

Information technology — Database languages — SQL — Multimedia application packages —

Part 3: Spatial

ICS 35.060

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW

BSI
Documents



Document Sharing Hub

National foreword

This British Standard reproduces verbatim ISO/IEC 13249-3:1999 and implements it as the UK national standard.

The UK participation in its preparation was entrusted to Technical Committee IST/40, Data management and interchange, which has the responsibility to:

- aid enquirers to understand the text;
- present to the responsible international/European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;
- monitor related international and European developments and promulgate them in the UK.

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

Cross-references

The British Standards which implement international or European publications referred to in this document may be found in the BSI Standards Catalogue under the section entitled “International Standards Correspondence Index”, or by using the “Find” facility of the BSI Standards Electronic Catalogue.

A British Standard does not purport to include all the necessary provisions of a contract. Users of British Standards are responsible for their correct application.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

Summary of pages

This document comprises a front cover, an inside front cover, the ISO/IEC title page, pages ii to xi, a blank page, pages 1 to 332, an inside back cover and a back cover.

The BSI copyright notice displayed in this document indicates when the document was last issued.

This British Standard, having been prepared under the direction of the DISC Board, was published under the authority of the Standards Committee and comes into effect on 15 March 2000

© BSI 03-2000

Amendments issued since publication

Amd. No.	Date	Comments

ISBN 0 580 34478 9

**Information technology — Database
languages — SQL Multimedia and
Application Packages —**

**Part 3:
Spatial**

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

Partie 3: Spatial

Reference number
ISO/IEC 13249-3:1999(E)



Contents	Page
Foreword	ix
Introduction	x
1 Scope	1
2 Normative references	3
3 Definitions, notations, and conventions	5
3.1 Definitions	5
3.1.1 Definitions provided in Part 1	5
3.1.2 Definitions provided in Part 3	5
3.1.3 Definitions taken from ISO/IEC 9075	6
3.1.4 Definitions taken from ISO 15046-7	7
3.1.5 Definitions taken from ISO 15046-11	9
3.2 Notations	10
3.2.1 Notations provided in Part 1	10
3.2.2 Notations provided in Part 3	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 Columns of type ST_Geometry	18
4.1.4 ST_Point	19
4.1.5 ST_Curve	19
4.1.6 ST_LineString	20
4.1.7 ST_CircularString	20
4.1.8 ST_CompoundCurve	21
4.1.9 ST_Surface	22
4.1.10 ST_CurvePolygon	22
4.1.11 ST_Polygon	23
4.1.12 ST_GeomCollection	23
4.1.13 ST_MultiPoint	23
4.1.14 ST_MultiCurve	24
4.1.15 ST_MultiLineString	24
4.1.16 ST_MultiSurface	25
4.1.17 ST_MultiPolygon	25
4.2 Spatial Reference System Type	26
4.2.1 ST_SpatialRefSys	26
4.3 Support Routines	27
4.3.1 ST_Geometry ARRAY Support Routines	27
5 Geometry Types	29
5.1 ST_Geometry Type and Routines	29
5.1.1 ST_Geometry Type	29
5.1.2 ST_Dimension Method	36
5.1.3 ST_CoordDim Method	37
5.1.4 ST_GeometryType Method	38
5.1.5 ST_SRID Methods	40
5.1.6 ST_Transform Method	41
5.1.7 ST_IsEmpty Method	42
5.1.8 ST_IsSimple Method	43
5.1.9 ST_IsValid Method	44
5.1.10 ST_Boundary Method	45

5.1.11	ST_Envelope Method.....	46
5.1.12	ST_ConvexHull Method	47
5.1.13	ST_Buffer Method	48
5.1.14	ST_Intersection Method	49
5.1.15	ST_Union Method.....	50
5.1.16	ST_Difference Method	51
5.1.17	ST_SymDifference Method.....	52
5.1.18	Return Types from ST_Intersection, ST_Union, ST_Difference, and ST_SymDifference	53
5.1.19	ST_Distance Method	56
5.1.20	ST_Equals Method.....	57
5.1.21	ST_Relate Method	58
5.1.22	ST_Disjoint Method.....	61
5.1.23	ST_Intersects Method.....	62
5.1.24	ST_Touches Method	63
5.1.25	ST_Crosses Method.....	64
5.1.26	ST_Within Method	66
5.1.27	ST_Contains Method	67
5.1.28	ST_Overlaps Method	68
5.1.29	ST_WKTTToSQL Method.....	69
5.1.30	ST_AsText Method.....	79
5.1.31	ST_WKBToSQL Method	80
5.1.32	ST_AsBinary Method.....	87
5.1.33	ST_GeomFromText Functions.....	88
5.1.34	ST_GeomFromWKB Functions.....	90
5.1.35	ST_OrderingEquals Function	92
5.1.36	SQL Transform Functions	93
6	Point Types	95
6.1	ST_Point Type and Routines	95
6.1.1	ST_Point Type	95
6.1.2	ST_Point Methods	98
6.1.3	ST_X Methods.....	100
6.1.4	ST_Y Methods.....	101
6.1.5	ST_ExplicitPoint Method	102
6.1.6	ST_PointFromText Functions	103
6.1.7	ST_PointFromWKB Functions.....	105
7	Curve Types.....	107
7.1	ST_Curve Type and Routines	107
7.1.1	ST_Curve Type	107
7.1.2	ST_Length Method.....	109
7.1.3	ST_StartPoint Method.....	110
7.1.4	ST_EndPoint Method	111
7.1.5	ST_IsClosed Method	112
7.1.6	ST_IsRing Method	113
7.1.7	ST_CurveToLine Method	114
7.2	ST_LineString Type and Routines.....	115
7.2.1	ST_LineString Type.....	115
7.2.2	ST_LineString Methods	118
7.2.3	ST_Points Methods	119
7.2.4	ST_NumPoints Method	121
7.2.5	ST_PointN Method	122
7.2.6	ST_StartPoint Method.....	123
7.2.7	ST_EndPoint Method	124
7.2.8	ST_LineFromText Functions.....	125
7.2.9	ST_LineFromWKB Functions	127
7.3	ST_CircularString Type and Routines	129
7.3.1	ST_CircularString Type	129
7.3.2	ST_CircularString Methods	133
7.3.3	ST_Points Methods	134
7.3.4	ST_NumPoints Method	136

7.3.5	ST_PointN Method	137
7.3.6	ST_MidPointRep Method	138
7.3.7	ST_StartPoint Method	139
7.3.8	ST_EndPoint Method	140
7.3.9	ST_CircularFromText Functions	141
7.3.10	ST_CircularFromWKB Functions	143
7.4	ST_CompoundCurve Type and Routines	145
7.4.1	ST_CompoundCurve Type	145
7.4.2	ST_CompoundCurve Methods	148
7.4.3	ST_Curves Methods	150
7.4.4	ST_NumCurves Method	152
7.4.5	ST_CurveN Method	153
7.4.6	ST_StartPoint Method	154
7.4.7	ST_EndPoint Method	155
7.4.8	ST_CompoundFromText Functions	156
7.4.9	ST_CompoundFromWKB Functions	158
8	Surface Types	161
8.1	ST_Surface Type and Routines	161
8.1.1	ST_Surface Type	161
8.1.2	ST_Area Method	163
8.1.3	ST_Perimeter Method	164
8.1.4	ST_Centroid Method	165
8.1.5	ST_PointOnSurface Method	166
8.2	ST_CurvePolygon Type and Routines	167
8.2.1	ST_CurvePolygon Type	167
8.2.2	ST_CurvePolygon Methods	171
8.2.3	ST_ExteriorRing Methods	173
8.2.4	ST_InteriorRings Methods	175
8.2.5	ST_NumInteriorRing Method	178
8.2.6	ST_InteriorRingN Method	179
8.2.7	ST_CurvePolyToPoly Method	180
8.2.8	ST_CPolyFromText Functions	181
8.2.9	ST_CPolyFromWKB Functions	183
8.3	ST_Polygon Type and Routines	185
8.3.1	ST_Polygon Type	185
8.3.2	ST_Polygon Methods	188
8.3.3	ST_ExteriorRing Methods	191
8.3.4	ST_InteriorRings Methods	192
8.3.5	ST_InteriorRingN Method	194
8.3.6	ST_PolyFromText Functions	195
8.3.7	ST_PolyFromWKB Functions	197
8.3.8	ST_BdPolyFromText Functions	199
8.3.9	ST_BdPolyFromWKB Functions	201
9	Geometry Collection Types	203
9.1	ST_GeomCollection Type and Routines	203
9.1.1	ST_GeomCollection Type	203
9.1.2	ST_GeomCollection Methods	206
9.1.3	ST_Geometries Methods	208
9.1.4	ST_NumGeometries Method	210
9.1.5	ST_GeometryN Method	211
9.1.6	ST_GeomCollFromTxt Functions	212
9.1.7	ST_GeomCollFromWKB Functions	214
9.2	ST_MultiPoint Type and Routines	216
9.2.1	ST_MultiPoint Type	216
9.2.2	ST_MultiPoint Methods	218
9.2.3	ST_Geometries Methods	219
9.2.4	ST_MPointFromText Functions	221
9.2.5	ST_MPointFromWKB Functions	223
9.3	ST_MultiCurve Type and Routines	225

9.3.1	ST_MultiCurve Type.....	225
9.3.2	ST_MultiCurve Methods	227
9.3.3	ST_IsClosed Method	228
9.3.4	ST_Length Method	229
9.3.5	ST_Geometries Methods	230
9.3.6	ST_MCurveFromText Functions.....	232
9.3.7	ST_MCurveFromWKB Functions.....	234
9.4	ST_MultiLineString Type and Routines	236
9.4.1	ST_MultiLineString Type	236
9.4.2	ST_MultiLineString Methods.....	238
9.4.3	ST_Geometries Methods	239
9.4.4	ST_MLineFromText Functions.....	241
9.4.5	ST_MLineFromWKB Functions	243
9.5	ST_MultiSurface Type and Routines.....	245
9.5.1	ST_MultiSurface Type.....	245
9.5.2	ST_MultiSurface Methods	247
9.5.3	ST_Area Method	248
9.5.4	ST_Perimeter Method	249
9.5.5	ST_Centroid Method	250
9.5.6	ST_PointOnSurface Method.....	251
9.5.7	ST_Geometries Methods	252
9.5.8	ST_MSurfaceFromTxt Functions.....	254
9.5.9	ST_MSurfaceFromWKB Functions.....	256
9.6	ST_MultiPolygon Type and Routines.....	258
9.6.1	ST_MultiPolygon Type.....	258
9.6.2	ST_MultiPolygon Methods	260
9.6.3	ST_Geometries Methods	261
9.6.4	ST_MPolyFromText Functions	263
9.6.5	ST_MPolyFromWKB Functions	265
9.6.6	ST_BdMPolyFromText Functions.....	267
9.6.7	ST_BdMPolyFromWKB Functions	269
10	Spatial Reference System Type.....	271
10.1	ST_SpatialRefSys Type and Routines	271
10.1.1	ST_SpatialRefSys Type	271
10.1.2	ST_SpatialRefSys Methods.....	273
10.1.3	ST_AsWKTSRS Method.....	278
10.1.4	ST_WKTSRSToSQL Method	279
10.1.5	ST_SRID Method	280
10.1.6	ST_Equals Method	281
10.1.7	ST_OrderingEquals Function	282
10.1.8	ST_WellKnownText SQL Transform Group.....	283
11	Support Routines	285
11.1	ST_Geometry ARRAY Support Routines.....	285
11.1.1	ST_MinDimension Function.....	285
11.1.2	ST_MaxDimension Function	287
11.1.3	ST_IsMixedDim Function	289
11.1.4	ST_CheckSRID Function	290
11.1.5	ST_CheckNulls Procedure	292
11.1.6	ST_CheckConsecDups Procedure	293
11.1.7	ST_ToPointAny Cast Function	294
11.1.8	ST_ToCurveAny Cast Function	296
11.1.9	ST_ToLineStringAny Cast Function	298
11.1.10	ST_ToCircularAny Cast Function.....	300
11.1.11	ST_ToCompoundAny Cast Function	302
11.1.12	ST_ToSurfaceAny Cast Function	304
11.1.13	ST_ToCurvePolyAny Cast Function	306
11.1.14	ST_ToPolygonAny Cast Function	308
12	Conformance	311
12.1	Requirements for conformance	311

12.2	Claims of conformance.....	311
13	Status Codes	315
Annex A		317
A.1	Implementation-defined Meta-variables.....	318
Annex B		319
Annex C		321
Bibliography		323
Index		325

Figures	Page
Figure C.1 — ST_Geometry Type Hierarchy Diagram	321

Tables	Page
Table 1 — Symbols	10
Table 2 — DE-9IM	14
Table 3 — Parameter Types	53
Table 4 — Return Type Sets.....	54
Table 5 — Return Type Matrix for the ST_Intersection Method	54
Table 6 — Return Type Matrix for the ST_Union Method.....	55
Table 7 — Return Type Matrix for the ST_Difference Method	55
Table 8 — Return Type Matrix for the ST_SymDifference Method	55
Table 9 — DE-9IM Mapping	60
Table 10 — Cell Values	60
Table 11 — SQLSTATE class and subclass values	315

Foreword

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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

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

ISO/IEC 13249 consists of the following parts, under the general title *Information technology — Database languages — SQL Multimedia and Application Packages*:

- *Part 1: Framework*
- *Part 2: Full text*
- *Part 3: Spatial*
- *Part 4: General purpose facilities*
- *Part 5: Still image*

Annexes A to C of this part of ISO/IEC 13249 are for information only.

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.

SQL/MM is structured as a multi-part standard. At present it consists of the following parts:

- Part 1: Framework
- Part 2: Full text
- Part 3: Spatial
- Part 4: General purpose facilities
- Part 5: Still image

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 referencing systems.
- 11) Clause 11, "Support Routines", defines supporting functions and procedures used by this part of ISO/IEC 13249.
- 12) Clause 12, "Conformance", defines the criteria for conformance to this part of ISO/IEC 13249.
- 13) Clause 13, "Status Codes", defines the SQLSTATE codes used in this part of ISO/IEC 13249.
- 14) 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 implementor shall provide in each case.
- 15) 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.

16) Annex C, "ST_Geometry Type Hierarchy", is an informative Annex. It visually describes the inheritance relationship between user-defined type in this part of ISO/IEC 13249.

17) 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, "Support Routines", subclauses begin a new page. Any resulting blank space is not significant.

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, geoenvironment, graphics, multimedia, and resource management applications.

Blank page

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 13249. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 13249 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative documents referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 9075-1:1999, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*.

ISO/IEC 9075-2:1999, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*.

ISO/IEC 9075-4:1999, *Information technology — Database languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM)*.

ISO/IEC 13249-1, *Information technology — Database languages — SQL Multimedia and Application*.

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

Blank page

3 Definitions, notations, and conventions

3.1 Definitions

3.1.1 Definitions provided in Part 1

This part of ISO/IEC 13249 makes use of all terms defined in ISO/IEC 13249-1.

3.1.2 Definitions provided in Part 3

For the purposes of this part of ISO/IEC 13249, the following 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 value with a geometric dimension of 2

3.1.2.4

boundary of a curve

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

3.1.2.5

boundary of a point

the empty set

3.1.2.6

boundary of a surface

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

3.1.2.7

closed curve

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

3.1.2.8

closure

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

3.1.2.9

dimension

geometric dimension

3.1.2.10

geometry

the shape and geographic location of a feature

3.1.2.11

intersection

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

3.1.2.12

linear ring

a linestring value that is closed and simple

3.1.2.13

linestring

a curve with linear interpolation between control points

3.1.2.14

non-closed curve

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

3.1.2.15

polygon

a surface that uses linear rings to define its boundary

3.1.2.16

point set

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

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

3.1.2.17

ring

an curve value that is closed and simple

3.1.2.18

spatially equals

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

NOTE 1 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 2 Since an infinite set of points cannot be tested, the internal representation of equal must test for equivalents between two, possibly quite different, representations. The test may be limited to the resolution of the spatial referencing 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.

3.1.2.19

topologically closed

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

3.1.3 Definitions taken from ISO/IEC 9075

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

- a) immediately contained.

3.1.4 Definitions taken from ISO 15046-7

This part of ISO/IEC 13249 makes use of the following terms defined in ISO 15046-7.

NOTE At the time of publication of this part of ISO 13249, ISO 15046-7 was at Committee Draft stage in the ISO process.

3.1.4.1

buffer

geometric object that contains all points whose distance from another geometric object is less than or equal to a given distance

3.1.4.2

computational topology, algebraic topology

topological concepts, structures and algebra that aid, enhance or define operations usually performed in computational geometry

3.1.4.3

connected

property of a geometric object implying that any two points on the object can be connected by a curve that remains within the object

3.1.4.4

convex hull of a geometric object

smallest convex set containing the geometric set associated with the geometric object

3.1.4.5

coordinate dimension

number of measurements or ordinates needed to describe a position in a coordinate system

3.1.4.6

coordinate

one of a set of N numbers designating the position of a point in N -dimensional space

3.1.4.7

coordinate system

set of (mathematical) rules for specifying how coordinates are to be assigned to points

3.1.4.8

coordinate reference system

coordinate system which is related to the earth by a datum

3.1.4.9

curve

bounded, connected 1-dimensional geometric primitive, representing the continuous image of a line, and therefore fully realizable as a 1-parameter set of points

3.1.4.10

end point

last point of a curve

3.1.4.11

geometric complex

set of disjoint geometric primitives such that the geometric boundary of each geometric primitive can be represented as the union of other geometric primitives within the geometric complex

3.1.4.12

geometric dimension

largest number n such that each direct position in a geometric set is associated with a subset that has the direct position in its interior and is similar (isomorphic) to $\mathbb{R}^n$, Euclidean n -space

3.1.4.13

geometric object

geometric primitive, collection of geometric primitives, or geometric complex treated as a single entity

3.1.4.14

geometric primitive

object representing a single, connected, homogeneous element of geometry

3.1.4.15

homomorphism

relationship between two objects (two complexes) such that there is a structure preserving function from one to the other

3.1.4.16

interior (of a geometric object)

set of all points that are on the geometric object but which are not on its geometric boundary

3.1.4.17

isomorphism

relationship between two objects (two complexes) such that there is a one to one, structure preserving function from one onto the other

3.1.4.18

point

0-dimensional geometric primitive, representing a position, but not having extent

3.1.4.19

start point

first point of a curve

3.1.4.20

surface

bounded, connected 2-dimensional geometric primitive, representing the continuous image of region of a plane, and therefore fully realizable locally as a 2-parameter set of points

3.1.5 Definitions taken from ISO 15046-11

This part of ISO/IEC 13249 makes use of the following terms defined in ISO 15046-11.

NOTE At the time of publication of this part of ISO 13249, ISO 15046-11 was at Committee Draft stage in the ISO process.

3.1.5.1

datum

any quantity or set of quantities that may serve as a reference or basis for the calculation of other quantities

3.1.5.2

ellipsoid

figure formed by the rotation of an ellipse about an axis

3.1.5.3

flattening

the ratio of the difference between the semi-major and semi-minor axis of an ellipsoid to the semi-major axis of an ellipsoid

3.1.5.4

geodetic coordinate system, ellipsoidal coordinate system

coordinate system in which position is specified by latitude, longitude and (in the three-dimensional case) ellipsoidal height

3.1.5.5

meridian

intersection between an ellipsoid and a plane containing the semi-minor axis of the ellipsoid

3.1.5.6

prime meridian, zero meridian

meridian from which the longitudes of other meridians are quantified

3.1.5.7

projected coordinate system

two-dimensional Cartesian coordinate system resulting from a map projection

3.1.5.8

semi-major axis

half of the longest diameter of an ellipsoid

3.1.5.9

semi-minor axis

half of the shortest diameter of an ellipsoid

3.1.5.10

unit

defined quantity in which dimensioned parameters are expressed

3.2 Notations

3.2.1 Notations provided in Part 1

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

3.2.2 Notations provided in Part 3

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.

4 Concepts

4.1 Geometry Types

The following geometry types are supported: 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. The geometry type hierarchy is visually described in Annex C, "ST_Geometry Type Hierarchy".

ST_Geometry, ST_Curve, and ST_Surface are not instantiable types. No implicitly defined 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 implicitly defined 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 instance 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 root type of all geometry types. 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 ($\mathbb{R}^2$).

NOTE A future edition of this part of ISO/IEC 13249 may deal with n -dimensional coordinate space ($\mathbb{R}^n$) 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 sub-type 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_Boundary: returns the boundary of an ST_Geometry value.
- 9) ST_Envelope: returns the bounding rectangle of an ST_Geometry value.
- 10) ST_ConvexHull: returns the convex hull of an ST_Geometry value.
- 11) ST_Buffer: returns the ST_Geometry value that represents all points whose distance from an ST_Geometry value is less than or equal to a specified distance.
- 12) 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:

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

- 13) 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:

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

- 14) 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:

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

- 15) 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:

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

- 16) ST_Distance: returns the minimum shortest distance between any two points in the two ST_Geometry values
- 17) ST_WKTToSQL: returns the ST_Geometry value for the specified well-known text representation.
- 18) ST_AsText: returns the well-known text representation for the specified ST_Geometry value.
- 19) ST_WKBToSQL: returns the ST_Geometry value for the specified well-known binary representation.
- 20) ST_AsBinary: returns the well-known binary representation for the specified ST_Geometry value.

4.1.1.2 Functions on ST_Geometry

- 1) ST_GeomFromText: transforms a CHARACTER LARGE OBJECT to an ST_Geometry value.
- 2) ST_GeomFromWKB: transforms a BINARY LARGE OBJECT 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.

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 must be 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 empty. 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(a), Boundary(a) and Exterior(a) 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 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	(Interior(a) $\cap$ Interior(b)). ST_Dimension	(Interior(a) $\cap$ Boundary(b)). ST_Dimension	(Interior(a) $\cap$ Exterior(b)). ST_Dimension
Boundary	(Boundary(a) $\cap$ Interior(b)). ST_Dimension	(Boundary(a) $\cap$ Boundary(b)). ST_Dimension	(Boundary(a) $\cap$ Exterior(b)). ST_Dimension
Exterior	(Exterior(a) $\cap$ Interior(b)). ST_Dimension	(Exterior(a) $\cap$ Boundary(b)). ST_Dimension	(Exterior(a) $\cap$ 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:

if $p = T$, then x .ST_Dimension $\in \{0, 1, 2\}$, i.e. $x \neq \emptyset$

if $p = F$, then x .ST_Dimension = -1, i.e. $x = \emptyset$

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

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

if $p = 2$, then x .ST_Dimension = 2

if $p = *$, then x .ST_Dimension $\in \{-1, 0, 1, 2\}$, i.e. any value

The pattern matrix can be represented as an <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:

$a.ST_Intersects(b) \Leftrightarrow$
 Case $a.ST_Disjoint(b)$ when 0 then 1 when 1 then 0 else NULL end

- 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: $a.ST_Touches(b) \Leftrightarrow (Interior(a) \cap Interior(b) = \emptyset) \wedge (a \cap b) \neq \emptyset$

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:

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

- 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 $(a.ST_Dimension = 0$ (zero) and $b.ST_Dimension = 0$ (zero)) or
 $(a.ST_Dimension = 2$ and $b.ST_Dimension = 2)$, then the null value

b) Otherwise:

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

Expressed in terms of the DE-9IM:

Case:

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

$$a.ST_Crosses(b) \Leftrightarrow \\ (Interior(a) \cap Interior(b) \neq \emptyset) \wedge (Interior(a) \cap Exterior(b) \neq \emptyset) \Leftrightarrow \\ a.ST_Relate(b, 'T*T*****')$$

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

$$a.ST_Crosses(b) \Leftrightarrow \\ (Interior(a) \cap Interior(b) \neq \emptyset) \wedge (Interior(b) \cap Exterior(a) \neq \emptyset) \Leftrightarrow \\ b.ST_Relate(a, 'T*T*****')$$

- d) If $a.ST_Dimension = 1$ (one) and $b.ST_Dimension = 1$ (one), then:

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

- 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:

$$a.ST_Within(b) \Leftrightarrow (a \cap b = a) \wedge (Interior(a) \cap Exterior(b) \neq \emptyset)$$

Expressed in terms of the DE-9IM:

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

- 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:

$$a.ST_Overlaps(b) \Leftrightarrow \\ (Interior(a).ST_Dimension = Interior(b).ST_Dimension = \\ (Interior(a) \cap Interior(b)).ST_Dimension) \wedge (a \cap b \neq a) \wedge (a \cap b \neq b)$$

- b) Otherwise: the null value

Expressed in terms of the DE-9IM:

Case:

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

$$\begin{aligned} a.ST_Overlaps(b) \Leftrightarrow & \\ & (Interior(a) \cap Interior(b) \neq \emptyset) \wedge \\ & (Interior(a) \cap Exterior(b) \neq \emptyset) \wedge \\ & (Exterior(a) \cap Interior(b) \neq \emptyset) \Leftrightarrow \\ & a.ST_Relate(b, 'T*T***T**') \end{aligned}$$

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

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

- c) Otherwise: the null value

4.1.3 Columns of type ST_Geometry

4.1.3.1 GEOMETRY_COLUMNS Table (or View)

For every catalog of an SQL-environment, for every schema thereof, for every table thereof, and for every column thereof the declared type of which is ST_Geometry, the GEOMETRY_COLUMNS table (or view) provides additional information on the spatial reference system of that column. There is a GEOMETRY_COLUMNS table descriptor in every schema that includes the descriptor of the ST_Geometry type; this table (or view) contains rows for each column whose declared type is ST_Geometry.

The following CREATE TABLE statement creates an appropriately structured table. This should be either an actual table or an updatable view so that insertion of geometry column information can be done directly with SQL.

```
CREATE TABLE GEOMETRY_COLUMNS (
    F_TABLE_CATALOG CHARACTER VARYING(256) NOT NULL,
    F_TABLE_SCHEMA CHARACTER VARYING(256) NOT NULL,
    F_TABLE_NAME CHARACTER VARYING(256) NOT NULL,
    F_GEOMETRY_COLUMN CHARACTER VARYING(256) NOT NULL,
    COORD_DIMENSION INTEGER,
    SRID INTEGER REFERENCES SPATIAL_REF_SYS,
    CONSTRAINT GC_PK PRIMARY KEY
        (F_TABLE_CATALOG, F_TABLE_SCHEMA, F_TABLE_NAME, F_GEOMETRY_COLUMN)
```

The meanings of the columns in the GEOMETRY_COLUMNS table (or view) are:

- a) The values F_TABLE_CATALOG, F_TABLE_SCHEMA, F_TABLE_NAME, and F_GEOMETRY_COLUMN are the catalog name, unqualified schema name, qualified identifier, and column name, respectively, of a column identified by a <column reference> (See Subclause 5.4, "Names and identifiers" in ISO/IEC 9075-2). The <column reference> identifies a geometry column in a feature table.
- b) COORD_DIMENSION is the number of coordinates used in the ST_Geometry values, usually corresponding to the number of dimensions in the spatial reference system.

- c) SRID is spatial reference system identifier used for the coordinate geometry in this table. It is a foreign key reference to the SPATIAL_REF_SYS table (or view).

4.1.3.2 Tables with column of type ST_Geometry

A table may be created with a column of ST_Geometry or one of its subtypes. Restrictions for these columns are modeled using constraints.

The following is the general form of such a table:

```
CREATE TABLE table-name (
  -- ...
  geometry-column-1 geometry-type,
  -- ...
  CONSTRAINT SRS-1 CHECK (
    geometry-column-1.ST_SRID IN (
      SELECT SRID
      FROM GEOMETRY_COLUMNS
      WHERE F_TABLE_CATALOG = 'catalog'
      AND F_TABLE_SCHEMA = 'schema'
      AND F_TABLE_NAME = 'table-name'
      AND F_GEOMETRY_COLUMN = 'geometry-column-1'))
  -- ...)
```

geometry-column-1 is the column name of a column of type ST_Geometry. *geometry-type* is ST_Geometry or one of its subtypes. In this general form, there is a constraint, *SRS-1* to restrict the ST_Geometry values in a column to a given spatial reference system.

4.1.4 ST_Point

The ST_Point type is a subtype of ST_Geometry. 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.

4.1.4.1 Methods on ST_Point

- 1) ST_Point: returns the specified ST_Point value.
- 2) ST_X: observes and mutates the x coordinate value of an ST_Point value.
- 3) ST_Y: observes and mutates the y coordinate value of an ST_Point value.
- 4) ST_ExplicitPoint: returns the coordinate values as a DOUBLE PRECISION ARRAY value.

4.1.4.2 Functions on ST_Point

- 1) ST_PointFromText: transforms a CHARACTER LARGE OBJECT to an ST_Point value.
- 2) ST_PointFromWKB: transforms a BINARY LARGE OBJECT to an ST_Point value.

4.1.5 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 simple if it does not pass through the same point twice.

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 a start point and an end point. An ST_Curve value is defined to be topologically closed.

4.1.5.1 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.

4.1.6 ST_LineString

The ST_LineString type is a subtype of ST_Curve. 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.

4.1.6.1 Methods on ST_LineString

- 1) ST_LineString: returns the specified ST_LineString value.
- 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.

4.1.6.2 Functions on ST_LineString

- 1) ST_LineFromText: transforms a CHARACTER LARGE OBJECT to an ST_LineString value.
- 2) ST_LineFromWKB: transforms a BINARY LARGE OBJECT to an ST_LineString value.

4.1.7 ST_CircularString

The ST_CircularString type is a subtype of ST_Curve. An ST_CircularString value has circular interpolation between ST_Point values. This type 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 and 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 colinear, 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.

4.1.7.1 Methods on ST_CircularString

- 1) ST_CircularString: returns the specified ST_CircularString value.
- 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.

4.1.7.2 Functions on ST_CircularString

- 1) ST_CircularFromText: transforms a CHARACTER LARGE OBJECT to an ST_CircularString value.
- 2) ST_CircularFromWKB: transforms a BINARY LARGE OBJECT to an ST_CircularString value.

4.1.8 ST_CompoundCurve

The ST_CompoundCurve type is a subtype of ST_Curve. The general notion of a compound curve is a sequence of contiguous curves such that adjacent curves are joined at their end points. The contributing curve types are limited to line strings and circular strings. Furthermore, the end point of each curve must be equal to the start point of the next curve in the list.

4.1.8.1 Methods on ST_CompoundCurve

- 1) ST_CompoundCurve: returns the specified ST_CompoundCurve value.
- 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.

4.1.8.2 Functions on ST_CompoundCurve

- 1) ST_CompoundFromText: transforms a CHARACTER LARGE OBJECT to an ST_CompoundCurve value.
- 2) ST_CompoundFromWKB: transforms a BINARY LARGE OBJECT to an ST_CompoundCurve value.

4.1.9 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.

4.1.9.1 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 then ST_Point value that is guaranteed to be on the ST_Surface value.

4.1.10 ST_CurvePolygon

The ST_CurvePolygon type is a subtype of ST_Surface. 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 may 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 connected component of the exterior.

ST_CurvePolygon values are simple.

4.1.10.1 Methods on ST_CurvePolygon

- 1) ST_CurvePolygon: returns the specified ST_CurvePolygon value.
- 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.

4.1.10.2 Functions on ST_CurvePolygon

- 1) ST_CPolyFromText: transforms a CHARACTER LARGE OBJECT to an ST_CurvePolygon value.
- 2) ST_CPolyFromWKB: transforms a BINARY LARGE OBJECT to an ST_CurvePolygon value.

4.1.11 ST_Polygon

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

4.1.11.1 Methods on ST_Polygon

- 1) ST_Polygon: returns the specified ST_Polygon value.

4.1.11.2 Functions on ST_Polygon

- 1) ST_PolyFromText: transforms a CHARACTER LARGE OBJECT to an ST_Polygon value.
- 2) ST_PolyFromWKB: transforms a BINARY LARGE OBJECT to an ST_Polygon value.
- 3) ST_BdPolyFromText: builds an ST_Polygon value from a well-known text representation of an ST_MultiLineString.
- 4) ST_BdPolyFromWKB: builds an ST_Polygon value from a well-known binary representation of an ST_MultiLineString.

4.1.12 ST_GeomCollection

The ST_GeomCollection type is a subtype of ST_Geometry. 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. Subclasses of ST_GeomCollection may restrict membership based on dimension or place other constraints on the degree of spatial overlap between elements.

4.1.12.1 Methods on ST_GeomCollection

- 1) ST_GeomCollection: returns the specified ST_GeomCollection value.
- 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.

4.1.12.2 Functions on ST_GeomCollection

- 1) ST_GeomCollFromTxt: transforms a CHARACTER LARGE OBJECT to an ST_GeomCollection value.
- 2) ST_GeomCollFromWKB: transforms a BINARY LARGE OBJECT to an ST_GeomCollection value.

4.1.13 ST_MultiPoint

The ST_MultiPoint type is a subtype of ST_GeomCollection. 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 no two ST_Point values in the ST_MultiPoint value are equal. The boundary of an ST_MultiPoint is the empty set.

4.1.13.1 Methods on ST_MultiPoint

- 1) ST_MultiPoint returns the specified ST_MultiPoint value.

4.1.13.2 Functions on ST_MultiPoint

- 1) ST_MPointFromText: transforms a CHARACTER LARGE OBJECT to an ST_MultiPoint value.
- 2) ST_MPointFromWKB: transforms a BINARY LARGE OBJECT to an ST_MultiPoint value.

4.1.14 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.

4.1.14.1 Methods on ST_MultiCurve

- 1) ST_MultiCurve: returns the specified ST_MultiCurve value.
- 2) ST_IsClosed: tests if an ST_MultiCurve value is closed.
- 3) ST_Length: returns the length of an ST_MultiCurve value.

4.1.14.2 Functions on ST_MultiCurve

- 1) ST_MCurveFromText: transforms a CHARACTER LARGE OBJECT to an ST_MultiCurve value.
- 2) ST_MCurveFromWKB: transforms a BINARY LARGE OBJECT to an ST_MultiCurve value.

4.1.15 ST_MultiLineString

The ST_MultiLineString type is a subtype of ST_Curve. The elements of an ST_MultiLineString are restricted to ST_LineString values.

4.1.15.1 Methods on ST_MultiLineString

- 1) ST_MultiLineString: returns the specified ST_MultiLineString value.

4.1.15.2 Functions on ST_MultiLineString

- 1) ST_MLineFromText: transforms a CHARACTER LARGE OBJECT to an ST_MultiLineString value.
- 2) ST_MLineFromWKB: transforms a BINARY LARGE OBJECT to an ST_MultiLineString value.

4.1.16 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 may not intersect. The boundaries of any two elements in an ST_MultiSurface may intersect at a finite number of ST_Point values.

4.1.16.1 Methods on ST_MultiSurface

- 1) ST_MultiSurface: returns the specified ST_MultiSurface value.
- 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.

4.1.16.2 Functions on ST_MultiSurface

- 1) ST_MSurfaceFromText: transforms a CHARACTER LARGE OBJECT to an ST_MultiSurface value.
- 2) ST_MSurfaceFromWKB: transforms a BINARY LARGE OBJECT to an ST_MultiSurface value.

4.1.17 ST_MultiPolygon

The ST_MultiPolygon type is a subtype of ST_MultiSurface. The elements of an ST_MultiPolygon value are restricted to ST_Polygon values. The interiors of distinct element of an ST_MultiPolygon do not intersect. The interiors of two ST_Polygon values that are elements of an ST_MultiPolygon may not intersect. The boundaries of any two ST_Polygon values that are elements of an ST_MultiPolygon may 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 may not have cut lines, spikes or punctures. An ST_MultiPolygon value is a 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 connected 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.

4.1.17.1 Methods on ST_MultiPolygon

- 1) ST_MultiPolygon: returns the specified ST_MultiPolygon value.

4.1.17.2 Functions on ST_MultiPolygon

- 1) ST_MPolyFromText: transforms a CHARACTER LARGE OBJECT to an ST_MultiPolygon value.
- 2) ST_MPolyFromWKB: transforms a BINARY LARGE OBJECT to an ST_MultiPolygon value.
- 3) ST_BdMPolyFromText: builds an ST_MultiPolygon value from a well-known text representation of an ST_MultiLineString.

- 4) ST_BdMPolyFromWKB: builds an ST_MultiPolygon value 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.

4.2.1.4 SPATIAL_REF_SYS Table (or View)

There is a SPATIAL_REF_SYS table (or view) in every schema containing the ST_Geometry and ST_SpatialRefSys type definitions. The SPATIAL_REF_SYS table (or view) contains information on each spatial reference system used to specify values of the given schema's ST_Geometry or ST_SpatialRefSys types.

The following CREATE TABLE statement creates an appropriately structured SPATIAL_REF_SYS. This should be either an actual table or an updatable view so that insertion of spatial reference system information can be done directly with SQL.

```
CREATE TABLE SPATIAL_REF_SYS (  
    SRID INTEGER NOT NULL PRIMARY KEY,  
    AUTH_NAME CHARACTER VARYING(256) ,  
    AUTH_SRID INTEGER,  
    SRTEXT CHARACTER VARYING(2048) )
```

The meanings of the columns in the SPATIAL_REF_SYS table (or view) are:

- a) SRID is the spatial reference system identifier.
- b) AUTH_NAME is the name of the standard or standards body that is being cited for this reference system.

- c) AUTH_SRID is the identifier of the spatial reference system as defined by the Authority cited in AUTH_NAME.
- d) SRTEXT is the <spatial reference system> of the spatial reference system.

4.3 Support Routines

4.3.1 ST_Geometry ARRAY Support Routines

4.3.1.1 Support Functions

- 1) ST_MinDimension: returns the minimum dimension value in an ST_Geometry ARRAY value.
- 2) ST_MaxDimension: returns the maximum dimension value in an ST_Geometry ARRAY value.
- 3) ST_IsMixedDim: tests if an ST_Geometry ARRAY value has ST_Geometry values with different dimensions.
- 4) ST_CheckSRID: if the elements in the ST_Geometry ARRAY value have mixed spatial reference systems, then raises an exception. Otherwise, returns the spatial reference system identifier of the ST_Geometry elements.

4.3.1.2 Supporting Procedures

- 1) ST_CheckNulls: raises an exception if an ST_Geometry ARRAY value is the null value, contains null elements, or has mixed spatial reference systems.
- 2) ST_CheckConsecDups: raises an exception if an ST_Geometry ARRAY value has consecutive duplicate values.

4.3.1.3 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.

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 root type for geometry types. 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(128)
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(asrid INTEGER)
RETURNS ST_Geometry
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
CALLED ON NULL INPUT,
```

```
METHOD ST_Transform(asrid 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_Intersection(ageometry ST_Geometry)
  RETURNS ST_Geometry
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

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

```

```

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

```

```

METHOD ST_SymDifference(ageometry ST_Geometry)
  RETURNS ST_Geometry
  SELF AS RESULT
  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_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_WKTToSQL(CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_Geometry
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

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

```

METHOD ST_WKBToSQL(BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_Geometry
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_AsBinary()
  RETURNS BINARY LARGE OBJECT(ST_MaxGeometryAsBinary)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT

```

Definitional Rules

- 1) *ST_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.
- 2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.
- 3) The attribute *ST_PrivateDimension* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST_PrivateDimension*.
- 4) The attribute *ST_PrivateCoordinateDimension* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST_PrivateCoordinateDimension*.

Description

- 1) The *ST_Geometry* type provides for public use:
 - a) a method *ST_Dimension()*,
 - b) a method *ST_CoordDim()*,
 - c) a method *ST_GeometryType()*,
 - d) a method *ST_SRID()*,
 - e) a method *ST_SRID(INTEGER)*,
 - f) a method *ST_Transform(INTEGER)*,
 - g) a method *ST_IsEmpty()*,
 - h) a method *ST_IsSimple()*,
 - i) a method *ST_IsValid()*,
 - j) a method *ST_Boundary()*,
 - k) a method *ST_Envelope()*,
 - l) a method *ST_ConvexHull()*,
 - m) a method *ST_Buffer(DOUBLE PRECISION)*,
 - n) a method *ST_Intersection(ST_Geometry)*,

- o) a method *ST_Union*(*ST_Geometry*),
 - p) a method *ST_Difference*(*ST_Geometry*),
 - q) a method *ST_SymDifference*(*ST_Geometry*),
 - r) a method *ST_Distance*(*ST_Geometry*),
 - s) a method *ST_Equals*(*ST_Geometry*),
 - t) a method *ST_Relate*(*ST_Geometry*, *CHARACTER*),
 - u) a method *ST_Disjoint*(*ST_Geometry*),
 - v) a method *ST_Intersects*(*ST_Geometry*),
 - w) a method *ST_Touches*(*ST_Geometry*),
 - x) a method *ST_Crosses*(*ST_Geometry*),
 - y) a method *ST_Within*(*ST_Geometry*),
 - z) a method *ST_Contains*(*ST_Geometry*),
 - aa) a method *ST_Overlaps*(*ST_Geometry*),
 - bb) a method *ST_WKTTToSQL*(*CHARACTER LARGE OBJECT*),
 - cc) a method *ST_AsText*(),
 - dd) a method *ST_WKBTToSQL*(*BINARY LARGE OBJECT*),
 - ee) a method *ST_AsBinary*(),
 - ff) a function *ST_GeomFromText*(*CHARACTER LARGE OBJECT*),
 - gg) a function *ST_GeomFromText*(*CHARACTER LARGE OBJECT*, *INTEGER*),
 - hh) a function *ST_GeomFromWKB*(*BINARY LARGE OBJECT*),
 - ii) a function *ST_GeomFromWKB*(*BINARY LARGE OBJECT*, *INTEGER*),
 - jj) an ordering function *ST_OrderingEquals*(*ST_Geometry*, *ST_Geometry*),
 - kk) an SQL Transform group *ST_WellKnownText*,
 - ll) an SQL Transform group *ST_WellKnownBinary*.
- 2) The *ST_PrivateDimension* attribute contains the dimension of the *ST_Geometry* value:
- Case:
- a) If the *ST_Geometry* value corresponds to the empty set, then the dimension is -1.
 - b) If the *ST_Geometry* value is 0-dimensional geometry, then the dimension is 0 (zero).
 - c) If the *ST_Geometry* value is 1-dimensional geometry, then the dimension is 1 (one).
 - d) If the *ST_Geometry* value is 2-dimensional geometry, then the dimension is 2.

- 3) The *ST_PrivateCoordinateDimension* attribute contains the coordinate dimension of the *ST_Geometry* value.
- 4) The *ST_PrivateCoordinateDimension* attribute shall be 2 for *ST_Geometry* values in two-dimensional coordinate space (R^2).
- 5) The *ST_PrivateDimension* attribute shall be less than or equal to the *ST_PrivateCoordinateDimension* attribute.
- 6) The *ST_Geometry* type is the root type of the geometry hierarchy.
- 7) All instantiable *ST_Geometry* subtypes are defined so that simple values of the geometry type are topologically closed.
- 8) The coordinate dimension shall be the same as the coordinate dimension of the spatial reference system for the *ST_Geometry* value.
- 9) An *ST_Geometry* value has an associated spatial reference system specified by a spatial reference system identifier.

5.1.2 ST_Dimension Method

Purpose

Return the dimension of the ST_Geometry value.

Definition

```
CREATE METHOD ST_Dimension()  
  RETURNS SMALLINT  
  FOR ST_Geometry  
  RETURN SELF.ST_PrivateDimension
```

Description

- 1) The method *ST_Dimension()* has no input parameters.
- 2) The null-call method *ST_Dimension()* returns the value of the *ST_PrivateDimension* attribute.