

Australian Standard™

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

Part 3: Spatial



This Australian Standard was prepared by Committee IT-027, Data Management & Interchange. It was approved on behalf of the Council of Standards Australia on 21 April 2005. This Standard was published on 3 June 2005.

The following are represented on Committee IT-027:

Association of Superannuation Funds of Australia
Australia Post
Australia Bankers Association
Australian Bureau of Statistics
Australian Computer Society
Australian Customs Service
Australian Electoral Commission
Australian Electric and Electronic Manufacturers Association
Australian Industry Group
Australian Information Industry Association
Australian Institute of Health and Welfare
Australian Taxation Office
Centrelink
Data Management Association Australia
Department of Defence
Department of Immigration, Multicultural and Indigenous Affairs
Health Insurance Commission

Keeping Standards up-to-date

Standards are living documents which reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued. Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments which may have been published since the Standard was purchased.

Detailed information about Standards can be found by visiting the Standards Web Shop at www.standards.com.au and looking up the relevant Standard in the on-line catalogue.

Alternatively, the printed Catalogue provides information current at 1 January each year, and the monthly magazine, *The Global Standard*, has a full listing of revisions and amendments published each month.

Australian Standards™ and other products and services developed by Standards Australia are published and distributed under contract by SAI Global, which operates the Standards Web Shop.

We also welcome suggestions for improvement in our Standards, and especially encourage readers to notify us immediately of any apparent inaccuracies or ambiguities. Contact us via email at mail@standards.org.au, or write to the Chief Executive, Standards Australia, GPO Box 5420, Sydney, NSW 2001.

This Standard was issued in draft form for comment as DR 05116.

Australian Standard™

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

Part 3: Spatial

First published as AS ISO/IEC 13249.3—2005.

COPYRIGHT

© Standards Australia

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher.

Published by Standards Australia GPO Box 5420, Sydney, NSW 2001, Australia

ISBN 0 7337 6721 4

PREFACE

This Standard was prepared by the Standards Australia Committee IT-027, Data Management & Interchange.

This Standard is identical with, and has been reproduced from, ISO/IEC 13249-3:2003, *Information technology—Database languages—SQL multimedia and application packages—Part 3: Spatial*.

The objective of this Standard is to define spatial data types and their associated routines for use in SQL databases by software developers.

This Standard is Part 3 of AS 3249, *Information technology—Database languages—SQL multimedia and application packages*, which is published in parts as follows:

Part 1: Framework

Part 2: Full-Text

Part 3: Spatial (this Standard)

Part 5: Still image

Part 6: Data mining

As this Standard is reproduced from an international standard, the following applies:

- (a) Its number appears on the cover and title page while the international standard number appears only on the cover.
- (b) In the source text ‘this part of ISO/IEC 13249’ should read ‘this Australian Standard’.
- (c) A full point substitutes for a comma when referring to a decimal marker.

References to International Standards should be replaced by references to Australian or Australian/New Zealand Standards, as follows:

<i>Reference to International Standard</i>		<i>Australian Standard</i>	
ISO/IEC		AS ISO/IEC	
9075	Information technology—Database languages—SQL (all parts)	9075	Information technology—Database languages—SQL (all parts)
13249	Information technology—Database languages—SQL multimedia and application packages	13249	Information technology—Database languages—SQL multimedia and application packages
13249-1	Part 1: Framework	13249.1	Part 1: Framework
ISO		AS/NZS ISO	
19111	Geographic information—Spatial referencing by coordinates	19111	Geographic information—Spatial referencing by coordinates

Only referenced documents that have been adopted as Australian or Australian/New Zealand Standards have been listed.

CONTENTS

Page

1	Scope.....	1
2	Normative references	3
2.1	ISO/IEC JTC 1 standards.....	3
2.2	ISO standards.....	3
2.3	IEC standards.....	3
2.4	Other international standards.....	3
3	Terms and definitions, notations and conventions.....	5
3.1	Terms and definitions.....	5
3.1.1	Terms and definitions provided in ISO/IEC 13249-1:2002.....	5
3.1.2	Terms and definitions provided in this part of ISO/IEC 13249.....	5
3.1.3	Terms and definitions taken from ISO/IEC 9075 (all parts).....	8
3.1.4	Terms and definitions taken from ISO 19107:2003.....	9
3.1.5	Terms and definitions taken from ISO 19111:2003.....	9
3.2	Notations.....	10
3.2.1	Notations provided in ISO/IEC 13249-1:2002.....	10
3.2.2	Notations provided in this part of ISO/IEC 13249.....	10
3.3	Conventions	10
4	Concepts.....	11
4.1	Geometry Types.....	11
4.1.1	ST_Geometry.....	11
4.1.2	Spatial Relationships using ST_Geometry.....	13
4.1.3	ST_Point.....	17
4.1.4	ST_Curve	17
4.1.5	ST_LineString.....	18
4.1.6	ST_CircularString.....	18
4.1.7	ST_CompoundCurve	19
4.1.8	ST_Surface	20
4.1.9	ST_CurvePolygon	20
4.1.10	ST_Polygon	20
4.1.11	ST_GeomCollection	21
4.1.12	ST_MultiPoint	22
4.1.13	ST_MultiCurve	22
4.1.14	ST_MultiLineString	22
4.1.15	ST_MultiSurface	23
4.1.16	ST_MultiPolygon	23
4.2	Spatial Reference System Type.....	25
4.2.1	ST_SpatialRefSys.....	25
4.3	Angle and Direction Types.....	26
4.3.1	ST_Angle.....	26
4.3.2	ST_Direction	27
4.4	Support Routines	29
4.4.1	ST_Geometry ARRAY Support Routines.....	29
4.5	Tables with columns using geometry types	30
4.6	The Spatial Information Schema	31
5	Geometry Types.....	33
5.1	ST_Geometry Type and Routines	33
5.1.1	ST_Geometry Type	33
5.1.2	ST_Dimension Method	42

5.1.3	ST_CoordDim Method.....	43
5.1.4	ST_GeometryType Method.....	44
5.1.5	ST_SRID Methods	46
5.1.6	ST_Transform Method	47
5.1.7	ST_IsEmpty Method	48
5.1.8	ST_IsSimple Method	49
5.1.9	ST_IsValid Method	50
5.1.10	ST_Boundary Method	51
5.1.11	ST_Envelope Method	52
5.1.12	ST_ConvexHull Method	53
5.1.13	ST_Buffer Methods	54
5.1.14	ST_Intersection Method	56
5.1.15	ST_Union Method.....	57
5.1.16	ST_Difference Method	58
5.1.17	ST_SymDifference Method.....	59
5.1.18	Return Types from ST_Intersection, ST_Union, ST_Difference, and ST_SymDifference	60
5.1.19	ST_Distance Methods.....	63
5.1.20	ST_Equals Method	65
5.1.21	ST_Relate Method	66
5.1.22	ST_Disjoint Method.....	69
5.1.23	ST_Intersects Method	70
5.1.24	ST_Touches Method	71
5.1.25	ST_Crosses Method.....	72
5.1.26	ST_Within Method.....	73
5.1.27	ST_Contains Method.....	74
5.1.28	ST_Overlaps Method.....	75
5.1.29	Empty Cast.....	76
5.1.30	ST_WKTToSQL Method	83
5.1.31	ST_AsText Method	84
5.1.32	ST_WKBToSQL Method.....	85
5.1.33	ST_AsBinary Method.....	86
5.1.34	ST_GMLToSQL Method	87
5.1.35	ST_AsGML Method	89
5.1.36	ST_GeomFromText Functions.....	90
5.1.37	ST_GeomFromWKB Functions.....	92
5.1.38	ST_GeomFromGML Functions	93
5.1.39	ST_Geometry Ordering Definition	95
5.1.40	SQL Transform Functions.....	96
5.1.41	<well-known text representation>	97
5.1.42	<well-known binary representation>.....	105
6	Point Types.....	113
6.1	ST_Point Type and Routines	113
6.1.1	ST_Point Type	113
6.1.2	ST_Point Methods.....	116
6.1.3	ST_X Methods.....	119
6.1.4	ST_Y Methods.....	120
6.1.5	ST_ExplicitPoint Method	121
6.1.6	ST_PointFromText Functions.....	122
6.1.7	ST_PointFromWKB Functions.....	123
6.1.8	ST_PointFromGML Functions	124
7	Curve Types.....	125
7.1	ST_Curve Type and Routines	125
7.1.1	ST_Curve Type	125
7.1.2	ST_Length Methods.....	127
7.1.3	ST_StartPoint Method.....	129
7.1.4	ST_EndPoint Method	130
7.1.5	ST_IsClosed Method	131
7.1.6	ST_IsRing Method	132
7.1.7	ST_CurveToLine Method.....	133

7.2	ST_LineString Type and Routines	134
7.2.1	ST_LineString Type	134
7.2.2	ST_LineString Methods.....	137
7.2.3	ST_Points Methods.....	140
7.2.4	ST_NumPoints Method.....	142
7.2.5	ST_PointN Method	143
7.2.6	ST_StartPoint Method.....	144
7.2.7	ST_EndPoint Method	145
7.2.8	ST_LineFromText Functions	146
7.2.9	ST_LineFromWKB Functions	147
7.2.10	ST_LineFromGML Functions.....	148
7.3	ST_CircularString Type and Routines	149
7.3.1	ST_CircularString Type	149
7.3.2	ST_CircularString Methods.....	153
7.3.3	ST_Points Methods.....	156
7.3.4	ST_NumPoints Method.....	158
7.3.5	ST_PointN Method	159
7.3.6	ST_MidPointRep Method.....	160
7.3.7	ST_StartPoint Method.....	161
7.3.8	ST_EndPoint Method	162
7.3.9	ST_CircularFromTxt Functions	163
7.3.10	ST_CircularFromWKB Functions.....	164
7.4	ST_CompoundCurve Type and Routines	165
7.4.1	ST_CompoundCurve Type	165
7.4.2	ST_CompoundCurve Methods	169
7.4.3	ST_Curves Methods.....	172
7.4.4	ST_NumCurves Method.....	174
7.4.5	ST_CurveN Method	175
7.4.6	ST_StartPoint Method.....	176
7.4.7	ST_EndPoint Method	177
7.4.8	ST_CompoundFromTxt Functions.....	178
7.4.9	ST_CompoundFromWKB Functions.....	179
8	Surface Types.....	181
8.1	ST_Surface Type and Routines	181
8.1.1	ST_Surface Type	181
8.1.2	ST_Area Methods.....	183
8.1.3	ST_Perimeter Methods	185
8.1.4	ST_Centroid Method.....	187
8.1.5	ST_PointOnSurface Method	188
8.2	ST_CurvePolygon Type and Routines.....	189
8.2.1	ST_CurvePolygon Type.....	189
8.2.2	ST_CurvePolygon Methods	193
8.2.3	ST_ExteriorRing Methods	197
8.2.4	ST_InteriorRings Methods	199
8.2.5	ST_NumInteriorRing Method	202
8.2.6	ST_InteriorRingN Method.....	203
8.2.7	ST_CurvePolyToPoly Method	204
8.2.8	ST_CPolyFromText Functions.....	205
8.2.9	ST_CPolyFromWKB Functions	206
8.3	ST_Polygon Type and Routines	207
8.3.1	ST_Polygon Type	207
8.3.2	ST_Polygon Methods.....	210
8.3.3	ST_ExteriorRing Methods	214
8.3.4	ST_InteriorRings Methods	215
8.3.5	ST_InteriorRingN Method.....	217
8.3.6	ST_PolyFromText Functions	218
8.3.7	ST_PolyFromWKB Functions	219
8.3.8	ST_PolyFromGML Functions.....	220
8.3.9	ST_BdPolyFromText Functions	221
8.3.10	ST_BdPolyFromWKB Functions	223

9	Geometry Collection Types.....	225
9.1	ST_GeomCollection Type and Routines.....	225
9.1.1	ST_GeomCollection Type.....	225
9.1.2	ST_GeomCollection Methods	229
9.1.3	ST_Geometries Methods	232
9.1.4	ST_NumGeometries Method	234
9.1.5	ST_GeometryN Method	235
9.1.6	ST_GeomCollFromTxt Functions	236
9.1.7	ST_GeomCollFromWKB Functions.....	237
9.1.8	ST_GeomCollFromGML Functions	238
9.2	ST_MultiPoint Type and Routines.....	239
9.2.1	ST_MultiPoint Type	239
9.2.2	ST_MultiPoint Methods	242
9.2.3	ST_Geometries Methods	245
9.2.4	ST_MPointFromText Functions	247
9.2.5	ST_MPointFromWKB Functions.....	248
9.2.6	ST_MPointFromGML Functions	249
9.3	ST_MultiCurve Type and Routines.....	250
9.3.1	ST_MultiCurve Type.....	250
9.3.2	ST_MultiCurve Methods	253
9.3.3	ST_IsClosed Method	256
9.3.4	ST_Length Methods.....	257
9.3.5	ST_Geometries Methods	259
9.3.6	ST_MCurveFromText Functions.....	261
9.3.7	ST_MCurveFromWKB Functions.....	262
9.4	ST_MultiLineString Type and Routines	263
9.4.1	ST_MultiLineString Type	263
9.4.2	ST_MultiLineString Methods.....	266
9.4.3	ST_Geometries Methods	269
9.4.4	ST_MLineFromText Functions.....	271
9.4.5	ST_MLineFromWKB Functions	272
9.4.6	ST_MLineFromGML Functions.....	273
9.5	ST_MultiSurface Type and Routines.....	274
9.5.1	ST_MultiSurface Type.....	274
9.5.2	ST_MultiSurface Methods	277
9.5.3	ST_Area Methods.....	280
9.5.4	ST_Perimeter Methods	282
9.5.5	ST_Centroid Method	284
9.5.6	ST_PointOnSurface Method.....	285
9.5.7	ST_Geometries Methods	286
9.5.8	ST_MSurfaceFromTxt Functions.....	288
9.5.9	ST_MSurfaceFromWKB Functions.....	289
9.6	ST_MultiPolygon Type and Routines.....	290
9.6.1	ST_MultiPolygon Type.....	290
9.6.2	ST_MultiPolygon Methods	293
9.6.3	ST_Geometries Methods	296
9.6.4	ST_MPolyFromText Functions	298
9.6.5	ST_MPolyFromWKB Functions	299
9.6.6	ST_MPolyFromGML Functions.....	300
9.6.7	ST_BdMPolyFromText Functions.....	301
9.6.8	ST_BdMPolyFromWKB Functions	303
10	Spatial Reference System Type.....	305
10.1	ST_SpatialRefSys Type and Routines	305
10.1.1	ST_SpatialRefSys Type	305
10.1.2	ST_SpatialRefSys Methods.....	307
10.1.3	ST_AsWKTSRS Method.....	308
10.1.4	ST_WKTSRSToSQL Method	309
10.1.5	ST_SRID Method	310
10.1.6	ST_Equals Method	311
10.1.7	ST_OrderingEquals Function	312

10.1.8	ST_WellKnownText SQL Transform Group.....	313
10.1.9	<spatial reference system>	314
11	Angle and Direction Types.....	319
11.1	ST_Angle Type and Routines	319
11.1.1	ST_Angle Type	319
11.1.2	ST_Angle Methods.....	324
11.1.3	ST_Radians Methods.....	331
11.1.4	ST_Degrees Methods.....	332
11.1.5	ST_DegreeComponent Method	333
11.1.6	ST_MinuteComponent Method	334
11.1.7	ST_SecondComponent Method.....	335
11.1.8	ST_String Methods	336
11.1.9	ST_Gradians Methods	338
11.1.10	ST_Add Method.....	339
11.1.11	ST_Subtract Method	340
11.1.12	ST_Multiply Method	341
11.1.13	ST_Divide Method.....	342
11.1.14	ST_AsText Method.....	343
11.1.15	ST_Angle Ordering Definition.....	344
11.1.16	SQL Transform Functions.....	345
11.2	ST_Direction Type and Routines.....	346
11.2.1	ST_Direction Type.....	346
11.2.2	ST_Direction Methods	351
11.2.3	ST_Radians Method.....	356
11.2.4	ST_AngleNAzimuth Methods.....	357
11.2.5	ST_AsText Method.....	358
11.2.6	ST_RadianBearing Method	359
11.2.7	ST_DegreesBearing Method	361
11.2.8	ST_DMSBearing Method	363
11.2.9	ST_RadianNAzimuth Method.....	365
11.2.10	ST_DegreesNAzimuth Method.....	366
11.2.11	ST_DMSNAzimuth Method.....	367
11.2.12	ST_RadianSAzimuth Method	368
11.2.13	ST_DegreesSAzimuth Method	369
11.2.14	ST_DMSSAzimuth Method	370
11.2.15	ST_AddAngle Method.....	371
11.2.16	ST_SubtractAngle Method	372
11.2.17	ST_Direction Ordering Definition	373
11.2.18	SQL Transform Functions.....	374
12	Support Routines	375
12.1	ST_Geometry ARRAY Support Routines.....	375
12.1.1	ST_MaxDimension Function.....	375
12.1.2	ST_CheckSRID Function.....	377
12.1.3	ST_CheckNulls Procedure	378
12.1.4	ST_CheckConsecDups Procedure.....	379
12.1.5	ST_ToPointAny Cast Function	380
12.1.6	ST_ToCurveAny Cast Function.....	382
12.1.7	ST_ToLineStringAny Cast Function	384
12.1.8	ST_ToCircularAny Cast Function	386
12.1.9	ST_ToCompoundAny Cast Function.....	388
12.1.10	ST_ToSurfaceAny Cast Function.....	390
12.1.11	ST_ToCurvePolyAny Cast Function	392
12.1.12	ST_ToPolygonAny Cast Function.....	394
13	SQL/MM Spatial Information Schema	397
13.1	Introduction	397
13.2	ST_GEOMETRY_COLUMNS view.....	398
13.3	ST_SPATIAL_REFERENCE_SYSTEMS view.....	399
13.4	ST_UNITS_OF_MEASURE view	400
13.5	ST_SIZINGS view	401

	<i>Page</i>
13.6 Short name views.....	402
14 SQL/MM Spatial Definition Schema.....	403
14.1 Introduction	403
14.2 ST_GEOMETRY_COLUMNS base table	404
14.3 ST_SPATIAL_REFERENCE_SYSTEMS base table	405
14.4 ST_UNITS_OF_MEASURE base table	407
14.5 ST_SIZINGS base table.....	408
15 Status Codes	409
16 Conformance	411
16.1 Requirements for conformance.....	411
16.2 Features of ISO/IEC 9075 required for this part of ISO/IEC 13249	411
16.3 Claims of conformance	411
Annex A.....	417
A.1 Implementation-defined Meta-variables.....	434
Annex B.....	435
Annex C.....	437
Annex D.....	439
Bibliography	441
Index	443

Figures	Page
Figure D.1 — Geometry Type Hierarchy Diagram.....	439

Tables	Page
Table 1 — Symbols	10
Table 2 — DE-9IM	14
Table 3 — Parameter Types	60
Table 4 — Return Type Sets.....	61
Table 5 — Return Type Matrix for the ST_Intersection Method	61
Table 6 — Return Type Matrix for the ST_Union Method.....	62
Table 7 — Return Type Matrix for the ST_Difference Method	62
Table 8 — Return Type Matrix for the ST_SymDifference Method	62
Table 9 — DE-9IM Mapping	68
Table 10 — Cell Values	68
Table 11 — <well-known binary representation> <uint32> Values.....	111
Table 12 — SQLSTATE class and subclass values.....	409

INTRODUCTION

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

This document 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:

- 1) Clause 1, "Scope", specifies the scope of this part of ISO/IEC 13249.
- 2) 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.
- 3) Clause 3, "Definitions, notations, and conventions", defines the notations and conventions used in this part of ISO/IEC 13249.
- 4) Clause 4, "Concepts", presents concepts used in the definition of this part of ISO/IEC 13249.
- 5) Clause 5, "Geometry Types", defines the geometry supertype.
- 6) Clause 6, "Point Types", defines primitive 0-dimensional geometry types.
- 7) Clause 7, "Curve Types", defines primitive 1-dimensional geometry types.
- 8) Clause 8, "Surface Types", defines primitive 2-dimensional geometry types.
- 9) Clause 9, "Geometry Collection Types", defines the geometry collection types.
- 10) Clause 10, "Spatial Reference System Types", defines the user-defined type to manage spatial reference systems.
- 11) Clause 11, "Angle and Direction Types", defines the angles and direction types.
- 12) Clause 12, "Support Routines", defines supporting functions and procedures used by this part of ISO/IEC 13249.
- 13) Clause 13, "SQL/MM Spatial Information Schema" defines the SQL/MM Spatial Information Schema.
- 14) Clause 14, "SQL/MM Spatial Definition Schema" defines the SQL/MM Spatial Definition Schema.
- 15) Clause 15, "Status Codes", defines the SQLSTATE codes used in this part of ISO/IEC 13249.
- 16) Clause 16, "Conformance", defines the criteria for conformance to this part of ISO/IEC 13249.
- 17) 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 shall provide in each case.
- 18) 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.
- 19) Annex C, "Incompatibilities with ISO/IEC 13249-3:1999", is an informative Annex. It lists incompatibilities with the previous version of this part of ISO/IEC 13249-3.
- 20) Annex D, "Geometry Type Hierarchy", is an informative Annex. It visually describes the inheritance relationship between user-defined types in this part of ISO/IEC 13249.
- 21) 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, Clauses begin a new odd-numbered page, and in Clause 5, "Geometry Types", through Clause 11, "Angle and Direction Types", subclauses begin a new page. Any resulting blank space is not significant.

X

NOTES

AUSTRALIAN STANDARD

**Information technology—Database languages—SQL multimedia
and application packages****Part 3:
Spatial****1 Scope**

This part of ISO/IEC 13249:

- a) introduces the Spatial part of ISO/IEC 13249,
- b) gives the references necessary for this part of ISO/IEC 13249,
- c) defines notations and conventions specific to this part of ISO/IEC 13249,
- d) defines concepts specific to this part of ISO/IEC 13249,
- e) defines spatial user-defined types and their associated routines.

The spatial user-defined types defined in this part 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 may 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, multimedia, and resource management applications.

Blank page

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.

2.1 ISO/IEC JTC 1 standards

ISO/IEC 13249-1:2002, *Information technology — Database languages — SQL multimedia and application packages — Part 1: Framework*

2.2 ISO standards

ISO/IEC 9075 (all parts), *Information technology — Database languages — SQL*

ISO 19107:2003, *Geographic information — Spatial schema*

ISO 19111:2003, *Geographic information — Spatial referencing by coordinates*

2.3 IEC standards

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

2.4 Other international standards

Open GIS Consortium, Inc., *OpenGIS® Geography Markup Language (GML), Revision 2.0*, 2001-02-20.
<http://www.opengis.net/gml/01-029/GML2.html>

The W3C Consortium, *Extensible Markup Language (XML) Version 1.0 (second edition)*, 2000-10-02.
<http://www.w3.org/TR/REC-xml>

Blank page

3 Terms and definitions, notations and conventions

3.1 Terms and definitions

3.1.1 Terms and definitions provided in ISO/IEC 13249-1:2002

This part of ISO/IEC 13249 makes use of all the terms and definitions given in ISO/IEC 13249-1.

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

a geometry with a geometric dimension of 0 (zero)

3.1.2.2

1-dimensional geometry

a geometry with a geometric dimension of 1 (one)

3.1.2.3

2-dimensional geometry

a geometry with a geometric dimension of 2

3.1.2.4

angle

the inclination to each other of two intersecting lines (which may 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

a 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 2π radians, or 400 gradians).

3.1.2.6

bearing

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

the 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

the empty set

3.1.2.9

boundary of a surface

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

3.1.2.10

clockwise

a sense of rotation followed by the hands of a conventional analog clock; or, equivalently, making a fist with the left hand, raising the thumb and pointing it toward your face, the remaining fingers on the left hand point in a clockwise direction around the thumb

3.1.2.11**closed curve**

a curve such that its start point is equal to its end point

3.1.2.12**closure**

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

3.1.2.13**counterclockwise**

a sense of rotation that is opposite to clockwise

3.1.2.14**degree**

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

3.1.2.15**degrees, minutes, and seconds representation (of an angle)**

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

3.1.2.16**dimension**

geometric dimension

3.1.2.17**direction**

the 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

3.1.2.18**distance between two geometries**

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

let g_1 and g_2 be two geometries,

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

3.1.2.19**distance between two points**

the minimum of the lengths of all curves connecting two points

3.1.2.20**geometry**

the shape and geographic location of a feature

3.1.2.21**GML**

Geography Markup Language as defined in *OpenGIS® Geography Markup Language (GML), Revision 2.0*

3.1.2.22**GML representation**

an XML element that is valid against an XML element definition in the Geometry Schema (geometry.xsd) as defined in *OpenGIS® Geography Markup Language (GML), Revision 2.0*

3.1.2.23**gradians**

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

3.1.2.24**heading**

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

3.1.2.25**intersection**

two geometries intersect if their point set representations, a and b intersect: $a \cap b \neq \emptyset$

3.1.2.26**linear ring**

a linestring that is closed and simple

3.1.2.27**linestring**

a curve with linear interpolation between control points

3.1.2.28**minute**

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

3.1.2.29**mod 2 union rule**

only elements that occur an odd number of times in a multiset are in the result set

NOTE 1 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 . Element 20 and 40 are not in R because they occur an even number of times in A .

3.1.2.30**non-closed curve**

a curve such that its start point is not equal to its end point

3.1.2.31**North azimuth**

an azimuthal heading whose rotation is measured clockwise from True North

3.1.2.32**pi**

a real number mathematical constant that represents the circumference of a circle with unit diameter. This number is transcendental and cannot be represented exactly in any algebraic form, so the precision is implementation-defined

3.1.2.33**polygon**

a surface that uses linear rings to define its boundary

3.1.2.34**point set**

the representation of a geometry as a finite set or infinite set of points

NOTE 2 Mathematical set intersection ($\cap$), set union ($\cup$) and set difference ($-$) operation work on point sets.

3.1.2.35**radians**

a unit of measurement for angles such that there are 2π radians in a circle

3.1.2.36**ring**

a curve that is closed and simple

3.1.2.37**rotation**

an angle with a specific sense, which may be either clockwise or counterclockwise

3.1.2.38**second**

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

3.1.2.39**South azimuth**

an azimuthal heading whose rotation is measured clockwise from True South

3.1.2.40**spatially equals**

the test for *spatially equals* on geometry values has a similar definition as 'equals' for mathematical sets

NOTE 3 If the point sets of two geometry values are equal, then the geometries are spatially equal. Two point sets, a and b , are equal if: $(a - b) \cup (b - a) = \emptyset$.

NOTE 4 Since an infinite set of points cannot be tested, the internal representation of equal shall test for equivalents between two, possibly quite different, representations. The test may be limited to 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 than the tolerance.

3.1.2.41**topologically closed**

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

3.1.2.42**True North**

a geographic reference direction towards the Earth's geographic North pole

3.1.2.43**True South**

a geographic reference direction towards the Earth's geographic South pole

3.1.2.44**unit of measure**

defined quantity in which dimensioned parameters are expressed

3.1.2.45**XML element**

an element as defined by *Extensible Markup Language (XML) Version 1.0 (second edition)*

3.1.3 Terms and definitions taken from ISO/IEC 9075 (all parts)

This part of ISO/IEC 13249 makes use of the following terms defined in ISO/IEC 9075 (all parts):

- a) immediately contained
- b) maximal supertype
- c) subtype family

3.1.4 Terms and definitions taken from ISO 19107:2003

This part of ISO/IEC 13249 makes use of the following terms defined in ISO 19107:2003.

- a) boundary
- b) buffer
- c) computational topology
- d) connected
- e) convex hull of a geometric object
- f) coordinate
- g) coordinate dimension
- h) coordinate system
- i) coordinate reference system
- j) curve
- k) end point
- l) exterior
- l) geometric complex
- m) geometric dimension
- n) geometric object
- o) geometric primitive
- p) homomorphism
- q) interior
- r) isomorphism
- s) point
- t) start point
- u) surface

3.1.5 Terms and definitions taken from ISO 19111:2003

This part of ISO/IEC 13249 makes use of the following terms defined in ISO 19111:2003.

- 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 13249-1:2002

The notations used in this part of ISO/IEC 13249 are defined in ISO/IEC 13249-1:2002.

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 and SQL-invoked routine 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.

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

Table 1 — Symbols

Symbols	Meaning
$\emptyset$	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
$\{x \mid \dots\}$	set of all x such that ...
$\wedge$	And
$\vee$	Or
$\neg$	Not

3.3 Conventions

The conventions used in this part of ISO/IEC 13249 are defined in ISO/IEC 13249-1:2002.

4 Concepts

4.1 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_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 D, "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_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.

4.1.1 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).

NOTE 5 A future edition of this part of ISO/IEC 13249 may deal with n -dimensional coordinate space (R^n) where n is greater than 2.

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 .

4.1.1.1 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.
- 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.
- 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 point, such as self intersection or self tangency. Subtypes of ST_Geometry will define the specific conditions that cause a value to be classified as simple.
- 8) ST_IsValid: tests if an ST_Geometry value is well formed.

- 9) ST_Boundary: returns the boundary of an ST_Geometry value.
- 10) ST_Envelope: returns the bounding rectangle of an ST_Geometry value.
- 11) ST_ConvexHull: returns the convex hull of an ST_Geometry value.
- 12) 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.
- 13) ST_Intersection: returns the ST_Geometry value that represents the point set intersection of two ST_Geometry values.

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

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

- 14) ST_Union: returns the ST_Geometry value that represents the point set union of two ST_Geometry values.

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

$$a.ST_Union(b) \Leftrightarrow \text{Closure}(a \cup b)$$

- 15) ST_Difference: returns the ST_Geometry value that represents the point set difference of two ST_Geometry values.

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

$$a.ST_Difference(b) \Leftrightarrow \text{Closure}(a - b)$$

- 16) ST_SymDifference: returns the ST_Geometry value that represents the point set symmetric difference of two ST_Geometry values.

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

$$\begin{aligned} a.ST_SymDifference(b) &\Leftrightarrow \text{Closure}(a - b) \cup \text{Closure}(b - a) \Leftrightarrow \\ &a.ST_Difference(b).ST_Union(b.ST_Difference(a)) \end{aligned}$$

- 17) ST_Distance: returns the distance between two geometries.
- 18) ST_WKTTToSQL: returns the ST_Geometry value for the specified well-known text representation.
- 19) ST_AsText: returns the well-known text representation for the specified ST_Geometry value.
- 20) ST_WKBToSQL: returns the ST_Geometry value for the specified well-known binary representation.
- 21) ST_AsBinary: returns the well-known binary representation for the specified ST_Geometry value.
- 22) ST_GMLToSQL: returns the ST_Geometry value for the specified GML representation.
- 23) ST_AsGML: returns the GML representation for the specified ST_Geometry value.

4.1.1.2 Functions on ST_Geometry

- 1) ST_GeomFromText: returns an ST_Geometry value, which is transformed from a CHARACTER LARGE OBJECT 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 that represents the well-known binary representation of an ST_Geometry.
- 3) ST_GeomFromGML: transforms a CHARACTER LARGE OBJECT containing a GML representation to an ST_Geometry value.

4.1.1.3 Ordering on ST_Geometry

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

4.1.1.4 SQL Transforms on ST_Geometry

- 1) ST_WellKnownText: is the SQL Transform group that transforms an ST_Geometry value to and from a CHARACTER LARGE OBJECT value.
- 2) ST_WellKnownBinary: is the SQL Transform group that transforms an ST_Geometry value to and from 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.

4.1.1.5 Empty Casts on ST_Geometry

- 1) Cast from an empty set of type ST_Geometry to an ST_Point value.
- 2) Cast from an empty set of type ST_Geometry to an ST_LineString value.
- 3) Cast from an empty set of type ST_Geometry to an ST_CircularString value.
- 4) Cast from an empty set of type ST_Geometry to an ST_CompoundCurve value.
- 5) Cast from an empty set of type ST_Geometry to an ST_CurvePolygon value.
- 6) Cast from an empty set of type ST_Geometry to an ST_Polygon value.
- 7) Cast from an empty set of type ST_Geometry to an ST_GeomCollection value.
- 8) Cast from an empty set of type ST_Geometry to an ST_MultiPoint value.
- 9) Cast from an empty set of type ST_Geometry to an ST_MultiCurve value.
- 10) Cast from an empty set of type ST_Geometry to an ST_MultiLineString value.
- 11) Cast from an empty set of type ST_Geometry to an ST_MultiSurface value.
- 12) Cast from an empty set of type ST_Geometry to an ST_MultiPolygon value.

4.1.2 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 two-dimensional coordinate space (R^2). In order to apply the concepts of interior, boundary and exterior to 0-dimensional geometry and 1-dimensional geometry values in R^2 , 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.

4.1.2.1 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 ($\emptyset$). A dimensionally extended nine intersection matrix (DE-9IM) is specified in Table 2 — DE-9IM.

Table 2 — DE-9IM

	Interior	Boundary	Exterior
Interior	$(\text{Interior}(a) \cap \text{Interior}(b)).$ ST_Dimension()	$(\text{Interior}(a) \cap \text{Boundary}(b)).$ ST_Dimension()	$(\text{Interior}(a) \cap \text{Exterior}(b)).$ ST_Dimension()
Boundary	$(\text{Boundary}(a) \cap \text{Interior}(b)).$ ST_Dimension()	$(\text{Boundary}(a) \cap \text{Boundary}(b)).$ ST_Dimension()	$(\text{Boundary}(a) \cap \text{Exterior}(b)).$ ST_Dimension()
Exterior	$(\text{Exterior}(a) \cap \text{Interior}(b)).$ ST_Dimension()	$(\text{Exterior}(a) \cap \text{Boundary}(b)).$ ST_Dimension()	$(\text{Exterior}(a) \cap \text{Exterior}(b)).$ 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 $x.\text{ST_Dimension}() \in \{0, 1, 2\}$, i.e. $x \neq \emptyset$
- b) if $p = F$, then $x.\text{ST_Dimension}() = -1$, i.e. $x = \emptyset$
- c) if $p = 0$, then $x.\text{ST_Dimension}() = 0$ (zero)
- d) if $p = 1$, then $x.\text{ST_Dimension}() = 1$ (one)
- e) if $p = 2$, then $x.\text{ST_Dimension}() = 2$
- f) if $p = *$, then $x.\text{ST_Dimension}() \in \{-1, 0, 1, 2\}$, 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 5.2, "<token> and <separator>" in ISO/IEC 9075-2 for the definition of <character string literal>.

4.1.2.2 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. 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.

- 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.

4.1.2.3 Spatial Methods using ST_Geometry

- 1) ST_Equals: tests if an ST_Geometry value is spatially equal to another ST_Geometry value.

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

$$\begin{aligned} a.ST_Equals(b) &\Leftrightarrow \\ (a - b) \cup (b - a) &= \emptyset \Leftrightarrow \\ a.ST_SymDifference(b).ST_IsEmpty() \end{aligned}$$

- 2) 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.
- 3) 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:

$$\begin{aligned} a.ST_Disjoint(b) &\Leftrightarrow \\ a \cap b &= \emptyset \end{aligned}$$

Expressed in terms of the DE-9IM:

$$\begin{aligned} a.ST_Disjoint(b) &\Leftrightarrow \\ (\text{Interior}(a) \cap \text{Interior}(b) = \emptyset) \wedge \\ (\text{Interior}(a) \cap \text{Boundary}(b) = \emptyset) \wedge \\ (\text{Boundary}(a) \cap \text{Interior}(b) = \emptyset) \wedge \\ (\text{Boundary}(a) \cap \text{Boundary}(b) = \emptyset) &\Leftrightarrow \\ a.ST_Relate(b, 'FF*FF****') \end{aligned}$$

- 4) 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:

$$\begin{aligned} a.ST_Intersects(b) &\Leftrightarrow \\ \text{Case } a.ST_Disjoint(b) &\text{ when 0 then 1 when 1 then 0 else NULL end} \end{aligned}$$

- 5) 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:

Case:

- a) If $a.ST_Dimension() = 0$ (zero) and $b.ST_Dimension() = 0$ (zero), then the null value
- b) Otherwise, $a.ST_Touches(b) \Leftrightarrow (\text{Interior}(a) \cap \text{Interior}(b) = \emptyset) \wedge (a \cap b) \neq \emptyset$

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:

$$\begin{aligned} a.ST_Touches(b) &\Leftrightarrow \\ (\text{Interior}(a) \cap \text{Interior}(b) = \emptyset) \wedge \\ ((\text{Boundary}(a) \cap \text{Interior}(b) \neq \emptyset) \vee \\ (\text{Interior}(a) \cap \text{Boundary}(b) \neq \emptyset) \vee \\ (\text{Boundary}(a) \cap \text{Boundary}(b) \neq \emptyset)) &\Leftrightarrow \\ \text{Case } a.ST_Relate(b, 'FT*****') &= 1 \text{ OR} \\ a.ST_Relate(b, 'F**T*****') &= 1 \text{ OR} \\ a.ST_Relate(b, 'F***T****') &= 1 \text{ when TRUE then 1 when FALSE then 0 else NULL end} \end{aligned}$$

- 6) 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:

$$a.ST_Crosses(b) \Leftrightarrow ((\text{Interior}(a) \cap \text{Interior}(b)).ST_Dimension() < \max(\text{Interior}(a).ST_Dimension(), \text{Interior}(b).ST_Dimension())) \wedge (a \cap b \neq a) \wedge (a \cap b \neq b)$$

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:

$$a.ST_Crosses(b) \Leftrightarrow (\text{Interior}(a) \cap \text{Interior}(b) \neq \emptyset) \wedge (\text{Interior}(a) \cap \text{Exterior}(b) \neq \emptyset) \Leftrightarrow a.ST_Relate(b, 'T*T*****')$$
- c) If $a.ST_Dimension() = 1$ (one) and $b.ST_Dimension() = 1$ (one), then:

$$a.ST_Crosses(b) \Leftrightarrow (\text{Interior}(a) \cap \text{Interior}(b)).ST_Dimension() = 0 \text{ (zero)} \Leftrightarrow a.ST_Relate(b, '0*****')$$

- 7) 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:

$$a.ST_Within(b) \Leftrightarrow (a \cap b = a) \wedge (\text{Interior}(a) \cap \text{Exterior}(b) = \emptyset)$$

Expressed in terms of the DE-9IM:

$$a.ST_Within(b) \Leftrightarrow (\text{Interior}(a) \cap \text{Interior}(b) \neq \emptyset) \wedge (\text{Interior}(a) \cap \text{Exterior}(b) = \emptyset) \wedge (\text{Boundary}(a) \cap \text{Exterior}(b) = \emptyset) \Leftrightarrow a.ST_Relate(b, 'T*F**F***')$$

- 8) 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:

$$a.ST_Contains(b) \Leftrightarrow b.ST_Within(a)$$

- 9) 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:

$$a.ST_Overlaps(b) \Leftrightarrow (\text{Interior}(a).ST_Dimension() = \text{Interior}(b).ST_Dimension() = (\text{Interior}(a) \cap \text{Interior}(b)).ST_Dimension()) \wedge (a \cap b \neq a) \wedge (a \cap b \neq b)$$
- b) Otherwise, the null value

Expressed in terms of the DE-9IM:

Case:

- a) If $(a.ST_Dimension() = 0 \text{ (zero)} \text{ and } b.ST_Dimension() = 0 \text{ (zero)})$ or $(a.ST_Dimension() = 2 \text{ and } b.ST_Dimension() = 2)$, then:

$$\begin{aligned}
 &a.ST_Overlaps(b) \Leftrightarrow \\
 &(\text{Interior}(a) \cap \text{Interior}(b) \neq \emptyset) \wedge \\
 &(\text{Interior}(a) \cap \text{Exterior}(b) \neq \emptyset) \wedge \\
 &(\text{Exterior}(a) \cap \text{Interior}(b) \neq \emptyset) \Leftrightarrow \\
 &a.ST_Relate(b, 'T*T***T**')
 \end{aligned}$$
- b) If $a.ST_Dimension() = 1 \text{ (one)} \text{ and } b.ST_Dimension() = 1 \text{ (one)}$, then:

$$\begin{aligned}
 &a.ST_Overlaps(b) \Leftrightarrow \\
 &((\text{Interior}(a) \cap \text{Interior}(b)).ST_Dimension() = 1 \text{ (one)}) \wedge \\
 &(\text{Interior}(a) \cap \text{Exterior}(b) \neq \emptyset) \wedge (\text{Exterior}(a) \cap \text{Interior}(b) \neq \emptyset) \Leftrightarrow \\
 &a.ST_Relate(b, '1*T***T**')
 \end{aligned}$$
- c) Otherwise, the null value

4.1.3 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 in two-dimensional coordinate space (R^2). An `ST_Point` has an x coordinate value and a y coordinate value. The boundary of an `ST_Point` value is the empty set. `ST_Point` values are simple.

4.1.3.1 Methods on ST_Point

- 1) `ST_Point`: returns an `ST_Point` value constructed from either:
 - a) the well-known text representation or the well-known binary representation of an `ST_Point` value;
 - b) 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.
- 6) `ST_ExplicitPoint`: returns the coordinate values as a `DOUBLE PRECISION ARRAY` value.

4.1.3.2 Functions on ST_Point

- 1) `ST_PointFromText`: returns an `ST_Point` value, which is transformed from a `CHARACTER LARGE OBJECT` 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` that represents the well-known binary representation of an `ST_Point`.
- 3) `ST_PointFromGML`: transforms a `CHARACTER LARGE OBJECT` containing a GML representation to an `ST_Point` value.

4.1.4 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 on 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.

4.1.4.1 Methods on ST_Curve

- 1) ST_Length: returns the length of an ST_Curve value.
- 2) ST_StartPoint: returns the ST_Point value that is the start point of an ST_Curve value.
- 3) ST_EndPoint: returns the ST_Point value that is the end point of an ST_Curve value.
- 4) ST_IsClosed: tests if an ST_Curve value is closed.
- 5) ST_IsRing: tests if an ST_Curve value is a ring.
- 6) ST_CurveToLine: returns the ST_LineString value approximating the ST_Curve value.

4.1.5 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.

4.1.5.1 Methods on ST_LineString

- 1) ST_LineString: returns an ST_LineString value constructed from either:
 - a) the well-known text representation or the well-known binary representation of an ST_LineString value;
 - b) 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.

4.1.5.2 Functions on ST_LineString

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

4.1.6 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.

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.

4.1.6.1 Methods on ST_CircularString

- 1) ST_CircularString: returns an ST_CircularString value constructed from either:
 - a) the well-known text representation or the well-known binary representation of an ST_CircularString value;
 - b) 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.

4.1.6.2 Functions on ST_CircularString

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

4.1.7 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.

4.1.7.1 Methods on ST_CompoundCurve

- 1) ST_CompoundCurve: returns an ST_CompoundCurve value constructed from either:
 - a) the well-known text representation or the well-known binary representation of an ST_CompoundCurve value;
 - b) 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.

4.1.7.2 Functions on ST_CompoundCurve

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

4.1.8 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 defined as a simple surface consisting of a single patch whose boundary is specified by one exterior ring and zero or more interior rings. Simple 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.

4.1.8.1 Methods on ST_Surface

- 1) ST_Area: returns the area of an ST_Surface value.
- 2) ST_Perimeter: returns the length of the perimeter of an ST_Surface value.
- 3) ST_Centroid: returns the ST_Point value that is the mathematical centroid of the ST_Surface value.
- 4) ST_PointOnSurface: returns the ST_Point value that is guaranteed to be on the ST_Surface value.

4.1.9 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, 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 but only as a tangent. 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.

4.1.9.1 Methods on ST_CurvePolygon

- 1) ST_CurvePolygon: returns an ST_CurvePolygon value constructed from either:
 - a) the well-known text representation or the well-known binary representation of an ST_CurvePolygon value;
 - b) 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.
- 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.

4.1.9.2 Functions on ST_CurvePolygon

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

4.1.10 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.

4.1.10.1 Methods on ST_Polygon

- 1) ST_Polygon: returns an ST_Polygon value constructed from either:
 - a) the well-known text representation or the well-known binary representation of an ST_Polygon value;
 - b) the specified ST_LineString values.

4.1.10.2 Functions on ST_Polygon

- 1) ST_PolyFromText: returns an ST_Polygon value, which is transformed from a CHARACTER LARGE OBJECT 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 that represents the well-known binary representation of an ST_Polygon.
- 3) ST_PolyFromGML: transforms a CHARACTER LARGE OBJECT containing a GML representation to 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.

4.1.11 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.

4.1.11.1 Methods on ST_GeomCollection

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

4.1.11.2 Functions on ST_GeomCollection

- 1) ST_GeomCollFromTxt: returns an ST_GeomCollection value, which is transformed from a CHARACTER LARGE OBJECT 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 that represents the well-known binary representation of an ST_GeomCollection.
- 3) ST_GeomCollFromGML: transforms a CHARACTER LARGE OBJECT containing a GML representation to an ST_GeomCollection value.

4.1.12 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 collection. 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.

4.1.12.1 Methods on ST_MultiPoint

- 1) ST_MultiPoint returns an ST_MultiPoint value constructed from either:
 - a) the well-known text representation or the well-known binary representation of an ST_MultiPoint value;
 - b) the specified ST_Point values.

4.1.12.2 Functions on ST_MultiPoint

- 1) ST_MPointFromText: returns an ST_MultiPoint value, which is transformed from a CHARACTER LARGE OBJECT 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 that represents the well-known binary representation of an ST_MultiPoint.
- 3) ST_MPointFromGML: transforms a CHARACTER LARGE OBJECT containing a GML representation to an ST_MultiPoint value.

4.1.13 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 collection. 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 on 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.

4.1.13.1 Methods on ST_MultiCurve

- 1) ST_MultiCurve: returns an ST_MultiCurve value constructed from either:
 - a) the well-known text representation or the well-known binary representation of an ST_MultiCurve value;
 - b) the specified ST_Curve values.
- 2) ST_IsClosed: tests if an ST_MultiCurve value is closed.
- 3) ST_Length: returns the length of an ST_MultiCurve value.

4.1.13.2 Functions on ST_MultiCurve

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

4.1.14 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.

4.1.14.1 Methods on ST_MultiLineString

- 1) ST_MultiLineString: returns an ST_MultiLineString value constructed from either:
 - a) the well-known text representation or the well-known binary representation of an ST_MultiLineString value;
 - b) the specified ST_LineString values.

4.1.14.2 Functions on ST_MultiLineString

- 1) ST_MLineFromText: returns an ST_MultiLineString value, which is transformed from a CHARACTER LARGE OBJECT 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 that represents the well-known binary representation of an ST_MultiLineString.
- 3) ST_MLineFromGML: transforms a CHARACTER LARGE OBJECT containing a GML representation to an ST_MultiLineString value.

4.1.15 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 collection. 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.

4.1.15.1 Methods on ST_MultiSurface

- 1) ST_MultiSurface: returns an ST_MultiSurface value constructed from either:
 - a) the well-known text representation or the well-known binary representation of an ST_MultiSurface value;
 - b) the specified ST_Surface values.
- 2) ST_Area: returns the area of an ST_MultiSurface value.
- 3) ST_Perimeter: returns the length of the perimeter of an ST_MultiSurface value.
- 4) ST_Centroid: returns the ST_Point value that is the mathematical centroid of the ST_MultiSurface value.
- 5) ST_PointOnSurface: returns the ST_Point value that is guaranteed to be on the ST_MultiSurface value.

4.1.15.2 Functions on ST_MultiSurface

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

4.1.16 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 distinct elements of an ST_MultiPolygon do not intersect. The interiors of 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 shall not cross and may touch at only a finite number of points. An ST_MultiPolygon value is defined to be topologically closed.

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.

ST_MultiPolygon values are simple.

4.1.16.1 Methods on ST_MultiPolygon

- 1) ST_MultiPolygon: returns an ST_MultiPolygon value constructed from either:
 - a) the well-known text representation or the well-known binary representation of an ST_MultiPolygon value;
 - b) the specified ST_Polygon values.

4.1.16.2 Functions on ST_MultiPolygon

- 1) ST_MPolyFromText: returns an ST_MultiPolygon value, which is transformed from a CHARACTER LARGE OBJECT that represents the well-known text representation of an ST_MultiPolygon.
- 2) ST_MPolyFromWKB: returns an ST_MultiPolygon value, which is transformed from a BINARY LARGE OBJECT that represents the well-known binary representation of an ST_MultiPolygon.
- 3) ST_MPolyFromGML: transforms a CHARACTER LARGE OBJECT containing a GML representation to 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.

4.2 Spatial Reference System Type

4.2.1 ST_SpatialRefSys

The ST_SpatialRefSys type encapsulates all aspects of spatial reference systems.

4.2.1.1 Methods on ST_SpatialRefSys

- 1) ST_SpatialRefSys: returns the specified ST_SpatialRefSys value.
- 2) ST_AsWKTSRS: returns the well-known text representation of a spatial reference system for the specified ST_SpatialRefSys value.
- 3) ST_WKTSRSToSQL: returns the ST_SpatialRefSys value of a well-known text representation of a spatial reference system.
- 4) ST_SRID: returns the integer identifier of an ST_SpatialRefSys value.
- 5) ST_Equals: tests if two ST_SpatialRefSys values are equal.

4.2.1.2 Ordering on ST_SpatialRefSys

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

4.2.1.3 SQL Transforms on ST_SpatialRefSys

- 1) ST_WellKnownText: is the SQL Transform group that transforms an ST_SpatialRefSys value to and from a CHARACTER LARGE OBJECT value.

4.3 Angle and Direction Types

The following types are supported: ST_Angle and ST_Direction.

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

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

4.3.1 ST_Angle

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

4.3.1.1 Methods on ST_Angle

- 1) ST_Angle: returns a specified ST_Angle value from either:
 - a) radians, gradians, or degrees;
 - b) degrees and minutes;
 - c) degrees, minutes, and seconds;
 - d) three points, represented as ST_Point values, such that the angle is the lesser of the two possible rotations measured between the direction from the first point to the second point and the direction from the first point to the third point;
 - e) two directions, represented as ST_Direction values, where the angle is the lesser of the two possible rotations measured between the two directions;
 - f) two lines, represented as ST_LineString values, such that the angle specified is the lesser of the two possible rotations measured between the respective directions of the two lines, where the direction of a line is the direction from its start point to its end point;
 - g) a well-known text representation.
- 2) ST_Radians: observes and mutates the radians attribute of an ST_Angle value.
- 3) ST_Degrees: observes and mutates the radians attribute of an ST_Angle value using decimal degrees.
- 4) ST_DegreeComponent: returns the integer value that represents the degrees part of the degrees, minutes, and seconds representation of the ST_Angle value.
- 5) ST_MinuteComponent: returns the integer value that represents the minutes part of the degrees, minutes, and seconds representation of the ST_Angle value.
- 6) ST_SecondComponent: returns the double precision value that represents the seconds part of the degrees, minutes, and seconds representation of the ST_Angle value.
- 7) ST_String: observes and mutates the radians attribute of an ST_Angle using a space separated string of degrees, minutes, and seconds.
- 8) ST_Gradians: observes and mutates the radians attribute of an ST_Angle value using gradians.
- 9) ST_Add: adds the value of an angle to the ST_Angle value.
- 10) ST_Subtract: subtracts the value of an angle from the ST_Angle value.
- 11) ST_Multiply: multiplies the ST_Angle value by a numeric value.
- 12) ST_Divide: divides the ST_Angle value by a non-zero, numeric value.
- 13) ST_AsText: returns the well-known text representation of an ST_Angle value.

4.3.1.2 Ordering on ST_Angle

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

4.3.1.3 SQL Transforms on ST_Angle

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

4.3.2 ST_Direction

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

4.3.2.1 Methods on ST_Direction

- 1) ST_Direction: returns a specified ST_Direction value from either:
 - a) radians;
 - b) 'N' (for North) or 'S' (for South), an angle, and 'E' (for East) or 'W' (for West);
 - c) 'N' (for North) or 'S' (for South) and an angle;
 - d) two points, represented as ST_Point values, such that the direction defined is the direction from the first point towards the second point;
 - e) a line, represented as an ST_LineString value, such that the direction defined is the direction from the start point of the line to the end point of the line;
 - f) a well-known text representation.
- 2) ST_AngleNAzimuth: observes and mutates the ST_PrivateAngleNAzimuth attribute of an ST_Direction value.
- 3) ST_RadianBearing: observes the ST_Direction value represented as a bearing with its angle part expressed in radians.
- 4) ST_DegreesBearing: returns the ST_Direction value represented as a bearing with its angle part expressed in decimal degrees.
- 5) ST_DMSBearing: returns the ST_Direction value represented as a bearing with its angle part expressed in degrees, minutes, and seconds.
- 6) ST_RadianNAzimuth: returns the ST_Direction value represented as a North azimuth with its angle part expressed in radians.
- 7) ST_DegreesNAzimuth: returns the ST_Direction value represented as a North azimuth with its angle part expressed in decimal degrees.
- 8) ST_DMSNAzimuth: returns the ST_Direction value represented as a North azimuth with its angle part expressed in degrees, minutes, and seconds.
- 9) ST_RadianSAzimuth: returns the ST_Direction value represented as a South azimuth with its angle part expressed in radians.
- 10) ST_DegreesSAzimuth: returns the ST_Direction value represented as a South azimuth with its angle part expressed in decimal degrees.
- 11) ST_DMSSAzimuth: returns the ST_Direction value represented as a South azimuth with its angle part expressed in degrees, minutes, and seconds.
- 12) ST_AddAngle: mutates the ST_Direction value by adding an angle.
- 13) ST_SubtractAngle: mutates the ST_Direction value by subtracting an angle.
- 14) ST_AsText: returns the well-known text representation of an ST_Direction value.

4.3.2.2 Ordering on ST_Direction

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

4.3.2.3 SQL Transforms on ST_Direction

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

4.4 Support Routines

4.4.1 ST_Geometry ARRAY Support Routines

4.4.1.1 Support Functions

- 1) ST_MaxDimension: returns the maximum geometric dimension value of the elements in an ST_Geometry ARRAY value.
- 2) ST_CheckSRID: if the elements in the ST_Geometry ARRAY value have mixed spatial reference systems, then raises an exception. Otherwise, the function returns the spatial reference system identifier of the elements of the ST_Geometry ARRAY value.

4.4.1.2 Supporting Procedures

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

4.4.1.3 Supporting Cast Functions

- 1) ST_ToPointAry Cast: casts an ST_Geometry ARRAY value to an ST_Point ARRAY value.
- 2) ST_ToCurveAry Cast: casts an ST_Geometry ARRAY value to an ST_Curve ARRAY value.
- 3) ST_ToLineStringAry Cast: casts an ST_Geometry ARRAY value to an ST_LineString ARRAY value.
- 4) ST_ToCircularAry Cast: casts an ST_Geometry ARRAY value to an ST_CircularString ARRAY value.
- 5) ST_ToCompoundAry Cast: casts an ST_Geometry ARRAY value to an ST_CompoundCurve ARRAY value.
- 6) ST_ToSurfaceAry Cast: casts an ST_Geometry ARRAY value to an ST_Surface ARRAY value.
- 7) ST_ToCurvePolyAry Cast: casts an ST_Geometry ARRAY value to an ST_CurvePolygon ARRAY value.
- 8) ST_ToPolygonAry Cast: casts an ST_Geometry ARRAY value to an ST_Polygon ARRAY value.

4.5 Tables with columns using geometry types

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

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

4.6 The Spatial Information Schema

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

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

Blank page

5 Geometry Types

5.1 ST_Geometry Type and Routines

5.1.1 ST_Geometry Type

Purpose

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

Definition

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

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

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

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

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

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

```

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

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

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

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

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

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

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

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

```

```

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

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

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

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

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

METHOD ST_Distance
  (ageometry ST_Geometry)
RETURNS DOUBLE PRECISION
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

METHOD ST_Distance
  (ageometry ST_Geometry,
   aunit CHARACTER VARYING(ST_MaxUnitNameLength))
RETURNS DOUBLE PRECISION
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

```

```

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

METHOD ST_Relate
  (ageometry ST_Geometry, amatrix CHARACTER(9))
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

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

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

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

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

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

```

```

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

```

```

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

```

```

METHOD ST_ToPoint()
  RETURNS ST_Point
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

METHOD ST_ToLineString()
  RETURNS ST_LineString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

METHOD ST_ToCircular()
  RETURNS ST_CircularString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

METHOD ST_ToCompound()
  RETURNS ST_CompoundCurve
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

METHOD ST_ToCurvePoly()
  RETURNS ST_CurvePolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

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

```

```

METHOD ST_ToGeomColl()
  RETURNS ST_GeomCollection
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_ToMultiPoint()
  RETURNS ST_MultiPoint
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_ToMultiCurve()
  RETURNS ST_MultiCurve
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_ToMultiLine()
  RETURNS ST_MultiLineString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_ToMultiSurface()
  RETURNS ST_MultiSurface
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_ToMultiPolygon()
  RETURNS ST_MultiPolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_WKTToSQL
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_Geometry
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_AsText()
  RETURNS CHARACTER LARGE OBJECT(ST_MaxGeometryAsText)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```