BS ISO/IEC 13249-3:2011

BSI Standards Publication

Information technology — Database languages — SQL multimedia and application packages

Part 3: Spatial

bsi.

...making excellence a habit.™

Document Sharing Hub

Full document details: BS ISO/IEC 13249-3:2011

National foreword

This British Standard is the UK implementation of ISO/IEC 13249-3:2011. It supersedes BS ISO/IEC 13249-3:2006 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/40, Data management and interchange.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© BSI 2011

ISBN 978 0 580 66744 2

ICS 35.060

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 June 2011.

Amendments issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 13249-3:2011

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

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

Documents
Document Sharing Hub

© ISO/IEC 2011

Full document details: BS ISO/IEC 13249-3:2011

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.

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

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 13249-3:2011

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

Documents
Document Sharing Hub

© ISO/IEC 2011 – All rights reserved

Full document details: BS ISO/IEC 13249-3:2011

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

Documents
Document Sharing Hub

© ISO/IEC 2011 – All rights reserved

Full document details: BS ISO/IEC 13249-3:2011

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

Documents
Document Sharing Hub

© ISO/IEC 2011 – All rights reserved

Full document details: BS ISO/IEC 13249-3:2011

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_Triangle Type 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

Documents
Document Sharing Hub

© ISO/IEC 2011 – All rights reserved

Full document details: BS ISO/IEC 13249-3:2011

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

Documents
Document Sharing Hub

© ISO/IEC 2011 – All rights reserved

Full document details: BS ISO/IEC 13249-3:2011

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

Documents
Document Sharing Hub

© ISO/IEC 2011 – All rights reserved

Full document details: BS ISO/IEC 13249-3:2011

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

Documents
Document Sharing Hub

© ISO/IEC 2011 – All rights reserved

Full document details: BS ISO/IEC 13249-3:2011

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

Documents
Document Sharing Hub

© ISO/IEC 2011 – All rights reserved

Full document details: BS ISO/IEC 13249-3:2011

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**

Documents
Document Sharing Hub

© ISO/IEC 2011 – All rights reserved

Full document details: BS ISO/IEC 13249-3:2011

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*

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

Documents
Document Sharing Hub

© ISO/IEC 2011 – All rights reserved

Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

INTERNATIONAL STANDARD ISO/IEC 13249-3:2011(E)

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

Part 3: Spatial

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

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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, $\mathrm{dist}(g_1, g_2) = \mathrm{Min}\ (\ \{\ \text{distance between } p \text{ and } q \mid p \in g_1 \wedge q \in g_2\ \}\ )$

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

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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
$\{ x \mid \ldots \}$	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.

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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:

    $a.\text{ST_Intersection}(b) \Leftrightarrow \text{Closure}(a \cap b)$

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.

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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

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
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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)

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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)

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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:

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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:

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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;

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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;

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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.

© ISO/IEC 2011 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011

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,

© ISO/IEC 2011 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 13249-3:2011