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.9 ST_ModLogLinkSplit Function

Purpose

Split a link in a logical network by creating a new node along an existing link, modifying the original link and adding a new link.

Definition

```
CREATE FUNCTION ST_ModLogLinkSplit
   (anetwork CHARACTER VARYING(ST_MaxNetworkName),
    alink INTEGER)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The function *ST_ModLogLinkSplit(CHARACTER VARYING, INTEGER)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*,
   b) an INTEGER value *alink*.
2) For the function *ST_ModLogLinkSplit(CHARACTER VARYING, INTEGER)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) If *alink* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) Otherwise, for rows in *ST_TOPO_NET*.*ST_NODE* and *ST_TOPO_NET*.*ST_LINK* where *NETWORK* is equal to *anetwork*:

      Case:

      i) If there is no row in *ST_TOPO_NET*.*ST_LINK* with an *LINK_ID* value equal to *alink*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent link*.
      ii) If the value of *ST_TOPO_NET*.*ST_LINK*.*GEOMETRY* where *LINK_ID* is equal to *alink* is not equal to the null value, then an exception condition is raised: *SQL/MM Spatial exception – not a logical link*.
      iii) Otherwise:
         1) generate a unique node id INTEGER value *newnode*,
         2) insert into *ST_TOPO_NET*.*ST_NODE* values (*anetwork*, *newnode*, NULL),

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3) select from *ST_TOPO_NET.ST_LINK END_NODE* into *oldend* where *LINK_ID* is equal to *alink*,

4) generate a unique link id INTEGER value *newlink*,

5) update *ST_TOPO_NET.ST_LINK* set the *END_NODE* value equal to *newnode* where *NETWORK* is equal to *anetwork* and *LINK_ID* is equal to *alink*,

6) insert into *ST_TOPO_NET.ST_LINK* values (*anetwork*, *newlink*, *newnode*, *oldend*, NULL),

7) return *newnode*.

iv) The new and modified links have the same direction as the link being split.

v) To determine the new link ID value, select *LINK_ID* from *ST_TOPO_NET.ST_LINK* where *START_NODE* is equal to *newnode*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.10 ST_NewGeoLinkSplit Function

Purpose

Split a link in a network with geometry by creating a new node along an existing link, deleting the original link and replacing it with two new links.

Definition

```
CREATE FUNCTION ST_NewGeoLinkSplit
   (anetwork CHARACTER VARYING(ST_MaxNetworkName),
    alink INTEGER,
    apoint ST_Point)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The function *ST_NewGeoLinkSplit(CHARACTER VARYING, INTEGER, ST_Point)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*,
   b) an INTEGER value *alink*,
   c) an *ST_Point* value *apoint*.
2) For the function *ST_NewGeoLinkSplit(CHARACTER VARYING, INTEGER, ST_Point)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) If *alink* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *apoint* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   f) Otherwise, for rows in *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork*:

      Case:

      i) If there is no row in *ST_TOPO_NET.ST_LINK* with an *LINK_ID* value equal to *alink*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent link*.
      ii) Let *G* be equal to the *ST_Curve* value of *ST_TOPO_NET.ST_LINK.GEOMETRY* where *LINK_ID* is equal to *alink*. If *G* is equal to the null value, then an exception condition is raised: *SQL/MM Spatial exception – link has null geometry*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

iii) If *apoint.ST_Within(G)* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – point not on link*.

iv) Otherwise:

1) generate a unique node id INTEGER value *newnode*,

2) insert into *ST_TOPO_NET.ST_NODE* values (*anetwork*, *newnode*, *apoint*),

3) select from *ST_TOPO_NET.ST_LINK START_NODE*, *END_NODE*, and *GEOMETRY* into *oldstart*, *oldend*, and *oldgeom* where *LINK_ID* is equal to *alink*,

4) delete from *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork* and *LINK_ID* is equal to *alink*,

5) generate two unique link id INTEGER values *newlink1* and *newlink2*,

6) create two new *ST_Curve* values, *newgeom1* and *newgeom2* from *oldgeom*, breaking *oldgeom* at the location defined by *apoint*,

7) insert into *ST_TOPO_NET.ST_LINK* values (*anetwork*, *newlink1*, *oldstart*, *newnode*, *newgeom1*),

8) insert into *ST_TOPO_NET.ST_LINK* values (*anetwork*, *newlink2*, *newnode*, *oldend*, *newgeom2*),

9) return *newnode*.

v) Both new links have the same direction as the link being split.

vi) To determine the two new link ID values, select *LINK_ID* from *ST_TOPO_NET.ST_LINK* where *START_NODE* or *END_NODE* is equal to *newnode*.

3) All non-null geometry values in the *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* base tables in rows which have the same *NETWORK* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.11 ST_ModGeoLinkSplit Function

Purpose

Split a link in a network with geometry by creating a new node along an existing link, modifying the original link and adding a new link.

Definition

```
CREATE FUNCTION ST_ModGeoLinkSplit
   (anetwork CHARACTER VARYING(ST_MaxNetworkName),
    alink INTEGER,
    apoint ST_Point)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The function *ST_ModGeoLinkSplit(CHARACTER VARYING, INTEGER, ST_Point)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*,
   b) an INTEGER value *alink*,
   c) an *ST_Point* value *apoint*.
2) For the function *ST_ModGeoLinkSplit(CHARACTER VARYING, INTEGER, ST_Point)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) If *alink* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *apoint* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   f) Otherwise, for rows in *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork*:

      Case:

      i) If there is no row in *ST_TOPO_NET.ST_LINK* with an *LINK_ID* value equal to *alink*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent link*.
      ii) Let *G* be equal to the *ST_Curve* value of *ST_TOPO_NET.ST_LINK.GEOMETRY* where *LINK_ID* is equal to *alink*. If *G* is equal to the null value, then an exception condition is raised: *SQL/MM Spatial exception – link has null geometry*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

iii) If *apoint.ST_Within(G)* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – point not on link*.

iv) Otherwise:

1) generate a unique node id INTEGER value *newnode*,

2) insert into *ST_TOPO_NET.ST_NODE* values (*anetwork*, *newnode*, *apoint*),

3) select from *ST_TOPO_NET.ST_LINK END_NODE* and *GEOMETRY* into *oldend* and *oldgeom* where *LINK_ID* is equal to *alink*,

4) generate a unique link id INTEGER value *newlink*,

5) create two new *ST_Curve* values, *newgeom1* and *newgeom2* from *oldgeom*, breaking *oldgeom* at the location defined by *apoint*,

6) update *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork* and *LINK_ID* is equal to *alink*:

A) set the *END_NODE* value equal to *newnode*,

B) set the *GEOMETRY* value equal to *newgeom1*,

7) insert into *ST_TOPO_NET.ST_LINK* values (*anetwork*, *newlink*, *newnode*, *oldend*, *newgeom2*),

8) return *newnode*.

v) The new and modified links have the same direction as the link being split.

vi) To determine the new link ID value, select *LINK_ID* from *ST_TOPO_NET.ST_LINK* where *START_NODE* is equal to *newnode*.

5) All non-null geometry values in the *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* base tables in rows which have the same *NETWORK* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.12 ST_NewLinkHeal Function

Purpose

Heal two links by deleting the node connecting them, deleting both links, and replacing them with a new link whose direction is the same as the first link provided.

Definition

```
CREATE FUNCTION ST_NewLinkHeal
   (anetwork CHARACTER VARYING(ST_MaxNetworkName),
    alink INTEGER,
    anotherlink INTEGER)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The function *ST_NewLinkHeal(CHARACTER VARYING, INTEGER, INTEGER)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*,
   b) an INTEGER value *alink*,
   c) an INTEGER value *anotherlink*.
2) For the function *ST_NewLinkHeal(CHARACTER VARYING, INTEGER, INTEGER)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) If *alink* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *anotherlink* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   f) Otherwise, for rows in *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork*:

      Case:

      i) If there is no row in *ST_TOPO_NET.ST_LINK* with an *LINK_ID* value equal to *alink*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent link*.
      ii) If there is no row in *ST_TOPO_NET.ST_LINK* with an *LINK_ID* value equal to *anotherlink*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent link*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

iii) Check if links are connected:

1) let *COMMONNODE* be an INTEGER value,

2) let *CASE* be an INTEGER value,

3) let *S1* and *E1* be INTEGER values equal to the *ST_TOPO_NET.ST_LINK.START_NODE* and *ST_TOPO_NET.END_NODE* values, respectively, where *LINK_ID* is equal to *alink*,

4) let *S2* and *E2* be INTEGER values equal to the *ST_TOPO_NET.ST_LINK.START_NODE* and *ST_TOPO_NET.END_NODE* values, respectively, where *LINK_ID* is equal to *anotherlink*,

5) Case:

A) if *E1* is equal to *S2*, then set *CASE* equal to 1 (one) and *COMMONNODE* equal to *E1*,

B) if *E1* is equal to *E2*, then set *CASE* equal to 2 and *COMMONNODE* equal to *E1*,

C) if *S1* is equal to *S2*, then set *CASE* equal to 3 and *COMMONNODE* equal to *S1*,

D) if *S1* is equal to *E2*, then set *CASE* equal to 4 and *COMMONNODE* equal to *S1*,

E) otherwise, an exception condition is raised: *SQL/MM Spatial exception – non-connected links*,

iv) If *COMMONNODE* is equal to the *ST_TOPO_NET.ST_LINK.START_NODE* or *ST_TOPO_NET.ST_LINK.END_NODE* value for any link other than the link whose *LINK_ID* is equal to either *alink* or *anotherlink*, then an exception condition is raised: *SQL/MM Spatial exception – other links connected*,

v) Otherwise:

1) delete from *ST_TOPO_NET.ST_NODE* where *NETWORK* is equal to *anetwork* and *NODE_ID* is equal to *COMMONNODE*,

2) select from *ST_TOPO_NET.ST_LINK START_NODE*, *END_NODE*, and *GEOMETRY* into *oldstart1*, *oldend1*, and *oldgeom1* where *LINK_ID* is equal to *alink*,

3) select from *ST_TOPO_NET.ST_LINK START_NODE*, *END_NODE*, and *GEOMETRY* into *oldstart2*, *oldend2*, and *oldgeom2* where *LINK_ID* is equal to *anotherlink*,

4) delete from *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork* and *LINK_ID* is equal to *alink*,

5) delete from *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork* and *LINK_ID* is equal to *anotherlink*,

6) generate a unique link id INTEGER value *newlink*,

7) case:

A) if *CASE* is equal to 1 (one) then:

I) Case:

1) if *oldgeom1* or *oldgeom2* is equal to the null value, then set *newgeom* equal to the null value,

2) otherwise, create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by connecting the end of *oldgeom1* to the start of *oldgeom2*,

II) insert into *ST_TOPO_NET.ST_LINK* values (*anetwork*, *newlink*, *oldstart1*, *oldend2*, *newgeom*),

B) if *CASE* is equal to 2 then:

I) Case:

1) if *oldgeom1* or *oldgeom2* is equal to the null value, then set *newgeom* equal to the null value,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) otherwise, create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by reversing the direction of *oldgeom2* and connecting it to the end of *oldgeom1*,

II) insert into *ST_TOPO_NET.ST_LINK* values (*anetwork*, *newlink*, *oldstart1*, *oldstart2*, *newgeom*),

C) if *CASE* is equal to 3 then:

I) case:

1) if *oldgeom1* or *oldgeom2* is equal to the null value, then set *newgeom* equal to the null value,

2) otherwise, create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by reversing the direction of *oldgeom2* and connecting *oldgeom1* to the new end,

II) insert into *ST_TOPO_NET.ST_LINK* values (*anetwork*, *newlink*, *oldend2*, *oldend1*, *newgeom*),

D) if *CASE* is equal to 4 then:

I) case:

1) if *oldgeom1* or *oldgeom2* is equal to the null value, then set *newgeom* equal to the null value,

2) otherwise, create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by connecting *oldgeom1* to the end of *oldgeom2*,

II) insert into *ST_TOPO_NET.ST_LINK* values (*anetwork*, *newlink*, *oldstart2*, *oldend1*, *newgeom*),

8) return *newlink*.

vi) The direction of the new link shall be the same as the direction of the first link supplied.

3) All non-null geometry values in the *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* base tables in rows which have the same *NETWORK* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.13 ST_ModLinkHeal Procedure

Purpose

Heal two links by deleting the node connecting them, modifying the first link provided, and deleting the second link.

Definition

```
CREATE PROCEDURE ST_ModLinkHeal
    (anetwork CHARACTER VARYING(ST_MaxNetworkName),
     alink INTEGER,
     anotherlink INTEGER)
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    CALLED ON NULL INPUT
    BEGIN
       --
       -- See Description
       --
    END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The procedure *ST_ModLinkHeal(CHARACTER VARYING, INTEGER, INTEGER)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*,
   b) an INTEGER value *alink*,
   c) an INTEGER value *anotherlink*.
2) For the procedure *ST_ModLinkHeal(CHARACTER VARYING, INTEGER, INTEGER)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) If *alink* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *anotherlink* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   f) Otherwise, for rows in *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork*:

      Case:

      i) If there is no row in *ST_TOPO_NET.ST_LINK* with an *LINK_ID* value equal to *alink*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent link*.
      ii) If there is no row in *ST_TOPO_NET.ST_LINK* with an *LINK_ID* value equal to *anotherlink*, then an exception condition is raised: *SQL/MM Spatial exception – non-existent link*.
      iii) Check if links are connected:

         1) let *COMMONNODE* be an INTEGER value,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) let *CASE* be an INTEGER value,

3) let *S1* and *E1* be INTEGER values equal to the *ST_TOPO_NET.ST_LINK.START_NODE* and *ST_TOPO_NET.END_NODE* values, respectively, where *LINK_ID* is equal to *alink*,

4) let *S2* and *E2* be INTEGER values equal to the *ST_TOPO_NET.ST_LINK.START_NODE* and *ST_TOPO_NET.END_NODE* values, respectively, where *LINK_ID* is equal to *anotherlink*,

5) case:

A) if *E1* is equal to *S2*, then set *CASE* equal to 1 (one) and *COMMONNODE* equal to *E1*,

B) if *E1* is equal to *E2*, then set *CASE* equal to 2 and *COMMONNODE* equal to *E1*,

C) if *S1* is equal to *S2*, then set *CASE* equal to 3 and *COMMONNODE* equal to *S1*,

D) if *S1* is equal to *E2*, then set *CASE* equal to 4 and *COMMONNODE* equal to *S1*,

E) otherwise, an exception condition is raised: *SQL/MM Spatial exception – non-connected links*,

iv) If *COMMONNODE* is equal to the *ST_TOPO_NET.ST_LINK.START_NODE* or *ST_TOPO_NET.ST_LINK.END_NODE* value for any link other than the link whose *LINK_ID* is equal to either *alink* or *anotherlink*, then an exception condition is raised: *SQL/MM Spatial exception – other links connected*,

v) Otherwise:

1) delete from *ST_TOPO_NET.ST_NODE* where *NETWORK* is equal to *anetwork* and *NODE_ID* is equal to *COMMONNODE*,

2) select from *ST_TOPO_NET.ST_LINK GEOMETRY* into *oldgeom1* where *LINK_ID* is equal to *alink*,

3) select from *ST_TOPO_NET.ST_LINK START_NODE*, *END_NODE*, and *GEOMETRY* into *oldstart2*, *oldend2*, and *oldgeom2* where *LINK_ID* is equal to *anotherlink*,

4) delete from *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork* and *LINK_ID* is equal to *anotherlink*,

5) case:

A) if *CASE* is equal to 1 (one) then:

I) case:

1) if *oldgeom1* or *oldgeom2* is equal to the null value, then set *newgeom* equal to the null value,

2) otherwise, create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by connecting the end of *oldgeom1* to the start of *oldgeom2*,

II) update *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork* and *LINK_ID* is equal to *alink*:

1) set the *END_NODE* value equal to *oldend2*,

2) set the *GEOMETRY* value equal to *newgeom*,

B) if *CASE* is equal to 2 then:

I) case:

1) if *oldgeom1* or *oldgeom2* is equal to the null value, then set *newgeom* equal to the null value,

2) otherwise, create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by reversing the direction of *oldgeom2* and connecting it to the end of *oldgeom1*,

II) update *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork* and *LINK_ID* is equal to *alink*:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

1) set the *END_NODE* value equal to *oldstart2*,

2) set the *GEOMETRY* value equal to *newgeom*,

C) if *CASE* is equal to 3 then:

I) case:

1) if *oldgeom1* or *oldgeom2* is equal to the null value, then set *newgeom* equal to the null value,

2) otherwise, create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by reversing the direction of *oldgeom2* and connecting *oldgeom1* to the new end,

II) update *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork* and *LINK_ID* is equal to *alink*:

1) set the *START_NODE* value equal to *oldend2*,

2) set the *GEOMETRY* value equal to *newgeom*,

D) if *CASE* is equal to 4 then:

I) case:

1) if *oldgeom1* or *oldgeom2* is equal to the null value, then set *newgeom* equal to the null value,

2) otherwise, create a new *ST_Curve* value *newgeom* from *oldgeom1* and *oldgeom2*, by connecting *oldgeom1* to the end of *oldgeom2*,

II) update *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork* and *LINK_ID* is equal to *alink*:

1) set the *START_NODE* value equal to *oldstart2*,

2) set the *GEOMETRY* value equal to *newgeom*.

3) All non-null geometry values in the *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* base tables in rows which have the same *NETWORK* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.14 ST_LogiNetFromTGeo Procedure

Purpose

Create a logical topology-network from a Topology-geometry.

Definition

```
CREATE PROCEDURE ST_LogiNetFromTGeo
   (anetwork CHARACTER VARYING(ST_MaxNetworkName),
    atopology CHARACTER VARYING(ST_MaxTopologyName))
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.
2) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The procedure *ST_LogiNetFromTGeo(CHARACTER VARYING, CHARACTER VARYING)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*,
   b) a CHARACTER VARYING value *atopology*.
2) For the procedure *ST_LogiNetFromTGeo(CHARACTER VARYING, CHARACTER VARYING)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topology name*.
   f) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   g) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.SCHEMATA WHERE SCHEMA_NAME = '*anetwork*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent schema*.
   h) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.SCHEMATA WHERE SCHEMA_NAME = '*atopology*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent schema*.
   i) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA = '*anetwork*' AND TABLE_NAME = '*ST_NODE*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent view*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

j) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA = '*anetwork*' AND TABLE_NAME = '*ST_LINK*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent view*.

k) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA = '*atopology*' AND TABLE_NAME = '*ST_NODE*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent view*.

l) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA = '*atopology*' AND TABLE_NAME = '*ST_EDGE*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent view*.

m) Otherwise, for rows in *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork*:

Case:

i) If SELECT COUNT(*) FROM *ST_TOPO_NET.ST_NODE* returns a value greater than 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-empty view*.

ii) If SELECT COUNT(*) FROM *ST_TOPO_NET.ST_LINK* returns a value greater than 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-empty view*.

iii) Otherwise:

1) for nodes in *atopology*, insert into *ST_TOPO_NET.ST_NODE* select *anetwork*, *NODE_ID*, NULL from *atopology*.*ST_NODE*,

2) for edges in *atopology*, insert into *ST_TOPO_NET.ST_LINK* select *anetwork*, *EDGE_ID*, *START_NODE*, *END_NODE*, NULL from *atopology*.*ST_EDGE*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.15 ST_SpatNetFromTGeo Procedure

Purpose

Create a spatial topology-network from a Topology-geometry.

Definition

```
CREATE PROCEDURE ST_SpatNetFromTGeo
   (anetwork CHARACTER VARYING(ST_MaxNetworkName),
    atopology CHARACTER VARYING(ST_MaxTopologyName))
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.
2) *ST_MaxTopologyName* is the implementation-defined maximum length of the CHARACTER VARYING topology name.

Description

1) The procedure *ST_SpatNetFromTGeo(CHARACTER VARYING, CHARACTER VARYING)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*,
   b) a CHARACTER VARYING value *atopology*.
2) For the procedure *ST_SpatNetFromTGeo(CHARACTER VARYING, CHARACTER VARYING)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) If *atopology* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If *atopology* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid topoloyg name*.
   f) If the user has not been granted SELECT privilege on *atopology*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – topology privilege denied*.
   g) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.SCHEMATA WHERE SCHEMA_NAME = '*anetwork*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent schema*.
   h) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.SCHEMATA WHERE SCHEMA_NAME = '*atopology*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent schema*.
   i) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA = '*anetwork*' AND TABLE_NAME = '*ST_NODE*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent view*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

j) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA = '*anetwork*' AND TABLE_NAME = '*ST_LINK*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent view*.

k) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA = '*atopology*' AND TABLE_NAME = '*ST_NODE*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent view*.

l) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA = '*atopology*' AND TABLE_NAME = '*ST_EDGE*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent view*.

m) Otherwise, for rows in *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork*:

Case:

i) If SELECT COUNT(*) FROM *ST_TOPO_NET.ST_NODE* returns a value greater than 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-empty view*.

ii) If SELECT COUNT(*) FROM *ST_TOPO_NET.ST_LINK* returns a value greater than 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-empty view*.

iii) Otherwise:

1) for nodes in *atopology*, insert into *ST_TOPO_NET.ST_NODE* select *anetwork*, *NODE_ID*, *GEOMETRY* from *ST_TOPO_GEO.ST_NODE*,

2) for edges in *atopology*, insert into *ST_TOPO_NET.ST_LINK* select *anetwork*, *EDGE_ID*, *START_NODE*, *END_NODE*, *GEOMETRY* from *ST_TOPO_GEO.ST_EDGE*.

3) All non-null geometry values in the *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* base tables in rows which have the same *NETWORK* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.16 ST_SpatNetFromGeom Procedure

Purpose

Create a topology-network from a collection of geometry values.

Definition

```
CREATE PROCEDURE ST_SpatNetFromGeom
    (anetwork CHARACTER VARYING(ST_MaxNetworkName),
     ageomcollection ST_GeomCollection)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The procedure *ST_SpatNetFromGeom(CHARACTER VARYING, ST_GeomCollection)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*,
   b) an ST_GeomCollection value *ageomcollection*.
2) For the procedure *ST_SpatNetFromGeom(CHARACTER VARYING, ST_GeomCollection)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) If *ageomcollection* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   e) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.SCHEMATA WHERE SCHEMA_NAME = '*anetwork*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent schema*.
   f) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA = '*anetwork*' AND TABLE_NAME = '*ST_NODE*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent view*.
   g) If SELECT COUNT(*) FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHEMA = '*anetwork*' AND TABLE_NAME = '*ST_LINK*' returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-existent view*.
   h) Otherwise, for rows in *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* where *NETWORK* is equal to *anetwork*:

      Case:

      i) If SELECT COUNT(*) FROM *ST_TOPO_NET.ST_NODE* returns a value greater than 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-empty view*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

ii) If SELECT COUNT(*) FROM *ST_TOPO_NET.ST_LINK* returns a value greater than 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – non-empty view*.

iii) Otherwise:

1) using the geometry values in *ageomcollection*, create a corresponding set of network primitives (nodes and links),

2) for each node:

A) let *nodeid* be the generated unique node id INTEGER value,

B) let *geometry* be the *ST_Point* value which specifies the node location,

C) insert into *ST_TOPO_NET.ST_NODE* values (*anetwork*, *nodeid*, *geometry*),

3) for each link:

A) let *linkid* be the generated unique link id INTEGER value,

B) let *startnode* be the INTEGER value equal to the node id of the node at the start of the link,

C) let *endnode* be the INTEGER value equal to the node id of the node at the end of the link,

D) let *geometry* be the *ST_Curve* value which represents the geometry of the link, in the same direction as the link,

E) insert into *ST_TOPO_NET.ST_LINK* values (*anetwork*, *linkid*, *startnode*, *endnode*, *geometry*).

3) All non-null geometry values in the *ST_TOPO_NET.ST_NODE* and *ST_TOPO_NET.ST_LINK* base tables in rows which have the same *NETWORK* column value shall have the same spatial reference system identifier.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.17 ST_ValidLogicalNet Function

Purpose

Return a table containing possible network inconsistencies for a logical topology-network.

Definition

```
CREATE FUNCTION ST_ValidLogicalNet
   (anetwork CHARACTER VARYING(ST_MaxNetworkName))
   RETURNS TABLE
      (error CHARACTER VARYING(30),
       primitive1 INTEGER,
       primitive2 INTEGER)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The function *ST_ValidLogicalNet(CHARACTER VARYING)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*.
2) For the function *ST_ValidLogicalNet(CHARACTER VARYING)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) Otherwise, for rows in *ST_TOPO_NET*.*ST_NODE* and *ST_TOPO_NET*.*ST_LINK* where *NETWORK* is equal to *anetwork*:

      Case:

      i) If SELECT COUNT(*) FROM *ST_TOPO_NET*.*ST_NODE* returns a value equal to 0 (zero) and SELECT COUNT(*) FROM *ST_TOPO_NET*.*ST_LINK* returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – empty network*.

      ii) Otherwise,

         1) let T be a table value consisting of the following three columns:

            A) a column *error* of type CHARACTER VARYING, which identifies the type of inconsistency found in *anetwork*,

            B) a column *primitive1* of type INTEGER, which contains the node or link id of the first offending network primitive,

            C) a column *primitive2* of type INTEGER, which contains the node or link id of the second offending network primitive,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) for each node with a non-null geometry value:

   A) let *E* be the CHARACTER VARYING value equal to 'node has geometry',

   B) let *N1* be the INTEGER value equal to *ST_TOPO_NET.ST_NODE.NODE_ID* for a node in *anetwork*,

   C) let *G1* be the *ST_Point* value equal to *ST_TOPO_NET.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *N1*,

   D) let *N2* be equal to the null value,

   E) for all values of *N1* such that *G1* is not equal to the null value, insert into *T* values (*E*, *N1*, *N2*),

3) for each link with a non-null geometry value:

   A) let *E* be the CHARACTER VARYING value equal to 'link has geometry',

   B) let *L1* be the INTEGER value equal to *ST_TOPO_NET.ST_LINK.LINK_ID* for a link in *anetwork*,

   C) let *G1* be the *ST_Curve* value equal to *ST_TOPO_NET.ST_LINK.GEOMETRY* where *LINK_ID* is equal to *L1*,

   D) let *L2* be equal to the null value,

   E) for all values of *L1* such that *G1* is not equal to the null value, insert into *T* values (*E*, *L1*, *L2*),

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

11.3.18 ST_ValidSpatialNet Function

Purpose

Return a table containing network inconsistencies for a spatial topology-network.

Definition

```
CREATE FUNCTION ST_ValidSpatialNet
   (atopology CHARACTER VARYING(ST_MaxNetworkName))
   RETURNS TABLE
      (error CHARACTER VARYING(30),
       primitive1 INTEGER,
       primitive2 INTEGER)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT
   STATIC DISPATCH
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxNetworkName* is the implementation-defined maximum length of the CHARACTER VARYING network name.

Description

1) The function *ST_ValidSpatialNet(CHARACTER VARYING)* takes the following input parameters:
   a) a CHARACTER VARYING value *anetwork*.
2) For the function *ST_ValidSpatialNet(CHARACTER VARYING)*:

   Case:

   a) If *anetwork* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If *anetwork* is not a valid schema name, then an exception condition is raised: *SQL/MM Spatial exception – invalid network name*.
   c) If the user has not been granted SELECT privilege on *anetwork*.*ST_NODE*, then an exception condition is raised: *SQL/MM Spatial exception – network privilege denied*.
   d) Otherwise, for rows in *ST_TOPO_NET*.*ST_NODE* and *ST_TOPO_NET*.*ST_LINK* where *NETWORK* is equal to *anetwork*:

      Case:

      i) If SELECT COUNT(*) FROM *ST_TOPO_NET*.*ST_NODE* returns a value equal to 0 (zero) and SELECT COUNT(*) FROM *ST_TOPO_NET*.*ST_LINK* returns a value equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – empty network*.

      ii) Otherwise,

         1) let *T* be a table value consisting of the following three columns:

            A) a column *error* of type CHARACTER VARYING, which identifies the type of inconsistency found in *anetwork*,

            B) a column *primitive1* of type INTEGER, which contains the node or link id of the first offending network primitive,

            C) a column *primitive2* of type INTEGER, which contains the node or link id of the second offending network primitive,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

2) for each node with a null geometry value:

   A) let *E* be the CHARACTER VARYING value equal to 'missing node geometry',

   B) let *N1* be the INTEGER value equal to *ST_TOPO_NET.ST_NODE.NODE_ID* for a node in *anetwork*,

   C) let *G1* be the *ST_Point* value equal to *ST_TOPO_NET.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *N1*,

   D) let *N2* be equal to the null value,

   E) for all values of *N1* such that *G1* is equal to the null value, insert into *T* values (*E*, *N1*, *N2*),

3) for each link with a null geometry value:

   A) let *E* be the CHARACTER VARYING value equal to 'missing link geometry',

   B) let *L1* be the INTEGER value equal to *ST_TOPO_NET.ST_LINK.LINK_ID* for a link in *anetwork*,

   C) let *G1* be the *ST_Curve* value equal to *ST_TOPO_NET.ST_LINK.GEOMETRY* where *LINK_ID* is equal to *L1*,

   D) let *L2* be equal to the null value,

   E) for all values of *L1* such that *G1* is equal to the null value, insert into *T* values (*E*, *L1*, *L2*),

4) for each link having a geometry with a start point not equal to the geometry of its start node:

   A) let *E* be the CHARACTER VARYING value equal to 'geometry start mismatch',

   B) let *L1* be the INTEGER value equal to *ST_TOPO_NET.ST_LINK*_ID for a link in *anetwork*,

   C) let *G1* be the *ST_Curve* value equal to *ST_TOPO_NET.ST_LINK.GEOMETRY* where *LINK_ID* is equal to *L1*,

   D) let *N2* be the INTEGER value equal to *ST_TOPO_NET.ST_LINK.START_NODE* where *LINK_ID* is equal to *L1*,

   E) let *G2* be the *ST_Point* value equal to *ST_TOPO_NET.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *N2*,

   F) for all values of *L1* and *N2* such that the value returned by *G2.ST_Equals(G1.ST_StartPoint())* is not equal to 1 (one), insert into *T* values (*E*, *L1*, *N2*),

5) for each link having a geometry with an end point not equal to the geometry of its end node:

   A) let *E* be the CHARACTER VARYING value equal to 'geometry end mismatch',

   B) let *L1* be the INTEGER value equal to *ST_TOPO_NET.ST_LINK*_ID for a link in *anetwork*,

   C) let *G1* be the *ST_Curve* value equal to *ST_TOPO_NET.ST_LINK.GEOMETRY* where *LINK_ID* is equal to *L1*,

   D) let *N2* be the INTEGER value equal to *ST_TOPO_NET.ST_LINK.END_NODE* where *LINK_ID* is equal to *L1*,

   E) let *G2* be the *ST_Point* value equal to *ST_TOPO_NET.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *N2*,

   F) for all values of *L1* and *N2* such that the value returned by *G2.ST_Equals(G1.ST_EndPoint())* is not equal to 1 (one), insert into *T* values (*E*, *L1*, *N2*),

6) if all geometries do not have the same spatial reference system identifier:

   A) let *E* be the CHARACTER VARYING value equal to 'mixed SRIDs',

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

B) let *N1* be the INTEGER value equal to *ST_TOPO_NET.ST_NODE.NODE_ID* for a node in *anetwork*,

C) let *G1* be the *ST_Point* value equal to *ST_TOPO_NET.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *N1*,

D) let *S1* be the INTEGER value returned by *G1.ST_SRID()*,

E) let *N2* be the INTEGER value equal to *ST_TOPO_NET.ST_NODE.NODE_ID* for another node in *anetwork*,

F) let *G2* be the *ST_Point* value equal to *ST_TOPO_NET.ST_NODE.GEOMETRY* where *NODE_ID* is equal to *N2*,

G) let *S2* be the INTEGER value returned by *G2.ST_SRID()*,

H) let *L3* be the INTEGER value equal to *ST_TOPO_NET.ST_LINK.LINK_ID* for a link in *anetwork*,

I) let *G3* be the *ST_Curve* value equal to *ST_TOPO_NET.ST_LINK.GEOMETRY* where *LINK_ID* is equal to *L3*,

J) let *S3* be the INTEGER value returned by *G3.ST_SRID()*,

K) if any value of *N2* or *L3* has a corresponding value of *S2* or *S3*, respectively, such that *S2* is not equal to *S1* or *S3* is not equal to *S1*, insert into *T* values (*E*, NULL, NULL).

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

12 General Routines

12.1 Shortest Path Routines

12.1.1 ST_ShortestUndPath Function

Purpose

Return a table containing IDs of undirected shortest paths between two specified points that shall be either a start or end point of simple ST_Geometry value in a referenced table.

Definition

```
CREATE FUNCTION ST_ShortestUndPath
   (paths_table_name DT1,
    path_id_column DT2,
    geometry_column DT2,
    edge_weight_column DT2,
    start_point ST_Point,
    end_point ST_Point)
   RETURNS TABLE
      (shortest_path INTEGER ARRAY[ST_MaxArrayElements],
       total_weight DOUBLE PRECISION)
   DETERMINISTIC
   BEGIN
      --
      -- See Description
      --
   END

CREATE FUNCTION ST_ShortestUndPath
   (paths_table_name DT1,
    path_id_column DT2,
    geometry_column DT2,
    start_point ST_Point,
    end_point ST_Point)
   RETURNS TABLE
      (shortest_path INTEGER ARRAY[ST_MaxArrayElements],
       total_weight DOUBLE PRECISION)
   DETERMINISTIC
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxArrayElements* is the implementation-defined maximum cardinality of an array for the number of geometric paths.
2) *DT1* is data type of variable-length character string with character set SQL_IDENTIFIER and implementation-defined maximum length.
3) *DT2* is data type of variable-length character string with character set SQL_IDENTIFIER and maximum length not less than 128 characters.

Description

1) The ST_ShortestUndPath functions are for public use.
2) The function *ST_ShortestUndPath* takes the following input parameters:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

a) a value *paths_table_name* of type *DT1*, which has the name of a referenced table consisting of at least three columns: an INTEGER type column which has a unique number to identify a path as path identifier, an *ST_Geometry* type column which has a spatial representation of a path with 1-dimensional geometry as path geometry, and a DOUBLE PRECISION type column which has weight value on the path as edge weight. Let *S* be *paths_table_name* value. Let *V* be the character string that is the value of TRIM(BOTH ' ' FROM *S*). If *V* value does not conform to the Format and Syntax Rules of <table name> specified in ISO/IEC 9075-2, or the table specified by paths_table_name value does not exist in the system, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

b) a value *path_id_column* of type *DT2*, which has the name of the column for path identifier in the table specified by *paths_table_name*. If the column specified by *path_id_column* does not exist in the table specified by *paths_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

c) a value *geometry_column* of type *DT2*, which has the name of the column for path geometry in the table specified by *paths_table_name*. If the column specified by *geometry_column* does not exist in the table specified by *paths_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

d) a value *edge_weight_column* of type *DT2*, which has the name of the column for edge weight in the table specified by *paths_table_name*. If the column specified by *edge_weight_column* does not exist in the table specified by *paths_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

e) an *ST_Point* value *start_point*, which is a start point for getting the shortest geometric paths. If *start_point* is not one of the points in path geometry, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

f) an *ST_Point* value *end_point*, which is an end point for acquiring the shortest geometric paths. If *end_point* is not one of the points in path geometry, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

3) The function *ST_ShortestUndPath* takes the following input parameters:

a) a value *paths_table_name* of type *DT1*, which has the name of a referenced table consisting of at least two columns:an INTEGER type column which has a unique number to identify a path as path identifier, and an *ST_Geometry* type column which has a spatial representation of a path with 1-dimensional geometry as path geometry. Let *S* be *paths_table_name* value. Let *V* be the character string that is the value of TRIM(BOTH ' ' FROM S). If *V* value does not conform to the Format and Syntax Rules of <table name> specified in ISO/IEC 9075-2, or the table specified by *paths_table_name* value does not exist in the system, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

b) a value *path_id_column* of type *DT2*, which has the name of the column for path identifier in the table specified by *paths_table_name*. If the column specified by *path_id_column* does not exist in the table specified by *paths_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

c) a value *geometry_column* of type *DT2*, which has the name of the column for path geometry in the table specified by *paths_table_name*. If the column specified by *geometry_column* does not exist in the table specified by *paths_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

d) an *ST_Point* value *start_point*, which is a start point for getting the shortest geometric paths. If *start_point* is not one of the points in path geometry, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

e) an *ST_Point* value *end_point*, which is an end point for acquiring the shortest geometric paths. If *end_point* is not one of the points in path geometry, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

4) The function *ST_ShortestUndPath* with input parameters (*paths_table_name*, *path_id_column*, *geometry_column*, *edge_weight_column*, *start_point*, *end_point*) returns the following value:

a) a table value consists of the following two columns:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

i) a column *shortest_path* of type ARRAY of INTEGER, which has a representation of shortest geometric paths using edge weight as geometric length from the start point to the end point by a sequence of path identifier defined by 2) a).

ii) a column *total_weight* of type DOUBLE PRECISION, which has the total geometric length of *shortest_path*.

5) The function *ST_ShortestUndPath* with input parameters (*paths_table_name*, *path_id_column*, *geometry_column*, *start_point*, *end_point*) returns the following value:

a) a table value consists of the following two columns:

i) a column *shortest_path* of type ARRAY of INTEGER, which has a representation of shortest geometric paths using length value of path geometry from the start point to the end point by a sequence of path identifier defined by 3) a).

ii) a column *total_weight* of type DOUBLE PRECISION, which has the total length of path geometries of *shortest_path*.

6) Case:

a) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_ShortestUndPath* is in the linear unit of measure identified by <linear unit>.

b) Otherwise, the value returned by *ST_ShortestUndPath* is in an implementation-defined unit of measure.

7) If the values in the column specified by *path_id_column* in the table specified by *paths_table_name* are not unique values, then an exception condition is raised: *SQL/MM Spatial exception – duplicate value*.

8) If the column specified by *geometry_column* contains a value of subtypes other than *ST_Curve*, then the row of the value shall be ignored.

9) If the table specified by *paths_table_name* has no rows, then the function *ST_ShortestUndPath* returns no rows.

10) If no contiguous paths is found from the start point to the end point in the table specified by *paths_table_name*, then the function *ST_ShortestUndPath* returns no rows.

11) If there are one or more shortest paths that have the same total length, then the function *ST_ShortestUndPath* returns the rows of those plural shortest paths.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

12.1.2 ST_ShortestDirPath Function

Purpose

Return a table containing IDs of directed shortest paths between two specified points of simple ST_Geometry value in a referenced table.

Definition

```
CREATE FUNCTION ST_ShortestDirPath
   (paths_table_name DT1,
    path_id_column DT2,
    geometry_column DT2,
    path_start_column DT2,
    edge_weight_column DT2,
    start_point ST_Point,
    end_point ST_Point)
   RETURNS TABLE
      (shortest_path INTEGER ARRAY[ST_MaxArrayElements],
       total_weight DOUBLE PRECISION)
   DETERMINISTIC
   BEGIN
      --
      -- See Description
      --
   END

CREATE FUNCTION ST_ShortestDirPath
   (paths_table_name DT1,
    path_id_column DT2,
    geometry_column DT2,
    path_start_column DT2,
    start_point ST_Point,
    end_point ST_Point)
   RETURNS TABLE
      (shortest_path INTEGER ARRAY[ST_MaxArrayElements],
       total_weight DOUBLE PRECISION )
   DETERMINISTIC
   BEGIN
      --
      -- See Description.
      --
   END
```

Definitional Rules

1) *ST_MaxArrayElements* is the implementation-defined maximum cardinality of an array for the number of geometric paths.
2) *DT1* is data type of variable-length character string with character set SQL_IDENTIFIER and implementation-defined maximum length.
3) *DT2* is data type of variable-length character string with character set SQL_IDENTIFIER and maximum length not less than 128 characters.

Description

1) The ST_ShortestDirPath functions are for public use.
2) The function *ST_ShortestDirPath* takes the following input parameters:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

a) a value *paths_table_name* of type *DT1*, which has the name of a referenced table consisting of at least four columns: an INTEGER type column which has a unique number to identify a path as path identifier, an *ST_Geometry* type column which has a spatial representation of a path with 1-dimensional geometry as path geometry, an *ST_Point* type column which is a start point of the path specified by path identifier and a DOUBLE PRECISION type column which has weight value on the path as edge weight. Let *S* be paths_table_name value. Let *V* be the character string that is the value of TRIM(BOTH ' ' FROM S). If *V* value does not conform to the Format and Syntax Rules of <table name> specified in ISO/IEC 9075-2, or the table specified by *paths_table_name* value does not exist in the system, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

b) a value *path_id_column* of type *DT2*, which has the name of the column for path identifier in the table specified by *paths_table_name*. If the column specified by *path_id_column* does not exist in the table specified by *paths_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

c) a value *geometry_column* of type *DT2*, which has the name of the column for path geometry in the table specified by *paths_table_name*. If the column specified by *geometry_column* does not exist in the table specified by *paths_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

d) a value *path_start_column* of type *DT2*, which has the name of the column for start point of path in the table specified by *paths_table_name*. If the column specified by *path_start_column* does not exist in the table specified by *paths_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

e) a value *edge_weight_column* of type *DT2*, which has the name of the column for edge weight in the table specified by *paths_table_name*. If the column specified by *edge_weight_column* does not exist in the table specified by *paths_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*..

f) an *ST_Point* value *start_point* , which is a start point for getting the shortest geometric paths. If the *ST_Point* value *start_point* is not one of the points in path geometry, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

g) an *ST_Point* value *end_point* , which is an end point for acquiring the shortest geometric paths. If the *ST_Point* value *end_point* is not one of the points in path geometry, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

3) The function *ST_ShortestDirPath* takes the following input parameters:

a) a value *paths_table_name* of type *DT1*, which has the name of a referenced table consisting of at least four columns: an INTEGER type column which has a unique number to identify a path as path identifier, an *ST_Geometry* type column which has a spatial representation of a path with 1-dimensional geometry as path geometry, and an *ST_Point* type column which is a start point of the path specified by path identifier. Let *S* be paths_table_name value. Let *V* be the character string that is the value of TRIM(BOTH ' ' FROM S). If *V* value does not conform to the Format and Syntax Rules of <table name> specified in ISO/IEC 9075-2, or the table specified by *paths_table_name* value does not exist in the system, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

b) a value *path_id_column* of type *DT2*, which has the name of the column for path identifier in the table specified by *paths_table_name*. If the column specified by *path_id_column* does not exist in the table specified by *paths_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

c) a value *geometry_column* of type *DT2*, which has the name of the column for path geometry in the table specified by *paths_table_name*. If the column specified by *geometry_column* does not exist in the table specified by *paths_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

d) a value *path_start_column* of type *DT2*, which has the name of the column for start point of path in the table specified by *paths_table_name*. If the column specified by *path_start_column* does not exist in the table specified by *paths_table_name*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

e) an *ST_Point* value *start_point*, which is a start point for getting the shortest geometric paths. If the *ST_Point* value *start_point* is not one of the points in path geometry, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

f) an *ST_Point* value *end_point*, which is an end point for acquiring the shortest geometric paths. If the *ST_Point* value *end_point* is not one of the points in path geometry, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

4) The function *ST_ShortestDirPath* with input parameters(*paths_table_name*, *path_id_column*, *geometry_column*, *path_start_column*, *edge_weight_column*, *start_point*, *end_point*) returns the following value:

a) a table value consists of the following two columns:

i) a column *shortest_path* of type ARRAY of INTEGER, which has a representation of shortest geometric paths using edge weight as geometric length from the start point to the end point by a sequence of path identifier defined by 2) a).

ii) a column *total_weight* of type DOUBLE PRECISION, which has the total geometric length of *shortest_path*.

5) The function *ST_ShortestDirPath* with input parameters(*paths_table_name*, *path_id_column*, *geometry_column*, *path_start_column*, *start_point*, *end_point*) returns the following value:

a) a table value consists of the following two columns:

i) a column *shortest_path* of type ARRAY of INTEGER, which has a representation of shortest geometric paths using length value of path geometry from the start point to the end point by a sequence of path identifier defined by 3) a).

ii) a column *total_weight* of type DOUBLE PRECISION, which has the total length value of path geometries of *shortest_path*.

6) Case:

a) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST_ShortestDirPath* is in the linear unit of measure identified by <linear unit>.

b) Otherwise, the value returned by *ST_ShortestDirPath* is in an implementation-defined unit of measure.

7) if the values in the column specified by *path_id_column* in the table specified by *paths_table_name* are not unique values, then an exception condition is raised: *SQL/MM Spatial exception – duplicate value*.

8) if the column specified by *geometry_column* contains a value of subtypes other than *ST_Curve*, then the row of the value shall be ignored.

9) if the table specified by *paths_table_name* has no row, then the function *ST_ShortestDirPath* returns no row.

10) if no contiguous paths found from the start point to the end point in the table specified by *paths_table_name*, then the function *ST_ShortestDirPath* returns no row.

11) if there are one or more shortest paths that have the same total length, then the function *ST_ShortestDirPath* returns the rows of those plural shortest paths.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

13 Spatial Reference System Type

13.1 ST_SpatialRefSys Type and Routines

13.1.1 ST_SpatialRefSys Type

Purpose

The ST_SpatialRefSys type encapsulates all aspects of spatial reference systems.

Definition

```
CREATE TYPE ST_SpatialRefSys
   AS (
      --
      -- See Description
      --
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_SpatialRefSys
      (awkt CHARACTER LARGE OBJECT(ST_MaxSRSAsText))
      RETURNS ST_SpatialRefSys
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_SpatialRefSys
      (ansrid INTEGER)
      RETURNS ST_SpatialRefSys
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_AsWKTSRS()
      RETURNS CHARACTER LARGE OBJECT(ST_MaxSRSAsText)
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_WKTSRSToSQL
      (awkt CHARACTER LARGE OBJECT(ST_MaxSRSAsText))
      RETURNS ST_SpatialRefSys
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   METHOD ST_SRID()
      RETURNS INTEGER
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_Equals
    (ansrs ST_SpatialRefSys)
    RETURNS INTEGER
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT
```

Definitional Rules

1) *ST_MaxSRSAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_SpatialRefSys* value.

Description

1) The *ST_SpatialRefSys* type provides for public use:
   a) a method *ST_SpatialRefSys(CHARACTER LARGE OBJECT)*,
   b) a method *ST_SpatialRefSys(INTEGER)*,
   c) a method *ST_AsWKTSRS()*,
   d) a method *ST_WKTSRSToSQL(CHARACTER LARGE OBJECT)*,
   e) a method *ST_SRID()*,
   f) a method *ST_Equals(ST_SpatialRefSys)*,
   g) an ordering function *ST_OrderingEquals(ST_SpatialRefSys, ST_SpatialRefSys)*,
   h) an SQL Transform group *ST_WellKnownText*.
2) The attribute definitions in the *ST_SpatialRefSys* type are implementation-dependent.

NOTE Implementations should refer to ISO 19111 as a model to follow for the implementation-dependent attribute definitions in the *ST_SpatialRefSys* type.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

13.1.2 ST_SpatialRefSys Methods

Purpose

Return a specified ST_SpatialRefSys value.

Definition

```
CREATE CONSTRUCTOR METHOD ST_SpatialRefSys
   (awkt CHARACTER LARGE OBJECT(ST_MaxSRSAsText))
   RETURNS ST_SpatialRefSys
   FOR ST_SpatialRefSys
   BEGIN
      --
      -- See Description
      --
   END

CREATE CONSTRUCTOR METHOD ST_SpatialRefSys
   (ansrid INTEGER)
   RETURNS ST_SpatialRefSys
   FOR ST_SpatialRefSys
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxSRSAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_SpatialRefSys* value.

Description

1) The method *ST_SpatialRefSys(CHARACTER LARGE OBJECT)* takes the following input parameters:
   a) a CHARACTER LARGE OBJECT value *awkt*.
2) The parameter *awkt* is the well-known text representation of an *ST_SpatialRefSys* value. If *awkt* is not producible in the BNF for <spatial reference system>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
3) The null-call type preserving SQL-invoked constructor method *ST_SpatialRefSys(CHARACTER LARGE OBJECT)* returns an *ST_SpatialRefSys* value representing the spatial reference system defined by *awkt*.
4) The method *ST_SpatialRefSys(INTEGER)* takes the following input parameters:
   a) an INTEGER value *ansrid*.
5) The null-call type preserving SQL-invoked constructor method *ST_SpatialRefSys(INTEGER)* returns an *ST_SpatialRefSys* value representing the spatial reference system defined by the spatial reference system identifier, *ansrid*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

13.1.3 ST_AsWKTSRS Method

Purpose

Return the well-known text representation of a spatial reference system.

Definition

```
CREATE METHOD ST_AsWKTSRS()
   RETURNS CHARACTER LARGE OBJECT(ST_MaxSRSAsText)
   FOR ST_SpatialRefSys
   BEGIN
      --
      -- See Description
      --
   END
```

Definitional Rules

1) *ST_MaxSRSAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_SpatialRefSys* value.

Description

1) The method *ST_AsWKTSRS()* has no input parameters.
2) The null-call method *ST_AsWKTSRS()* returns a CHARACTER LARGE OBJECT value containing the well-known text representation of SELF. Values shall be produced in the BNF for <spatial reference system>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

13.1.4 ST_WKTSRSToSQL Method

Purpose

Return the ST_SpatialRefSys value repsentedby the specified well-known text representation for a spatial reference system.

Definition

```
CREATE METHOD ST_WKTSRSToSQL
    (awkt CHARACTER LARGE OBJECT(ST_MaxSRSAsText))
    RETURNS ST_SpatialRefSys
    FOR ST_SpatialRefSys
    RETURN SELF.ST_SpatialRefSys(awkt)
```

Definitional Rules

1) *ST_MaxSRSAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_SpatialRefSys*.

Description

1) The method *ST_WKTSRSToSQL(CHARACTER LARGE OBJECT)* takes the following input parameters:

   a) a CHARACTER LARGE OBJECT *awkt*.

2) The parameter *awkt* is the well-known text representation of an *ST_SpatialRefSys* value. If *awkt* is not producible in the BNF for <spatial reference system>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

3) The null-call method *ST_WKTSRSToSQL(CHARACTER LARGE OBJECT)* returns an *ST_SpatialRefSys* value represented by *awkt*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

13.1.5 ST_SRID Method

Purpose

Return a spatial reference system identifier of an ST_SpatialRefSys value.

Definition

```
CREATE METHOD ST_SRID()
   RETURNS INTEGER
   FOR ST_SpatialRefSys
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_SRID()* has no input parameters.

2) The null-call method *ST_SRID()* returns an INTEGER value representing a unique identifier. This unique identifier is called the *spatial reference system identifier*. A spatial reference system identifier that is equal to 0 (zero) is implementation-defined. A spatial reference system identifier that is not equal to 0 (zero) is implementation-dependent.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

13.1.6 ST_Equals Method

Purpose

Test if two ST_SpatialRefSys values are equal.

Definition

```
CREATE METHOD ST_Equals
   (ansrs ST_SpatialRefSys)
   RETURNS INTEGER
   FOR ST_SpatialRefSys
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_Equals(ST_SpatialRefSys)* takes the following input parameters:
   a) an *ST_SpatialRefSys* value *ansrs*.
2) For the null-call method *ST_Equals(ST_SpatialRefSys)*:
   Case:
   a) If SELF is equal to *ansrs*, then return 1 (one).
   b) Otherwise, return 0 (zero).
3) The method *ST_Equals(ST_SpatialRefSys)* is implementation-defined.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

13.1.7 ST_OrderingEquals Function

Purpose

Define the equals ordering for the ST_SpatialRefSys type.

Definition

```
CREATE FUNCTION ST_OrderingEquals
   (ansrs ST_SpatialRefSys,
    anothersrs ST_SpatialRefSys)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN
      CASE
         WHEN ansrs.ST_Equals(anothersrs) = 1 THEN
            0
         ELSE
            1
      END

CREATE ORDERING FOR ST_SpatialRefSys
   EQUALS ONLY BY RELATIVE
      WITH FUNCTION ST_OrderingEquals(ST_SpatialRefSys, ST_SpatialRefSys)
```

Description

1) The function *ST_OrderingEquals(ST_SpatialRefSys, ST_SpatialRefSys)* takes the following input parameters:
   a) an *ST_SpatialRefSys* value *ansrs*,
   b) an *ST_SpatialRefSys* value *anothersrs*.
2) For the null-call function *ST_OrderingEquals(ST_SpatialRefSys, ST_SpatialRefSys)*:

   Case:
   a) If the value expression *ansrs.ST_Equals(anothersrs)* is 1 (one), then return 0 (zero).
   b) Otherwise, return 1 (one).
3) Use the function *ST_OrderingEquals(ST_SpatialRefSys, ST_SpatialRefSys)* to define ordering for the *ST_SpatialRefSys* type.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

13.1.8 ST_WellKnownText SQL Transform Group

Purpose

Define SQL transform functions for the ST_SpatialRefSys type.

Definition

```
CREATE TRANSFORM FOR ST_SpatialRefSys
   ST_WellKnownText
      (TO SQL WITH METHOD ST_WKTSRSToSQL
         (CHARACTER LARGE OBJECT(ST_MaxSRSAsText)),
       FROM SQL WITH METHOD ST_AsWKTSRS())
```

Definitional Rules

1) *ST_MaxSRSAsText* is the implementation-defined maximum length of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_SpatialRefSys*.

Description

1) Use the method *ST_WKTSRSToSQL(CHARACTER LARGE OBJECT)* and the method *ST_AsWKTSRS()* to define the transform group *ST_WellKnownText*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

13.1.9 <spatial reference system>

Purpose

This subclause contains the definition of <spatial reference system>.

Description

1) The well-known text representation of an *ST_SpatialRefSys* value is defined by the following BNF for <spatial reference system>.

```
<spatial reference system> ::=
     <projected cs>
   | <geographic cs>
   | <geocentric cs>

<projected cs> ::=
     PROJCS <left delimiter>
        <double quote> <name> <double quote> <comma>
        <geographic cs> <comma>
        <projection> <comma>
        { <parameter> <comma> }...
        <linear unit>
        <right delimiter>

<geographic cs> ::=
     GEOGCS <left delimiter>
        <double quote> <name> <double quote> <comma>
        <datum> <comma>
        <prime meridian> <comma>
        <angular unit>
        [ <linear unit> ]
        <right delimiter>

<geocentric cs> ::=
     GEOCCS <left delimiter>
        <double quote> <name> <double quote> <comma>
        <datum> <comma>
        <prime meridian> <comma>
        <linear unit>
        <right delimiter>

<projection> ::=
     PROJECTION <left delimiter>
        <double quote> <projection name> <double quote>
        <right delimiter>

<parameter> ::=
     PARAMETER <left delimiter>
        <double quote> <parameter name> <double quote> <comma>
        <value>
        <right delimiter>

<value> ::= <number>

<datum> ::=
     DATUM <left delimiter>
        <double quote> <datum name> <double quote> <comma>
        <spheroid>
        <right delimiter>
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
<spheroid> ::=
     SPHEROID <left delimiter>
        <double quote> <spheroid name> <double quote> <comma>
        <semi-major axis> <comma>
        <inverse flattening>
        <right delimiter>

<semi-major axis> ::= <number>

<inverse flattening> ::= <number>

<prime meridian> ::=
     PRIMEM <left delimiter>
        <double quote> <prime meridian name> <double quote> <comma>
        <longitude>
        <right delimiter>

<longitude> ::= <number>

<angular unit> ::= <unit>

<linear unit> ::= <unit>

<unit> ::=
     UNIT <left delimiter>
        <double quote> <unit name> <double quote> <comma>
        <conversion factor>
        <right delimiter>

<conversion factor> ::= <number>

<datum name> ::= <letters>

<parameter name> ::= <letters>

<prime meridian name> ::= <letters>

<projection name> ::= <letters>

<spheroid name> ::= <letters>

<unit name> ::= <letters>

<name> ::= <letters>

<letters> ::= <letter>...

<letter> ::=
     <simple Latin letter>
   | <digit>
   | <special>

<special> ::=
     <left paren>
   | <right paren>
   | <minus>
   | <underscore>
   | <period>
   | <quote>
   | <space>

<number> ::=
     <exact numeric literal>
   | <approximate numeric literal>

<left delimiter> ::=
     <left paren>
   | <left bracket>
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
<right delimiter> ::=
     <right paren>
   | <right bracket>

<exact numeric literal> ::=
     !! See Subclause 5.3, "<literal>", in Part 2 of ISO/IEC 9075

<approximate numeric literal> ::=
     !! See Subclause 5.3, "<literal>", in Part 2 of ISO/IEC 9075

<simple Latin letter> ::=
     !! See Subclause 5.1, "<SQL terminal character>", in
        Part 2 of ISO/IEC 9075

<digit> ::=
     !! See Subclause 5.1, "<SQL terminal character>", in
        Part 2 of ISO/IEC 9075

<double quote> ::=
     !! See Subclause 5.1, "<SQL terminal character>", in
        Part 2 of ISO/IEC 9075

<comma> ::=
     !! See Subclause 5.1, "<SQL terminal character>", in
        Part 2 of ISO/IEC 9075

<left bracket> ::=
     !! See Subclause 5.1, "<SQL terminal character>", in
        Part 2 of ISO/IEC 9075

<right bracket> ::=
     !! See Subclause 5.1, "<SQL terminal character>", in
        Part 2 of ISO/IEC 9075

<left paren> ::=
     !! See Subclause 5.1, "<SQL terminal character>", in
        Part 2 of ISO/IEC 9075

<right paren> ::=
     !! See Subclause 5.1, "<SQL terminal character>", in
        Part 2 of ISO/IEC 9075

<minus> ::=
     !! See Subclause 5.1, "<SQL terminal character>", in
        Part 2 of ISO/IEC 9075

<underscore> ::=
     !! See Subclause 5.1, "<SQL terminal character>", in
        Part 2 of ISO/IEC 9075

<period> ::=
     !! See Subclause 5.1, "<SQL terminal character>", in
        Part 2 of ISO/IEC 9075

<quote> ::=
     !! See Subclause 5.1, "<SQL terminal character>", in
        Part 2 of ISO/IEC 9075

<space> ::=
     !! See Subclause 5.1, "<SQL terminal character>", in
        Part 2 of ISO/IEC 9075
```

a) Case:

i) If <left paren> is used as a <left delimiter>, then <right paren> shall be used as the corresponding <right delimiter>.

ii) If <left bracket> is used as a <left delimiter>, then <right bracket> shall be used as the corresponding <right delimiter>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

b) A <spatial reference system> is a geographic (latitude-longitude), a projected (X, Y), or a geocentric (X, Y, Z) coordinate system.

The coordinate reference system support for *ST_Geometry* values with m coordinate values is implementation-defined.

The coordinate system is composed of several items. Each item has a keyword in upper case followed by the defining, comma-delimited, parameters of the item in brackets. Some items are composed of other items in a nested structure.

c) The list of keywords are DATUM, GEOCCS, GEOGCS, PROJCS, PARAMETER, PRIMEM, PROJECTION, SPHEROID, and UNIT.

d) <projected cs> is a projected coordinate system. <projection name> is an implementation-defined name of a parameter.

e) <geographic cs> is a geographic coordinate system.

f) <geocentric cs> is a geocentric coordinate system.

g) <name> is the name of the coordinate system.

h) <projection> is the projection system of a <projected cs>.

i) <parameter> is a parameter of the <projection> to define the <projected cs>. <parameter name> is an implementation-defined name of a parameter.

j) <datum> is the horizontal datum used by the <geographic cs> or the <geocentric cs>. <datum name> is an implementation-defined name of a datum.

k) <spheroid> is the spheroid of a datum defined by a semi-major axis, <semi-major axis> , and an inverse flattening, <inverse flattening>. <spheroid>s are implementation-defined.

l) <prime meridian> is the longitude used by the <geographic cs> or the <geocentric cs>. The <semi-major axis> is greater than 0 (zero) and it is measured in meters. <prime meridian>s are implementation-defined.

m) <angular unit> specifies an angular unit and <linear unit> defines a linear unit. <conversion factor> specifies the number of meters for a linear unit or the number of radians for an angular unit per unit. <conversion factor> is greater than zero. <angular unit>s and <linear unit>s are implementation-defined.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

14 Angle and Direction Types

14.1 ST_Angle Type and Routines

14.1.1 ST_Angle Type

Purpose

The ST_Angle type is used to express circular measurement or to measure the degree of separation of two intersecting lines.

Definition

```
CREATE TYPE ST_Angle
   AS (
      ST_PrivateRadians DOUBLE PRECISION DEFAULT NULL
   )
   INSTANTIABLE
   NOT FINAL

   CONSTRUCTOR METHOD ST_Angle
      (angle DOUBLE PRECISION)
      RETURNS ST_Angle
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_Angle
      (units CHARACTER(1),
       angle DOUBLE PRECISION)
      RETURNS ST_Angle
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_Angle
      (degrees INTEGER, minutes DOUBLE PRECISION)
      RETURNS ST_Angle
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

   CONSTRUCTOR METHOD ST_Angle
      (degrees INTEGER, minutes INTEGER, seconds DOUBLE PRECISION)
      RETURNS ST_Angle
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CONSTRUCTOR METHOD ST_Angle
   (atpoint ST_Point, apoint ST_Point, anotherpoint ST_Point)
   RETURNS ST_Angle
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Angle
   (adirection ST_Direction,
    anotherdirection ST_Direction)
   RETURNS ST_Angle
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Angle
   (aline ST_LineString,
    anotherline ST_LineString)
   RETURNS ST_Angle
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Angle
   (awkt CHARACTER VARYING(ST_MaxAngleAsText))
   RETURNS ST_Angle
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Radians()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Radians
   (radians DOUBLE PRECISION)
   RETURNS ST_Angle
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_Degrees()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_Degrees
   (degrees DOUBLE PRECISION)
   RETURNS ST_Angle
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_DegreeComponent()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_MinuteComponent()
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_SecondComponent()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_String()
   RETURNS CHARACTER VARYING(ST_MaxAngleString)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_String
   (numdecdigits INTEGER)
   RETURNS CHARACTER VARYING(ST_MaxAngleString)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_String
   (dms CHARACTER VARYING(ST_MaxAngleString))
   RETURNS ST_Angle
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_Gradians()
   RETURNS DOUBLE PRECISION
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
METHOD ST_Gradians
   (gradians DOUBLE PRECISION)
   RETURNS ST_Angle
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   CALLED ON NULL INPUT,

METHOD ST_Add
   (anangle ST_Angle)
   RETURNS ST_Angle
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Subtract
   (anangle ST_Angle)
   RETURNS ST_Angle
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Multiply
   (afactor DOUBLE PRECISION)
   RETURNS ST_Angle
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_Divide
   (adivisor DOUBLE PRECISION)
   RETURNS ST_Angle
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

METHOD ST_AsText()
   RETURNS CHARACTER VARYING(ST_MaxAngleAsText)
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

Definitional Rules

1) *ST_MaxAngleString* is the implementation-defined maximum length of the CHARACTER VARYING used for the character string representation of the *ST_PrivateRadians* attribute of an *ST_Angle* value.

2) *ST_MaxAngleAsText* is the implementation-defined maximum length of the CHARACTER VARYING used for the well-known tet representation of an *ST_Angle* value.

3) The attribute *ST_PrivateRadians* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST_PrivateRadians*.

© ISO/IEC 2011 – All rights reserved

Description

1) The *ST_Angle* type provides for public use:
   a) a method *ST_Angle(DOUBLE PRECISION)*,
   b) a method *ST_Angle(CHARACTER, DOUBLE PRECISION)*,
   c) a method *ST_Angle(INTEGER, DOUBLE PRECISION)*,
   d) a method *ST_Angle(INTEGER, INTEGER, DOUBLE PRECISION)*,
   e) a method *ST_Angle(ST_Point, ST_Point, ST_Point)*,
   f) a method *ST_Angle(ST_Direction, ST_Direction)*,
   g) a method *ST_Angle(ST_LineString, ST_LineString)*,
   h) a method *ST_Angle(CHARACTER VARYING)*,
   i) a method *ST_Radians()*,
   j) a method *ST_Radians(DOUBLE PRECISION)*,
   k) a method *ST_Degrees()*,
   l) a method *ST_Degrees(DOUBLE PRECISION)*,
   m) a method *ST_DegreeComponent()*,
   n) a method *ST_MinuteComponent()*,
   o) a method *ST_SecondComponent()*,
   p) a method *ST_String()*,
   q) a method *ST_String(INTEGER)*,
   r) a method *ST_String(CHARACTER VARYING)*,
   s) a method *ST_Gradians()*,
   t) a method *ST_Gradians(DOUBLE PRECISION)*,
   u) a method *ST_Add(ST_Angle)*,
   v) a method *ST_Subtract(ST_Angle)*,
   w) a method *ST_Multiply(DOUBLE PRECISION)*,
   x) a method *ST_Divide(DOUBLE PRECISION)*,
   y) a method *ST_AsText()*,
   z) an ordering function *ST_OrderingCompare(ST_Angle, ST_Angle)*,
   aa) an SQL Transform group *ST_WellKnownText*,
   ab) an SQL Transform group *ST_WellKnownBinary*.
2) The attribute *ST_PrivateRadians* contains the measurement of the angle expressed in radians.
3) The direction of an angle is not specified.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

14.1.2 ST_Angle Methods

Purpose

Return a specified ST_Angle value.

Definition

```
CREATE CONSTRUCTOR METHOD ST_Angle
   (angle DOUBLE PRECISION)
   RETURNS ST_Angle
   FOR ST_Angle
   RETURN NEW ST_Angle().ST_PrivateRadians(angle)

CREATE CONSTRUCTOR METHOD ST_Angle
   (units CHARACTER(1),
    angle DOUBLE PRECISION)
   RETURNS ST_Angle
   FOR ST_Angle
   BEGIN
      IF (units <> 'R') AND
         (units <> 'D') AND
         (units <> 'G') THEN
         SIGNAL SQLSTATE '2FF02'
            SET MESSAGE_TEXT = 'invalid argument: invalid angle units';
      ELSE
         RETURN NEW ST_Angle().ST_PrivateRadians(
            CASE
               WHEN units = 'R' THEN angle
               WHEN units = 'D' THEN (ST_ApproximatePi * angle) / 180
               WHEN units = 'G' THEN (ST_ApproximatePi * angle) / 200
            END);
      END IF;
   END

CREATE CONSTRUCTOR METHOD ST_Angle
   (degrees INTEGER,
    minutes DOUBLE PRECISION)
   RETURNS ST_Angle
   FOR ST_Angle
   BEGIN
      IF (minutes < 0) OR
         (minutes >= 60) THEN
         SIGNAL SQLSTATE '2FF02'
            SET MESSAGE_TEXT = 'invalid argument: minutes out of range';
      ELSE
         IF degrees >= 0 THEN
            RETURN NEW ST_Angle().ST_PrivateRadians(
               (ST_ApproximatePi * (degrees+(minutes/60))) / 180);
         ELSE
            RETURN NEW ST_Angle().ST_PrivateRadians(
               (ST_ApproximatePi * (degrees-(minutes/60))) / 180);
         END IF;
      END IF;
   END
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CREATE CONSTRUCTOR METHOD ST_Angle
   (degrees INTEGER,
    minutes INTEGER,
    seconds DOUBLE PRECISION)
   RETURNS ST_Angle
   FOR ST_Angle
   BEGIN
      IF (minutes < 0) OR
         (minutes >= 60) THEN
         SIGNAL SQLSTATE '2FF02'
            SET MESSAGE_TEXT = 'invalid argument: minutes out of range';
      ELSEIF(seconds < 0) OR
            (seconds >= 60) THEN
         SIGNAL SQLSTATE '2FF02'
            SET MESSAGE_TEXT = 'invalid argument: seconds out of range';
      ELSE
         IF degrees >= 0 THEN
            RETURN NEW ST_Angle().ST_PrivateRadians(
               (ST_ApproximatePi *
                  (degrees+(minutes/60)+(seconds/3600))) / 180);
         ELSE
            RETURN NEW ST_Angle().ST_PrivateRadians(
               (ST_ApproximatePi *
                  (degrees-(minutes/60)-(seconds/3600))) / 180);
         END IF;
      END IF;
   END
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CREATE CONSTRUCTOR METHOD ST_Angle
   (atpoint ST_Point,
    apoint ST_Point,
    anotherpoint ST_Point)
   RETURNS ST_Angle
   FOR ST_Angle
   BEGIN
      -- check atpoint
      IF atpoint.ST_IsEmpty() = 1 THEN
         SIGNAL SQLSTATE '2FF17'
            SET MESSAGE_TEXT = 'empty point value';
      END IF;
      IF atpoint.ST_IsValid() = 0 THEN
         SIGNAL SQLSTATE '2FF18'
            SET MESSAGE_TEXT = 'point value not well formed';
      END IF;

      -- check apoint
      IF apoint.ST_IsEmpty() = 1 THEN
         SIGNAL SQLSTATE '2FF17'
            SET MESSAGE_TEXT = 'empty point value';
      END IF;
      IF apoint.ST_IsValid() = 0 THEN
         SIGNAL SQLSTATE '2FF18'
            SET MESSAGE_TEXT = 'point value not well formed';
      END IF;

      -- check anotherpoint
      IF anotherpoint.ST_IsEmpty() = 1 THEN
         SIGNAL SQLSTATE '2FF17'
            SET MESSAGE_TEXT = 'empty point value';
      END IF;
      IF anotherpoint.ST_IsValid() = 0 THEN
         SIGNAL SQLSTATE '2FF18'
            SET MESSAGE_TEXT = 'point value not well formed';
      END IF;

      -- check if points are coincident
      IF atpoint.ST_Equals(apoint) = 1 THEN
         -- points are co-located so direction is not defined
         SIGNAL SQLSTATE '2FF19'
            SET MESSAGE_TEXT = 'points are equal';
      END IF;

      IF atpoint.ST_Equals(anotherpoint) = 1 THEN
         -- points are co-located so direction is not defined
         SIGNAL SQLSTATE '2FF19'
            SET MESSAGE_TEXT = 'points are equal';
      END IF;
      --
      -- See Description
      --
   END
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

```
CREATE CONSTRUCTOR METHOD ST_Angle
   (adirection ST_Direction,
    anotherdirection ST_Direction)
   RETURNS ST_Angle
   FOR ST_Angle
   RETURN NEW ST_Angle(
      CASE
         WHEN (anotherdirection.ST_Radians()-adirection.ST_Radians()) <
            (adirection.ST_Radians()-anotherdirection.ST_Radians()) THEN
            anotherdirection.ST_Radians()-adirection.ST_Radians()
         ELSE
            adirection.ST_Radians()-anotherdirection.ST_Radians()
      END)
CREATE CONSTRUCTOR METHOD ST_Angle
   (aline ST_LineString,
    anotherline ST_LineString)
   RETURNS ST_Angle
   FOR ST_Angle
   BEGIN
      -- check if aline is valid
      IF aline.ST_NumPoints() <> 2 THEN
         -- not a line
         SIGNAL SQLSTATE '2FF20'
            SET MESSAGE_TEXT = 'linestring is not a line';
      ELSEIF aline.ST_IsClosed() = 1 THEN
         -- not a line
         SIGNAL SQLSTATE '2FF21'
            SET MESSAGE_TEXT = 'degenerate line has no direction';
      END IF;

      -- check if anotherline is valid
      IF anotherline.ST_NumPoints() <> 2 THEN
         -- not a line
         SIGNAL SQLSTATE '2FF20'
            SET MESSAGE_TEXT = 'linestring is not a line';
      ELSEIF anotherline.ST_IsClosed() = 1 THEN
         -- not a line
         SIGNAL SQLSTATE '2FF21'
            SET MESSAGE_TEXT = 'degenerate line has no direction';
      END IF;
      RETURN NEW ST_Angle(
         NEW ST_Direction(aline), NEW ST_Direction(anotherline));
   END
CREATE CONSTRUCTOR METHOD ST_Angle
   (awkt CHARACTER VARYING(ST_MaxAngleAsText))
   RETURNS ST_Angle
   FOR ST_Angle
   BEGIN
      --
      -- See Description
      --
   END
```

Description

1) The method *ST_Angle(DOUBLE PRECISION)* takes the following input parameters:

   a) a DOUBLE PRECISION value *angle*.

2) The null-call type preserving SQL-invoked constructor method *ST_Angle(DOUBLE PRECISION)* returns an *ST_Angle* value with the attribute *ST_PrivateRadians* set to *angle*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

3) The method *ST_Angle(CHARACTER, DOUBLE PRECISION)* takes the following input parameters:

   a) a CHARACTER value *units*,

   b) a DOUBLE PRECISION value *angle*.

4) Let *IND* be the CHARACTER value specified by *units*.

5) For the null-call type preserving SQL-invoked constructor method *ST_Angle(CHARACTER, DOUBLE PRECISION)* return an *ST_Angle* value with:

   Case:

   a) If IND is not 'R', 'D' or 'G', then an exception condition is raised: *SQL/MM Spatial exception – invalid argument: invalid angle units*.

   b) Otherwise, return an *ST_Angle* value with:

      Case:

      i) If *IND* is equal to 'R', then the attribute *ST_PrivateRadians* set to *angle*.

      i) If *IND* is equal to 'D', then the attribute *ST_PrivateRadians* set to *(pi * angle) / 180*.

      iii) If *IND* is equal to 'G', then the attribute *ST_PrivateRadians* set to *(pi * angle) / 200*.

6) The method *ST_Angle(INTEGER, DOUBLE PRECISION)* takes the following input parameters:

   a) an INTEGER value *degrees*,

   b) a DOUBLE PRECISION value *minutes*.

7) Let *D* be the INTEGER value specified by *degrees*, and *M* the DOUBLE PRECISION value specified by *minutes*.

8) If *M* is less than 0 or *M* is greater than or equal to 60, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument: minutes out of range*.

9) For the null-call type preserving SQL-invoked constructor method *ST_Angle(INTEGER, DOUBLE PRECISION)* return an *ST_Angle* value with:

   Case:

   a) If *D* >= 0 (zero), then the attribute *ST_PrivateRadians* set to the angle measurement represented by the combination of *degrees* and *minutes,* expressed in radians, equal to:

      *(pi * (D + M / 60)) / 180*

   b) Otherwise, the attribute *ST_PrivateRadians* set to the angle measurement represented by the combination of *degrees* and *minutes*, expressed in radians, equal to:

      *(pi * (D - M / 60)) / 180*

10) The method *ST_Angle(INTEGER, INTEGER, DOUBLE PRECISION)* takes the following input parameters:

   a) an INTEGER value *degrees*,

   b) an INTEGER value *minutes*,

   c) a DOUBLE PRECISION value *seconds*.

11) Let *D* be the INTEGER value specified by *degrees*, *M* the INTEGER value specified by *minutes*, and *S* the DOUBLE PRECISION value specified by *seconds*.

12) If *M* is less than 0 or *M* is greater than or equal to 60, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument: minutes out of range*.

13) If *S* is less than 0 or *S* is greater than or equal to 60, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument: seconds out of range*.

14) For the null-call type preserving SQL-invoked constructor method *ST_Angle(INTEGER, INTEGER, DOUBLE PRECISION)* return an *ST_Angle* value with:

   Case:

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

a) If *D* >= 0 (zero), then the attribute *ST_PrivateRadians* set to the angle measurement represented by the combination of *degrees*, *minutes*, and *seconds*, expressed in radians, equal to:

*(pi * (D + M / 60 + S / 3600)) / 180*

b) Otherwise, the attribute *ST_PrivateRadians* set to the angle measurement represented by the combination of *degrees*, *minutes*, and *seconds*, expressed in radians, equal to:

*(pi * (D - M / 60 - S / 3600)) / 180*

15) The method *ST_Angle(ST_Point, ST_Point, ST_Point)* takes the following input parameters:

a) an *ST_Point* value *atpoint*,

b) an *ST_Point* value *apoint*,

c) an *ST_Point* value *anotherpoint*.

16) Let *D1* be the direction from *atpoint* to *apoint* and let *D2* be the direction from *atpoint* to *anotherpoint*.

17) For the null-call type preserving SQL-invoked constructor method *ST_Angle(ST_Point, ST_Point, ST_Point)*:

Case:

a) If *atpoint* is the empty set, then an exception condition is raised: *SQL/MM Spatial exception – empty point value*.

b) If *atpoint* is not well formed, then an exception condition is raised: *SQL/MM Spatial exception – point value not well formed*.

c) If *apoint* is the empty set, then an exception condition is raised: *SQL/MM Spatial exception – empty point value*.

d) If *apoint* is not well formed, then an exception condition is raised: *SQL/MM Spatial exception – point value not well formed*.

e) If *anotherpoint* is the empty set, then an exception condition is raised: *SQL/MM Spatial exception – empty point value*.

f) If *anotherpoint* is not well formed, then an exception condition is raised: *SQL/MM Spatial exception – point value not well formed*.

g) If *atpoint* is equal to *apoint*, then an exception condition is raised: *SQL/MM Spatial exception – points are equal*.

h) If *atpoint* is equal to *anotherpoint*, then an exception condition is raised: *SQL/MM Spatial exception – points are equal*.

i) Otherwise, return an ST_Angle value with the attribute ST_PrivateRadians set to the lesser of:

i) the clockwise angle measured in radians from *D1* to *D2*, or

ii) the clockwise angle measured in radians from *D2* to *D1*.

18) The method *ST_Angle(ST_Direction, ST_Direction)* takes the following input parameters:

a) an *ST_Direction* value *adirection*,

b) an *ST_Direction* value *anotherdirection*.

19) For the null-call type preserving SQL-invoked constructor method *ST_Angle(ST_Direction, ST_Direction)* return an *ST_Angle* value with the attribute *ST_PrivateRadians* set to the lesser of:

a) the clockwise angle measured in radians from *adirection* to *anotherdirection*, or

b) the clockwise angle measured in radians from *anotherdirection* to *adirection*.

20) The method *ST_Angle(ST_LineString, ST_LineString)* takes the following input parameters:

a) an *ST_LineString* value *aline*,

b) an *ST_LineString* value *anotherline*,

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

21) Let *P1* be the point value returned by *aline.ST_StartPoint()* and *P2* be the point value returned by *aline.ST_EndPoint()*. Let *D1* be the direction from *P1* to *P2*.

22) Let *P3* be the point value returned by *anotherline.ST_StartPoint()* and *P4* be the point value returned by *anotherline.ST_EndPoint()* Then let *D2* be the direction from *P3* to *P4*.

23) For the null-call type preserving SQL-invoked constructor method *ST_Angle(ST_LineString, ST_LineString)*:

    Case:

    a) If *aline* is not a line, then an exception condition is raised: *SQL/MM Spatial exception – linestring is not a line*.

    b) If *aline* is closed, then an exception condition is raised: *SQL/MM Spatial exception – degenerate line has no direction*.

    c) If *anotherline* is not a line, then an exception condition is raised: *SQL/MM Spatial exception – linestring is not a line*.

    d) If *anotherline* is closed, then an exception condition is raised: *SQL/MM Spatial exception – degenerate line has no direction*.

    e) Otherwise, return an *ST_Angle* value with the attribute *ST_PrivateRadians* set to the lesser of:

        i) the clockwise angle measured in radians from *D1* to *D2*, or

        ii) the clockwise angle measured in radians from *D2* to *D1*.

24) The method *ST_Angle(CHARACTER VARYING)* takes the following input parameters:

    a) a CHARACTER VARYING value *awkt*.

25) The well-known tet representation of an *ST_Angle* value is defined by the following BNF for <angle text representation>:

```
<angle text representation> ::=
     ANGLE <angle text>

<angle text> ::=
     DEGREES <left paren> <degrees> <right paren>
   | GRADIANS <left paren> <gradians> <right paren>
   | RADIANS <left paren> <radians> <right paren>

<degrees> ::=
     <number>

<gradians> ::=
     <number>

<radians> ::=
     <number>
```

    a) <angle text representation> is the well-known tet representation for an *ST_Angle* value that is produced by <angle text>.

    b) <angle text> produces the *ST_Angle* value from <degrees>, <gradians>, or <radians>

    c) Case:

        i) Let *DEGREES* be the DOUBLE PRECISION value specified by <degrees>. <degrees> produces an *ST_Angle* value as the result of the value expression: *NEW ST_Angle*('D', *DEGREES*).

        ii) Let *GRADIANS* be the DOUBLE PRECISION value specified by <gradians>. <gradians> produces an *ST_Angle* value as the result of the value expression: *NEW ST_Angle*('G', *GRADIANS*).

        iii) Let *RADIANS* be the DOUBLE PRECISION value specified by <radians>. <radians> produces an *ST_Angle* value as the result of the value expression: *NEW ST_Angle*('R', *RADIANS*).

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

26) The parameter *awkt* is the well-known tet representation of an *ST_Angle* value. If *awkt* is not producible in the BNF for <angle text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

27) The null-call type-preserving SQL-invoked constructor method *ST_Angle(CHARACTER VARYING)* returns an *ST_Angle* value represented by *awkt*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

14.1.3 ST_Radians Methods

Purpose

Observe and mutate the radians attribute of an ST_Angle value.

Definition

```
CREATE METHOD ST_Radians()
   RETURNS DOUBLE PRECISION
   FOR ST_Angle
   RETURN SELF.ST_PrivateRadians

CREATE METHOD ST_Radians
   (radians DOUBLE PRECISION)
   RETURNS ST_Angle
   FOR ST_Angle
   BEGIN
      IF radians IS NULL THEN
         SIGNAL SQLSTATE '2FF03'
            SET MESSAGE_TEXT = 'null argument';
      ELSE
         RETURN
            CASE
               WHEN SELF IS NULL THEN
                  NULL
               ELSE
                  SELF.ST_PrivateRadians(radians)
            END;
      END IF;
   END
```

Description

1) The method *ST_Radians()* has no input parameters.
2) The null-call method *ST_Radians()* returns the value of the *ST_PrivateRadians* attribute.
3) The method *ST_Radians(DOUBLE PRECISION)* takes the following input parameters:
   a) a DOUBLE PRECISION value *radians*.
4) For the type preserving method *ST_Radians(DOUBLE PRECISION)*:

   Case:

   a) If *radians* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If SELF is the null value, then return the null value.
   c) Otherwise, return the result of the value expression: *SELF.ST_PrivateRadians(radians)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

14.1.4 ST_Degrees Methods

Purpose

Observe and mutate the radians attribute of an ST_Angle value using decimal degrees.

Definition

```
CREATE METHOD ST_Degrees()
   RETURNS DOUBLE PRECISION
   FOR ST_Angle
   RETURN (SELF.ST_PrivateRadians / ST_ApproximatePi) * 180

CREATE METHOD ST_Degrees
   (degrees DOUBLE PRECISION)
   RETURNS ST_Angle
   FOR ST_Angle
   BEGIN
      IF degrees IS NULL THEN
         SIGNAL SQLSTATE '2FF03'
            SET MESSAGE_TEXT = 'null argument';
      ELSE
         RETURN
            CASE
               WHEN SELF IS NULL THEN
                  NULL
               ELSE
                  SELF.ST_PrivateRadians(
                     (ST_ApproximatePi * degrees) / 180)
            END;
      END IF;
   END
```

Definitional Rules

1) *ST_ApproximatePi* is the implementation-defined meta-variable representing pi.

Description

1) The method *ST_Degrees()* has no input parameters.
2) The null-call method *ST_Degrees()* returns the value of the *ST_PrivateRadians* attribute converted to decimal degrees.
3) The method *ST_Degrees(DOUBLE PRECISION)* takes the following input parameters:
   a) a DOUBLE PRECISION value *degrees*.
4) For the type preserving method *ST_Degrees(DOUBLE PRECISION)*:

   Case:

   a) If *degrees* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
   b) If SELF is the null value, then return the null value.
   c) Otherwise, return the result of the value expression: *SELF.ST_PrivateRadians((ST_ApproximatePi * degrees) / 180)*.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

14.1.5 ST_DegreeComponent Method

Purpose

Observe the INTEGER degrees part of the degrees, minutes, and seconds representation of the radians attribute of an ST_Angle value.

Definition

```
CREATE METHOD ST_DegreeComponent()
   RETURNS INTEGER
   FOR ST_Angle
   RETURN FLOOR(SELF.ST_Degrees())
```

Description

1) The method *ST_DegreeComponent()* has no input parameters.
2) The null-call method *ST_DegreeComponent()* returns the INTEGER degree part of the degrees, minutes, and seconds representation of the value of the *ST_PrivateRadians* attribute.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved

14.1.6 ST_MinuteComponent Method

Purpose

Observe the INTEGER minutes part of the degrees, minutes, and seconds representation of the radians attribute of an ST_Angle value.

Definition

```
CREATE METHOD ST_MinuteComponent()
   RETURNS INTEGER
   FOR ST_Angle
   RETURN FLOOR(ABS((SELF.ST_Degrees()-SELF.ST_DegreeComponent())*60))
```

Description

1) The method *ST_MinuteComponent()* has no input parameters.
2) The null-call method *ST_MinuteComponent()* returns the INTEGER minutes part of the degrees, minutes, and seconds representation of the value of the *ST_PrivateRadians* attribute.

IECNORM.COM : Click to view the full PDF of ISO/IEC 13249-3:2011

© ISO/IEC 2011 – All rights reserved