

---

---

**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*

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

**PDF disclaimer**

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

Adobe is a trademark of Adobe Systems Incorporated.

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

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

© ISO/IEC 2003

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

ISO copyright office  
Case postale 56 • CH-1211 Geneva 20  
Tel. + 41 22 749 01 11  
Fax + 41 22 749 09 47  
E-mail [copyright@iso.org](mailto:copyright@iso.org)  
Web [www.iso.org](http://www.iso.org)

Published in Switzerland

Contents	Page
Foreword .....	ix
Introduction .....	x
<b>1 Scope</b> .....	<b>1</b>
<b>2 Normative references</b> .....	<b>3</b>
2.1 ISO/IEC JTC 1 standards .....	3
2.2 ISO standards .....	3
2.3 IEC standards .....	3
2.4 Other international standards .....	3
<b>3 Terms and definitions, notations and conventions</b> .....	<b>5</b>
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
<b>4 Concepts</b> .....	<b>11</b>
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
<b>5 Geometry Types</b> .....	<b>33</b>
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_WKTTToSQL 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_ToPointAry Cast Function.....	380
12.1.6	ST_ToCurveAry Cast Function.....	382
12.1.7	ST_ToLineStringAry Cast Function.....	384
12.1.8	ST_ToCircularAry Cast Function.....	386
12.1.9	ST_ToCompoundAry Cast Function.....	388
12.1.10	ST_ToSurfaceAry Cast Function.....	390
12.1.11	ST_ToCurvePolyAry Cast Function.....	392
12.1.12	ST_ToPolygonAry 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

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

## Foreword

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

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

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

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

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

This second edition cancels and replaces the first edition (ISO/IEC 13249-3:1999), which has been technically revised.

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

- *Part 1: Framework*
- *Part 2: Full-Text*
- *Part 3: Spatial*
- *Part 5: Still image*
- *Part 6: Data mining*

## 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.

## 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

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 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

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 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  $2\pi$  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  $2\pi$  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:

$$a.ST\_SymDifference(b) \Leftrightarrow \text{Closure}(a - b) \cup \text{Closure}(b - a) \Leftrightarrow a.ST\_Difference(b).ST\_Union(b.ST\_Difference(a))$$

- 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  $x.ST\_Dimension() \in \{0, 1, 2\}$ , i.e.  $x \neq \emptyset$
- b) if  $p = F$ , then  $x.ST\_Dimension() = -1$ , i.e.  $x = \emptyset$
- c) if  $p = 0$ , then  $x.ST\_Dimension() = 0$  (zero)
- d) if  $p = 1$ , then  $x.ST\_Dimension() = 1$  (one)
- e) if  $p = 2$ , then  $x.ST\_Dimension() = 2$
- f) if  $p = *$ , then  $x.ST\_Dimension() \in \{-1, 0, 1, 2\}$ , 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 ( $\mathbb{R}^2$ ). 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.

STANDARDISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 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.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

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

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

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

#### 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.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

Blank page

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 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,

```

STANDARDS190.COM · Click to view the full PDF of ISO/IEC 13249-3:2003

```

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,
```

STANDARDS150.COM: Click to view the full PDF of ISO/IEC 13249-3:2003

```
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,

```

```

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

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

METHOD ST_GMLToSQL
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
  RETURNS ST_Geometry
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_AsGML()
  RETURNS CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT

```

### Definitional Rules

- 1) *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.
- 2) *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.
- 3) *ST\_MaxGeometryAsGML* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the GML representation of an *ST\_Geometry* value.
- 4) *ST\_MaxTypeNameLength* is the implementation-defined maximum length used the character string representation of a type name.
- 5) *ST\_MaxUnitNameLength* is the implementation-defined maximum length for the character representation of a unit indication.
- 6) The attribute *ST\_PrivateDimension* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST\_PrivateDimension*.
- 7) The attribute *ST\_PrivateCoordinateDimension* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST\_PrivateCoordinateDimension*.

### 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\_Buffer(DOUBLE PRECISION, CHARACTER VARYING)*,
- o) a method *ST\_Intersection(ST\_Geometry)*,
- p) a method *ST\_Union(ST\_Geometry)*,
- q) a method *ST\_Difference(ST\_Geometry)*,
- r) a method *ST\_SymDifference(ST\_Geometry)*,
- s) a method *ST\_Distance(ST\_Geometry)*,
- t) a method *ST\_Distance(ST\_Geometry, CHARACTER VARYING)*,
- u) a method *ST\_Equals(ST\_Geometry)*,
- v) a method *ST\_Relate(ST\_Geometry, CHARACTER)*,
- w) a method *ST\_Disjoint(ST\_Geometry)*,
- x) a method *ST\_Intersects(ST\_Geometry)*,
- y) a method *ST\_Touches(ST\_Geometry)*,
- z) a method *ST\_Crosses(ST\_Geometry)*,
- aa) a method *ST\_Within(ST\_Geometry)*,
- ab) a method *ST\_Contains(ST\_Geometry)*,
- ac) a method *ST\_Overlaps(ST\_Geometry)*,
- ad) a method *ST\_WKTTtoSQL(CHARACTER LARGE OBJECT)*,
- ae) a method *ST\_AsText()*,
- af) a method *ST\_WKBtoSQL(BINARY LARGE OBJECT)*,
- ag) a method *ST\_AsBinary()*,
- ah) a method *ST\_GMLtoSQL(CHARACTER LARGE OBJECT)*,
- ai) a method *ST\_AsGML()*,
- aj) a function *ST\_GeomFromText(CHARACTER LARGE OBJECT)*,
- ak) a function *ST\_GeomFromText(CHARACTER LARGE OBJECT, INTEGER)*,
- al) a function *ST\_GeomFromWKB(BINARY LARGE OBJECT)*,
- am) a function *ST\_GeomFromWKB(BINARY LARGE OBJECT, INTEGER)*,
- an) a function *ST\_GeomFromGML(CHARACTER LARGE OBJECT)*,
- ao) a function *ST\_GeomFromGML(CHARACTER LARGE OBJECT, INTEGER)*,
- ap) an ordering function *ST\_OrderingEquals(ST\_Geometry, ST\_Geometry)*,
- aq) an SQL Transform group *ST\_WellKnownText*,
- ar) an SQL Transform group *ST\_WellKnownBinary*,
- as) an SQL Transform group *ST\_GML*,
- at) an implicit cast of an empty set of type *ST\_Geometry* to an *ST\_Point* value,

- au) an implicit cast of an empty set of type *ST\_Geometry* to an *ST\_LineString* value,
  - av) an implicit cast of an empty set of type *ST\_Geometry* to an *ST\_CircularString* value,
  - aw) an implicit cast of an empty set of type *ST\_Geometry* to an *ST\_CompoundCurve* value,
  - ax) an implicit cast of an empty set of type *ST\_Geometry* to an *ST\_CurvePolygon* value,
  - ay) an implicit cast of an empty set of type *ST\_Geometry* to an *ST\_Polygon* value,
  - az) an implicit cast of an empty set of type *ST\_Geometry* to an *ST\_GeomCollection* value,
  - ba) an implicit cast of an empty set of type *ST\_Geometry* to an *ST\_MultiPoint* value,
  - bb) an implicit cast of an empty set of type *ST\_Geometry* to an *ST\_MultiCurve* value,
  - bc) an implicit cast of an empty set of type *ST\_Geometry* to an *ST\_MultiLineString* value,
  - bd) an implicit cast of an empty set of type *ST\_Geometry* to an *ST\_MultiSurface* value,
  - be) an implicit cast of an empty set of type *ST\_Geometry* to an *ST\_MultiPolygon* value.
- 2) The *ST\_PrivateDimension* attribute contains the dimension of the *ST\_Geometry* value:
- Case:
- a) If the *ST\_Geometry* value corresponds to the empty set, then the dimension is -1.
  - b) If the *ST\_Geometry* value is 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 value of the *ST\_PrivateDimension* attribute shall be less than or equal to the value of the *ST\_PrivateCoordinateDimension* attribute.
- 6) All instantiable *ST\_Geometry* subtypes are defined so that simple values of the geometry type are topologically closed.
- 7) The coordinate dimension shall be the same as the coordinate dimension of the spatial reference system for the *ST\_Geometry* value.
- 8) 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.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 5.1.3 ST\_CoordDim Method

#### Purpose

Return the coordinate dimension of the ST\_Geometry value.

#### Definition

```
CREATE METHOD ST_CoordDim()  
  RETURNS SMALLINT  
  FOR ST_Geometry  
  RETURN SELF.ST_PrivateCoordinateDimension
```

#### Description

- 1) The method *ST\_CoordDim()* has no input parameters.
- 2) The null-call method *ST\_CoordDim()* returns the value of the *ST\_PrivateCoordinateDimension* attribute.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

#### 5.1.4 ST\_GeometryType Method

##### Purpose

Return the geometry type of the ST\_Geometry value.

##### Definition

```
CREATE METHOD ST_GeometryType()
  RETURNS CHARACTER VARYING(ST_MaxTypeNameLength)
  FOR ST_Geometry
  RETURN
  CASE
    WHEN SELF IS OF (ST_Point) THEN
      'ST_Point'
    WHEN SELF IS OF (ST_LineString) THEN
      'ST_LineString'
    WHEN SELF IS OF (ST_CircularString) THEN
      'ST_CircularString'
    WHEN SELF IS OF (ST_CompoundCurve) THEN
      'ST_CompoundCurve'
    WHEN SELF IS OF (ST_Polygon) THEN
      'ST_Polygon'
    WHEN SELF IS OF (ST_CurvePolygon) THEN
      'ST_CurvePolygon'
    WHEN SELF IS OF (ST_GeomCollection) THEN
      CASE
        WHEN SELF IS OF (ST_MultiPoint) THEN
          'ST_MultiPoint'
        WHEN SELF IS OF (ST_MultiLineString) THEN
          'ST_MultiLineString'
        WHEN SELF IS OF (ST_MultiCurve) THEN
          'ST_MultiCurve'
        WHEN SELF IS OF (ST_MultiPolygon) THEN
          'ST_MultiPolygon'
        WHEN SELF IS OF (ST_MultiSurface) THEN
          'ST_MultiSurface'
        ELSE
          'ST_GeomCollection'
      END
    -- ELSE
    --
    -- See Description
    --
  END
```

##### Definitional Rules

- 1) *ST\_MaxTypeNameLength* is the implementation-defined maximum length used the character string representation of a type name.

##### Description

- 1) The method *ST\_GeometryType()* has no input parameters.

- 2) For the null-call method *ST\_GeometryType()*:

Case:

- a) If SELF is of type *ST\_Point*, then return the value: 'ST\_Point'.
- b) If SELF is of type *ST\_LineString*, then return the value: 'ST\_LineString'.
- c) If SELF is of type *ST\_CircularString*, then the value 'ST\_CircularString'.
- d) If SELF is of type *ST\_CompoundCurve*, then the value 'ST\_CompoundCurve'.

- e) If SELF is of type *ST\_Polygon*, then return the value: 'ST\_Polygon'.
- f) If SELF is of type *ST\_CurvePolygon*, then return the value: 'ST\_CurvePolygon'.
- g) If SELF is of type *ST\_GeomCollection*, then:
  - Case:
    - i) If SELF is of type *ST\_MultiPoint*, then return the value 'ST\_MultiPoint'.
    - ii) If SELF is of type *ST\_MultiLineString*, then return the value 'ST\_MultiLineString'.
    - iii) If SELF is of type *ST\_MultiCurve*, then return the value 'ST\_MultiCurve'.
    - iv) If SELF is of type *ST\_MultiPolygon*, then return the value 'ST\_MultiPolygon'.
    - v) If SELF is of type *ST\_MultiSurface*, then return the value 'ST\_MultiSurface'.
    - vi) Otherwise, return the value: 'ST\_GeomCollection'.
  - h) Otherwise, the method *ST\_GeometryType()* returns an implementation-defined CHARACTER VARYING value for a user-defined type not defined in this part of ISO/IEC 13249.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 5.1.5 ST\_SRID Methods

#### Purpose

Observe and mutate the spatial reference system identifier of the *ST\_Geometry* value.

#### Definition

```
CREATE METHOD ST_SRID()
  RETURNS INTEGER
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END

CREATE METHOD ST_SRID
  (ansrid INTEGER)
  RETURNS ST_Geometry
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END
```

#### Description

- 1) The method *ST\_SRID()* has no input parameters.
- 2) The null-call method *ST\_SRID()* returns the spatial reference system identifier for the *ST\_Geometry* value.
- 3) The method *ST\_SRID(INTEGER)* takes the following input parameters:
  - a) an INTEGER value *ansrid*.
- 4) The parameter *ansrid* is a spatial reference system identifier.
- 5) The type preserving method *ST\_SRID(INTEGER)* returns an *ST\_Geometry* value with the spatial reference system identifier set to *ansrid*.

### 5.1.6 ST\_Transform Method

#### Purpose

Return an ST\_Geometry value transformed to the specified spatial reference system.

#### Definition

```
CREATE METHOD ST_Transform
  (ansrid INTEGER)
  RETURNS ST_Geometry
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END
```

#### Description

- 1) The method *ST\_Transform(INTEGER)* takes the following input parameters:
  - a) an INTEGER value *ansrid*.
- 2) The parameter *ansrid* is a spatial reference system identifier.
- 3) For the null-call type preserving method *ST\_Transform(INTEGER)*:
 

Case:

  - a) If the spatial reference system identifier of SELF is equal to *ansrid*, then return SELF.
  - b) If SELF is an empty set, then return SELF with the spatial reference system identifier equal to *ansrid*.
  - c) If SELF cannot be transformed to the spatial reference system specified by *ansrid*, then an exception condition is raised: *SQL/MM Spatial exception – failed to transform geometry*
  - d) Otherwise, return an *ST\_Geometry* value as the result of an implementation-defined transform of SELF from the spatial reference system of SELF to the spatial reference system specified by *ansrid*. The value returned has the spatial reference system identifier equal to *ansrid*.

### 5.1.7 ST\_IsEmpty Method

#### Purpose

Test if an *ST\_Geometry* value corresponds to the empty set.

#### Definition

```
CREATE METHOD ST_IsEmpty()  
  RETURNS INTEGER  
  FOR ST_Geometry  
  BEGIN  
    --  
    -- See Description  
    --  
  END
```

#### Description

- 1) The method *ST\_IsEmpty()* has no input parameters.
- 2) For the null-call method *ST\_IsEmpty()*:  
Case:
  - a) If the *ST\_Geometry* value corresponds to the empty set, then return 1 (one).
  - b) Otherwise, return 0 (zero).
- 3) The Description sections in the subclasses defining the instantiable subtypes of *ST\_Geometry* include the specific conditions that cause a value of these types to correspond to the empty set.

### 5.1.8 ST\_IsSimple Method

#### Purpose

Test if an *ST\_Geometry* value has no anomalous geometric points, such as self intersection or self tangency.

#### Definition

```
CREATE METHOD ST_IsSimple()
  RETURNS INTEGER
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END
```

#### Description

- 1) The method *ST\_IsSimple()* has no input parameters.
- 2) For the null-call method *ST\_IsSimple()*:
 

Case:

  - a) If SELF is an empty set, then return 1 (one).
  - b) If the *ST\_Geometry* value is simple, then return 1 (one).
  - c) Otherwise, return 0 (zero).
- 3) The Description sections in the subclauses defining the instantiable subtypes of *ST\_Geometry* include the specific conditions that cause a value of these types to be classified as simple.

### 5.1.9 ST\_IsValid Method

#### Purpose

Test if an *ST\_Geometry* value is well formed.

#### Definition

```
CREATE METHOD ST_IsValid()  
  RETURNS INTEGER  
  FOR ST_Geometry  
  BEGIN  
    --  
    -- See Description  
    --  
  END
```

#### Description

- 1) The method *ST\_IsValid()* has no input parameters.
- 2) For the null-call method *ST\_IsValid()*:  
Case:
  - a) If SELF is an empty *set*, then return 1 (one).
  - b) If the *ST\_Geometry* value is well formed, then return 1 (one).
  - c) Otherwise, return 0 (zero).
- 3) The Description sections in the subclauses defining the instantiable subtypes of *ST\_Geometry* include the specific conditions that cause a value of these types to be classified as well formed.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 5.1.10 ST\_Boundary Method

#### Purpose

Return the boundary of the ST\_Geometry value.

#### Definition

```
CREATE METHOD ST_Boundary()  
  RETURNS ST_Geometry  
  FOR ST_Geometry  
  BEGIN  
    --  
    -- See Description  
    --  
  END
```

#### Description

- 1) The method *ST\_Boundary()* has no input parameters.
- 2) For the null-call method *ST\_Boundary()*:  
Case:
  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the closure of the boundary of the *ST\_Geometry* value:  
Closure(Boundary(SELF)).
- 3) The spatial reference system identifier of the returned *ST\_Geometry* value is equal to the spatial reference system identifier of SELF.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 5.1.11 ST\_Envelope Method

#### Purpose

Return the bounding rectangle for the ST\_Geometry value.

#### Definition

```
CREATE METHOD ST_Envelope()
  RETURNS ST_Polygon
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END
```

#### Description

- 1) The method *ST\_Envelope()* has no input parameters.
- 2) For the null-call method *ST\_Envelope()* returns the bounding rectangle the *ST\_Geometry* value:
  - a) If SELF is an empty set, then return the null value.
  - b) Let *MINX* be the minimum x coordinate value in the *ST\_Geometry* value. Let *MINY* be the minimum y coordinate value in the *ST\_Geometry* value. Let *MAXX* be the maximum x coordinate value in the *ST\_Geometry* value. Let *MAXY* be the minimum y coordinate value in the *ST\_Geometry* value.
  - c) Let *ETOL* be an implementation-defined envelope tolerance. *ETOL* shall be greater than zero.
  - d) If *MINX* is equal to *MAXX*, then set *MINX* to *MINX - ETOL* and set *MAXX* to *MAXX + ETOL*.
  - e) If *MINY* is equal to *MAXY*, then set *MINY* to *MINY - ETOL* and set *MAXY* to *MAXY + ETOL*.
  - f) Return the bounding rectangle constructed as follows:

```
NEW ST_Polygon(
  NEW ST_LineString(
    ARRAY[
      NEW ST_Point(MINX, MINY, SELF.ST_SRID()),
      NEW ST_Point(MAXX, MINY, SELF.ST_SRID()),
      NEW ST_Point(MAXX, MAXY, SELF.ST_SRID()),
      NEW ST_Point(MINX, MAXY, SELF.ST_SRID()),
      NEW ST_Point(MINX, MINY, SELF.ST_SRID())],
    SELF.ST_SRID()),
  SELF.ST_SRID())
```

- 3) The spatial reference system identifier of the returned *ST\_Polygon* value is equal to the spatial reference system identifier of SELF.

**5.1.12 ST\_ConvexHull Method****Purpose**

Return the convex hull of the *ST\_Geometry* value.

**Definition**

```
CREATE METHOD ST_ConvexHull()
  RETURNS ST_Geometry
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END
```

**Description**

- 1) The method *ST\_ConvexHull()* has no input parameters.
- 2) For the null-call method *ST\_ConvexHull()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return an *ST\_Geometry* value representing the convex hull of the *ST\_Geometry* value.
- 3) The spatial reference system identifier of the returned *ST\_Geometry* value is equal to the spatial reference system identifier of SELF.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 5.1.13 ST\_Buffer Methods

#### Purpose

Return a buffer around the ST\_Geometry value.

#### Definition

```
CREATE METHOD ST_Buffer
  (adistance DOUBLE PRECISION)
  RETURNS ST_Geometry
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END

CREATE METHOD ST_Buffer
  (adistance DOUBLE PRECISION,
   aunit CHARACTER VARYING(ST_MaxUnitNameLength))
  RETURNS ST_Geometry
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END
```

#### Definitional Rules

- 1) *ST\_MaxUnitNameLength* is the implementation-defined maximum length for the character representation of a unit indication.

#### Description

- 1) The method *ST\_Buffer(DOUBLE PRECISION)* takes the following input parameters:
  - a) a DOUBLE PRECISION value *adistance*.
- 2) The parameter *adistance* is measured in an implementation-defined linear unit of measure in the spatial reference system of SELF.
- 3) For the null-call method *ST\_Buffer(DOUBLE PRECISION)*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return an *ST\_Geometry* value that represents all points whose distance from SELF is less than or equal to *adistance*.
- 4) The spatial reference system identifier of the returned *ST\_Geometry* value is equal to the spatial reference system identifier of SELF.
- 5) The method *ST\_Buffer(DOUBLE PRECISION, CHARACTER VARYING)* takes the following input parameters:
  - a) a DOUBLE PRECISION value *adistance*,
  - b) a CHARACTER VARYING value *aunit*.
- 6) For the null-call method *ST\_Buffer(DOUBLE PRECISION, CHARACTER VARYING)*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return an *ST\_Geometry* value that represents all points whose distance from SELF is less than or equal to *adistance*.

- 7) The DOUBLE PRECISION value *adistance* is measured in the units indicated by *unit*.
- 8) The values for *unit* shall be a supported <unit name>.
- 9) The value for *unit* is a supported <unit name> if and only if the value of *unit* is equal to the value of the UNIT\_NAME column of one of the rows where the value of the UNIT\_TYPE column is equal to 'LINEAR' in the ST\_UNITS\_OF\_MEASURE view.
- 10) If the unit specified by *unit* is not supported by the implementation to compute the an *ST\_Geometry* value that represents all points whose distance from SELF is less than or equal to *adistance*, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

**5.1.14 ST\_Intersection Method****Purpose**

Return an *ST\_Geometry* value that represents the point set intersection of two *ST\_Geometry* values.

**Definition**

```
CREATE METHOD ST_Intersection
  (ageometry ST_Geometry)
  RETURNS ST_Geometry
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END
```

**Description**

- 1) The method *ST\_Intersection(ST\_Geometry)* takes the following input parameters:
  - a) an *ST\_Geometry* value *ageometry*.
- 2) The null-call method *ST\_Intersection(ST\_Geometry)* returns an *ST\_Geometry* value that represents the point set intersection:  $\text{Closure}(\text{SELF} \cap \textit{ageometry})$ .

NOTE 6 For the list of subtypes returned by *ST\_Intersection(ST\_Geometry)*, see Table 5 — Return Type Matrix for the *ST\_Intersection* Method in Subclause 5.1.18, "Return Types from *ST\_Intersection*, *ST\_Union*, *ST\_Difference*, and *ST\_SymDifference*".

- 3) The spatial reference system identifier of the returned *ST\_Geometry* value is equal to the spatial reference system identifier of *SELF*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

**5.1.15 ST\_Union Method****Purpose**

Return an *ST\_Geometry* value that represents the point set union of two *ST\_Geometry* values.

**Definition**

```
CREATE METHOD ST_Union
  (ageometry ST_Geometry)
  RETURNS ST_Geometry
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END
```

**Description**

- 1) The method *ST\_Union(ST\_Geometry)* takes the following input parameters:
  - a) an *ST\_Geometry* value *ageometry*.
- 2) The null-call method *ST\_Union(ST\_Geometry)* returns an *ST\_Geometry* value that represents the point set union:  $\text{Closure}(\text{SELF} \cup \textit{ageometry})$ .

NOTE 7 For the list of subtypes returned by *ST\_Union(ST\_Geometry)*, see Table 6 — Return Type Matrix for the *ST\_Union* Method in Subclause 5.1.18, "Return Types from *ST\_Intersection*, *ST\_Union*, *ST\_Difference*, and *ST\_SymDifference*".

- 3) The spatial reference system identifier of the returned *ST\_Geometry* value is equal to the spatial reference system identifier of *SELF*.

**5.1.16 ST\_Difference Method****Purpose**

Return an *ST\_Geometry* value that represents the point set difference of two *ST\_Geometry* values.

**Definition**

```
CREATE METHOD ST_Difference
  (ageometry ST_Geometry)
  RETURNS ST_Geometry
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END
```

**Description**

- 1) The method *ST\_Difference(ST\_Geometry)* takes the following input parameters:
  - a) an *ST\_Geometry* value *ageometry*.
- 2) The null-call method *ST\_Difference(ST\_Geometry)* returns an *ST\_Geometry* value that represents the point set difference: *Closure(SELF — ageometry)*.

NOTE 8 For the list of subtypes returned by *ST\_Difference(ST\_Geometry)*, see Table 7 — Return Type Matrix for the *ST\_Difference* Method in Subclause 5.1.18, "Return Types from *ST\_Intersection*, *ST\_Union*, *ST\_Difference*, and *ST\_SymDifference*".

- 3) The spatial reference system identifier of the returned *ST\_Geometry* value is equal to the spatial reference system identifier of *SELF*.

**5.1.17 ST\_SymDifference Method****Purpose**

Return an *ST\_Geometry* value that represents the point set symmetric difference of two *ST\_Geometry* values.

**Definition**

```
CREATE METHOD ST_SymDifference
  (ageometry ST_Geometry)
  RETURNS ST_Geometry
  FOR ST_Geometry
  RETURN SELF.ST_Difference(ageometry) .
         ST_Union(ageometry.ST_Difference(SELF))
```

**Description**

- 1) The method *ST\_SymDifference(ST\_Geometry)* takes the following input parameters:
  - a) an *ST\_Geometry* value *ageometry*.
- 2) The null-call method *ST\_SymDifference(ST\_Geometry)* returns an *ST\_Geometry* value that represents the point set symmetric difference:  $\text{Closure}(\text{Closure}(\text{SELF} - \text{ageometry}) \cup \text{Closure}(\text{ageometry} - \text{SELF}))$ .

NOTE 9 For the list of subtypes returned by *ST\_SymDifference(ST\_Geometry)*, see Table 8 — Return Type Matrix for the *ST\_SymDifference* Method in Subclause 5.1.18, "Return Types from *ST\_Intersection*, *ST\_Union*, *ST\_Difference*, and *ST\_SymDifference*".

- 3) The spatial reference system identifier of the returned *ST\_Geometry* value is equal to the spatial reference system identifier of *SELF*.

**5.1.18 Return Types from ST\_Intersection, ST\_Union, ST\_Difference, and ST\_SymDifference**

The return types from the *ST\_Intersection*, *ST\_Union*, *ST\_Difference*, and *ST\_SymDifference* methods on the *ST\_Geometry* type are defined in this subclause. These methods take two *ST\_Geometry* values, the subject parameter and an additional parameter and return *ST\_Geometry* values. The parameter type code for the possible parameter types is described in Table 3 — Parameter Types. For any given method, the type of the return value shall be one of the possible subtypes of *ST\_Geometry* as specified in the return type set. The return type set codes are described in Table 4 — Return Type Sets.

A matrix for each method is presented with the parameter type code of the subject parameter down the column and the parameter type code of the additional parameter across the row. Each cell of the matrix contains the return set code for the two parameter types. The matrix for the *ST\_Intersection* method is in Table 5 — Return Type Matrix for the ST\_Intersection Method. The matrix for the *ST\_Union* method is in Table 6 — Return Type Matrix for the ST\_Union Method. The matrix for the *ST\_Difference* method is in Table 7 — Return Type Matrix for the ST\_Difference Method. The matrix for the *ST\_SymDifference* method is in Table 8 — Return Type Matrix for the ST\_SymDifference Method.

The elements in return values of type *ST\_GeomCollection* have no implied order.

**Table 3 — Parameter Types**

Parameter Type	
Code	Type
∅	empty set of any type
P	<i>ST_Point</i>
C	<i>ST_Curve</i>
S	<i>ST_Surface</i>
MP	<i>ST_MultiPoint</i>
MC	<i>ST_MultiCurve</i>
MS	<i>ST_MultiSurface</i>
GC	<i>ST_GeomCollection</i>

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

Table 4 — Return Type Sets

Return Type Sets	
Code	Set of Types
R00	empty set of type <i>ST_Point</i>
R01	<i>ST_Point</i>
R02	<i>ST_Curve</i>
R03	<i>ST_Surface</i>
R04	<i>ST_MultiPoint</i> ,
R05	<i>ST_MultiCurve</i>
R06	<i>ST_MultiSurface</i>
R07	<i>ST_GeomCollection</i>
R08	empty set of type <i>ST_Point</i> , <i>ST_Curve</i> , <i>ST_MultiCurve</i>
R09	empty set of type <i>ST_Point</i> , <i>ST_Point</i>
R10	empty set of type <i>ST_Point</i> , <i>ST_MultiPoint</i>
R11	empty set of type <i>ST_Point</i> , <i>ST_Point</i> , <i>ST_Curve</i> , <i>ST_MultiPoint</i> , <i>ST_MultiCurve</i> , <i>ST_GeomCollection</i> of <i>ST_Point</i> and <i>ST_Curve</i> values
R12	empty set of type <i>ST_Point</i> , <i>ST_Point</i> , <i>ST_Curve</i> , <i>ST_Surface</i> , <i>ST_MultiPoint</i> , <i>ST_MultiCurve</i> , <i>ST_MultiSurface</i> , <i>ST_GeomCollection</i>
R13	empty set of type <i>ST_Point</i> , <i>ST_Point</i> , <i>ST_MultiPoint</i>
R14	empty set of type <i>ST_Point</i> , <i>ST_Surface</i> , <i>ST_MultiSurface</i>
R15	<i>ST_Curve</i> , <i>ST_GeomCollection</i> of <i>ST_Point</i> and <i>ST_Curve</i> values
R16	<i>ST_Curve</i> , <i>ST_MultiCurve</i>
R17	<i>ST_MultiCurve</i> , <i>ST_GeomCollection</i> of <i>ST_Point</i> and <i>ST_Curve</i> values
R18	<i>ST_MultiSurface</i> , <i>ST_GeomCollection</i> of <i>ST_Curve</i> and <i>ST_Surface</i> values
R19	<i>ST_MultiSurface</i> , <i>ST_GeomCollection</i> of <i>ST_Point</i> and <i>ST_Surface</i> values
R20	<i>ST_Point</i> , <i>ST_MultiPoint</i>
R21	<i>ST_Surface</i> , <i>ST_GeomCollection</i> of <i>ST_Curve</i> and <i>ST_Surface</i> values
R22	<i>ST_Surface</i> , <i>ST_GeomCollection</i> of <i>ST_Point</i> and <i>ST_Surface</i> values
R23	<i>ST_Surface</i> , <i>ST_MultiSurface</i>

Table 5 — Return Type Matrix for the *ST\_Intersection* Method

a ∩ b									
a \ b	∅	P	C	S	MP	MC	MS	GC	
∅	R00								
P	R00	R09							
C	R00	R09	R11	R11	R13	R11	R11	R11	
S	R00	R09	R11	R12	R13	R11	R12	R12	
MP	R00	R09	R13	R13	R13	R13	R13	R13	
MC	R00	R09	R11	R11	R13	R11	R11	R11	
MS	R00	R09	R11	R12	R13	R11	R12	R12	
GC	R00	R09	R11	R12	R13	R11	R12	R12	

**Table 6 — Return Type Matrix for the ST\_Union Method**

$a \cup b$									
$a \backslash b$	$\emptyset$	P	C	S	MP	MC	MS	GC	
$\emptyset$	R00	R01	R02	R03	R04	R05	R06	R07	
P	R01	R20	R15	R22	R04	R17	R19	R07	
C	R02	R15	R16	R21	R15	R16	R18	R07	
S	R03	R22	R21	R23	R22	R21	R23	R07	
MP	R04	R04	R15	R22	R04	R17	R19	R07	
MC	R05	R17	R16	R21	R17	R16	R18	R07	
MS	R06	R19	R18	R23	R19	R18	R23	R07	
GC	R07	R07	R07	R07	R07	R07	R07	R07	

**Table 7 — Return Type Matrix for the ST\_Difference Method**

$a - b$									
$a \backslash b$	$\emptyset$	P	C	S	MP	MC	MS	GC	
$\emptyset$	R00	R00	R00	R00	R00	R00	R00	R00	
P	R01	R09							
C	R02	R02	R08	R08	R02	R08	R08	R08	
S	R03	R03	R03	R14	R14	R14	R14	R14	
MP	R04	R13							
MC	R05	R05	R08	R08	R05	R08	R08	R08	
MS	R06	R06	R06	R14	R06	R06	R06	R14	
GC	R07	R12							

**Table 8 — Return Type Matrix for the ST\_SymDifference Method**

$(a - b) \cup (b - a)$									
$a \backslash b$	$\emptyset$	P	C	S	MP	MC	MS	GC	
$\emptyset$	R00	R01	R02	R03	R04	R05	R06	R07	
P	R01	R10	R15	R22	R10	R17	R19	R12	
C	R02	R15	R08	R21	R15	R08	R18	R12	
S	R03	R22	R21	R14	R22	R21	R14	R12	
MP	R04	R10	R15	R22	R10	R17	R19	R12	
MC	R05	R17	R08	R21	R17	R08	R18	R12	
MS	R06	R19	R18	R14	R19	R18	R14	R12	
GC	R07	R12							

### 5.1.19 ST\_Distance Methods

#### Purpose

Return the distance between two geometries.

#### Definition

```
CREATE METHOD ST_Distance
  (ageometry ST_Geometry)
  RETURNS DOUBLE PRECISION
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END

CREATE METHOD ST_Distance
  (ageometry ST_Geometry,
   aunit CHARACTER VARYING(ST_MaxUnitNameLength))
  RETURNS DOUBLE PRECISION
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END
```

#### Definitional Rules

- 1) *ST\_MaxUnitNameLength* is the implementation-defined maximum length for the character representation of a unit indication.

#### Description

- 1) The method *ST\_Distance(ST\_Geometry)* takes the following input parameters:
  - a) an *ST\_Geometry* value *ageometry*.
- 2) For the null-call method *ST\_Distance(ST\_Geometry)*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) If *ageometry* is an empty set, then return the null value.
  - c) If SELF and *ageometry* spatially intersect, then return 0 (zero).
  - d) Otherwise, return the distance between two geometries, SELF and *ageometry*, is calculated in the spatial reference system of SELF. The distance between the two points is calculated using an implementation-defined algorithm.
- 3) Case:
  - a) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST\_Distance(ST\_Geometry)* is in the linear unit of measure identified by <linear unit>.
  - b) Otherwise, the value returned by *ST\_Distance(ST\_Geometry)* is in an implementation-defined unit of measure.
- 4) The method *ST\_Distance(ST\_Geometry, CHARACTER VARYING)* takes the following input parameters:
  - a) an *ST\_Geometry* value *ageometry*;
  - b) a CHARACTER VARYING value *aunit*.

5) For the null-call method *ST\_Distance*(*ST\_Geometry*, *CHARACTER VARYING*):

Case:

- a) If SELF is an empty set, then return the null value.
  - b) If *ageometry* is an empty set, then return the null value.
  - c) If SELF and *ageometry* spatially intersect, then return 0 (zero).
  - d) Otherwise, return the distance between two geometries, SELF and *ageometry*, is calculated in the spatial reference system of SELF. The distance between the two points is calculated using an implementation-defined algorithm.
- 6) The value returned by *ST\_Distance*(*ST\_Geometry*, *CHARACTER VARYING*) is measured in the units indicated by *unit*.
- 7) The values for *unit* shall be a supported <unit name>.
- 8) The value for *unit* is a supported <unit name> if and only if the value of *unit* is equal to the value of the UNIT\_NAME column of one of the rows where the value of the UNIT\_TYPE column is equal to 'LINEAR' in the ST\_UNITS\_OF\_MEASURE view.
- 9) If the unit specified by *unit* is not supported by the implementation to compute the distance between SELF and *ageometry*, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 5.1.20 ST\_Equals Method

#### Purpose

Test if an ST\_Geometry value is spatially equal to another ST\_Geometry value.

#### Definition

```
CREATE METHOD ST_Equals  
  (ageometry ST_Geometry)  
  RETURNS INTEGER  
  FOR ST_Geometry  
  RETURN SELF.ST_SymDifference(ageometry).ST_IsEmpty()
```

#### Description

- 1) The method *ST\_Equals(ST\_Geometry)* takes the following input parameters:
  - a) an *ST\_Geometry* value *ageometry*.
- 2) The null-call method *ST\_Equals(ST\_Geometry)* returns the result of the value expression:  
*SELF.ST\_SymDifference(ageometry).ST\_IsEmpty()*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 5.1.21 ST\_Relate Method

## Purpose

Test if an ST\_Geometry value is spatially related to another ST\_Geometry value.

## Definition

```

CREATE METHOD ST_Relate
  (ageometry ST_Geometry,
   amatrix CHARACTER(9))
RETURNS INTEGER
FOR ST_Geometry
BEGIN
  DECLARE counter INTEGER;
  DECLARE intersectdim INTEGER;

  -- If any value in amatrix is not the list of
  -- possible values: 'T', 'F', '0', '1', '2', '*', then
  -- raise an exception.
  SET counter = 1;
  WHILE counter <= 9 DO
    IF SUBSTRING(amatrix FROM counter FOR 1)
      NOT IN ( 'T', 'F', '0', '1', '2', '*' ) THEN
      SIGNAL SQLSTATE '2FF04'
        SET MESSAGE_TEXT = 'invalid intersection matrix';
    END IF;
    SET counter = counter + 1;
  END WHILE;
  -- Process each of the 9 intersections
  SET counter = 1;
  WHILE counter <= 9 DO
    -- Set intersectdim to the dimension of the current intersection
    CASE counter
      WHEN 1 THEN
        SET intersectdim =
          Intersection(Interior(SELF), Interior(ageometry)).
          ST_Dimension(); -- See Description
      WHEN 2 THEN
        SET intersectdim =
          Intersection(Interior(SELF), Boundary(ageometry)).
          ST_Dimension(); -- See Description
      WHEN 3 THEN
        SET intersectdim =
          Intersection(Interior(SELF), Exterior(ageometry)).
          ST_Dimension(); -- See Description
      WHEN 4 THEN
        SET intersectdim =
          Intersection(Boundary(SELF), Interior(ageometry)).
          ST_Dimension(); -- See Description
      WHEN 5 THEN
        SET intersectdim =
          Intersection(Boundary(SELF), Boundary(ageometry)).
          ST_Dimension(); -- See Description
      WHEN 6 THEN
        SET intersectdim =
          Intersection(Boundary(SELF), Exterior(ageometry)).
          ST_Dimension(); -- See Description
      WHEN 7 THEN
        SET intersectdim =
          Intersection(Exterior(SELF), Interior(ageometry)).
          ST_Dimension(); -- See Description
    
```

```

WHEN 8 THEN
  SET intersectdim =
    Intersection(Exterior(SELF), Boundary(ageometry)).
    ST_Dimension(); -- See Description
WHEN 9 THEN
  SET intersectdim =
    Intersection(Exterior(SELF), Exterior(ageometry)).
    ST_Dimension(); -- See Description
END CASE;
-- If intersectdim is not in the result set as defined by the
-- current amatrix position, then return 0 (zero).
CASE SUBSTRING(amatrix FROM counter FOR 1)
WHEN 'T' THEN
  IF intersectdim NOT IN ( 0, 1, 2 ) THEN
    RETURN 0;
  END IF;
WHEN 'F' THEN
  IF intersectdim <> -1 THEN
    RETURN 0;
  END IF;
WHEN '0' THEN
  IF intersectdim <> 0 THEN
    RETURN 0;
  END IF;
WHEN '1' THEN
  IF intersectdim <> 1 THEN
    RETURN 0;
  END IF;
WHEN '2' THEN
  IF intersectdim <> 2 THEN
    RETURN 0;
  END IF;
WHEN '*' THEN
  IF intersectdim NOT IN ( -1, 0, 1, 2 ) THEN
    RETURN 0;
  END IF;
END CASE;
SET counter = counter + 1;
END WHILE;
-- If the dimension of each intersection matches the amatrix, then
-- return 1 (one).
RETURN 1;
END

```

### Definitional Rules

- 1) Let  $G1$  and  $G2$  be *ST\_Geometry* values.
- 2) *Interior(G1)* represents the interior of  $G1$ .
- 3) *Boundary(G1)* represents the boundary of  $G1$ .
- 4) *Exterior(G1)* represents the exterior of  $G1$ .
- 5) *Intersection(G1, G2)* returns the point set intersection of  $G1$  and  $G2$ .

NOTE 10 interior, boundary, exterior and point set intersection are described in Subclause 4.1.2.1, "The Dimensionally Extended 9 Intersection Model".

### Description

- 1) The method *ST\_Relate(ST\_Geometry, CHARACTER)* takes the following input parameters:
  - a) an *ST\_Geometry* value *ageometry*,
  - b) a CHARACTER value *amatrix*.

2) For null-call method *ST\_Relate(ST\_Geometry, CHARACTER)*:

Case:

- a) If *SELF* is an empty set, then return the null value.
- b) If *ageometry* is an empty set, then return the null value.
- c) If any character in *amatrix* is not 'T', 'F', '0', '1', '2', or '\*\*', then an exception condition is raised:  
*SQL/MM Spatial exception – invalid intersection matrix.*
- d) Otherwise:
  - i) Each character in *amatrix* corresponds to a cell in the DE9IM pattern matrix. This mapping is defined in Table 9 — DE-9IM .

**Table 9 — DE-9IM Mapping**

Position	DE-9IM Cell
1	$(\text{Interior}(\text{SELF}) \cap \text{Interior}(\text{ageometry})).\text{ST\_Dimension}()$
2	$(\text{Interior}(\text{SELF}) \cap \text{Boundary}(\text{ageometry})).\text{ST\_Dimension}()$
3	$(\text{Interior}(\text{SELF}) \cap \text{Exterior}(\text{ageometry})).\text{ST\_Dimension}()$
4	$(\text{Boundary}(\text{SELF}) \cap \text{Interior}(\text{ageometry})).\text{ST\_Dimension}()$
5	$(\text{Boundary}(\text{SELF}) \cap \text{Boundary}(\text{ageometry})).\text{ST\_Dimension}()$
6	$(\text{Boundary}(\text{SELF}) \cap \text{Exterior}(\text{ageometry})).\text{ST\_Dimension}()$
7	$(\text{Exterior}(\text{SELF}) \cap \text{Interior}(\text{ageometry})).\text{ST\_Dimension}()$
8	$(\text{Exterior}(\text{SELF}) \cap \text{Boundary}(\text{ageometry})).\text{ST\_Dimension}()$
9	$(\text{Exterior}(\text{SELF}) \cap \text{Exterior}(\text{ageometry})).\text{ST\_Dimension}()$

See Subclause 4.1.2.1, "The Dimensionally Extended 9 Intersection Model" for a detailed description of the DE-9IM.

- ii) Each character value in *amatrix* specifies the set of acceptable values for an intersection at a given cell. The meaning for any cell is described in Table 10 — Cell Values.

**Table 10 — Cell Values**

Cell Value	Intersection Set Results
'T'	{ 0, 1, 2 }
'F'	{ -1 }
'0'	{ 0 }
'1'	{ 1 }
'2'	{ 2 }
'**'	{ -1, 0, 1, 2 }

- iii) Let *RESULT* be the value returned by this method. Set *RESULT* to 1 (one).
- iv) For *COUNTER* varying from 1 (one) to 9:
  - 1) Let *INTERSECTDIM* be the result of the DE-9IM Intersection at position *COUNTER*.
  - 2) Let *SVI* be the character value at *COUNTER* and let *SRI* be the intersection set results corresponding to *SVI*.
  - 3) If *INTERSECTDIM* is not in the set *SRI*, then set *RESULT* to 0 (zero).
- v) Return *RESULT*.

### 5.1.22 ST\_Disjoint Method

#### Purpose

Test if an ST\_Geometry value is spatially disjoint from another ST\_Geometry value.

#### Definition

```
CREATE METHOD ST_Disjoint
  (ageometry ST_Geometry)
  RETURNS INTEGER
  FOR ST_Geometry
  RETURN SELF.ST_Relate(ageometry, 'FF*FF****')
```

#### Description

- 1) The method *ST\_Disjoint(ST\_Geometry)* takes the following input parameters:
  - a) an *ST\_Geometry* value *ageometry*.
- 2) The null-call method *ST\_Disjoint(ST\_Geometry)* returns the result of the value expression: *SELF.ST\_Relate(ageometry, 'FF\*FF\*\*\*\*')*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 5.1.23 ST\_Intersects Method

#### Purpose

Test if an ST\_Geometry value spatially intersects another ST\_Geometry value.

#### Definition

```
CREATE METHOD ST_Intersects
  (ageometry ST_Geometry)
  RETURNS INTEGER
  FOR ST_Geometry
  RETURN
    CASE SELF.ST_Disjoint(ageometry)
      WHEN 1 THEN
        0
      WHEN 0 THEN
        1
      ELSE
        NULL
    END
```

#### Description

- 1) The method *ST\_Intersects(ST\_Geometry)* takes the following input parameters:
  - a) an *ST\_Geometry* value *ageometry*.
- 2) For the null-call method *ST\_Intersects(ST\_Geometry)*:
 

Case:

  - a) If *SELF.ST\_Disjoint(ageometry)* is equal to 1 (one), then return 0 (zero).
  - b) If *SELF.ST\_Disjoint(ageometry)* is equal to 0 (zero), then return 1 (one).
  - c) Otherwise, return the null value.

### 5.1.24 ST\_Touches Method

#### Purpose

Test if an ST\_Geometry value spatially touches another ST\_Geometry value.

#### Definition

```

CREATE METHOD ST_Touches
  (ageometry ST_Geometry)
  RETURNS INTEGER
  FOR ST_Geometry
  RETURN
  CASE
    WHEN (SELF.ST_Dimension() = 0 AND
          ageometry.ST_Dimension() = 0) THEN
      NULL
    ELSE
      -- Use ST_Relate to determine result of the touch operation
      -- on SELF and ageometry
      CASE (SELF.ST_Relate(ageometry, 'FT*****') = 1 OR
            SELF.ST_Relate(ageometry, 'F**T*****') = 1 OR
            SELF.ST_Relate(ageometry, 'F***T*****') = 1)
        WHEN TRUE THEN
          1
        WHEN FALSE THEN
          0
        ELSE
          NULL
      END
    END
  END

```

#### Description

- 1) The method *ST\_Touches(ST\_Geometry)* takes the following input parameters:
  - a) an *ST\_Geometry* value *ageometry*.
- 2) For the null-call method *ST\_Touches(ST\_Geometry)*:
 

Case:

  - a) If the dimension of *SELF* is equal to 0 (zero) and the dimension of *ageometry* is equal to 0 (zero), then return the null value.
  - b) Otherwise,
    - i) Let *BVE* be the result of the value expression: *SELF.ST\_Relate(ageometry, 'FT\*\*\*\*\*') = 1* OR *SELF.ST\_Relate(ageometry, 'F\*\*T\*\*\*\*\*') = 1* OR *SELF.ST\_Relate(ageometry, 'F\*\*\*T\*\*\*\*\*') = 1*.
    - ii) Case:
      - 1) If *BVE* is *True*, then return 1 (one).
      - 2) If *BVE* is *False*, then return 0 (zero).
      - 3) Otherwise, return the null value.

### 5.1.25 ST\_Crosses Method

#### Purpose

Test if an ST\_Geometry value spatially crosses another ST\_Geometry value.

#### Definition

```
CREATE METHOD ST_Crosses
  (ageometry ST_Geometry)
  RETURNS INTEGER
  FOR ST_Geometry
  RETURN
  CASE
    WHEN (SELF.ST_Dimension() = 0 AND
          ageometry.ST_Dimension() = 1) THEN
      SELF.ST_Relate(ageometry, 'T*T*****')
    WHEN (SELF.ST_Dimension() = 0 AND
          ageometry.ST_Dimension() = 2) THEN
      SELF.ST_Relate(ageometry, 'T*T*****')
    WHEN (SELF.ST_Dimension() = 1 AND
          ageometry.ST_Dimension() = 1) THEN
      SELF.ST_Relate(ageometry, '0*****')
    WHEN (SELF.ST_Dimension() = 1 AND
          ageometry.ST_Dimension() = 2) THEN
      SELF.ST_Relate(ageometry, 'T*T*****')
    ELSE
      NULL
  END
```

#### Description

1) The method *ST\_Crosses(ST\_Geometry)* takes the following input parameters:

a) an *ST\_Geometry* value *ageometry*.

2) For the null-call method *ST\_Crosses(ST\_Geometry)*:

Case:

- a) If the dimension of SELF is equal to 0 (zero) and the dimension of *ageometry* is equal to 1 (one), then return the result of the value expression: *SELF.ST\_Relate(ageometry, 'T\*T\*\*\*\*\*')*.
- b) If the dimension of SELF is equal to 0 (zero) and the dimension of *ageometry* is equal to 2, then return the result of the value expression: *SELF.ST\_Relate(ageometry, 'T\*T\*\*\*\*\*')*.
- c) If the dimension of SELF is equal to 1 (one) and the dimension of *ageometry* is equal to 1 (one), then return the result of the value expression: *SELF.ST\_Relate(ageometry, '0\*\*\*\*\*')*.
- d) If the dimension of SELF is equal to 1 (one) and the dimension of *ageometry* is equal to 2, then return the result of the value expression: *SELF.ST\_Relate(ageometry, 'T\*T\*\*\*\*\*')*.
- e) Otherwise, return the null value.

### 5.1.26 ST\_Within Method

#### Purpose

Test if an ST\_Geometry value is spatially within another ST\_Geometry value.

#### Definition

```
CREATE METHOD ST_Within
  (ageometry ST_Geometry)
  RETURNS INTEGER
  FOR ST_Geometry
  RETURN SELF.ST_Relate(ageometry, 'T**F***')
```

#### Description

- 1) The method *ST\_Within(ST\_Geometry)* takes the following input parameters:
  - a) an *ST\_Geometry* value *ageometry*.
- 2) The null-call method *ST\_Within(ST\_Geometry)* returns the result of the value expression: *SELF.ST\_Relate(ageometry, 'T\*\*F\*\*\*')*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 5.1.27 ST\_Contains Method

#### Purpose

Test if an ST\_Geometry value spatially contains another ST\_Geometry value.

#### Definition

```
CREATE METHOD ST_Contains
  (ageometry ST_Geometry)
  RETURNS INTEGER
  FOR ST_Geometry
  RETURN ageometry.ST_Within(SELf)
```

#### Description

- 1) The method *ST\_Contains(ST\_Geometry)* takes the following input parameters:
  - a) an *ST\_Geometry* value *ageometry*.
- 2) The null-call method *ST\_Contains(ST\_Geometry)* returns the result of the value expression: *ageometry.ST\_Within(SELf)*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 5.1.28 ST\_Overlaps Method

#### Purpose

Test if an ST\_Geometry value spatially overlaps another ST\_Geometry value.

#### Definition

```
CREATE METHOD ST_Overlaps
  (ageometry ST_Geometry)
  RETURNS INTEGER
  FOR ST_Geometry
  RETURN
  CASE
    WHEN (SELF.ST_Dimension() = 0 AND
          ageometry.ST_Dimension() = 0) THEN
      SELF.ST_Relate(ageometry, 'T*T***T**')
    WHEN (SELF.ST_Dimension() = 1 AND
          ageometry.ST_Dimension() = 1) THEN
      SELF.ST_Relate(ageometry, '1*T***T**')
    WHEN (SELF.ST_Dimension() = 2 AND
          ageometry.ST_Dimension() = 2) THEN
      SELF.ST_Relate(ageometry, 'T*T***T**')
    ELSE
      NULL
  END
```

#### Description

1) The method *ST\_Overlaps(ST\_Geometry)* takes the following input parameters:

a) an *ST\_Geometry* value *ageometry*.

2) For the null-call method *ST\_Overlaps(ST\_Geometry)*:

Case:

- a) If the dimension of SELF is equal to 0 (zero) and the dimension of *ageometry* is equal to 0 (zero), then return the result of the value expression: *SELF.ST\_Relate(ageometry, 'T\*T\*\*\*T\*\*')*.
- b) If the dimension of SELF is equal to 1 (one) and the dimension of *ageometry* is equal to 1 (one), then return the result of the value expression: *SELF.ST\_Relate(ageometry, '1\*T\*\*\*T\*\*')*.
- c) If the dimension of SELF is equal to 2 and the dimension of *ageometry* is equal to 2, then return the result of the value expression: *SELF.ST\_Relate(ageometry, 'T\*T\*\*\*T\*\*')*.
- d) Otherwise, return the null value.

**5.1.29 Empty Cast****Purpose**

Cast an empty set of type ST\_Geometry to a specific instantiable subtype of ST\_Geometry.

**Definition**

```

CREATE METHOD ST_ToPoint()
  RETURNS ST_Point
  FOR ST_Geometry
  BEGIN
    IF SELF IS OF (ST_Point) THEN
      RETURN SELF;
    END IF;
    IF SELF.ST_IsEmpty() = 0 THEN
      SIGNAL SQLSTATE '2FF16'
        SET MESSAGE_TEXT = 'not an empty set';
    END IF;
    RETURN NEW ST_Point();
  END

CREATE CAST(ST_Geometry AS ST_Point)
  WITH METHOD ST_ToPoint()
  AS ASSIGNMENT

CREATE METHOD ST_ToLineString()
  RETURNS ST_LineString
  FOR ST_Geometry
  BEGIN
    IF SELF IS OF (ST_LineString) THEN
      RETURN SELF;
    END IF;
    IF SELF.ST_IsEmpty() = 0 THEN
      SIGNAL SQLSTATE '2FF16'
        SET MESSAGE_TEXT = 'not an empty set';
    END IF;
    RETURN NEW ST_LineString();
  END

CREATE CAST(ST_Geometry AS ST_LineString)
  WITH METHOD ST_ToLineString()
  AS ASSIGNMENT

CREATE METHOD ST_ToCircular()
  RETURNS ST_CircularString
  FOR ST_Geometry
  BEGIN
    IF SELF IS OF (ST_CircularString) THEN
      RETURN SELF;
    END IF;
    IF SELF.ST_IsEmpty() = 0 THEN
      SIGNAL SQLSTATE '2FF16'
        SET MESSAGE_TEXT = 'not an empty set';
    END IF;
    RETURN NEW ST_CircularString();
  END

CREATE CAST(ST_Geometry AS ST_CircularString)
  WITH METHOD ST_ToCircular()
  AS ASSIGNMENT

```

```

CREATE METHOD ST_ToCompound()
  RETURNS ST_CompoundCurve
  FOR ST_Geometry
  BEGIN
    IF SELF IS OF (ST_CompoundCurve) THEN
      RETURN SELF;
    END IF;
    IF SELF.ST_IsEmpty() = 0 THEN
      SIGNAL SQLSTATE '2FF16'
        SET MESSAGE_TEXT = 'not an empty set';
    END IF;
    RETURN NEW ST_CompoundCurve();
  END

CREATE CAST(ST_Geometry AS ST_CompoundCurve)
  WITH METHOD ST_ToCompound()
  AS ASSIGNMENT

CREATE METHOD ST_ToCurvePoly()
  RETURNS ST_CurvePolygon
  FOR ST_Geometry
  BEGIN
    IF SELF IS OF (ST_CurvePolygon) THEN
      RETURN SELF;
    END IF;
    IF SELF.ST_IsEmpty() = 0 THEN
      SIGNAL SQLSTATE '2FF16'
        SET MESSAGE_TEXT = 'not an empty set';
    END IF;
    RETURN NEW ST_CurvePolygon();
  END

CREATE CAST(ST_Geometry AS ST_CurvePolygon)
  WITH METHOD ST_ToCurvePoly()
  AS ASSIGNMENT

CREATE METHOD ST_ToPolygon()
  RETURNS ST_Polygon
  FOR ST_Geometry
  BEGIN
    IF SELF IS OF (ST_Polygon) THEN
      RETURN SELF;
    END IF;
    IF SELF.ST_IsEmpty() = 0 THEN
      SIGNAL SQLSTATE '2FF16'
        SET MESSAGE_TEXT = 'not an empty set';
    END IF;
    RETURN NEW ST_Polygon();
  END

CREATE CAST(ST_Geometry AS ST_Polygon)
  WITH METHOD ST_ToPolygon()
  AS ASSIGNMENT

```

```

CREATE METHOD ST_ToGeomColl()
  RETURNS ST_GeomCollection
  FOR ST_Geometry
  BEGIN
    IF SELF IS OF (ST_GeomCollection) THEN
      RETURN SELF;
    END IF;
    IF SELF.ST_IsEmpty() = 0 THEN
      SIGNAL SQLSTATE '2FF16'
        SET MESSAGE_TEXT = 'not an empty set';
    END IF;
    RETURN NEW ST_GeomCollection();
  END

CREATE CAST(ST_Geometry AS ST_GeomCollection)
  WITH METHOD ST_ToGeomColl()
  AS ASSIGNMENT

CREATE METHOD ST_ToMultiPoint()
  RETURNS ST_MultiPoint
  FOR ST_Geometry
  BEGIN
    IF SELF IS OF (ST_MultiPoint) THEN
      RETURN SELF;
    END IF;
    IF SELF.ST_IsEmpty() = 0 THEN
      SIGNAL SQLSTATE '2FF16'
        SET MESSAGE_TEXT = 'not an empty set';
    END IF;
    RETURN NEW ST_MultiPoint();
  END

CREATE CAST(ST_Geometry AS ST_MultiPoint)
  WITH METHOD ST_ToMultiPoint()
  AS ASSIGNMENT

CREATE METHOD ST_ToMultiCurve()
  RETURNS ST_MultiCurve
  FOR ST_Geometry
  BEGIN
    IF SELF IS OF (ST_MultiCurve) THEN
      RETURN SELF;
    END IF;
    IF SELF.ST_IsEmpty() = 0 THEN
      SIGNAL SQLSTATE '2FF16'
        SET MESSAGE_TEXT = 'not an empty set';
    END IF;
    RETURN NEW ST_MultiCurve();
  END

CREATE CAST(ST_Geometry AS ST_MultiCurve)
  WITH METHOD ST_ToMultiCurve()
  AS ASSIGNMENT

```

```

CREATE METHOD ST_ToMultiLine()
  RETURNS ST_MultiLineString
  FOR ST_Geometry
  BEGIN
    IF SELF IS OF (ST_MultiLineString) THEN
      RETURN SELF;
    END IF;
    IF SELF.ST_IsEmpty() = 0 THEN
      SIGNAL SQLSTATE '2FF16'
        SET MESSAGE_TEXT = 'not an empty set';
    END IF;
    RETURN NEW ST_MultiLineString();
  END

CREATE CAST(ST_Geometry AS ST_MultiLineString)
  WITH METHOD ST_ToMultiLine()
  AS ASSIGNMENT

CREATE METHOD ST_ToMultiSurface()
  RETURNS ST_MultiSurface
  FOR ST_Geometry
  BEGIN
    IF SELF IS OF (ST_MultiSurface) THEN
      RETURN SELF;
    END IF;
    IF SELF.ST_IsEmpty() = 0 THEN
      SIGNAL SQLSTATE '2FF16'
        SET MESSAGE_TEXT = 'not an empty set';
    END IF;
    RETURN NEW ST_MultiSurface();
  END

CREATE CAST(ST_Geometry AS ST_MultiSurface)
  WITH METHOD ST_ToMultiSurface()
  AS ASSIGNMENT

CREATE METHOD ST_ToMultiPolygon()
  RETURNS ST_MultiPolygon
  FOR ST_Geometry
  BEGIN
    IF SELF IS OF (ST_MultiPolygon) THEN
      RETURN SELF;
    END IF;
    IF SELF.ST_IsEmpty() = 0 THEN
      SIGNAL SQLSTATE '2FF16'
        SET MESSAGE_TEXT = 'not an empty set';
    END IF;
    RETURN NEW ST_MultiPolygon();
  END

CREATE CAST(ST_Geometry AS ST_MultiPolygon)
  WITH METHOD ST_ToMultiPolygon()
  AS ASSIGNMENT

```

### Description

- 1) The method *ST\_ToPoint()* has no input parameters.
- 2) For the null-call method *ST\_ToPoint()*:
  - a) If *SELF* is of type *ST\_Point*, then return *SELF*.
  - b) If *SELF* is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.
  - c) Otherwise, return an empty set of type *ST\_Point*.

- 3) Use the method *ST\_ToPoint()* to define an implicitly invocable cast function to cast an *ST\_Geometry* value to an *ST\_Point* value.
- 4) The method *ST\_ToLineString()* has no input parameters.
- 5) For the null-call method *ST\_ToLineString()*:
  - a) If *SELF* is of type *ST\_LineString*, then return *SELF*.
  - b) If *SELF* is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.
  - c) Otherwise, return an empty set of type *ST\_LineString*.
- 6) Use the method *ST\_ToLineString()* to define an implicitly invocable cast function to cast an *ST\_Geometry* value to an *ST\_LineString* value.
- 7) The method *ST\_ToCircular()* has no input parameters.
- 8) For the null-call method *ST\_ToCircular()*:
  - a) If *SELF* is of type *ST\_CircularString*, then return *SELF*.
  - b) If *SELF* is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.
  - c) Otherwise, return an empty set of type *ST\_CircularString*.
- 9) Use the method *ST\_ToCircular()* to define an implicitly invocable cast function to cast an *ST\_Geometry* value to an *ST\_CircularString* value.
- 10) The method *ST\_ToCompound()* has no input parameters.
- 11) For the null-call method *ST\_ToCompound()*:
  - a) If *SELF* is of type *ST\_CompoundCurve*, then return *SELF*.
  - b) If *SELF* is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.
  - c) Otherwise, return an empty set of type *ST\_CompoundCurve*.
- 12) Use the method *ST\_ToCompound()* to define an implicitly invocable cast function to cast an *ST\_Geometry* value to an *ST\_CompoundCurve* value.
- 13) The method *ST\_ToCurvePoly()* has no input parameters.
- 14) For the null-call method *ST\_ToCurvePoly()*:
  - a) If *SELF* is of type *ST\_CurvePolygon*, then return *SELF*.
  - b) If *SELF* is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.
  - c) Otherwise, return an empty set of type *ST\_CurvePolygon*.
- 15) Use the method *ST\_ToCurvePoly()* to define an implicitly invocable cast function to cast an *ST\_Geometry* value to an *ST\_CurvePolygon* value.
- 16) The method *ST\_ToPolygon()* has no input parameters.
- 17) For the null-call method *ST\_ToPolygon()*:
  - a) If *SELF* is of type *ST\_Polygon*, then return *SELF*.
  - b) If *SELF* is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.
  - c) Otherwise, return an empty set of type *ST\_Polygon*.
- 18) Use the method *ST\_ToPolygon()* to define an implicitly invocable cast function to cast an *ST\_Geometry* value to an *ST\_Polygon* value.
- 19) The method *ST\_ToGeomColl()* has no input parameters.

- 20) For the null-call method *ST\_ToGeomColl()*:
- If *SELF* is of type *ST\_GeomCollection*, then return *SELF*.
  - If *SELF* is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.
  - Otherwise, return an empty set of type *ST\_GeomCollection*.
- 21) Use the method *ST\_ToGeomColl()* to define an implicitly invocable cast function to cast an *ST\_Geometry* value to an *ST\_GeomCollection* value.
- 22) The method *ST\_ToMultiPoint()* has no input parameters.
- 23) For the null-call method *ST\_ToMultiPoint()*:
- If *SELF* is of type *ST\_MultiPoint*, then return *SELF*.
  - If *SELF* is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.
  - Otherwise, return an empty set of type *ST\_MultiPoint*.
- 24) Use the method *ST\_ToMultiPoint()* to define an implicitly invocable cast function to cast an *ST\_Geometry* value to an *ST\_MultiPoint* value.
- 25) The method *ST\_ToMultiCurve()* has no input parameters.
- 26) For the null-call method *ST\_ToMultiCurve()*:
- If *SELF* is of type *ST\_MultiCurve*, then return *SELF*.
  - If *SELF* is not an empty set value, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.
  - Otherwise, return an empty set of type *ST\_MultiCurve*.
- 27) Use the method *ST\_ToMultiCurve()* to define an implicitly invocable cast function to cast an *ST\_Geometry* value to an *ST\_MultiCurve* value.
- 28) The method *ST\_ToMultiLine()* has no input parameters.
- 29) For the null-call method *ST\_ToMultiLine()*:
- If *SELF* is of type *ST\_MultiLineString*, then return *SELF*.
  - If *SELF* is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.
  - Otherwise, return an empty set of type *ST\_MultiLineString*.
- 30) Use the method *ST\_ToMultiLine()* to define an implicitly invocable cast function to cast an *ST\_Geometry* value to an *ST\_MultiLineString* value.
- 31) The method *ST\_ToMultiSurface()* has no input parameters.
- 32) For the null-call method *ST\_ToMultiSurface()*:
- If *SELF* is of type *ST\_MultiSurface*, then return *SELF*.
  - If *SELF* is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.
  - Otherwise, return an empty set of type *ST\_MultiSurface*.
- 33) Use the method *ST\_ToMultiSurface()* to define an implicitly invocable cast function to cast an *ST\_Geometry* value to an *ST\_MultiSurface* value.
- 34) The method *ST\_ToMultiPolygon()* has no input parameters.

- 35) For the null-call method *ST\_ToMultiPolygon()*:
- a) If *SELF* is of type *ST\_MultiPolygon*, then return *SELF*.
  - b) If *SELF* is not an empty set, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.
  - c) Otherwise, return an empty set of type *ST\_MultiPolygon*.
- 36) Use the method *ST\_ToMultiPolygon()* to define an implicitly invocable cast function to cast an *ST\_Geometry* value to an *ST\_MultiPolygon* value.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 5.1.30 ST\_WKTTToSQL Method

#### Purpose

Return a specified ST\_Geometry value.

#### Definition

```
CREATE METHOD ST_WKTTToSQL
  (awkt CHARACTER LARGE OBJECT (ST_MaxGeometryAsText))
  RETURNS ST_Geometry
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END
```

#### Definitional Rules

- 1) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

#### Description

- 1) The method *ST\_WKTTToSQL*(CHARACTER LARGE OBJECT) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) The parameter *awkt* is the well-known text representation of an *ST\_Geometry* value. If *awkt* is not producible in the BNF for <well-known text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
- 3) The null-call method *ST\_WKTTToSQL*(CHARACTER LARGE OBJECT) returns an *ST\_Geometry* value represented by *awkt*.

### 5.1.31 ST\_AsText Method

#### Purpose

Return the well-known text representation of an ST\_Geometry value.

#### Definition

```
CREATE METHOD ST_AsText()  
  RETURNS CHARACTER LARGE OBJECT(ST_MaxGeometryAsText)  
  FOR ST_Geometry  
  BEGIN  
    --  
    -- See Description  
    --  
  END
```

#### Definitional Rules

- 1) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

#### Description

- 1) The method *ST\_AsText()* has no input parameters.
- 2) The null-call method *ST\_AsText()* returns a CHARACTER LARGE OBJECT value containing the well-known text representation of SELF. Values shall be produced in the BNF for <well-known text representation>.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

**5.1.32 ST\_WKBTtoSQL Method****Purpose**

Return a specified ST\_Geometry value.

**Definition**

```
CREATE METHOD ST_WKBTtoSQL
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_Geometry
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END
```

**Definitional Rules**

- 1) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.

**Description**

- 1) The method *ST\_WKBTtoSQL(BINARY LARGE OBJECT)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 2) The parameter *awkb* is the well-known binary representation of an *ST\_Geometry* value. If *awkb* is not producible in the BNF for <well-known binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
- 3) The null-call method *ST\_WKBTtoSQL(BINARY LARGE OBJECT)* returns an *ST\_Geometry* value represented by *awkb*.

### 5.1.33 ST\_AsBinary Method

#### Purpose

Return the well-known binary representation of an ST\_Geometry value.

#### Definition

```
CREATE METHOD ST_AsBinary()  
  RETURNS BINARY LARGE OBJECT(ST_MaxGeometryAsBinary)  
  FOR ST_Geometry  
  BEGIN  
    --  
    -- See Description  
    --  
  END
```

#### Definitional Rules

- 1) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.

#### Description

- 1) The method *ST\_AsBinary()* has no input parameters.
- 2) The null-call method *ST\_AsBinary()* returns a BINARY LARGE OBJECT value containing the well-known binary representation of SELF. Values shall be produced in the BNF for <well-known binary representation>.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 5.1.34 ST\_GMLToSQL Method

#### Purpose

Return a specified ST\_Geometry value.

#### Definition

```
CREATE METHOD ST_GMLToSQL
  (agml CHARACTER LARGE OBJECT (ST_MaxGeometryAsGML))
  RETURNS ST_Geometry
  FOR ST_Geometry
  BEGIN
    --
    -- See Description
    --
  END
```

#### Definitional Rules

- 1) *ST\_MaxGeometryAsGML* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the GML representation of an *ST\_Geometry* value.

#### Description

- 1) The method *ST\_GMLToSQL*(CHARACTER LARGE OBJECT) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*.
- 2) The parameter *agml* is the GML representation of an *ST\_Geometry* value. The instantiable subtypes are mapped to XML elements in the GML representation as follows:
  - a) an *ST\_Point* value is represented as a Point XML element.
  - b) an *ST\_LineString* value is represented as a LineString XML element.
  - c) an *ST\_CircularString* value is approximated as an *ST\_LineString* value using the *ST\_CurveToLine* method and the resulting *ST\_LineString* value is represented as a LineString XML element.
  - d) an *ST\_CompoundCurve* value is approximated as an *ST\_LineString* value using the *ST\_CurveToLine* method and the resulting *ST\_LineString* value is represented as a LineString XML element.
  - e) an *ST\_CurvePolygon* value is approximated as an *ST\_Polygon* value using the *ST\_CurvePolyToPoly* method and the resulting *ST\_Polygon* value is represented as a Polygon XML element.
  - f) an *ST\_Polygon* value is represented as a Polygon XML element.
  - g) an *ST\_GeomCollection* value is represented as a GeometryCollection XML element. *ST\_CircularString* values contained in the *ST\_GeomCollection* value are approximated as *ST\_LineString* values using the *ST\_CurveToLine* method and the resulting *ST\_LineString* values are represented as LineString XML elements within the GeometryCollection XML element. *ST\_CompoundCurve* values contained in the *ST\_GeomCollection* value are approximated as *ST\_LineString* values using the *ST\_CurveToLine* method and the resulting *ST\_LineString* values are represented as LineString XML elements within the GeometryCollection XML element. *ST\_CurvePolygon* values contained in the *ST\_GeomCollection* value are approximated as *ST\_Polygon* values using the *ST\_CurvePolyToPoly* method and the resulting *ST\_Polygon* values are represented as Polygon XML elements within the GeometryCollection XML element.
  - h) an *ST\_MultiPoint* value is represented as a MultiPoint XML element.

- i) an *ST\_MultiCurve* value is represented as a MultiLineString XML element. *ST\_CircularString* values contained in the *ST\_MultiCurve* value are approximated as *ST\_LineString* values using the *ST\_CurveToLine* method and the resulting *ST\_LineString* values are represented as LineString XML elements within the MultiLineString XML element. *ST\_CompoundCurve* values contained in the *ST\_MultiCurve* value are approximated as *ST\_LineString* values using the *ST\_CurveToLine* method and the resulting *ST\_LineString* values are represented as LineString XML elements within the MultiLineString XML element.
  - j) an *ST\_MultiLineString* value is represented as a MultiLineString XML element.
  - k) an *ST\_MultiSurface* value is represented as a MultiPolygon XML element. *ST\_CurvePolygon* values contained in the *ST\_MultiSurface* value are approximated as *ST\_Polygon* values using the *ST\_CurvePolyToPoly* method and the resulting *ST\_Polygon* values are represented as Polygon XML elements within the MultiPolygon XML element.
  - l) an *ST\_MultiPolygon* value is represented as a MultiPolygon XML element.
- 3) The *srsname* attribute of the XML element identifies its spatial referencing system. Select the row in the *SPATIAL\_REF\_SYS* view where the *srid* is equal to *SELF.ST\_SRID()*. For the selected row, let *AN* be the value of the *AUTH\_NAME* column, *AI* be the value of the *AUTH\_ID* column and *SRT* be the value of the *SRTEXT* column.
- Case:
- a) If the *AN* is not the null value and *AI* is not the null value then the *srsname* attribute is specified as:
 

```
srsname='AN:AI'
```
  - b) Otherwise, the *srsname* attribute is specified as:
 

```
srsname='SRT'
```
- 4) The null-call method *ST\_GMLToSQL*(*CHARACTER LARGE OBJECT*) returns an *ST\_Geometry* value represented by *agml*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 5.1.35 ST\_AsGML Method

#### Purpose

Return the GML representation of an *ST\_Geometry* value.

#### Definition

```
CREATE METHOD ST_AsGML()  
  RETURNS CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML)  
  FOR ST_Geometry  
  BEGIN  
    --  
    -- See Description  
    --  
  END
```

#### Definitional Rules

- 1) *ST\_MaxGeometryAsGML* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the GML representation of an *ST\_Geometry* value.

#### Description

- 1) The method *ST\_AsGML()* has no input parameters.
- 2) The null-call method *ST\_AsGML()* returns a CHARACTER LARGE OBJECT value containing a GML representation.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 5.1.36 ST\_GeomFromText Functions

#### Purpose

Return a specified ST\_Geometry value.

#### Definition

```
CREATE FUNCTION ST_GeomFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_Geometry
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN ST_GeomFromText(awkt, 0)

CREATE FUNCTION ST_GeomFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_Geometry
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  BEGIN
    --
    -- See Description
    --
  END
```

#### Definitional Rules

- 1) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

#### Description

- 1) The function *ST\_GeomFromText*(CHARACTER LARGE OBJECT) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call function *ST\_GeomFromText*(CHARACTER LARGE OBJECT):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_Geometry* value. If *awkt* is not producible in the BNF for <well-known text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_Geometry* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_GeomFromText*(CHARACTER LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_GeomFromText*(CHARACTER LARGE OBJECT, INTEGER):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_Geometry* value. If *awkt* is not producible in the BNF for <well-known text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

- b) Return an *ST\_Geometry* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 5.1.37 ST\_GeomFromWKB Functions

#### Purpose

Return a specified ST\_Geometry value.

#### Definition

```
CREATE FUNCTION ST_GeomFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_Geometry
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN ST_GeomFromWKB(awkb, 0)

CREATE FUNCTION ST_GeomFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_Geometry
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  BEGIN
    --
    -- See Description
    --
  END
```

#### Definitional Rules

- 1) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.

#### Description

- 1) The function *ST\_GeomFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 2) For the null-call function *ST\_GeomFromWKB(BINARY LARGE OBJECT)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_Geometry* value. If *awkb* is not producible in the BNF for <well-known binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_Geometry* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_GeomFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_GeomFromWKB(BINARY LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_Geometry* value. If *awkb* is not producible in the BNF for <well-known binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_Geometry* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.

### 5.1.38 ST\_GeomFromGML Functions

#### Purpose

Return a specified ST\_Geometry value.

#### Definition

```
CREATE FUNCTION ST_GeomFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
  RETURNS ST_Geometry
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  BEGIN
  --
  -- See Description
  --
  END

CREATE FUNCTION ST_GeomFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
   ansrid INTEGER)
  RETURNS ST_Geometry
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  BEGIN
  --
  -- See Description
  --
  END
```

#### Definitional Rules

- 1) *ST\_MaxGeometryAsGML* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the GML representation of an *ST\_Geometry* value.

#### Description

- 1) The function *ST\_GeomFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*.
- 2) For the null-call function *ST\_GeomFromGML(CHARACTER LARGE OBJECT)*:
  - a) If the parameter *agml* does not contain a GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_Geometry* value represented by *agml* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_GeomFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*,
  - b) an INTEGER value *ansrid*.

- 4) For the null-call function *ST\_GeomFromGML*(*CHARACTER LARGE OBJECT*, *INTEGER*):
- a) If the parameter *agml* does not contain a GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_Geometry* value represented by *agml* with the spatial reference system identifier set to *ansrid*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

**5.1.39 ST\_Geometry Ordering Definition****Purpose**

Test if two ST\_Geometry values are equal.

**Definition**

```

CREATE FUNCTION ST_OrderingEquals
  (ageometry ST_Geometry,
   anothergeometry ST_Geometry)
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  STATIC DISPATCH
  RETURN
  CASE
    WHEN ageometry.ST_Equals(anothergeometry) = 1 THEN
      0
    ELSE
      1
  END

CREATE ORDERING FOR ST_Geometry
  EQUALS ONLY BY RELATIVE WITH
  FUNCTION ST_OrderingEquals(ST_Geometry, ST_Geometry)

```

**Description**

- 1) The function *ST\_OrderingEquals(ST\_Geometry, ST\_Geometry)* takes the following input parameters:
  - a) an *ST\_Geometry* value *ageometry*,
  - b) an *ST\_Geometry* value *anothergeometry*.
- 2) For the null-call function *ST\_OrderingEquals(ST\_Geometry, ST\_Geometry)*:
 

Case:

  - a) If the result of the value expression: *ageometry.ST\_Equals(anothergeometry)* is 1 (one), then return 0 (zero).
  - b) Otherwise, return 1 (one)
- 3) Use the function *ST\_OrderingEquals(ST\_Geometry, ST\_Geometry)* to define ordering for the *ST\_Geometry* type.

### 5.1.40 SQL Transform Functions

#### Purpose

Define SQL transform functions for the *ST\_Geometry* type.

#### Definition

```
CREATE TRANSFORM FOR ST_Geometry
  ST_WellKnownText
    (TO SQL WITH METHOD ST_WKTTToSQL
     (CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))),
    FROM SQL WITH METHOD ST_AsText())
  ST_WellKnownBinary
    (TO SQL WITH METHOD ST_WKBTToSQL
     (BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))),
    FROM SQL WITH METHOD ST_AsBinary())
  ST_GML
    (TO SQL WITH METHOD ST_GMLToSQL
     (CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))),
    FROM SQL WITH METHOD ST_AsGML())
```

#### Definitional Rules

- 1) *ST\_MaxGeometryAsText* is the implementation-defined maximum 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) *ST\_MaxGeometryAsGML* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the GML representation of an *ST\_Geometry* value.

#### Description

- 1) Use the method *ST\_WKTTToSQL(CHARACTER LARGE OBJECT)* and the method *ST\_AsText()* to define the transform group *ST\_WellKnownText*.
- 2) Use the method *ST\_WKBTToSQL(BINARY LARGE OBJECT)* and the method *ST\_AsBinary()* to define the transform group *ST\_WellKnownBinary*.
- 3) Use the method *ST\_GMLToSQL(CHARACTER LARGE OBJECT)* and the method *ST\_AsGML()* to define the transform group *ST\_GML*.

**5.1.41 <well-known text representation>****Purpose**

This subclause contains the definition of <well-known text representation>.

**Description**

- 1) The well-known text representation of an *ST\_Geometry* value is defined by the following BNF for <well-known text representation>.

```

<well-known text representation> ::=
    <point text representation>
  | <curve text representation>
  | <surface text representation>
  | <collection text representation>

<point text representation> ::=
    POINT <point text>

<curve text representation> ::=
    <linestring text representation>
  | <circularstring text representation>
  | <compoundcurve text representation>

<linestring text representation> ::=
    LINESTRING <linestring text body>

<circularstring text representation> ::=
    CIRCULARSTRING <circularstring text>

<compoundcurve text representation> ::=
    COMPOUNDCURVE <compoundcurve text>

<surface text representation> ::=
    <curvopolygon text representation>

<curvopolygon text representation> ::=
    CURVEPOLYGON <curvopolygon text body>
  | <polygon text representation>

<polygon text representation> ::=
    POLYGON <polygon text body>

<collection text representation> ::=
    <multipoint text representation>
  | <multicurve text representation>
  | <multisurface text representation>
  | <geometrycollection text representation>

<multipoint text representation> ::=
    MULTIPOINT <multipoint text>

<multicurve text representation> ::=
    MULTICURVE <multicurve text>
  | <multilinestring text representation>

<multilinestring text representation> ::=
    MULTILINESTRING <multilinestring text>

<multisurface text representation> ::=
    MULTISURFACE <multisurface text>
  | <multipolygon text representation>

<multipolygon text representation> ::=
    MULTIPOLYGON <multipolygon text>

<geometrycollection text representation> ::=
    GEOMETRYCOLLECTION <geometrycollection text>

```

```

<linestring text body> ::=
  <linestring text>

<curvepolygon text body> ::=
  <curvepolygon text>

<polygon text body> ::=
  <polygon text>

<point text> ::=
  <empty set>
  | <left paren> <point> <right paren>

<point> ::= <x> <y>

<x> ::= <number>

<y> ::= <number>

<linestring text> ::=
  <empty set>
  | <left paren> <point>
    { <comma> <point> }... <right paren>

<circularstring text> ::=
  <empty set>
  | <left paren> <point>
    { <comma> <point> }... <right paren>

<compoundcurve text> ::=
  <empty set>
  | <left paren> <single curve text>
    { <comma> <single curve text> }... <right paren>

<single curve text> ::=
  <linestring text body>
  | <circularstring text representation>

<curve text> ::=
  <linestring text body>
  | <circularstring text representation>
  | <compoundcurve text representation>

<ring text> ::=
  <linestring text body>
  | <circularstring text representation>
  | <compoundcurve text representation>

<surface text> ::=
  CURVEPOLYGON <curvepolygon text body>
  | <polygon text body>

<curvepolygon text> ::=
  <empty set>
  | <left paren> <ring text>
    { <comma> <ring text> }... <right paren>

<polygon text> ::=
  <empty set>
  | <left paren> <linestring text>
    { <comma> <linestring text> }... <right paren>

<multipoint text> ::=
  <empty set>
  | <left paren> <point text>
    { <comma> <point text > }... <right paren>

```

```

<multicurve text> ::=
  <empty set>
  | <left paren> <curve text>
    { <comma> <curve text> }... <right paren>

<multilinestring text> ::=
  <empty set>
  | <left paren> <linestring text body>
    { <comma> <linestring text body> }... <right paren>

<multisurface text> ::=
  <empty set>
  | <left paren> <surface text>
    { <comma> <surface text> }... <right paren>

<multipolygon text> ::=
  <empty set>
  | <left paren> <polygon text body>
    { <comma> <polygon text body> }... <right paren body>

<geometrycollection text> ::=
  <empty set>
  | <left paren> <well-known text representation>
    { <comma> <well-known text representation> }... <right paren>

<empty set> ::= EMPTY

```

## a) Case:

- i) If <well-known text representation> immediately contains a <point text representation>, then <well-known text representation> produces an *ST\_Point* value specified by the immediately contained <point text representation>.
- ii) If <well-known text representation> immediately contains a <curve text representation>, then <well-known text representation> produces an *ST\_Curve* value specified by the immediately contained <curve text representation>.
- iii) If <well-known text representation> immediately contains a <surface text representation>, then <well-known text representation> produces an *ST\_Surface* value specified by the immediately contained <surface text representation>.
- iv) Otherwise, <well-known text representation> produces an *ST\_GeomCollection* value specified by the immediately contained <collection text representation>.

b) <point text representation> is the well-known text representation for an *ST\_Point* value that is produced by <point text>.

## c) Case:

- i) If <curve text representation> immediately contains a <linestring text representation>, then <curve text representation> produces an *ST\_LineString* value specified by the immediately contained <linestring text representation>.
- ii) If <curve text representation> immediately contains a <circularstring text representation>, then <curve text representation> produces an *ST\_CircularString* value specified by the immediately contained <circularstring text representation>.
- iii) Otherwise, <curve text representation> produces an *ST\_CompoundCurve* value specified by the immediately contained <compoundcurve text representation>.

d) <linestring text representation> is the well-known text representation for an *ST\_LineString* value. <linestring text representation> produces an *ST\_LineString* value specified by the immediately contained <linestring text body>.

e) <circularstring text representation> is the well-known text representation for an *ST\_CircularString* value. Let *APA* be the *ST\_Point* ARRAY value produced by a <circularstring text>.

Case:

- i) If the cardinality of *APA* is 0 (zero), then <circularstring text representation> produces an empty set of type *ST\_CircularString*.
- ii) Otherwise, <circularstring text representation> produces an *ST\_CircularString* value as the result of the value expression: `NEW ST_CircularString(APA)`.

f) <compoundcurve text representation> is the well-known text representation for an *ST\_CompoundCurve* value. Let *ACA* be the *ST\_Curve* ARRAY value produced by a <compoundcurve text>.

Case:

- i) If the cardinality of *ACA* is 0 (zero), then <compoundcurve text representation> produces an empty set of type *ST\_CompoundCurve*.
- ii) Otherwise, <compoundcurve text representation> produces an *ST\_CompoundCurve* value as the result of the value expression: `NEW ST_CompoundCurve(ACA)`.

g) <surface text representation> produces an *ST\_Surface* value specified by the immediately contained <curvopolygon text representation>.

h) <curvopolygon text representation> is the well-known text representation for an *ST\_CurvePolygon* value.

Case:

- i) If <curvopolygon text representation> immediately contains a <curvopolygon text body>, then <curvopolygon text representation> produces an *ST\_CurvePolygon* value specified by the immediately contained <curvopolygon text body>.
- ii) Otherwise, <curvopolygon text representation> produces an *ST\_Polygon* value specified by the immediately contained <polygon text representation>.

i) <polygon text representation> is the well-known text representation for an *ST\_Polygon* value. <polygon text representation> produces an *ST\_Polygon* value specified by the immediately contained <polygon text body>.

j) Case:

- i) If <collection text representation> immediately contains a <multipoint text representation>, then <collection text representation> produces an *ST\_MultiPoint* value specified by the immediately contained <multipoint text representation>.
- ii) If <collection text representation> immediately contains a <multicurve text representation>, then <collection text representation> produces an *ST\_MultiCurve* value specified by the immediately contained <multicurve text representation>.
- iii) If <collection text representation> immediately contains a <multisurface text representation>, then <collection text representation> produces an *ST\_MultiSurface* value specified by the immediately contained <multisurface text representation>.
- iv) Otherwise, <collection text representation> produces an *ST\_GeomCollection* value specified by the immediately contained <geometrycollection text representation>.

k) <multipoint text representation> is the well-known text representation for an *ST\_MultiPoint* value. Let *APA* be the *ST\_Point* ARRAY value produced by a <multipoint text>.

Case:

- i) If the cardinality of *APA* is 0 (zero), then <multipoint text representation> produces an empty set of type *ST\_MultiPoint*.
- ii) Otherwise, <multipoint text representation> produces an *ST\_MultiPoint* value as the result of the value expression: `NEW ST_MultiPoint(APA)`.

l) Case:

- i) If <multicurve text representation> immediately contains a <multicurve text>, then <multicurve text representation> produces an *ST\_MultiCurve* value. Let *ACA* be the *ST\_Curve* ARRAY value produced by a <multicurve text>.

Case:

- 1) If the cardinality of *ACA* is 0 (zero), then <multicurve text representation> produces an empty set of type *ST\_MultiCurve*.
- 2) Otherwise, <multicurve text representation> produces an *ST\_MultiCurve* value as the result of the value expression: *NEW ST\_MultiCurve(ACA)*.

- ii) Otherwise, <multicurve text representation> produces an *ST\_MultiLineString* value specified by the immediately contained <multilinestring text representation>.

- m) <multilinestring text representation> is the well-known text representation for an *ST\_MultiLineString* value. Let *ALSA* be the *ST\_LineString* ARRAY value produced by a <multilinestring text>.

Case:

- i) If the cardinality of *ALSA* is 0 (zero), then <multilinestring text representation> produces an empty set of type *ST\_MultiLineString*.
- ii) Otherwise, <multilinestring text representation> produces an *ST\_MultiLineString* value as the result of the value expression: *NEW ST\_MultiLineString(ALSA)*.

n) Case:

- i) If <multisurface text representation> immediately contains a <multisurface text>, then <multisurface text representation> produces an *ST\_MultiSurface* value. Let *ASA* be the *ST\_Surface* ARRAY value produced by a <multisurface text>.

Case:

- 1) If the cardinality of *ASA* is 0 (zero), then <multisurface text representation> produces an empty set of type *ST\_MultiSurface*.
- 2) Otherwise, <multisurface text representation> produces an *ST\_MultiSurface* value as the result of the value expression: *NEW ST\_MultiSurface(ASA)*.

- ii) Otherwise, <multisurface text representation> produces an *ST\_MultiPolygon* value specified by the immediately contained <multipolygon text representation>.

- o) <multipolygon text representation> is the well-known text representation for an *ST\_MultiPolygon* value. Let *APA* be the *ST\_Polygon* ARRAY value produced by a <multipolygon text>.

Case:

- i) If the cardinality of *APA* is 0 (zero), then <multipolygon text representation> produces an empty set of type *ST\_MultiPolygon*.
- ii) Otherwise, <multipolygon text representation> produces an *ST\_MultiPolygon* value as the result of the value expression: *NEW ST\_MultiPolygon(APA)*.

- p) <geometrycollection text representation> is the well-known text representation for an *ST\_GeomCollection*. Let *AGA* be the *ST\_Geometry* ARRAY value produced by a <geometrycollection text>.

Case:

- i) If the cardinality of *AGA* is 0 (zero), then <geometrycollection text representation> produces an empty set of type *ST\_GeomCollection*.
- ii) Otherwise, <geometrycollection text representation> produces an *ST\_GeomCollection* value as the result of the value expression: *NEW ST\_GeomCollection(AGA)*.

q) Let *APA* be the *ST\_Point* ARRAY value produced by a <linestring text> in <linestring text body>.

Case:

- i) If the cardinality of *APA* is 0 (zero), then <linestring text body> produces an empty set of type *ST\_LineString*.
- ii) Otherwise, <linestring text body> produces an *ST\_LineString* value as the result of the value expression: `NEW ST_LineString(APA)`.

r) Let *ACA* be the *ST\_Curve* ARRAY value produced by a <curvepolygon text> in <curvepolygon text body>.

Case:

- i) If the cardinality of *ACA* is 0 (zero), then <curvepolygon text body> produces an empty set of type *ST\_CurvePolygon*.
- ii) If the cardinality of *ACA* is 1 (one), then let *AER* be the element of *ACA*. <curvepolygon text body> produces an *ST\_CurvePolygon* value as the result of the value expression: `NEW ST_CurvePolygon(AER)`.
- iii) Otherwise, let *AER* be the first element in *ACA* and let *AIR* be the sublist of *ACA* containing the other elements of *ACA*. <curvepolygon text body> produces an *ST\_CurvePolygon* value as the result of the value expression: `NEW ST_CurvePolygon(AER, AIR)`.

s) Let *ALSA* be the *ST\_LineString* ARRAY value produced by a <polygon text> in <polygon text body>.

Case:

- i) If the cardinality of *ALSA* is 0 (zero), then <polygon text representation><polygon text body> produces an empty set of type *ST\_Polygon*.
- ii) If the cardinality of *ALSA* is 1 (one), then let *ALS* be the element of *ALSA*. <polygon text body> produces an *ST\_Polygon* value as the result of the value expression: `NEW ST_Polygon(ALS)`.
- iii) Otherwise, let *AER* be the first element in *ALSA* and let *AIR* be the sublist of *ALSA* containing the other elements of *ALSA*. <polygon text body> produces an *ST\_Polygon* value as the result of the value expression: `NEW ST_Polygon(AER, AIR)`.

t) Case:

- i) If <point text> immediately contains an <empty set>, then <point text> produces an empty *ST\_Point* value.
- ii) Otherwise, <point text> produces the *ST\_Point* value from <point>.

u) Let *XC* be the DOUBLE PRECISION value specified by <x> in <point> and *YC* be the DOUBLE PRECISION value specified by <y> in <point>. <point> produces an *ST\_Point* value as the result of the value expression: `NEW ST_Point(XC, YC)`.

v) Case:

- i) If <linestring text> immediately contains an <empty set>, then <linestring text> produces an empty *ST\_Point* ARRAY value.
- ii) Otherwise, <linestring text> produces an *ST\_Point* ARRAY value that contains the *ST\_Point* values specified by the immediately contained <point>s.

w) Case:

- i) If <circularstring text> immediately contains an <empty set>, then <circularstring text> produces an empty *ST\_Point* ARRAY value.
- ii) Otherwise, <circularstring text> produces an *ST\_Point* ARRAY value that contains the *ST\_Point* values specified by the immediately contained <point>s.

- x) Case:
- i) If <compoundcurve text> immediately contains an <empty set>, then <compoundcurve text> produces an empty *ST\_Curve* ARRAY value.
  - ii) Otherwise, <compoundcurve text> produces an *ST\_Curve* ARRAY value that contains the *ST\_Curve* values specified by the immediately contained <single curve text>s.
- y) Case:
- i) If <single curve text> immediately contains a <linestring text body>, then <single curve text> produces an *ST\_LineString* value specified by the immediately contained <linestring text body>.
  - ii) Otherwise, <single curve text> produces an *ST\_CircularString* value specified by the immediately contained <circularstring text representation>.
- z) Case:
- i) If <curve text> immediately contains a <linestring text body>, then <curve text> produces an *ST\_LineString* value specified by the immediately contained <linestring text body>.
  - ii) If <curve text> immediately contains a <circularstring text representation>, then <curve text> produces an *ST\_CircularString* value specified by the immediately contained <circularstring text>.
  - iii) Otherwise, <curve text> produces an *ST\_CompoundCurve* value specified by the immediately contained <compoundcurve text representation>.
- aa) Case:
- i) If <ring text> immediately contains a <linestring text body>, then <ring text> produces an *ST\_LineString* value specified by the immediately contained <linestring text body>.
  - ii) If <ring text> immediately contains a <circularstring text representation>, then <ring text> produces an *ST\_CircularString* value specified by the immediately contained <circularstring text representation>.
  - iii) Otherwise, <ring text> produces an *ST\_CompoundCurve* value specified by the immediately contained <compoundcurve text representation>.
- ab) Case:
- i) If <surface text> immediately contains a <curvepolygon text body>, then <surface text> produces an *ST\_CurvePolygon* value specified by the immediately contained <curvepolygon text body>.
  - ii) Otherwise, <surface text> produces an *ST\_Polygon* value specified by the immediately contained <polygon text body>.
- ac) Case:
- i) If <curvepolygon text> immediately contains an <empty set>, then <curvepolygon text> produces an empty *ST\_Curve* ARRAY value.
  - ii) Otherwise, <curvepolygon text> produces an *ST\_Curve* ARRAY value that contains the *ST\_Curve* values specified by the immediately contained <ring text>s.
- ad) Case:
- i) If <polygon text> immediately contains an <empty set>, then <polygon text> produces an empty *ST\_LineString* ARRAY value.
  - ii) Otherwise, <polygon text> produces an *ST\_LineString* ARRAY value that contains the *ST\_LineString* values specified by the immediately contained <linestring text>s.
- ae) Case:
- i) If <multipoint text> immediately contains an <empty set>, then <multipoint text> produces an empty *ST\_Point* ARRAY value.

- ii) Otherwise, <multipoint text> produces an *ST\_Point* ARRAY value that contains the *ST\_Point* values specified by the immediately contained <point text>s.
- af) Case:
- i) If <multicurve text> immediately contains an <empty set>, then <multicurve text> produces an empty *ST\_Curve* ARRAY value.
  - ii) Otherwise, <multicurve text> produces an *ST\_Curve* ARRAY value that contains the *ST\_Curve* values specified by the immediately contained <curve text>s.
- ag) Case:
- i) If <multilinestring text> immediately contains an <empty set>, then <multilinestring text> produces an empty *ST\_LineString* ARRAY value.
  - ii) Otherwise, <multilinestring text> produces an *ST\_LineString* ARRAY value that contains the *ST\_LineString* values specified by the immediately contained <linestring text body>s.
- ah) Case:
- i) If <multisurface text> immediately contains an <empty set>, then <multisurface text> produces an empty *ST\_Surface* ARRAY value.
  - ii) Otherwise, <multisurface text> produces an *ST\_Polygon* ARRAY value that contains the *ST\_Surface* values from the immediately contained <surface text>s.
- ai) Case:
- i) If <multipolygon text> immediately contains an <empty set>, then <multipolygon text> produces an empty *ST\_Polygon* ARRAY value.
  - ii) Otherwise, <multipolygon text> produces an *ST\_Polygon* ARRAY value that contains the *ST\_Polygon* values from the immediately contained <polygon text body>s.
- aj) Case:
- i) If <geometrycollection text> immediately contains an <empty set>, then <geometrycollection text> produces an empty *ST\_Geometry* ARRAY.
  - ii) Otherwise, an *ST\_Geometry* ARRAY value that contains the *ST\_Geometry* values from the immediately contained <well-known text representation>s.

**5.1.42 <well-known binary representation>****Purpose**

This subclause contains the definition of <well-known binary representation>.

**Description**

- 1) The well-known binary representation of an *ST\_Geometry* value is defined by the following BNF for <well-known binary representation>.

```

<well-known binary representation> ::=
    <point binary representation>
  | <curve binary representation>
  | <surface binary representation>
  | <collection binary representation>

<point binary representation> ::=
    <byte order> <wkbpnt> [ <wkbpnt binary> ]

<curve binary representation> ::=
    <linestring binary representation>
  | <circularstring binary representation>
  | <compoundcurve binary representation>

<linestring binary representation> ::=
    <byte order> <wkblinestring> [ <num> <wkbpnt binary>... ]

<circularstring binary representation> ::=
    <byte order> <wkbcircularstring> [ <num> <wkbpnt binary>... ]

<compoundcurve binary representation> ::=
    <byte order> <wkbcompoundcurve> [ <num> <wkbcurve binary>... ]

<surface binary representation> ::=
    <curvepolygon binary representation>

<curvepolygon binary representation> ::=
    <byte order> <wkbcurvepolygon> [ <num> <wkbring binary>... ]
  | <polygon binary representation>

<polygon binary representation> ::=
    <byte order> <wkbpolygon> [ <num> <wkblinestring binary>... ]

<collection binary representation> ::=
    <multipoint binary representation>
  | <multicurve binary representation>
  | <multisurface binary representation>
  | <geometrycollection binary representation>

<multipoint binary representation> ::=
    <byte order> <wkbmultipoint>
  [ <num> <point binary representation>... ]

<multicurve binary representation> ::=
    <byte order> <wkbmulticurve>
  [ <num> <curve binary representation>... ]
  | <multilinestring binary representation>

<multilinestring binary representation> ::=
    <byte order> <wkbmultilinestring>
  [ <num> <linestring binary representation>... ]

<multisurface binary representation> ::=
    <byte order> <wkbmultisurface>
  [ <num> <surface binary representation>... ]
  | <multipolygon binary representation>

```

```

<multipolygon binary representation> ::=
  <byte order> <wkbmultipolygon>
    [ <num> <polygon binary representation>... ]

<geometrycollection binary representation> ::=
  <byte order> <wkbgeometrycollection>
    [ <num> <well-known binary representation>... ]

<wkbcurve binary> ::=
  <linestring binary representation>
  | <circularstring binary representation>

<wkbring binary> ::=
  <linestring binary representation>
  | <circularstring binary representation>
  | <compoundcurve binary representation>

<wkbpoint binary> ::= <wkbx> <wkby>

<wkbx> ::= <double>
<wkby> ::= <double>

<num> ::= <uint32>

<wkblinearring> ::= <num> <wkbpoint binary>...

<wkbpoint> ::= <uint32>

<wkblinestring> ::= <uint32>

<wkbcircularstring> ::= <uint32>

<wkbcompoundcurve> ::= <uint32>

<wkbpolygon> ::= <uint32>

<wkbcurvepolygon> ::= <uint32>

<wkbmultipoint> ::= <uint32>

<wkbmultilinestring> ::= <uint32>

<wkbmulticurve> ::= <uint32>

<wkbmultisurface> ::= <uint32>

<wkbmultipolygon> ::= <uint32>

<wkbgeometrycollection> ::= <uint32>

<byte order> ::=
  <big endian>
  | <little endian>

<big endian> ::= !! See Description
<little endian> ::= !! See Description
<byte> ::= !! See Description
<uint32> ::= !! See Description
<double> ::= !! See Description

```

## a) Case:

- i) If <well-known binary representation> immediately contains a <point binary representation>, then <well-known binary representation> produces an *ST\_Point* value specified by the immediately contained <point binary representation>.
- ii) If <well-known binary representation> immediately contains a <curve binary representation>, then <well-known binary representation> produces an *ST\_Curve* value specified by the immediately contained <curve binary representation>.

- iii) If <well-known binary representation> immediately contains a <surface binary representation>, then <well-known binary representation> produces an *ST\_Surface* value specified by the immediately contained <surface binary representation>.
  - iv) Otherwise, <well-known binary representation> produces an *ST\_GeomCollection* value specified by the immediately contained <collection binary representation>.
- b) Case:
- i) If <point binary representation> immediately contains a <wkbpoint binary>, then <point binary representation> is the well-known binary representation for an *ST\_Point* value that is produced by <wkbpoint binary>.
  - ii) Otherwise, <point binary representation> produces an empty set of type *ST\_Point*.
- c) Case:
- i) If <curve binary representation> immediately contains a <linestring binary representation>, then <curve binary representation> produces an *ST\_LineString* value specified by the immediately contained <linestring binary representation>.
  - ii) If <curve binary representation> immediately contains a <circularstring binary representation>, then <curve binary representation> produces an *ST\_CircularString* value specified by the immediately contained <circularstring binary representation>.
  - iii) Otherwise, <curve binary representation> produces an *ST\_CompoundCurve* value specified by the immediately contained <compoundcurve binary representation>.
- d) Case:
- i) If <linestring binary representation> immediately contains <num>, then <linestring binary representation> is the well-known binary representation for an *ST\_LineString* value. Let *APA* be an *ST\_Point* ARRAY value with cardinality of <num> that contains the *ST\_Point* values specified by the immediately contained <wkbpoint binary>s. <linestring binary representation> produces an *ST\_LineString* value as the result of the value expression: *NEW ST\_LineString(APA)*.
  - ii) Otherwise, <linestring binary representation> produces an empty set of type *ST\_LineString*.
- e) Case:
- i) If <circularstring binary representation> immediately contains <num>, then <circularstring binary representation> is the well-known binary representation for an *ST\_CircularString* value. Let *APA* be an *ST\_Point* ARRAY value with cardinality of <num> that contains the *ST\_Point* values specified by the immediately contained <wkbpoint binary>s. <linestring binary representation> produces an *ST\_CircularString* value as the result of the value expression: *NEW ST\_CircularString(APA)*.
  - ii) Otherwise, <circularstring binary representation> produces an empty set of type *ST\_CircularString*.
- f) Case:
- i) If <compoundcurve binary representation> immediately contains <num>, then <compoundcurve binary representation> is the well-known binary representation for an *ST\_CompoundCurve* value. Let *ACA* be an *ST\_Curve* ARRAY value with cardinality of <num> that contains the *ST\_Curve* values specified by the immediately contained <wkbcurve binary>s. <compoundcurve binary representation> produces an *ST\_CompoundCurve* value as the result of the value expression: *NEW ST\_CompoundCurve(ACA)*.
  - ii) Otherwise, <compoundcurve binary representation> produces an empty set of type *ST\_CompoundCurve*.
- g) <surface binary representation> produces an *ST\_Surface* value specified by the immediately contained <curvopolygon binary representation>.

h) Case:

- i) If  $\langle \text{curvepolygon binary representation} \rangle$  immediately contains a  $\langle \text{num} \rangle$ , then  $\langle \text{curvepolygon binary representation} \rangle$  produces an *ST\_CurvePolygon*. Let *ACA* be an *ST\_Curve* ARRAY value with cardinality of  $\langle \text{num} \rangle$  that contains the *ST\_Curve* values specified by the immediately contained  $\langle \text{wkbring binary} \rangle$ s.

Case:

- 1) If the cardinality of *ACA* is 0 (zero), then  $\langle \text{curvepolygon binary representation} \rangle$  produces an empty set of type *ST\_CurvePolygon*.
  - 2) If the cardinality of *ACA* is 1 (one), then let *AER* be the element of *ACA*.  $\langle \text{curvepolygon binary representation} \rangle$  produces an *ST\_CurvePolygon* value as the result of the value expression: `NEW ST_CurvePolygon(AER)`.
  - 3) Otherwise, let *AER* be the first element in *ACA* and let *AIR* be the sublist of *ACA* containing the other elements of *ACA*.  $\langle \text{curvepolygon binary representation} \rangle$  produces an *ST\_CurvePolygon* value as the result of the value expression: `NEW ST_CurvePolygon(AER, AIR)`.
- ii) If  $\langle \text{curvepolygon binary representation} \rangle$  immediately contains a  $\langle \text{polygon binary representation} \rangle$ , then  $\langle \text{curvepolygon binary representation} \rangle$  produces an *ST\_Polygon* value specified by the immediately contained  $\langle \text{polygon binary representation} \rangle$ .
- iii) Otherwise,  $\langle \text{curvepolygon binary representation} \rangle$  produces an empty set of type *ST\_CurvePolygon*.

i) Case:

- i) If  $\langle \text{polygon binary representation} \rangle$  immediately contains  $\langle \text{num} \rangle$ , then  $\langle \text{polygon binary representation} \rangle$  is the well-known binary representation for an *ST\_Polygon* value. Let *ALSA* be an *ST\_LineString* ARRAY value with cardinality of  $\langle \text{num} \rangle$  that contains the *ST\_LineString* values specified by the immediately contained  $\langle \text{wkblinestring binary} \rangle$ s.

Case:

- 1) If the cardinality of *ALSA* is 0 (zero), then  $\langle \text{polygon binary representation} \rangle$  produces an empty set of type *ST\_Polygon*.
  - 2) If the cardinality of *ALSA* is 1 (one), then let *ALS* be the element of *ALSA*.  $\langle \text{polygon binary representation} \rangle$  produces an *ST\_Polygon* value as the result of the value expression: `NEW ST_Polygon(ALS)`.
  - 3) Otherwise, let *AER* be the first element in *ALSA* and let *AIR* be the sublist of *ALSA* containing the other elements of *ALSA*.  $\langle \text{polygon binary representation} \rangle$  produces an *ST\_Polygon* value as the result of the value expression: `NEW ST_Polygon(AER, AIR)`.
- ii) Otherwise,  $\langle \text{polygon binary representation} \rangle$  produces an empty set of type *ST\_Polygon*.

j) Case:

- i) If  $\langle \text{collection binary representation} \rangle$  immediately contains a  $\langle \text{multipoint binary representation} \rangle$ , then  $\langle \text{collection binary representation} \rangle$  produces an *ST\_MultiPoint* value specified by the immediately contained  $\langle \text{multipoint binary representation} \rangle$ .
- ii) If  $\langle \text{collection binary representation} \rangle$  immediately contains a  $\langle \text{multicurve binary representation} \rangle$ , then  $\langle \text{collection binary representation} \rangle$  produces an *ST\_MultiCurve* value specified by the immediately contained  $\langle \text{multicurve binary representation} \rangle$ .
- iii) If  $\langle \text{collection binary representation} \rangle$  immediately contains a  $\langle \text{multisurface binary representation} \rangle$ , then  $\langle \text{collection binary representation} \rangle$  produces an *ST\_MultiSurface* value specified by the immediately contained  $\langle \text{multisurface binary representation} \rangle$ .
- iv) Otherwise,  $\langle \text{collection binary representation} \rangle$  produces an *ST\_GeomCollection* value specified by the immediately contained  $\langle \text{geometrycollection binary representation} \rangle$ .

## k) Case:

- i) If <multipoint binary representation> immediately contains <num>, then <multipoint binary representation> is the well-known binary representation for an *ST\_MultiPoint* value. Let *APA* be the *ST\_Point* ARRAY value with cardinality of <num> that contains the *ST\_Point* values specified by the immediately contained <point binary representation>s. <multipoint binary representation> produces an *ST\_MultiPoint* value as the result of the value expression: *NEW ST\_MultiPoint(APA)*.
- ii) Otherwise, <multipoint binary representation> produces an empty set of type *ST\_MultiPoint*.

## l) Case:

- i) If <multicurve binary representation> immediately contains a <num>, then <multicurve binary representation> produces an *ST\_MultiCurve* value. Let *ACA* be the *ST\_Curve* ARRAY value with cardinality of <num> that contains the *ST\_Curve* values specified by the immediately contained <curve binary representation>s. <multicurve binary representation> produces an *ST\_MultiCurve* value as the result of the value expression: *NEW ST\_MultiCurve(ACA)*.
- ii) If <multicurve binary representation> immediately contains a <multilinestring binary representation>, then <multicurve binary representation> produces an *ST\_MultiLineString* value specified by the immediately contained <multilinestring binary representation>.
- iii) Otherwise, <multicurve binary representation> produces an empty set of type *ST\_MultiCurve*.

## m) Case:

- i) If <multilinestring binary representation> immediately contains <num>, then <multilinestring binary representation> is the well-known binary representation for an *ST\_MultiLineString* value. Let *ALSA* be the *ST\_LineString* ARRAY value with cardinality of <num> that contains the *ST\_LineString* values specified by the immediately contained <linestring binary representation>s. <multilinestring binary representation> produces an *ST\_MultiLineString* value as the result of the value expression: *NEW ST\_MultiLineString(ALSA)*.
- ii) Otherwise, <multilinestring binary representation> produces an empty set of type *ST\_MultiLineString*.

## n) Case:

- i) If <multisurface binary representation> immediately contains a <num>, then <multisurface binary representation> produces an *ST\_MultiSurface* value. Let *ASA* be the *ST\_Surface* ARRAY value with cardinality of <num> that contains the *ST\_Surface* values specified by the immediately contained <surface binary representation>s. <multisurface binary representation> produces an *ST\_MultiSurface* value as the result of the value expression: *NEW ST\_MultiSurface(ASA)*.
- ii) If <multisurface binary representation> immediately contains a <multipolygon binary representation>, then <multisurface binary representation> produces an *ST\_MultiPolygon* value specified by the immediately contained <multipolygon binary representation>.
- iii) Otherwise, <multisurface binary representation> produces an empty set of type *ST\_MultiSurface*.

## o) Case:

- i) If <multipolygon binary representation> immediately contains <num>, then <multipolygon binary representation> is the well-known binary representation for an *ST\_MultiPolygon* value. Let *APA* be the *ST\_Polygon* ARRAY value with cardinality of <num> that contains the *ST\_Polygon* values specified by the immediately contained <polygon binary representation>s. <multipolygon binary representation> produces an *ST\_MultiPolygon* value as the result of the value expression: *NEW ST\_MultiPolygon(APA)*.
- ii) Otherwise, <multipolygon binary representation> produces an empty set of type *ST\_MultiPolygon*.

p) Case:

- i) If <geometrycollection binary representation> immediately contains <num>, then <geometrycollection binary representation> is the well-known binary representation for an *ST\_GeomCollection*. Let *AGA* be the *ST\_Geometry* ARRAY value with cardinality of <num> that contains the *ST\_Geometry* values specified by the immediately contained <well-known binary representation>s. <geometrycollection binary representation> produces an *ST\_GeomCollection* value as the result of the value expression: *NEW ST\_GeomCollection(AGA)*.
- ii) Otherwise, <geometrycollection binary representation> produces an empty set of type *ST\_GeomCollection*.

q) Case:

- i) If <wkbcurve binary> immediately contains a <linestring binary representation>, then <wkbcurve binary> produces an *ST\_LineString* value specified by the immediately contained <linestring binary representation>.
- ii) Otherwise, <wkbcurve binary> produces an *ST\_CircularString* value specified by the immediately contained <circularstring binary representation>.

r) Case:

- i) If <wkbring binary> immediately contains a <linestring binary representation>, then <wkbring binary> produces an *ST\_LineString* value specified by the immediately contained <linestring binary representation>.
- ii) If <wkbring binary> immediately contains a <circularstring binary representation>, then <wkbring binary> produces an *ST\_CircularString* value specified by the immediately contained <circularstring binary representation>.
- iii) Otherwise, <wkbring binary> produces an *ST\_CompoundCurve* value specified by the immediately contained <compoundcurve binary representation>.

s) Let *XC* be the DOUBLE PRECISION value specified by <wkbx> in <wkbpoint binary> and *YC* be the DOUBLE PRECISION value specified by <wkby> in <wkbpoint binary>. <wkbpoint binary> produces an *ST\_Point* value as the result of the value expression: *NEW ST\_Point(XC, YC)*.

t) <wkbx> is a <double> representing the x coordinate value of an *ST\_Point* value.

u) <wkby> is a <double> representing the y coordinate value of an *ST\_Point* value.

v) <num> is an <uint32> that represent the number of elements in a repeating group.

w) <wkblinestring binary> produces an *ST\_Point* ARRAY value with cardinality of <num> that contains the *ST\_Point* values specified by the immediately contained <wkbpoint binary>s.

- x) The <well-known binary representation> <uint32> values are defined in Table 11 – <well-known binary representation> <uint32> Values.

**Table 11 – <well-known binary representation> <uint32> Values**

<well-known binary representation>	<uint32> Value
<wkbpolygon>	1 (one)
<wkblinestring>	2
<wkbcircularstring>	1000001
<wkbcompoundcurve>	1000002
<wkbpolygon>	3
<wkbcurvepolygon>	1000003
<wkbmultipoint>	4
<wkbmulticurve>	1000004
<wkbmultilinestring>	5
<wkbmultisurface>	1000005
<wkbmultipolygon>	6
<wkbgeometrycollection>	7

- y) <byte order> indicates the binary representation of <uint32> and <double> values that follow <byte order>.
- z) <big endian> is a <byte order> represented by a <byte> with the value 0 (zero). <uint32> is Big Endian (most significant octet first). <double> is Big Endian (sign bit is in the first octet).
- aa) <little endian> is a <byte order> represented by a <byte> with the value 1 (one). <uint32> is Little Endian (most significant octet last). <double> is Little Endian (sign bit is in the last octet).
- ab) <byte> is an 8 bit (1 (one) octet) data type that encodes an unsigned integer in the range [0, 255].
- ac) <uint32> is a 32 bit (4 octets) data type that encodes an unsigned integer in the range [0, 4294967295].
- ad) <double> is a 64 bit (8 octets) double precision data type that encodes a double precision format using the IEC 559:1989.
- ae) <well-known binary representation> provides a portable representation of a geometry value as a contiguous stream of octets in a BINARY LARGE OBJECT value. The serialized *ST\_Geometry* is either represented in Big Endian format or Little Endian format. Conversion between Big Endian format or Little Endian format is a simple operation involving reversing the order of octets within each <uint32> or <double> value in the BINARY LARGE OBJECT.

Blank page

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 6 Point Types

### 6.1 ST\_Point Type and Routines

#### 6.1.1 ST\_Point Type

##### Purpose

The ST\_Point type is a 0-dimensional geometry and represents a single location in two-dimensional coordinate space.

##### Definition

```

CREATE TYPE ST_Point
  UNDER ST_Geometry
  AS (
    ST_PrivateX DOUBLE PRECISION DEFAULT NULL,
    ST_PrivateY DOUBLE PRECISION DEFAULT NULL
  )
  INSTANTIABLE
  NOT FINAL

  CONSTRUCTOR METHOD ST_Point
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_Point
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

  CONSTRUCTOR METHOD ST_Point
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
     ansrid INTEGER)
  RETURNS ST_Point
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

  CONSTRUCTOR METHOD ST_Point
    (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_Point
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

  CONSTRUCTOR METHOD ST_Point
    (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
     ansrid INTEGER)
  RETURNS ST_Point
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

CONSTRUCTOR METHOD ST_Point
  (xcoord DOUBLE PRECISION,
   ycoord DOUBLE PRECISION)
RETURNS ST_Point
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Point
  (xcoord DOUBLE PRECISION,
   ycoord DOUBLE PRECISION,
   ansrid INTEGER)
RETURNS ST_Point
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

METHOD ST_X()
RETURNS DOUBLE PRECISION
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

METHOD ST_X
  (xcoord DOUBLE PRECISION)
RETURNS ST_Point
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
CALLED ON NULL INPUT,

METHOD ST_Y()
RETURNS DOUBLE PRECISION
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

METHOD ST_Y
  (ycoord DOUBLE PRECISION)
RETURNS ST_Point
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
CALLED ON NULL INPUT,

METHOD ST_ExplicitPoint()
RETURNS DOUBLE PRECISION ARRAY[2]
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT

```

STANDARDS190.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

**Definitional Rules**

- 1) *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.
- 2) *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.
- 3) The attribute *ST\_PrivateX* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST\_PrivateX*.
- 4) The attribute *ST\_PrivateY* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST\_PrivateY*.

**Description**

- 1) The *ST\_Point* type provides for public use:
  - a) a method *ST\_Point*(CHARACTER LARGE OBJECT),
  - b) a method *ST\_Point*(CHARACTER LARGE OBJECT, INTEGER),
  - c) a method *ST\_Point*(BINARY LARGE OBJECT),
  - d) a method *ST\_Point*(BINARY LARGE OBJECT, INTEGER),
  - e) a method *ST\_Point*(DOUBLE PRECISION, DOUBLE PRECISION),
  - f) a method *ST\_Point*(DOUBLE PRECISION, DOUBLE PRECISION, INTEGER),
  - g) a method *ST\_X*(),
  - h) a method *ST\_X*(DOUBLE PRECISION),
  - i) a method *ST\_Y*(),
  - j) a method *ST\_Y*(DOUBLE PRECISION),
  - k) a method *ST\_ExplicitPoint*(),
  - l) a function *ST\_PointFromText*(CHARACTER LARGE OBJECT),
  - m) a function *ST\_PointFromText*(CHARACTER LARGE OBJECT, INTEGER),
  - n) a function *ST\_PointFromWKB*(BINARY LARGE OBJECT),
  - o) a function *ST\_PointFromWKB*(BINARY LARGE OBJECT, INTEGER),
  - p) a function *ST\_PointFromGML*(CHARACTER LARGE OBJECT),
  - q) a function *ST\_PointFromGML*(CHARACTER LARGE OBJECT, INTEGER),
- 2) The *ST\_PrivateX* attribute contains the x coordinate value.
- 3) The *ST\_PrivateY* attribute contains the y coordinate value.
- 4) An *ST\_Point* value is a 0-dimensional geometry that represents a single location.
- 5) The dimension of an *ST\_Point* value is 0 (zero).
- 6) The coordinate dimension of an *ST\_Point* value is 2.
- 7) The boundary of an *ST\_Point* value is the empty set.
- 8) An *ST\_Point* value is simple.
- 9) An *ST\_Point* value returned by the constructor function corresponds to the empty set.
- 10) An *ST\_Point* value is not well formed if either:
  - a) the *ST\_PrivateX* attribute is the null value and the *ST\_PrivateY* attribute is not the null value, or
  - b) the *ST\_PrivateY* attribute is the null value and the *ST\_PrivateX* attribute is not the null value.

## 6.1.2 ST\_Point Methods

### Purpose

Return an *ST\_Point* value constructed from either the well-known text representation or the well-known binary representation of an *ST\_Point* value, or the specified coordinate values.

### Definition

```

CREATE CONSTRUCTOR METHOD ST_Point
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_Point
  FOR ST_Point
  RETURN ST_PointFromText(awkt)

CREATE CONSTRUCTOR METHOD ST_Point
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_Point
  FOR ST_Point
  RETURN ST_PointFromText(awkt, ansrid)

CREATE CONSTRUCTOR METHOD ST_Point
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_Point
  FOR ST_Point
  RETURN ST_PointFromWKB(awkb)

CREATE CONSTRUCTOR METHOD ST_Point
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_Point
  FOR ST_Point
  RETURN ST_PointFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_Point
  (xcoord DOUBLE PRECISION,
   ycoord DOUBLE PRECISION)
  RETURNS ST_Point
  FOR ST_Point
  RETURN NEW ST_Point(xcoord, ycoord, 0)

CREATE CONSTRUCTOR METHOD ST_Point
  (xcoord DOUBLE PRECISION,
   ycoord DOUBLE PRECISION,
   ansrid INTEGER)
  RETURNS ST_Point
  FOR ST_Point
  RETURN SELF.
  -- Return an ST_Point value with
  ST_PrivateDimension(0).           -- dimension = 0,
  ST_PrivateCoordinateDimension(2). -- coordinate dimension = 2,
  ST_SRID(ansrid).                 -- SRID = ansrid,
  ST_X(xcoord).                    -- x coordinate = xcoord,
  ST_Y(ycoord).                    -- y coordinate = ycoord

```

### Definitional Rules

- 1) *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.
- 2) *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.

**Description**

- 1) The method *ST\_Point*(*CHARACTER LARGE OBJECT*) takes the following input parameter:
  - a) a *CHARACTER LARGE OBJECT* value *awkt*.
- 2) For the null-call type preserving SQL-invoked constructor method *ST\_Point*(*CHARACTER LARGE OBJECT*):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_Point* value. If *awkt* is not producible in the BNF for <point text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_Point* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The method *ST\_Point*(*CHARACTER LARGE OBJECT*, *INTEGER*) takes the following input parameters:
  - a) a *CHARACTER LARGE OBJECT* value *awkt*,
  - b) an *INTEGER* value *ansrid*.
- 4) For the null-call type preserving SQL-invoked constructor method *ST\_Point*(*CHARACTER LARGE OBJECT*, *INTEGER*):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_Point* value. If *awkt* is not producible in the BNF for <point text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_Point* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.
- 5) The method *ST\_Point*(*BINARY LARGE OBJECT*) takes the following input parameter:
  - a) a *BINARY LARGE OBJECT* value *awkb*.
- 6) For the null-call type preserving SQL-invoked constructor method *ST\_Point*(*BINARY LARGE OBJECT*):
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_Point* value. If *awkb* is not producible in the BNF for <point binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_Point* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 7) The method *ST\_Point*(*BINARY LARGE OBJECT*, *INTEGER*) takes the following input parameters:
  - a) a *BINARY LARGE OBJECT* value *awkb*,
  - b) an *INTEGER* value *ansrid*.
- 8) For the null-call type preserving SQL-invoked constructor method *ST\_Point*(*BINARY LARGE OBJECT*, *INTEGER*):
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_Point* value. If *awkb* is not producible in the BNF for <point binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_Point* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.
- 9) The method *ST\_Point*(*DOUBLE PRECISION*, *DOUBLE PRECISION*) takes the following input parameters:
  - a) a *DOUBLE PRECISION* value *xcoord*,

- b) a DOUBLE PRECISION value *ycoord*.
- 10) The null-call type preserving SQL-invoked constructor method *ST\_Point(DOUBLE PRECISION, DOUBLE PRECISION)* returns the value expression: *NEW ST\_Point(xcoord, ycoord, 0)*.
- 11) The method *ST\_Point(DOUBLE PRECISION, DOUBLE PRECISION, INTEGER)* takes the following input parameters:
- a) a DOUBLE PRECISION value *xcoord*,
  - b) a DOUBLE PRECISION value *ycoord*,
  - c) an INTEGER value *ansrid*.
- 12) The null-call type preserving SQL-invoked constructor method *ST\_Point(DOUBLE PRECISION, DOUBLE PRECISION, INTEGER)* returns an *ST\_Point* value with:
- a) The dimension set to 0 (zero).
  - b) The coordinate dimension value set to 2.
  - c) The spatial reference system identifier set to *ansrid*.
  - d) Using the method *ST\_X(DOUBLE PRECISION)*, the x coordinate value is set to *xcoord*.
  - e) Using the method *ST\_Y(DOUBLE PRECISION)*, the y coordinate value is set to *ycoord*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 6.1.3 ST\_X Methods

#### Purpose

Observe and mutate the x coordinate value of an ST\_Point value.

#### Definition

```

CREATE METHOD ST_X()
  RETURNS DOUBLE PRECISION
  FOR ST_Point
  RETURN SELF.ST_PrivateX

CREATE METHOD ST_X
  (xcoord DOUBLE PRECISION)
  RETURNS ST_Point
  FOR ST_Point
  BEGIN
    IF xcoord IS NULL THEN
      SIGNAL SQLSTATE '2FF03'
        SET MESSAGE_TEXT = 'null argument';
    ELSE
      RETURN
        CASE
          WHEN SELF IS NULL THEN
            NULL
          ELSE
            SELF.ST_PrivateX(xcoord)
        END;
    END IF;
  END

```

#### Description

- 1) The method *ST\_X()* has no input parameters.
- 2) The null-call method *ST\_X()* returns the value of the *ST\_PrivateX* attribute.
- 3) The method *ST\_X(DOUBLE PRECISION)* takes the following input parameters:
  - a) a DOUBLE PRECISION value *xcoord*.
- 4) For the type preserving method *ST\_X(DOUBLE PRECISION)*:
 

Case:

  - a) If *xcoord* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
  - b) If SELF is the null value, then return the null value.
  - c) Otherwise, return the value expression: *SELF.ST\_PrivateX(xcoord)*.

### 6.1.4 ST\_Y Methods

#### Purpose

Observe and mutate the y coordinate value of an ST\_Point value.

#### Definition

```

CREATE METHOD ST_Y()
  RETURNS DOUBLE PRECISION
  FOR ST_Point
  RETURN SELF.ST_PrivateY

CREATE METHOD ST_Y
  (ycoord DOUBLE PRECISION)
  RETURNS ST_Point
  FOR ST_Point
  BEGIN
    IF ycoord IS NULL THEN
      SIGNAL SQLSTATE '2FF03'
        SET MESSAGE_TEXT = 'null argument';
    ELSE
      RETURN
        CASE
          WHEN SELF IS NULL THEN
            NULL
          ELSE
            SELF.ST_PrivateY(ycoord)
        END;
    END IF;
  END

```

#### Description

- 1) The method *ST\_Y()* has no input parameters.
- 2) The null-call method *ST\_Y()* returns the value of the *ST\_PrivateY* attribute.
- 3) The method *ST\_Y(DOUBLE PRECISION)* takes the following input parameters:
  - a) a DOUBLE PRECISION value *ycoord*.
- 4) For the type preserving method *ST\_Y(DOUBLE PRECISION)*:
 

Case:

  - a) If *ycoord* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
  - b) If SELF is the null value, then return the null value.
  - c) Otherwise, return the value expression: *SELF.ST\_PrivateY(ycoord)*.

### 6.1.5 ST\_ExplicitPoint Method

#### Purpose

Return the coordinate values as a DOUBLE PRECISION ARRAY value.

#### Definition

```
CREATE METHOD ST_ExplicitPoint()
  RETURNS DOUBLE PRECISION ARRAY[2]
  FOR ST_Point
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      ARRAY[SELF.ST_X(), SELF.ST_Y()]
  END
```

#### Description

- 1) The method *ST\_ExplicitPoint()* has no input parameters.
- 2) For the null-call method *ST\_ExplicitPoint()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return an array of type DOUBLE PRECISION with the first element representing the x coordinate value and the second element representing the y coordinate value.

### 6.1.6 ST\_PointFromText Functions

#### Purpose

Return a specified ST\_Point value.

#### Definition

```
CREATE FUNCTION ST_PointFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_Point
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt) AS ST_Point)

CREATE FUNCTION ST_PointFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_Point
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt, ansrid) AS ST_Point)
```

#### 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.

#### Description

- 1) The function *ST\_PointFromText*(CHARACTER LARGE OBJECT) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call function *ST\_PointFromText*(CHARACTER LARGE OBJECT):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_Point* value. If *awkt* is not producible in the BNF for <point text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_Point* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_PointFromText*(CHARACTER LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_PointFromText*(CHARACTER LARGE OBJECT, INTEGER):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_Point* value. If *awkt* is not producible in the BNF for <point text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_Point* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.

### 6.1.7 ST\_PointFromWKB Functions

#### Purpose

Return a specified ST\_Point value.

#### Definition

```
CREATE FUNCTION ST_PointFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_Point
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_Point)

CREATE FUNCTION ST_PointFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_Point
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_Point)
```

#### Definitional Rules

- 1) *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.

#### Description

- 1) The function *ST\_PointFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 2) For the null-call function *ST\_PointFromWKB(BINARY LARGE OBJECT)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_Point* value. If *awkb* is not producible in the BNF for <point binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_Point* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_PointFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_PointFromWKB(BINARY LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_Point* value. If *awkb* is not producible in the BNF for <point binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_Point* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.

### 6.1.8 ST\_PointFromGML Functions

#### Purpose

Return a specified ST\_Point value.

#### Definition

```
CREATE FUNCTION ST_PointFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
  RETURNS ST_Point
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromGML(agml) AS ST_Point)

CREATE FUNCTION ST_PointFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
   ansrid INTEGER)
  RETURNS ST_Point
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromGML(agml, ansrid) AS ST_Point)
```

#### Definitional Rules

- 1) *ST\_MaxGeometryAsGML* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the GML representation of an *ST\_Geometry* value.

#### Description

- 1) The function *ST\_PointFromGML*(CHARACTER LARGE OBJECT) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*.
- 2) For the null-call function *ST\_PointFromGML*(CHARACTER LARGE OBJECT):
  - a) If the parameter *agml* does not contain a Point XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_Point* value represented by *agml* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_PointFromGML*(CHARACTER LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_PointFromGML*(CHARACTER LARGE OBJECT, INTEGER):
  - a) If the parameter *agml* does not contain a Point XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_Point* value represented by *agml* with the spatial reference system identifier set to *ansrid*.

## 7 Curve Types

### 7.1 ST\_Curve Type and Routines

#### 7.1.1 ST\_Curve Type

##### Purpose

The ST\_Curve type is a supertype for 1-dimensional geometry types and represents a continuous locus of points from the start point to the end point. Subtypes of ST\_Curve specify the form of interpolation between points.

##### Definition

```

CREATE TYPE ST_Curve
  UNDER ST_Geometry
  NOT INSTANTIABLE
  NOT FINAL

METHOD ST_Length()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

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

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

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

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

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

```

```

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

```

### Definitional Rules

- 1) *ST\_MaxUnitNameLength* is the implementation-defined maximum length for the character representation of a unit indication.

### Description

- 1) The *ST\_Curve* type provides for public use:
  - a) a method *ST\_Length()*,
  - b) a method *ST\_Length(CHARACTER VARYING)*,
  - c) a method *ST\_StartPoint()*,
  - d) a method *ST\_EndPoint()*,
  - e) a method *ST\_IsClosed()*,
  - f) a method *ST\_IsRing()*,
  - g) a method *ST\_CurveToLine()*.
- 2) An *ST\_Curve* value is a 1-dimensional geometry that is defined as a sequence of *ST\_Point* values.
- 3) Subtypes of the *ST\_Curve* type specifies the form of interpolation between *ST\_Point* values.
- 4) An *ST\_Curve* value is defined to be topologically closed.
- 5) An *ST\_Curve* value is the homomorphic image of a real, closed interval:
 
$$\text{Domain} = [a, b] = \{ x \in \mathbb{R} \mid a \leq x \leq b \}$$
 under a mapping  $f: [a, b] \rightarrow \mathbb{R}^2$ .
- 6) An *ST\_Curve* value is not simple if any interior point has the same location as another interior point or a point on the boundary:
 
$$\forall c \in \text{ST\_Curve}, [a, b] = c.\text{Domain}$$

$$c.\text{ST\_IsSimple}() \Leftrightarrow (\forall x_1, x_2 \in (a, b) \ x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2)) \wedge (\forall x_1, x_2 \in [a, b] \ x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2))$$
- 7) The dimension of an *ST\_Curve* value is 1 (one).
- 8) The start point of an *ST\_Curve* value is returned by the method *ST\_StartPoint()*.
- 9) The end point of an *ST\_Curve* value is returned by the method *ST\_EndPoint()*.
- 10) If the start point of an *ST\_Curve* value is equal to the end point of the *ST\_Curve* value, then the *ST\_Curve* value is closed.
- 11) The boundary of a closed *ST\_Curve* value is the empty set.
- 12) The boundary of an *ST\_Curve* value that is not closed consists of the start point and end point of the *ST\_Curve* value.
- 13) If an *ST\_Curve* value is simple and closed, then it is called a *ring*.

## 7.1.2 ST\_Length Methods

### Purpose

Return the length measurement of an ST\_Curve value.

### Definition

```
CREATE METHOD ST_Length()
  RETURNS DOUBLE PRECISION
  FOR ST_Curve
  BEGIN
    --
    -- See Description
    --
  END

CREATE METHOD ST_Length
  (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
  RETURNS DOUBLE PRECISION
  FOR ST_Curve
  BEGIN
    --
    -- See Description
    --
  END
```

### Definitional Rules

- 1) *ST\_MaxUnitNameLength* is the implementation-defined maximum length for the character representation of a unit indication.

### Description

- 1) The method *ST\_Length()* has no input parameters.
- 2) For the null-call method *ST\_Length()*:
  - Case:
    - a) If SELF is an empty set, then return the null value.
    - b) Otherwise, return the implementation-defined length of SELF as measured in its spatial reference system.
- 3) Case:
  - a) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST\_Length()* is in the linear unit of measure identified by <linear unit>.
  - b) Otherwise, the value returned by *ST\_Length()* is in an implementation-defined unit of measure.
- 4) The method *ST\_Length(CHARACTER VARYING)* takes the following input parameter:
  - a) a CHARACTER VARYING value *aunit*.
- 5) For the null-call method *ST\_Length(CHARACTER VARYING)*:
  - Case:
    - a) If SELF is an empty set, then return the null value.
    - b) Otherwise, return the length of SELF as measured in its spatial reference system.
- 6) The value returned by *ST\_Length(CHARACTER VARYING)* is in the units indicated by *aunit*.
- 7) The values for *aunit* shall be a supported <unit name>.

- 8) The value for *unit* is a supported <unit name> if and only if the value of *unit* is equal to the value of the UNIT\_NAME column of one of the rows where the value of the UNIT\_TYPE column is equal to 'LINEAR' in the ST\_UNITS\_OF\_MEASURE view.
- 9) If the unit specified by *unit* is not supported by the implementation to compute the length of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 7.1.3 ST\_StartPoint Method

#### Purpose

Return an ST\_Point value that is the start point of an ST\_Curve value.

#### Definition

```
CREATE METHOD ST_StartPoint()  
  RETURNS ST_Point  
  FOR ST_Curve  
  BEGIN  
    --  
    -- See Description  
    --  
  END
```

#### Description

- 1) The method *ST\_StartPoint()* has no input parameters.
- 2) For the null-call method *ST\_StartPoint()*:  
Case:
  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return an *ST\_Point* value that is the start point of SELF.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

#### 7.1.4 ST\_EndPoint Method

##### Purpose

Return an ST\_Point value that is the end point of an ST\_Curve value.

##### Definition

```
CREATE METHOD ST_EndPoint()  
  RETURNS ST_Point  
  FOR ST_Curve  
  BEGIN  
    --  
    -- See Description  
    --  
  END
```

##### Description

- 1) The method *ST\_EndPoint()* has no input parameters.
- 2) For the null-call method *ST\_EndPoint()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the *ST\_Point* value that is the end point of SELF.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 7.1.5 ST\_IsClosed Method

#### Purpose

Test if an ST\_Curve value is closed.

#### Definition

```
CREATE METHOD ST_IsClosed()
  RETURNS INTEGER
  FOR ST_Curve
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      0
    ELSE
      SELF.ST_Boundary().ST_IsEmpty()
  END
```

#### Description

- 1) The method *ST\_IsClosed()* has no input parameters.
- 2) For the null-call method *ST\_IsClosed()*:

Case:

- a) If SELF is an empty set, then return 0 (zero).
- b) If the boundary of the *ST\_MultiCurve* value is the empty set, then 1 (one).
- c) Otherwise, 0 (zero).

### 7.1.6 ST\_IsRing Method

#### Purpose

Test if an ST\_Curve value is a ring.

#### Definition

```
CREATE METHOD ST_IsRing()  
  RETURNS INTEGER  
  FOR ST_Curve  
  RETURN  
  CASE  
    WHEN SELF.ST_IsEmpty() = 1 THEN  
      0  
    WHEN (SELF.ST_IsSimple() = 1 AND SELF.ST_IsClosed() = 1) THEN  
      1  
    ELSE  
      0  
  END
```

#### Description

- 1) The method *ST\_IsRing()* has no input parameters.
- 2) For the null-call method *ST\_IsRing()*:

Case:

- a) If SELF is an empty set, then return 0 (zero).
- b) If SELF is simple and SELF is closed, then return 1 (one).
- c) Otherwise 0 (zero).

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 7.1.7 ST\_CurveToLine Method

#### Purpose

Return the ST\_LineString value approximation of an ST\_Curve value.

#### Definition

```
CREATE METHOD ST_CurveToLine()  
  RETURNS ST_LineString  
  FOR ST_Curve  
  BEGIN  
    --  
    -- See Description  
    --  
  END
```

#### Description

- 1) The method *ST\_CurveToLine()* has no input parameters.
- 2) For the null-call method *ST\_CurveToLine()*:

Case:

- a) If SELF is an empty set, then return an empty set of type *ST\_LineString*.
- b) Otherwise, return the implementation-defined *ST\_LineString* value approximation of the *ST\_Curve* value.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 7.2 ST\_LineString Type and Routines

### 7.2.1 ST\_LineString Type

#### Purpose

The ST\_LineString type is a subtype of the ST\_Curve type and represents a continuous locus of points from the start point to the end point with a linear interpolation between points.

#### Definition

```

CREATE TYPE ST_LineString
  UNDER ST_Curve
  AS (
    ST_PrivatePoints ST_Point
    ARRAY[ST_MaxGeometryArrayElements] DEFAULT ARRAY[]
  )
  INSTANTIABLE
  NOT FINAL

CONSTRUCTOR METHOD ST_LineString
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_LineString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_LineString
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_LineString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_LineString
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_LineString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_LineString
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_LineString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

CONSTRUCTOR METHOD ST_LineString
  (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_LineString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_LineString
  (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
  RETURNS ST_LineString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_Points()
  RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements]
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_Points
  (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_LineString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

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

METHOD ST_PointN
  (aposition INTEGER)
  RETURNS ST_Point
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_StartPoint()
  RETURNS ST_Point,

OVERRIDING METHOD ST_EndPoint()
  RETURNS ST_Point

```

**Definitional Rules**

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.

- 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) *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.
- 4) The attribute *ST\_PrivatePoints* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST\_PrivatePoints*.

#### Description

- 1) The *ST\_LineString* type provides for public use:
  - a) a method *ST\_LineString*(CHARACTER LARGE OBJECT),
  - b) a method *ST\_LineString*(CHARACTER LARGE OBJECT, INTEGER),
  - c) a method *ST\_LineString*(BINARY LARGE OBJECT),
  - d) a method *ST\_LineString*(BINARY LARGE OBJECT, INTEGER),
  - e) a method *ST\_LineString*(*ST\_Point* ARRAY),
  - f) a method *ST\_LineString*(*ST\_Point* ARRAY, INTEGER),
  - g) a method *ST\_Points*(),
  - h) a method *ST\_Points*(*ST\_Point* ARRAY),
  - i) a method *ST\_NumPoints*(),
  - j) a method *ST\_PointN*(INTEGER),
  - k) an overriding method *ST\_StartPoint*(),
  - l) an overriding method *ST\_EndPoint*(),
  - m) a function *ST\_LineFromText*(CHARACTER LARGE OBJECT),
  - n) a function *ST\_LineFromText*(CHARACTER LARGE OBJECT, INTEGER),
  - o) a function *ST\_LineFromWKB*(BINARY LARGE OBJECT),
  - p) a function *ST\_LineFromWKB*(BINARY LARGE OBJECT, INTEGER),
  - q) a function *ST\_LineFromGML*(CHARACTER LARGE OBJECT),
  - r) a function *ST\_LineFromGML*(CHARACTER LARGE OBJECT, INTEGER),
- 2) The *ST\_PrivatePoints* attribute contains the collection of *ST\_Point* values.
- 3) The *ST\_PrivatePoints* attribute shall not be the null value. The elements in the *ST\_PrivatePoints* attribute shall not be the null value.
- 4) If the cardinality of the *ST\_PrivatePoints* attribute is greater than or equal to two, then the *ST\_LineString* value is well formed.
- 5) All the *ST\_Point* values in the *ST\_PrivatePoints* attribute shall be in the same spatial reference system as the *ST\_LineString* value.
- 6) The coordinate dimension of an *ST\_LineString* value is 2.
- 7) The type *ST\_LineString* is a subtype of *ST\_Curve* with linear interpolation between points. Each consecutive pair of points defines a *line segment*.
- 8) If the cardinality of the *ST\_PrivatePoints* attribute is two, then the *ST\_LineString* value is called a *line*.
- 9) If an *ST\_LineString* value is simple and closed, then it is called a *linear ring*.
- 10) An *ST\_LineString* value returned by the constructor function corresponds to the empty set.
- 11) An *ST\_LineString* value with the cardinality of the *ST\_PrivatePoints* attribute equal to 0 (zero) corresponds to the empty set.

## 7.2.2 ST\_LineString Methods

### Purpose

Return an ST\_LineString value constructed from either the well-known text representation or the well-known binary representation of an ST\_LineString value, or the specified ST\_Point values.

### Definition

```

CREATE CONSTRUCTOR METHOD ST_LineString
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_LineString
  FOR ST_LineString
  RETURN ST_LineFromText(awkt)

CREATE CONSTRUCTOR METHOD ST_LineString
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_LineString
  FOR ST_LineString
  RETURN ST_LineFromText(awkt, ansrid)

CREATE CONSTRUCTOR METHOD ST_LineString
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_LineString
  FOR ST_LineString
  RETURN ST_LineFromWKB(awkb)

CREATE CONSTRUCTOR METHOD ST_LineString
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_LineString
  FOR ST_LineString
  RETURN ST_LineFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_LineString
  (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_LineString
  FOR ST_LineString
  RETURN SELF.ST_SRID(0).ST_Points(apointarray)

CREATE CONSTRUCTOR METHOD ST_LineString
  (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
  RETURNS ST_LineString
  FOR ST_LineString
  RETURN SELF.ST_SRID(ansrid).ST_Points(apointarray)

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

### Description

- 1) The method *ST\_LineString(CHARACTER LARGE OBJECT)* takes the following input parameter:
  - a) a CHARACTER LARGE OBJECT value *awkt*.

- 2) For the null-call type preserving SQL-invoked constructor method *ST\_LineString*(*CHARACTER LARGE OBJECT*):
- The parameter *awkt* is the well-known text representation of an *ST\_LineString* value. If *awkt* is not producible in the BNF for <linestring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - Return an *ST\_LineString* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The method *ST\_LineString*(*CHARACTER LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *CHARACTER LARGE OBJECT* value *awkt*,
  - an *INTEGER* value *ansrid*.
- 4) For the null-call type preserving SQL-invoked constructor method *ST\_LineString*(*CHARACTER LARGE OBJECT*, *INTEGER*):
- The parameter *awkt* is the well-known text representation of an *ST\_LineString* value. If *awkt* is not producible in the BNF for <linestring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - Return an *ST\_LineString* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.
- 5) The method *ST\_LineString*(*BINARY LARGE OBJECT*) takes the following input parameter:
- a *BINARY LARGE OBJECT* value *awkb*.
- 6) For the null-call type preserving SQL-invoked constructor method *ST\_LineString*(*BINARY LARGE OBJECT*):
- The parameter *awkb* is the well-known binary representation of an *ST\_LineString* value. If *awkb* is not producible in the BNF for <linestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_LineString* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 7) The method *ST\_LineString*(*BINARY LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *BINARY LARGE OBJECT* value *awkb*,
  - an *INTEGER* value *ansrid*.
- 8) For the null-call type preserving SQL-invoked constructor method *ST\_LineString*(*BINARY LARGE OBJECT*, *INTEGER*):
- The parameter *awkb* is the well-known binary representation of an *ST\_LineString* value. If *awkb* is not producible in the BNF for <linestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_LineString* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.
- 9) The method *ST\_LineString*(*ST\_Point ARRAY*) takes the following input parameters:
- an *ST\_Point ARRAY* value *apointarray*.
- 10) The null-call type preserving SQL-invoked constructor method *ST\_LineString*(*ST\_Point ARRAY*) returns an *ST\_LineString* value with:
- The spatial reference system identifier set to 0 (zero).

- b) Using the method *ST\_Points(ST\_Point ARRAY)*, the *ST\_PrivatePoints* attribute set to *apointarray*, the *ST\_PrivateDimension* attribute set to 1 (one), and the *ST\_PrivateCoordinateDimension* attribute set to 2.
- 11) The method *ST\_LineString(ST\_Point ARRAY, INTEGER)* takes the following input parameters:
- a) an *ST\_Point ARRAY* value *apointarray*,
  - b) an *INTEGER* value *ansrid*.
- 12) The null-call type preserving SQL-invoked constructor method *ST\_LineString(ST\_Point ARRAY, INTEGER)* returns an *ST\_LineString* value with:
- a) The spatial reference system identifier set to *ansrid*.
  - b) Using the method *ST\_Points(ST\_Point ARRAY)*, the *ST\_PrivatePoints* attribute set to *apointarray*, the *ST\_PrivateDimension* attribute set to 1 (one), and the *ST\_PrivateCoordinateDimension* attribute set to 2.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 7.2.3 ST\_Points Methods

#### Purpose

Observe and mutate the *ST\_PrivatePoints* attribute of an *ST\_LineString* value.

#### Definition

```

CREATE METHOD ST_Points()
  RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements]
  FOR ST_LineString
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      SELF.ST_PrivatePoints
  END

CREATE METHOD ST_Points
  (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_LineString
  FOR ST_LineString
  BEGIN
    -- If apointarray is the null value, contains null elements, or
    -- contains consecutive duplicate points, then raise an exception.
    CALL ST_CheckConsecDups(apointarray);
    -- If SELF is the null value, then return the null value.
    IF SELF IS NULL THEN
      RETURN CAST (NULL AS ST_LineString);
    END IF;
    -- Check that there are no mixed spatial reference
    -- systems between SELF and apointarray.
    IF SELF.ST_SRID() <> ST_CheckSRID(apointarray) THEN
      SIGNAL SQLSTATE '2FF10'
      SET MESSAGE_TEXT = 'mixed spatial reference systems';
    END IF;
    RETURN
      SELF.ST_PrivateDimension(1).
      ST_PrivateCoordinateDimension(2).
      ST_PrivatePoints(apointarray);
  END

```

#### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.

#### Description

- 1) The method *ST\_Points()* has no input parameters.
- 2) For the null-call method *ST\_Points()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the *ST\_PrivatePoints* attribute of SELF.
- 3) The method *ST\_Points(ST\_Point ARRAY)* takes the following input parameters:
  - a) an *ST\_Point* ARRAY value *apointarray*.
- 4) For the type preserving method *ST\_Points(ST\_Point ARRAY)*:
  - a) Call the procedure *ST\_CheckConsecDups(ST\_Geometry ARRAY)* to check if *apointarray* is the null value, contains null elements, or contains consecutive duplicate points.

## b) Case:

- i) If SELF is the null value, then return the null value.
- ii) If the spatial reference system of SELF is not equal to *ST\_CheckSRID(apointarray)*, then an exception condition is raised: *SQL/MM Spatial exception – mixed spatial reference systems*.
- iii) Otherwise, return an *ST\_LineString* value with:
  - 1) The dimension set to 1 (one).
  - 2) The coordinate dimension set to 2.
  - 3) The *ST\_PrivatePoints* attribute set to *apointarray*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

#### 7.2.4 ST\_NumPoints Method

##### Purpose

Return the cardinality of the ST\_PrivatePoints attribute of an ST\_LineString value.

##### Definition

```
CREATE METHOD ST_NumPoints()  
  RETURNS INTEGER  
  FOR ST_LineString  
  RETURN  
  CASE  
    WHEN SELF.ST_IsEmpty() = 1 THEN  
      NULL  
    ELSE  
      CARDINALITY(SELF.ST_PrivatePoints)  
  END
```

##### Description

- 1) The method *ST\_NumPoints()* has no input parameters.
- 2) For the null-call method *ST\_NumPoints()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the cardinality of the *ST\_PrivatePoints* attribute.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 7.2.5 ST\_PointN Method

#### Purpose

Return the specified element in the ST\_PrivatePoints attribute of an ST\_LineString value.

#### Definition

```
CREATE METHOD ST_PointN
(aosition INTEGER)
RETURNS ST_Point
FOR ST_LineString
BEGIN
  IF SELF.ST_IsEmpty() = 1 THEN
    RETURN CAST (NULL AS ST_Point);
  END IF;
  IF aosition < 1 OR
    aosition > SELF.ST_NumPoints() THEN
    BEGIN
      SIGNAL SQLSTATE '01F01'
      SET MESSAGE_TEXT = 'invalid position';
      RETURN CAST (NULL AS ST_Point);
    END;
  END IF;
  RETURN SELF.ST_PrivatePoints[aosition];
END
```

#### Description

1) The method *ST\_PointN(INTEGER)* takes the following input parameters:

a) an INTEGER value *aosition*.

2) For the null-call method *ST\_PointN(INTEGER)*:

Case:

a) If SELF is an empty set, then return the null value.

b) If *aosition* is less than 1 (one) or greater than the cardinality of the *ST\_PrivatePoints* attribute, then:

i) A completion condition is raised: *SQL/MM Spatial warning – invalid position*.

ii) Return the null value.

c) Otherwise, return an *ST\_Point* value at element *aosition* in the *ST\_PrivatePoints* attribute of SELF.

### 7.2.6 ST\_StartPoint Method

#### Purpose

Return the start point of an ST\_LineString value.

#### Definition

```
CREATE METHOD ST_StartPoint()  
  RETURNS ST_Point  
  FOR ST_LineString  
  RETURN  
  CASE  
    WHEN SELF.ST_IsEmpty() = 1 THEN  
      NULL  
    ELSE  
      SELF.ST_Points()[1]  
  END
```

#### Description

- 1) The method *ST\_StartPoint()* has no input parameters.
- 2) For the null-call method *ST\_StartPoint()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the result of the value expression: SELF.ST\_Points()[1].

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

**7.2.7 ST\_EndPoint Method****Purpose**

Return the end point of an ST\_LineString value.

**Definition**

```
CREATE METHOD ST_EndPoint()
  RETURNS ST_Point
  FOR ST_LineString
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      SELF.ST_Points()[SELF.ST_NumPoints()]
  END
```

**Description**

- 1) The method *ST\_EndPoint()* has no input parameters.
- 2) For the null-call method *ST\_EndPoint()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the result of the value expression: *SELF.ST\_Points()[SELF.ST\_NumPoints()]*.

## 7.2.8 ST\_LineFromText Functions

### Purpose

Return a specified ST\_LineString value.

### Definition

```
CREATE FUNCTION ST_LineFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_LineString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt) AS ST_LineString)

CREATE FUNCTION ST_LineFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_LineString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt, ansrid) AS ST_LineString)
```

### 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.

### Description

- 1) The function *ST\_LineFromText*(CHARACTER LARGE OBJECT) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call function *ST\_LineFromText*(CHARACTER LARGE OBJECT):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_LineString* value. If *awkt* is not producible in the BNF for <linestring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_LineString* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_LineFromText*(CHARACTER LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_LineFromText*(CHARACTER LARGE OBJECT, INTEGER):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_LineString* value. If *awkt* is not producible in the BNF for <linestring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_LineString* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.

## 7.2.9 ST\_LineFromWKB Functions

### Purpose

Return a specified ST\_LineString value.

### Definition

```
CREATE FUNCTION ST_LineFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_LineString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_LineString)

CREATE FUNCTION ST_LineFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_LineString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_LineString)
```

### Definitional Rules

- 1) *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.

### Description

- 1) The function *ST\_LineFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 2) For the null-call function *ST\_LineFromWKB(BINARY LARGE OBJECT)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_LineString* value. If *awkb* is not producible in the BNF for <linestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_LineString* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_LineFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_LineFromWKB(BINARY LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_LineString* value. If *awkb* is not producible in the BNF for <linestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_LineString* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.

## 7.2.10 ST\_LineFromGML Functions

### Purpose

Return a specified ST\_LineString value.

### Definition

```
CREATE FUNCTION ST_LineFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
  RETURNS ST_LineString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromGML(agml) AS ST_LineString)

CREATE FUNCTION ST_LineFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
   ansrid INTEGER)
  RETURNS ST_LineString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromGML(agml, ansrid) AS ST_LineString)
```

### Definitional Rules

- 1) *ST\_MaxGeometryAsGML* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the GML representation of an *ST\_Geometry* value.

### Description

- 1) The function *ST\_LineFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*.
- 2) For the null-call function *ST\_LineFromGML(CHARACTER LARGE OBJECT)*:
  - a) If the parameter *agml* does not contain a LineString XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_LineString* value represented by *agml* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_LineFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_LineFromGML(CHARACTER LARGE OBJECT, INTEGER)*:
  - a) If the parameter *agml* does not contain a LineString XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_LineString* value represented by *agml* with the spatial reference system identifier set to *ansrid*.

## 7.3 ST\_CircularString Type and Routines

### 7.3.1 ST\_CircularString Type

#### Purpose

The ST\_CircularString type is a subtype of the ST\_Curve type and represents a continuous locus of points from the start point to the end point with a circular interpolation between points.

#### Definition

```

CREATE TYPE ST_CircularString
  UNDER ST_Curve
  AS (
    ST_PrivatePoints ST_Point
    ARRAY[ST_MaxGeometryArrayElements] DEFAULT ARRAY[]
  )
  INSTANTIABLE
  NOT FINAL

  CONSTRUCTOR METHOD ST_CircularString
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_CircularString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

  CONSTRUCTOR METHOD ST_CircularString
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
     ansrid INTEGER)
  RETURNS ST_CircularString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

  CONSTRUCTOR METHOD ST_CircularString
    (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_CircularString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

  CONSTRUCTOR METHOD ST_CircularString
    (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
     ansrid INTEGER)
  RETURNS ST_CircularString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

CONSTRUCTOR METHOD ST_CircularString
  (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_CircularString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CircularString
  (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
  RETURNS ST_CircularString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_Points()
  RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements]
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_Points
  (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_CircularString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

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

METHOD ST_PointN
  (aposition INTEGER)
  RETURNS ST_Point
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_MidPointRep()
  RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements]
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_StartPoint()
  RETURNS ST_Point,

```

OVERRIDING METHOD `ST_EndPoint()`  
 RETURNS `ST_Point`

### Definitional Rules

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

### Description

- 1) The `ST_CircularString` type provides for public use:
  - a) a method `ST_CircularString(CHARACTER LARGE OBJECT)`,
  - b) a method `ST_CircularString(CHARACTER LARGE OBJECT, INTEGER)`,
  - c) a method `ST_CircularString(BINARY LARGE OBJECT)`,
  - d) a method `ST_CircularString(BINARY LARGE OBJECT, INTEGER)`,
  - e) a method `ST_CircularString(ST_Point ARRAY)`,
  - f) a method `ST_CircularString(ST_Point ARRAY, INTEGER)`,
  - g) a method `ST_Points()`,
  - h) a method `ST_Points(ST_Point ARRAY)`,
  - i) a method `ST_NumPoints()`,
  - j) a method `ST_PointN(INTEGER)`,
  - k) a method `ST_MidPointRep()`,
  - l) an overriding method `ST_StartPoint()`,
  - m) an overriding method `ST_EndPoint()`,
  - n) a function `ST_CircularFromTxt(CHARACTER LARGE OBJECT)`,
  - o) a function `ST_CircularFromTxt(CHARACTER LARGE OBJECT, INTEGER)`,
  - p) a function `ST_CircularFromWKB(BINARY LARGE OBJECT)`,
  - q) a function `ST_CircularFromWKB(BINARY LARGE OBJECT, INTEGER)`.
- 2) The `ST_PrivatePoints` attribute contains the collection of `ST_Point` values.
- 3) The `ST_PrivatePoints` attribute shall not be the null value. The elements in the `ST_PrivatePoints` attribute shall not be the null value.
- 4) All the `ST_Point` values in the `ST_PrivatePoints` attribute shall be in the same spatial reference system as the `ST_CircularString` value.
- 5) The coordinate dimension of an `ST_CircularString` value is 2.
- 6) An `ST_CircularString` value consists of one or more circular arc segments connected end to end. The first segment is defined by three points. The first point is the start point of the arc segment. The second point is any intermediate point on the arc segment other than the start or end point. The third point is the end point of the arc segment. Subsequent segments are defined by their intermediate and end points only, as the start point is implicitly defined as the previous segment's end point. In the special case where a segment is a complete circle, that is, the start and end points are coincident, then the intermediate point shall be the midpoint of the segment.

7) Let  $NSEG$  be the number of circular arc segments in the  $ST\_CircularString$  value. If  $SELF.NumPoints$  is equal to  $2 * NSEG + 1$ , then the  $ST\_CircularString$  value is well formed.

8) A circular arc segment is the locus of points defined as follows:

Case:

a) If the start, intermediate, and end points of an arc segment are not collinear, then the circular arc segment is the locus of points a distance  $R$  from the center of the arc, beginning at the start point, passing through the intermediate point, and ending at the end point of the circular arc segment. The distance  $R$  is the radius of the circular arc segment, and is equal to the distance from the center of the circular arc segment and the start, intermediate, or end points. The center of the circular arc segment is defined as follows:

Case:

i) If the segment is a complete circle, then the center is located at the midpoint of the line connecting the start point and the intermediate point.

ii) Otherwise, let  $CHORD1$  be the line connecting the start point of a circular arc segment and the intermediate point on the segment. Let  $CHORD2$  be the line connecting the intermediate point with the end point of this arc segment. Then the center of the circular arc segment is located at the intersection of the perpendicular bisectors of  $CHORD1$  and  $CHORD2$ .

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

9) If the cardinality of the attribute  $ST\_PrivatePoints$  is three, then the  $ST\_CircularString$  value is considered a *circular arc*.

10) If an  $ST\_CircularString$  value is simple and closed, then it is considered a *circular ring*.

11) An  $ST\_CircularString$  value returned by the constructor function corresponds to the empty set.

12) An  $ST\_CircularString$  value with the cardinality of the  $ST\_PrivatePoints$  attribute equal to 0 (zero) corresponds to the empty set.

### 7.3.2 ST\_CircularString Methods

#### Purpose

Return an ST\_CircularString value constructed from either the well-known text representation or the well-known binary representation of an ST\_CircularString value, or the specified ST\_Point values.

#### Definition

```

CREATE CONSTRUCTOR METHOD ST_CircularString
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_CircularString
  FOR ST_CircularString
  RETURN ST_CircularFromTxt(awkt)

CREATE CONSTRUCTOR METHOD ST_CircularString
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_CircularString
  FOR ST_CircularString
  RETURN ST_CircularFromTxt(awkt, ansrid)

CREATE CONSTRUCTOR METHOD ST_CircularString
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_CircularString
  FOR ST_CircularString
  RETURN ST_LineFromWKB(awkb)

CREATE CONSTRUCTOR METHOD ST_CircularString
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_CircularString
  FOR ST_CircularString
  RETURN ST_LineFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_CircularString
  (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_CircularString
  FOR ST_CircularString
  RETURN SELF.ST_SRID(0).ST_Points(apointarray)

CREATE CONSTRUCTOR METHOD ST_CircularString
  (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
  RETURNS ST_CircularString
  FOR ST_CircularString
  RETURN SELF.ST_SRID(ansrid).ST_Points(apointarray)

```

#### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

#### Description

- 1) The method *ST\_CircularString(CHARACTER LARGE OBJECT)* takes the following input parameter:
  - a) a CHARACTER LARGE OBJECT value *awkt*.

- 2) For the null-call type preserving SQL-invoked constructor method *ST\_CircularString*(*CHARACTER LARGE OBJECT*):
- The parameter *awkt* is the well-known text representation of an *ST\_CircularString* value. If *awkt* is not producible in the BNF for <circularstring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - Return an *ST\_CircularString* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The method *ST\_CircularString*(*CHARACTER LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *CHARACTER LARGE OBJECT* value *awkt*,
  - an *INTEGER* value *ansrid*.
- 4) For the null-call type preserving SQL-invoked constructor method *ST\_CircularString*(*CHARACTER LARGE OBJECT*, *INTEGER*):
- The parameter *awkt* is the well-known text representation of an *ST\_CircularString* value. If *awkt* is not producible in the BNF for <circularstring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - Return an *ST\_CircularString* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.
- 5) The method *ST\_CircularString*(*BINARY LARGE OBJECT*) takes the following input parameter:
- a *BINARY LARGE OBJECT* value *awkb*.
- 6) For the null-call type preserving SQL-invoked constructor method *ST\_CircularString*(*BINARY LARGE OBJECT*):
- The parameter *awkb* is the well-known binary representation of an *ST\_CircularString* value. If *awkb* is not producible in the BNF for <circularstring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_CircularString* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 7) The method *ST\_CircularString*(*BINARY LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *BINARY LARGE OBJECT* value *awkb*,
  - an *INTEGER* value *ansrid*.
- 8) For the null-call type preserving SQL-invoked constructor method *ST\_CircularString*(*BINARY LARGE OBJECT*, *INTEGER*):
- The parameter *awkb* is the well-known binary representation of an *ST\_CircularString* value. If *awkb* is not producible in the BNF for <circularstring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_CircularString* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.
- 9) The method *ST\_CircularString*(*ST\_Point ARRAY*) takes the following input parameters:
- an *ST\_Point ARRAY* value *apointarray*
- 10) The null-call type preserving SQL-invoked constructor method *ST\_CircularString*(*ST\_Point ARRAY*) returns an *ST\_CircularString* value with:
- The spatial reference system identifier set to 0 (zero).

- b) Using the method *ST\_Points(ST\_Point ARRAY)*, the attribute *ST\_PrivatePoints* array set to *apointarray*, the *ST\_PrivateDimension* attribute set to 1 (one), and the *ST\_PrivateCoordinateDimension* attribute set to 2.
- 11) The method *ST\_CircularString(ST\_Point ARRAY, INTEGER)* takes the following input parameters:
- a) an *ST\_Point ARRAY* value *apointarray*,
  - b) an *INTEGER* value *ansrid*.
- 12) The null-call type preserving SQL-invoked constructor method *ST\_CircularString(ST\_Point ARRAY, INTEGER)* returns an *ST\_CircularString* value with:
- a) The spatial reference system identifier set to *ansrid*.
  - b) Using the method *ST\_Points(ST\_Point ARRAY)*, the attribute *ST\_PrivatePoints* array set to *apointarray*, the *ST\_PrivateDimension* attribute set to 1 (one), and the *ST\_PrivateCoordinateDimension* attribute set to 2.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 7.3.3 ST\_Points Methods

#### Purpose

Observe and mutate the attribute *ST\_PrivatePoints* of an *ST\_CircularString* value.

#### Definition

```

CREATE METHOD ST_Points()
  RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements]
  FOR ST_CircularString
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      SELF.ST_PrivatePoints
  END

CREATE METHOD ST_Points
  (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_CircularString
  FOR ST_CircularString
  BEGIN
    -- If apointarray is the null value, contains null elements, or
    -- contains consecutive duplicate points, then raise an exception.
    CALL ST_CheckConsecDups(apointarray);
    -- If SELF is the null value, then return the null value.
    IF SELF IS NULL THEN
      RETURN CAST (NULL AS ST_CircularString);
    END IF;
    -- Check that there are no mixed spatial reference
    -- systems between SELF and apointarray.
    IF SELF.ST_SRID() <> ST_CheckSRID(apointarray) THEN
      SIGNAL SQLSTATE '2FF10'
      SET MESSAGE_TEXT = 'mixed spatial reference systems';
    END IF;
    RETURN
      SELF.ST_PrivateDimension(1).
      ST_PrivateCoordinateDimension(2).
      ST_PrivatePoints(apointarray);
  END

```

#### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.

#### Description

- 1) The method *ST\_Points()* has no input parameters.
- 2) For the null-call method *ST\_Points()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the attribute *ST\_PrivatePoints* of SELF.
- 3) The method *ST\_Points(ST\_Point ARRAY)* takes the following input parameters:
  - a) an *ST\_Point ARRAY* value *apointarray*.
- 4) For the type preserving method *ST\_Points(ST\_Point ARRAY)*:
  - a) Call the procedure *ST\_CheckConsecDups(ST\_Geometry ARRAY)* to check if *apointarray* is the null value or contains null elements.

## b) Case:

- i) If SELF is the null value, then return the null value.
- ii) If the spatial reference system of SELF is not equal to *ST\_CheckSRID(apointarray)*, then an exception condition is raised: *SQL/MM Spatial exception – mixed spatial reference systems*.
- iii) Otherwise, return an *ST\_CircularString* value with:
  - 1) the dimension set to 1 (one).
  - 2) The coordinate dimension set to 2.
  - 3) the attribute *ST\_PrivatePoints* set to *apointarray*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

#### 7.3.4 ST\_NumPoints Method

##### Purpose

Return the cardinality of the ST\_PrivatePoints attribute of an ST\_CircularString value.

##### Definition

```
CREATE METHOD ST_NumPoints()  
  RETURNS INTEGER  
  FOR ST_CircularString  
  RETURN  
  CASE  
    WHEN SELF.ST_IsEmpty() = 1 THEN  
      NULL  
    ELSE  
      CARDINALITY(SELF.ST_PrivatePoints)  
  END
```

##### Description

- 1) The method *ST\_NumPoints()* has no input parameters.
- 2) For the null-call method *ST\_NumPoints()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the cardinality of the *ST\_PrivatePoints* attribute.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 7.3.5 ST\_PointN Method

#### Purpose

Return the specified element in the ST\_PrivatePoints attribute of an ST\_CircularString value.

#### Definition

```
CREATE METHOD ST_PointN
  (aposition INTEGER)
  RETURNS ST_Point
  FOR ST_CircularString
  BEGIN
    IF SELF.ST_IsEmpty() = 1 THEN
      RETURN CAST (NULL AS ST_Point);
    END IF;
    IF aposition < 1 OR
       aposition > SELF.ST_NumPoints() THEN
      BEGIN
        SIGNAL SQLSTATE '01F01'
          SET MESSAGE_TEXT = 'invalid position';
        RETURN CAST (NULL AS ST_Point);
      END;
    END IF;
    RETURN SELF.ST_PrivatePoints[aposition];
  END
```

#### Description

- 1) The method *ST\_PointN(INTEGER)* takes the following input parameters:
  - a) an INTEGER value *aposition*.
- 2) For the null-call method *ST\_PointN(INTEGER)*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) If *aposition* is less than 1 (one) or greater than the cardinality of the attribute *ST\_PrivatePoints*, then:
    - i) A completion condition is raised: *SQL/MM Spatial warning – invalid position*.
    - ii) Return the null value.
  - c) Otherwise, return an *ST\_Point* value at element *aposition* in the attribute *ST\_PrivatePoints* of SELF.

### 7.3.6 ST\_MidPointRep Method

#### Purpose

Return an ST\_Point ARRAY which uniquely identifies an ST\_CircularString value, including the start, mid, and end points of each curve segment.

#### Definition

```
CREATE METHOD ST_MidPointRep()
  RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements]
  FOR ST_CircularString
  BEGIN
    --
    -- See Description
    --
  END
```

#### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.

#### Description

- 1) The method *ST\_MidPointRep()* has no input parameters.
- 2) For the null-call method *ST\_MidPointRep()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return an *ST\_Points* ARRAY such that:
    - i) For the first circular arc segment of the curve, the points returned are the start, mid, and end points of the segment.
    - ii) For all subsequent segments, the points returned are the mid and end points of the respective segment.

### 7.3.7 ST\_StartPoint Method

#### Purpose

Return the start point of an ST\_CircularString value.

#### Definition

```
CREATE METHOD ST_StartPoint()
  RETURNS ST_Point
  FOR ST_CircularString
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      SELF.ST_Points()[1]
  END
```

#### Description

- 1) The method *ST\_StartPoint()* has no input parameters.
- 2) For the null-call method *ST\_StartPoint()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the result of the value expression: *SELF.ST\_Points()[1]*.

### 7.3.8 ST\_EndPoint Method

#### Purpose

Return the end point of an ST\_CircularString value.

#### Definition

```
CREATE METHOD ST_EndPoint()  
  RETURNS ST_Point  
  FOR ST_CircularString  
  RETURN  
  CASE  
    WHEN SELF.ST_IsEmpty() = 1 THEN  
      NULL  
    ELSE  
      SELF.ST_Points()[SELF.ST_NumPoints()]  
  END
```

#### Description

- 1) The method *ST\_EndPoint()* has no input parameters.
- 2) For the null-call method *ST\_EndPoint()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the result of the value expression: *SELF.ST\_Points()[SELF.ST\_NumPoints()]*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 7.3.9 ST\_CircularFromTxt Functions

#### Purpose

Return a specified ST\_CircularString value.

#### Definition

```
CREATE FUNCTION ST_CircularFromTxt
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_CircularString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt) AS ST_CircularString)

CREATE FUNCTION ST_CircularFromTxt
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_CircularString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt, ansrid) AS ST_CircularString)
```

#### 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.

#### Description

- 1) The function *ST\_CircularFromTxt*(CHARACTER LARGE OBJECT) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call function *ST\_CircularFromTxt*(CHARACTER LARGE OBJECT):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_CircularString* value. If *awkt* is not producible in the BNF for <circularstring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_CircularString* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_CircularFromTxt*(CHARACTER LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_CircularFromTxt*(CHARACTER LARGE OBJECT, INTEGER):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_CircularString* value. If *awkt* is not producible in the BNF for <circularstring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_CircularString* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.

### 7.3.10 ST\_CircularFromWKB Functions

#### Purpose

Return a specified ST\_CircularString value.

#### Definition

```
CREATE FUNCTION ST_CircularFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_CircularString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_CircularString)

CREATE FUNCTION ST_CircularFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_CircularString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_CircularString)
```

#### Definitional Rules

- 1) *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.

#### Description

- 1) The function *ST\_CircularFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 2) For the null-call function *ST\_CircularFromWKB(BINARY LARGE OBJECT)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_CircularString* value. If *awkb* is not producible in the BNF for <circlestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_CircularString* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_CircularFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_CircularFromWKB(BINARY LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_CircularString* value. If *awkb* is not producible in the BNF for <circlestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_CircularString* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.

## 7.4 ST\_CompoundCurve Type and Routines

### 7.4.1 ST\_CompoundCurve Type

#### Purpose

The general notion of a compound curve is a sequence of contiguous curves such that adjacent curves are joined at their end points. The contributing curve types are limited to ST\_LineString and ST\_CircularString values. Furthermore, the end point of each curve shall be coincident with the start point of the next curve in the list.

#### Definition

```

CREATE TYPE ST_CompoundCurve
  UNDER ST_Curve
  AS (
    ST_PrivateCurves ST_Curve
      ARRAY[ST_MaxGeometryArrayElements] DEFAULT ARRAY[]
  )
  INSTANTIABLE
  NOT FINAL

CONSTRUCTOR METHOD ST_CompoundCurve
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_CompoundCurve
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CompoundCurve
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   anrid INTEGER)
  RETURNS ST_CompoundCurve
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CompoundCurve
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_CompoundCurve
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CompoundCurve
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   anrid INTEGER)
  RETURNS ST_CompoundCurve
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

CONSTRUCTOR METHOD ST_CompoundCurve(acurve ST_Curve)
  RETURNS ST_CompoundCurve
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CompoundCurve
  (acurve ST_Curve,
   ansrid INTEGER)
  RETURNS ST_CompoundCurve
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CompoundCurve
  (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_CompoundCurve
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CompoundCurve
  (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
  RETURNS ST_CompoundCurve
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_Curves()
  RETURNS ST_Curve ARRAY[ST_MaxGeometryArrayElements]
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_Curves
  (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_CompoundCurve
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

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

```

```

METHOD ST_CurveN
  (aposition INTEGER)
  RETURNS ST_Curve
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_StartPoint ()
  RETURNS ST_Point,

OVERRIDING METHOD ST_EndPoint ()
  RETURNS ST_Point

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.
- 4) The attribute *ST\_PrivateCurves* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST\_PrivateCurves*.

### Description

- 1) The *ST\_CompoundCurve* type provides for public use:
  - a) a method *ST\_CompoundCurve*(CHARACTER LARGE OBJECT),
  - b) a method *ST\_CompoundCurve*(CHARACTER LARGE OBJECT, INTEGER),
  - c) a method *ST\_CompoundCurve*(BINARY LARGE OBJECT),
  - d) a method *ST\_CompoundCurve*(BINARY LARGE OBJECT, INTEGER),
  - e) a method *ST\_CompoundCurve*(*ST\_Curve*),
  - f) a method *ST\_CompoundCurve*(*ST\_Curve*, INTEGER),
  - g) a method *ST\_CompoundCurve*(*ST\_Curve* ARRAY),
  - h) a method *ST\_CompoundCurve*(*ST\_Curve* ARRAY, INTEGER),
  - i) a method *ST\_Curves*(),
  - j) a method *ST\_Curves*(*ST\_Curve* ARRAY),
  - k) a method *ST\_NumCurves*(),
  - l) a method *ST\_CurveN*(INTEGER),
  - m) an overriding method *ST\_StartPoint*(),
  - n) an overriding method *ST\_EndPoint*(),
  - o) a function *ST\_CompoundFromTxt*(CHARACTER LARGE OBJECT),
  - p) a function *ST\_CompoundFromTxt*(CHARACTER LARGE OBJECT, INTEGER),
  - q) a function *ST\_CompoundFromWKB*(BINARY LARGE OBJECT),
  - r) a function *ST\_CompoundFromWKB*(BINARY LARGE OBJECT, INTEGER).
- 2) The *ST\_PrivateCurves* attribute contains a collection of *ST\_Curve* values.
- 3) If each *ST\_Curve* value in the *ST\_PrivateCurves* attribute is well formed, then the *ST\_CompoundCurve* value is well formed.

- 4) All the *ST\_Curve* values in the *ST\_PrivateCurves* attribute are in the same spatial reference system as the *ST\_CompoundCurve* value.
- 5) The *ST\_PrivateCurves* attribute shall not be the null value. The elements in the *ST\_PrivateCurves* attribute shall not be the null value.
- 6) The coordinate dimension of an *ST\_CompoundCurve* value is 2.
- 7) A *ST\_CompoundCurve* value consists of one or more curves connected end to end. The contributing curve types are limited to *ST\_LineString* and *ST\_CircularString* values. Furthermore, the end point of each curve shall be coincident with the start point of the next curve in the list.
- 8) If an *ST\_CompoundCurve* value is simple and closed, then it is considered a ring.
- 9) An *ST\_CompoundCurve* value returned by the constructor function corresponds to the empty set.
- 10) An *ST\_CompoundCurve* value with the cardinality of the attribute *ST\_PrivateCurves* equal to 0 (zero) corresponds to the empty set.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 7.4.2 ST\_CompoundCurve Methods

### Purpose

Return an ST\_CompoundCurve value constructed from either the well-known text representation or the well-known binary representation of an ST\_CompoundCurve value, or the specified ST\_Curve values.

### Definition

```

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_CompoundCurve
  FOR ST_CompoundCurve
  RETURN ST_CompoundFromTxt(awkt)

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_CompoundCurve
  FOR ST_CompoundCurve
  RETURN ST_CompoundFromTxt(awkt, ansrid)

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_CompoundCurve
  FOR ST_CompoundCurve
  RETURN ST_CompoundFromWKB(awkb)

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_CompoundCurve
  FOR ST_CompoundCurve
  RETURN ST_CompoundFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
  (acurve ST_Curve)
  RETURNS ST_CompoundCurve
  FOR ST_CompoundCurve
  RETURN SELF.ST_SRID(0).ST_Curves(ARRAY[acurve])

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
  (acurve ST_Curve,
   ansrid INTEGER)
  RETURNS ST_CompoundCurve
  FOR ST_CompoundCurve
  RETURN SELF.ST_SRID(ansrid).ST_Curves(ARRAY[acurve])

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
  (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_CompoundCurve
  FOR ST_CompoundCurve
  RETURN SELF.ST_SRID(0).ST_Curves(acurvearray)

CREATE CONSTRUCTOR METHOD ST_CompoundCurve
  (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
  RETURNS ST_CompoundCurve
  FOR ST_CompoundCurve
  RETURN SELF.ST_SRID(ansrid).ST_Curves(acurvearray)

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.

- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

### Description

- 1) The method *ST\_CompoundCurve*(CHARACTER LARGE OBJECT) takes the following input parameter:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call type preserving SQL-invoked constructor method *ST\_CompoundCurve*(CHARACTER LARGE OBJECT):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_CompoundCurve* value. If *awkt* is not producible in the BNF for <compoundcurve text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_CompoundCurve* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The method *ST\_CompoundCurve*(CHARACTER LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call type preserving SQL-invoked constructor method *ST\_CompoundCurve*(CHARACTER LARGE OBJECT, INTEGER):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_CompoundCurve* value. If *awkt* is not producible in the BNF for <compoundcurve text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_CompoundCurve* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.
- 5) The method *ST\_CompoundCurve*(BINARY LARGE OBJECT) takes the following input parameter:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 6) For the null-call type preserving SQL-invoked constructor method *ST\_CompoundCurve*(BINARY LARGE OBJECT):
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_CompoundCurve* value. If *awkb* is not producible in the BNF for <compoundcurve binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_CompoundCurve* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 7) The method *ST\_CompoundCurve*(BINARY LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.
- 8) For the null-call type preserving SQL-invoked constructor method *ST\_CompoundCurve*(BINARY LARGE OBJECT, INTEGER):
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_CompoundCurve* value. If *awkb* is not producible in the BNF for <compoundcurve binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

- b) Return an *ST\_CompoundCurve* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.
- 9) The method *ST\_CompoundCurve(ST\_Curve)* takes the following input parameters:
- b) an *ST\_Curve* value *acurve*.
- 10) The null-call type preserving SQL-invoked constructor method *ST\_CompoundCurve(ST\_Curve)* returns an *ST\_CompoundCurve* value with:
- a) The spatial reference system identifier set to 0 (zero).
- b) Using the method *ST\_Curves(ST\_Curve ARRAY)*, the *ST\_PrivateCurves* attribute set to *ARRAY[acurve]*, the *ST\_PrivateDimension* attribute set to 1 (one), and the *ST\_PrivateCoordinateDimension* attribute set to 2.
- 11) The method *ST\_CompoundCurve(ST\_Curve, INTEGER)* takes the following input parameters:
- a) an *ST\_Curve* value *acurve*,
- b) an *INTEGER* value *ansrid*.
- 12) The null-call type preserving SQL-invoked constructor method *ST\_CompoundCurve(ST\_Curve, INTEGER)* returns an *ST\_CompoundCurve* value with:
- a) The spatial reference system identifier set to *ansrid*.
- b) Using the method *ST\_Curves(ST\_Curve ARRAY)*, the *ST\_PrivateCurves* attribute set to *ARRAY[acurve]*, the *ST\_PrivateDimension* attribute set to 1 (one), and the *ST\_PrivateCoordinateDimension* attribute set to 2.
- 13) The method *ST\_CompoundCurve(ST\_Curve ARRAY)* takes the following input parameters:
- a) an *ST\_Curve ARRAY* value *acurvearray*.
- 14) The null-call type preserving SQL-invoked constructor method *ST\_CompoundCurve(ST\_Curve ARRAY)* returns an *ST\_CompoundCurve* value with:
- a) The spatial reference system identifier set to 0 (zero).
- b) Using the method *ST\_Curves(ST\_Curve ARRAY)*, the *ST\_PrivateCurves* attribute set to *acurvearray*, the *ST\_PrivateDimension* attribute set to 1 (one), and the *ST\_PrivateCoordinateDimension* attribute set to 2.
- 15) The method *ST\_CompoundCurve(ST\_Curve ARRAY, INTEGER)* takes the following input parameters:
- a) an *ST\_Curve ARRAY* value *acurvearray*,
- b) an *INTEGER* value *ansrid*.
- 16) The null-call type preserving SQL-invoked constructor method *ST\_CompoundCurve(ST\_Curve ARRAY, INTEGER)* returns an *ST\_CompoundCurve* value with:
- a) The spatial reference system identifier set to *ansrid*.
- b) Using the method *ST\_Curves(ST\_Curve ARRAY)*, the *ST\_PrivateCurves* attribute set to *acurvearray*, the *ST\_PrivateDimension* attribute set to 1 (one), and the *ST\_PrivateCoordinateDimension* attribute set to 2.

### 7.4.3 ST\_Curves Methods

#### Purpose

Observe and mutate the ST\_PrivateCurves attribute of an ST\_CompoundCurve value.

#### Definition

```

CREATE METHOD ST_Curves()
  RETURNS ST_Curve ARRAY[ST_MaxGeometryArrayElements]
  FOR ST_CompoundCurve
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      SELF.ST_PrivateCurves
  END

CREATE METHOD ST_Curves
  (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_CompoundCurve
  FOR ST_CompoundCurve
  BEGIN
    DECLARE counter INTEGER;

    -- If acurvearray is the null value or contains null elements,
    -- then raise an exception.
    CALL ST_CheckNulls(acurvearray);
    -- If SELF is the null value, then return the null value.
    IF SELF IS NULL THEN
      RETURN CAST (NULL AS ST_CompoundCurve);
    END IF;
    -- Check that there are no mixed spatial reference
    -- systems between SELF and acurvearray.
    IF SELF.ST_SRID() <> ST_CheckSRID(acurvearray) THEN
      SIGNAL SQLSTATE '2FF10'
        SET MESSAGE_TEXT = 'mixed spatial reference systems';
    END IF;
    -- If any ST_Curve value in acurvearray is an ST_CompoundCurve
    -- value, then raise an exception.
    SET counter = 1;
    WHILE counter <= CARDINALITY(acurvearray) DO
      IF acurvearray[counter] IS OF (ST_CompoundCurve) THEN
        SIGNAL SQLSTATE '2FF02'
          SET MESSAGE_TEXT = 'invalid argument';
      END IF;
      SET counter = counter + 1;
    END WHILE;
    -- If the start point of any curve is not coincident with the end
    -- point of the previous curve, then raise an exception
    SET counter = 2;
    WHILE counter <= CARDINALITY(acurvearray) DO
      IF acurvearray[counter].ST_StartPoint() <>
        acurvearray[counter-1].ST_EndPoint() THEN
        SIGNAL SQLSTATE '2FF11'
          SET MESSAGE_TEXT = 'non-contiguous curves';
      END IF;
      SET counter = counter + 1;
    END WHILE;
    -- If SELF is the null value, then return the null value. Otherwise,
    -- return an ST_CompoundCurve value with the ST_PrivateCurves

```

```

-- attribute set to acurvearray.
RETURN
    SELF.ST_PrivateDimension(1).
        ST_PrivateCoordinateDimension(2).
        ST_PrivateCurves(acurvearray);
END

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.

### Description

- 1) The method *ST\_Curves()* has no input parameters.
- 2) For the null-call method *ST\_Curves()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the *ST\_PrivateCurves* attribute of SELF.
- 3) The method *ST\_Curves(ST\_Curve ARRAY)* takes the following input parameters:
  - a) an *ST\_Curve* ARRAY value *acurvearray*.
- 4) For the type preserving method *ST\_Curves(ST\_Curve ARRAY)*:
  - a) Call the procedure *ST\_CheckNulls(ST\_Geometry ARRAY)* to check if *acurvearray* is the null value or contains null elements.
  - b) Case:
    - i) If SELF is the null value, then return the null value.
    - ii) If the spatial reference system of SELF is not equal to *ST\_CheckSRID(acurvearray)*, then an exception condition is raised: *SQL/MM Spatial exception – mixed spatial reference systems*.
    - iii) If any *ST\_Curve* value in *acurvearray* is an *ST\_CompoundCurve* value, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
    - iv) If the start point of any *ST\_Curve* value in *acurvearray* is not equal to the end point of the previous *ST\_Curve* value in *acurvearray*, then an exception condition is raised: *SQL/MM Spatial exception – non-contiguous curves*.
    - v) Otherwise, return an *ST\_CompoundCurve* value with:
      - 1) The dimension set to 1 (one).
      - 2) The coordinate dimension set to 2.
      - 3) The *ST\_PrivateCurves* attribute set to *acurvearray*.

#### 7.4.4 ST\_NumCurves Method

##### Purpose

Return the cardinality of the ST\_PrivateCurves attribute of an ST\_CompoundCurve value.

##### Definition

```
CREATE METHOD ST_NumCurves ()
  RETURNS INTEGER
  FOR ST_CompoundCurve
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      CARDINALITY(SELF.ST_PrivateCurves)
  END
```

##### Description

- 1) The method *ST\_NumCurves()* has no input parameters.
- 2) For the null-call method *ST\_NumCurves()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the cardinality of the *ST\_PrivateCurves* attribute.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 7.4.5 ST\_CurveN Method

#### Purpose

Return the specified element in the ST\_PrivateCurves attribute of an ST\_CompoundCurve value.

#### Definition

```
CREATE METHOD ST_CurveN
  (aposition INTEGER)
  RETURNS ST_Curve
  FOR ST_CompoundCurve
  BEGIN
    IF SELF.ST_IsEmpty() = 1 THEN
      RETURN CAST (NULL AS ST_Curve);
    END IF;
    IF aposition < 1 OR
       aposition > CARDINALITY(SELF.ST_PrivateCurves) THEN
      BEGIN
        SIGNAL SQLSTATE '01F01'
          SET MESSAGE_TEXT = 'invalid position';
        RETURN CAST (NULL AS ST_Curve);
      END;
    END IF;
    RETURN SELF.ST_PrivateCurves[aposition];
  END
```

#### Description

- 1) The method *ST\_CurveN(INTEGER)* takes the following input parameters:
  - a) an INTEGER value *aposition*.
- 2) For the null-call method *ST\_CurveN(INTEGER)*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) If *aposition* is less than 1 (one) or greater than the cardinality of the *ST\_PrivateCurves* attribute, then:
    - i) A completion condition is raised: *SQL/MM Spatial warning – invalid position*.
    - ii) Return the null value.
  - c) Otherwise, return an *ST\_Curve* value at element *aposition* in the *ST\_PrivateCurves* attribute of SELF.

#### 7.4.6 ST\_StartPoint Method

##### Purpose

Return an ST\_Point value that is the start point of an ST\_CompoundCurve value.

##### Definition

```
CREATE METHOD ST_StartPoint()  
  RETURNS ST_Point  
  FOR ST_CompoundCurve  
  RETURN  
  CASE  
    WHEN SELF.ST_IsEmpty() = 1 THEN  
      NULL  
    ELSE  
      SELF.ST_Curves()[1].ST_Points()[1]  
  END
```

##### Description

- 1) The method *ST\_StartPoint()* has no input parameters.
- 2) For the null-call method *ST\_StartPoint()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the result of the value expression: *SELF.ST\_Curves()[1].ST\_Points()[1]*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

#### 7.4.7 ST\_EndPoint Method

##### Purpose

Return an ST\_Point value that is the end point of an ST\_CompoundCurve value.

##### Definition

```
CREATE METHOD ST_EndPoint()
  RETURNS ST_Point
  FOR ST_CompoundCurve
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      SELF.ST_Curves()[SELF.ST_NumCurves()].
        ST_Points[SELF.ST_Curves()[SELF.ST_NumCurves()].
          ST_NumPoints()]
  END
```

##### Description

1) The method *ST\_EndPoint()* has no input parameters.

2) For the null-call method *ST\_EndPoint()*:

Case:

a) If SELF is an empty set, then return the null value.

b) Otherwise, return the result of the value expression:

SELF.ST\_Curves()[SELF.ST\_NumCurves()].ST\_Points()[SELF.ST\_Curves()[SELF.ST\_NumCurves()].ST\_NumPoints()].

### 7.4.8 ST\_CompoundFromTxt Functions

#### Purpose

Return a specified ST\_CompoundCurve value.

#### Definition

```
CREATE FUNCTION ST_CompoundFromTxt
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_CompoundCurve
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt) AS ST_CompoundCurve)

CREATE FUNCTION ST_CompoundFromTxt
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_CompoundCurve
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt, ansrid) AS ST_CompoundCurve)
```

#### 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.

#### Description

- 1) The function *ST\_CompoundFromTxt*(CHARACTER LARGE OBJECT) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call function *ST\_CompoundFromTxt*(CHARACTER LARGE OBJECT):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_CompoundCurve* value. If *awkt* is not producible in the BNF for <compoundcurve text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_CompoundCurve* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_CompoundFromTxt*(CHARACTER LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_CompoundFromTxt*(CHARACTER LARGE OBJECT, INTEGER):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_CompoundCurve* value. If *awkt* is not producible in the BNF for <compoundcurve text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_CompoundCurve* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.

### 7.4.9 ST\_CompoundFromWKB Functions

#### Purpose

Return a specified ST\_CompoundCurve value.

#### Definition

```
CREATE FUNCTION ST_CompoundFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_CompoundCurve
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_CompoundCurve)

CREATE FUNCTION ST_CompoundFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_CompoundCurve
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_CompoundCurve)
```

#### Definitional Rules

- 1) *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.

#### Description

- 1) The function *ST\_CompoundFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 2) For the null-call function *ST\_CompoundFromWKB(BINARY LARGE OBJECT)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_CompoundCurve* value. If *awkb* is not producible in the BNF for <compoundcurve binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_CompoundCurve* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_CompoundFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_CompoundFromWKB(BINARY LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_CompoundCurve* value. If *awkb* is not producible in the BNF for <compoundcurve binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_CompoundCurve* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.

Blank page

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 8 Surface Types

### 8.1 ST\_Surface Type and Routines

#### 8.1.1 ST\_Surface Type

##### Purpose

The ST\_Surface type is a supertype for 2-dimensional geometry types.

##### Definition

```

CREATE TYPE ST_Surface
  UNDER ST_Geometry
  NOT INSTANTIABLE
  NOT FINAL

METHOD ST_Area()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

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

METHOD ST_Perimeter()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

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

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

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

```

### Definitional Rules

- 1) *ST\_MaxUnitNameLength* is the implementation-defined maximum length for the character representation of a unit indication.

### Description

- 1) The *ST\_Surface* type provides for public use:
  - a) a method *ST\_Area()*,
  - b) a method *ST\_Area(CHARACTER VARYING)*,
  - c) a method *ST\_Perimeter()*,
  - d) a method *ST\_Perimeter(CHARACTER VARYING)*,
  - e) a method *ST\_Centroid()*,
  - f) a method *ST\_PointOnSurface()*.
- 2) An *ST\_Surface* value is a 2-dimensional *ST\_Geometry* value that consists of a single connected interior that is associated with one exterior ring and zero or more interior rings. *ST\_Surface* values in three-dimensional coordinate space are isomorphic to planar *ST\_Surface* values. Stitching together simple surfaces along their boundaries forms polyhedral *ST\_Surface* values and polyhedral surfaces in three-dimensional coordinate space may not be planar.
- 3) The dimension of an *ST\_Surface* value is 2.
- 4) The boundary of an *ST\_Surface* value is the collection of the exterior ring and interior rings.
- 5) An *ST\_Surface* value is simple.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 8.1.2 ST\_Area Methods

#### Purpose

Return the area measurement of an ST\_Surface value.

#### Definition

```
CREATE METHOD ST_Area()
  RETURNS DOUBLE PRECISION
  FOR ST_Surface
  BEGIN
    --
    -- See Description
    --
  END

CREATE METHOD ST_Area
  (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
  RETURNS DOUBLE PRECISION
  FOR ST_Surface
  BEGIN
    --
    -- See Description
    --
  END
```

#### Definitional Rules

- 1) *ST\_MaxUnitNameLength* is the implementation-defined maximum length for the character representation of a unit indication.

#### Description

- 1) The method *ST\_Area()* has no input parameters.
- 2) For the null-call method *ST\_Area()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the implementation-defined area of SELF as measured in its spatial reference system.
- 3) Case:
  - a) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST\_Area()* is in the linear unit of measure identified by <linear unit> squared.
  - b) Otherwise, the value returned by *ST\_Area()* is in an implementation-defined unit of measure.
- 4) The method *ST\_Area(CHARACTER VARYING)* takes the following input parameter:
  - a) a CHARACTER VARYING value *aunit*.
- 5) For the null-call method *ST\_Area(CHARACTER VARYING)*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the area of SELF as measured in its spatial reference system.
- 6) The value returned by *ST\_Area(CHARACTER VARYING)* is in the units indicated by *aunit*.
- 7) The values for *aunit* shall be a supported <unit name>.

- 8) The value for *unit* is a supported <unit name> if and only if the value of *unit* is equal to the value of the UNIT\_NAME column of one of the rows where the value of the UNIT\_TYPE column is equal to 'LINEAR' in the ST\_UNITS\_OF\_MEASURE view.
- 9) If the unit specified by *unit* is not supported by the implementation to compute the area of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 8.1.3 ST\_Perimeter Methods

#### Purpose

Return the length measurement of the boundary of an ST\_Surface value.

#### Definition

```
CREATE METHOD ST_Perimeter()
  RETURNS DOUBLE PRECISION
  FOR ST_Surface
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      SELF.ST_Boundary().ST_Length()
  END

CREATE METHOD ST_Perimeter
(aunit CHARACTER VARYING(ST_MaxUnitNameLength))
  RETURNS DOUBLE PRECISION
  FOR ST_Surface
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      SELF.ST_Boundary().ST_Length(aunit)
  END
```

#### Definitional Rules

- 1) *ST\_MaxUnitNameLength* is the implementation-defined maximum length for the character representation of a unit indication.

#### Description

- 1) The method *ST\_Perimeter()* has no input parameters.
- 2) For the null-call method *ST\_Perimeter()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the implementation-defined length of the boundary of SELF as measured in its spatial reference system.
- 3) Case:
  - a) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST\_Perimeter()* is in the linear unit of measure identified by <linear unit> squared.
  - b) Otherwise, the value returned by *ST\_Perimeter()* is in an implementation-defined unit of measure.
- 4) The method *ST\_Perimeter(CHARACTER VARYING)* takes the following input parameter:
  - a) a CHARACTER VARYING value *aunit*.
- 5) For the null-call method *ST\_Perimeter(CHARACTER VARYING)*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the length of the boundary of SELF as measured in its spatial reference system.
- 6) The value returned by *ST\_Perimeter(CHARACTER VARYING)* is in the units indicated by *aunit*.

- 7) The values for *unit* shall be a supported <unit name>.
- 8) The value for *unit* is a supported <unit name> if and only if the value of *unit* is equal to the value of the UNIT\_NAME column of one of the rows where the value of the UNIT\_TYPE column is equal to 'LINEAR' in the ST\_UNITS\_OF\_MEASURE view.
- 9) If the unit specified by *unit* is not supported by the implementation to compute the length of the boundary of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

#### 8.1.4 ST\_Centroid Method

##### Purpose

Return mathematical centroid of the *ST\_Surface* value.

##### Definition

```
CREATE METHOD ST_Centroid()
  RETURNS ST_Point
  FOR ST_Surface
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    -- ELSE
    --
    -- See Description
    --
  END
```

##### Description

- 1) The method *ST\_Centroid()* has no input parameters.
- 2) For the null-call method *ST\_Centroid()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the mathematical centroid of the *ST\_Surface* value. The result is not guaranteed to spatially intersect the *ST\_Surface* value.

The spatial reference system identifier of the returned *ST\_Point* value is equal to the spatial reference system identifier of SELF.

**8.1.5 ST\_PointOnSurface Method****Purpose**

Return an *ST\_Point* value guaranteed to spatially intersect the *ST\_Surface* value.

**Definition**

```
CREATE METHOD ST_PointOnSurface()
  RETURNS ST_Point
  FOR ST_Surface
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    -- ELSE
    --
    -- See Description
    --
  END
```

**Description**

- 1) The method *ST\_PointOnSurface()* has no input parameters.
- 2) For the null-call method *ST\_PointOnSurface()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return an *ST\_Point* value guaranteed to spatially intersect the *ST\_Surface* value.

The spatial reference system identifier of the returned *ST\_Point* value is equal to the spatial reference system identifier of SELF.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 8.2 ST\_CurvePolygon Type and Routines

### 8.2.1 ST\_CurvePolygon Type

#### Purpose

The ST\_CurvePolygon type is a subtype of the ST\_Surface type and values represent a planar surface whose boundary is specified by one exterior ring and zero or more interior rings. Each interior ring defines a hole in the curve polygon.

#### Definition

```

CREATE TYPE ST_CurvePolygon
  UNDER ST_Surface
  AS (
    ST_PrivateExteriorRing ST_Curve,
    ST_PrivateInteriorRings ST_Curve
      ARRAY[ST_MaxGeometryArrayElements] DEFAULT ARRAY[]
  )
  INSTANTIABLE
  NOT FINAL

CONSTRUCTOR METHOD ST_CurvePolygon
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_CurvePolygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CurvePolygon
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   anrid INTEGER)
  RETURNS ST_CurvePolygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CurvePolygon
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_CurvePolygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CurvePolygon
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   anrid INTEGER)
  RETURNS ST_CurvePolygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

CONSTRUCTOR METHOD ST_CurvePolygon
  (acurve ST_Curve)
  RETURNS ST_CurvePolygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CurvePolygon
  (acurve ST_Curve,
   acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_CurvePolygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CurvePolygon
  (acurve ST_Curve,
   ansrid INTEGER)
  RETURNS ST_CurvePolygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_CurvePolygon
  (acurve ST_Curve,
   acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
  RETURNS ST_CurvePolygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_ExteriorRing()
  RETURNS ST_Curve
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_ExteriorRing(acurve ST_Curve)
  RETURNS ST_CurvePolygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD ST_InteriorRings()
  RETURNS ST_Curve ARRAY[ST_MaxGeometryArrayElements]
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

METHOD ST_InteriorRings
  (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_CurvePolygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

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

METHOD ST_InteriorRingN
  (aposition INTEGER)
  RETURNS ST_Curve
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

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

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.
- 4) The attribute *ST\_PrivateExteriorRing* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST\_PrivateExteriorRing*.
- 5) The attribute *ST\_PrivateInteriorRings* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST\_PrivateInteriorRings*.

### Description

- 1) The *ST\_CurvePolygon* type provides for public use:
  - a) a method *ST\_CurvePolygon*(CHARACTER LARGE OBJECT),
  - b) a method *ST\_CurvePolygon*(CHARACTER LARGE OBJECT, INTEGER),
  - c) a method *ST\_CurvePolygon*(BINARY LARGE OBJECT),
  - d) a method *ST\_CurvePolygon*(BINARY LARGE OBJECT, INTEGER),
  - e) a method *ST\_CurvePolygon*(*ST\_Curve*),
  - f) a method *ST\_CurvePolygon*(*ST\_Curve*, *ST\_Curve* ARRAY),
  - g) a method *ST\_CurvePolygon*(*ST\_Curve*, INTEGER),
  - h) a method *ST\_CurvePolygon*(*ST\_Curve*, *ST\_Curve* ARRAY, INTEGER),

- i) a method *ST\_ExteriorRing()*,
  - j) a method *ST\_ExteriorRing(ST\_Curve)*,
  - k) a method *ST\_InteriorRings()*,
  - l) a method *ST\_InteriorRings(ST\_Curve ARRAY)*,
  - m) a method *ST\_NumInteriorRing()*,
  - n) a method *ST\_InteriorRingN(INTEGER)*,
  - o) a method *ST\_CurvePolyToPoly()*,
  - p) a function *ST\_CPolyFromText(CHARACTER LARGE OBJECT)*,
  - q) a function *ST\_CPolyFromText(CHARACTER LARGE OBJECT, INTEGER)*,
  - r) a function *ST\_CPolyFromWKB(BINARY LARGE OBJECT)*,
  - s) a function *ST\_CPolyFromWKB(BINARY LARGE OBJECT, INTEGER)*.
- 2) The *ST\_PrivateExteriorRing* attribute is an *ST\_Curve* value that is a ring.
  - 3) The *ST\_PrivateInteriorRings* attribute is a collection of *ST\_Curve* values. Each *ST\_Curve* value in the collection is a ring.
  - 4) The *ST\_PrivateExteriorRing* attribute shall not be the null value.
  - 5) The *ST\_PrivateInteriorRings* attribute shall not be the null value. The elements in the *ST\_PrivateInteriorRings* attribute shall not be the null value. If the *ST\_CurvePolygon* value does not have interior rings, then the *ST\_PrivateInteriorRings* attribute is set to an empty *ST\_Curve ARRAY* value.
  - 6) All the *ST\_Curve* values in the *ST\_PrivateExteriorRing* attribute and *ST\_PrivateInteriorRings* attribute shall be in the same spatial reference system as the *ST\_CurvePolygon* value.
  - 7) The coordinate dimension of an *ST\_CurvePolygon* value is 2.
  - 8) The ring in the *ST\_PrivateExteriorRing* attribute and the rings in the *ST\_PrivateInteriorRings* attribute represent the boundary of the *ST\_CurvePolygon* value.
  - 9) An *ST\_CurvePolygon* value is topologically closed.
  - 10) The rings in the boundary may only spatially intersect at a finite number of points:
 
$$\forall p \in ST\_CurvePolygon, \forall c_1, c_2 \in Boundary(p), c_1 \neq c_2,$$

$$\forall a_1, a_2 \in ST\_Point, a_1, a_2 \in c_1, a_1 \neq a_2, [ a_1 \in c_2 \Rightarrow a_2 \notin c_2 ]$$
  - 11) An *ST\_CurvePolygon* value shall not have cut lines, spikes or punctures:
 
$$\forall p \in ST\_CurvePolygon, p = Closure(Interior(p))$$
  - 12) The interior of every *ST\_CurvePolygon* value is a connected point set.
  - 13) The exterior of an *ST\_CurvePolygon* with one or more holes is not connected. Each hole defines a connected component of the exterior.
  - 14) An *ST\_CurvePolygon* is a topologically closed point set.
  - 15) An *ST\_CurvePolygon* value returned by the constructor function corresponds to the empty set.
  - 16) An *ST\_CurvePolygon* value corresponds to the empty set if the *ST\_PrivateExteriorRing* attribute corresponds to the empty set.
  - 17) An *ST\_CurvePolygon* value is well formed only if all the *ST\_Curve* values in the *ST\_PrivateExteriorRing* attribute and *ST\_PrivateInteriorRings* attribute are well formed.

## 8.2.2 ST\_CurvePolygon Methods

### Purpose

Return an ST\_CurvePolygon value constructed from either the well-known text representation or the well-known binary representation of an ST\_CurvePolygon value, or the specified ST\_Curve values.

### Definition

```

CREATE CONSTRUCTOR METHOD ST_CurvePolygon
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_CurvePolygon
  FOR ST_CurvePolygon
  RETURN ST_CPolyFromText(awkt)

CREATE CONSTRUCTOR METHOD ST_CurvePolygon
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_CurvePolygon
  FOR ST_CurvePolygon
  RETURN ST_CPolyFromText(awkt, ansrid)

CREATE CONSTRUCTOR METHOD ST_CurvePolygon
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_CurvePolygon
  FOR ST_CurvePolygon
  RETURN ST_CPolyFromWKB(awkb)

CREATE CONSTRUCTOR METHOD ST_CurvePolygon
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_CurvePolygon
  FOR ST_CurvePolygon
  RETURN ST_CPolyFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_CurvePolygon
  (acurve ST_Curve)
  RETURNS ST_CurvePolygon
  FOR ST_CurvePolygon
  RETURN SELF.ST_SRID(0).ST_ExteriorRing(acurve).
    ST_InteriorRings(CAST(ARRAY[] AS
      ST_Curve ARRAY[ST_MaxGeometryArrayElements]))

CREATE CONSTRUCTOR METHOD ST_CurvePolygon
  (acurve ST_Curve,
   acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_CurvePolygon
  FOR ST_CurvePolygon
  RETURN SELF.ST_SRID(0).ST_ExteriorRing(acurve).
    ST_InteriorRings(acurvearray)

CREATE CONSTRUCTOR METHOD ST_CurvePolygon
  (acurve ST_Curve,
   ansrid INTEGER)
  RETURNS ST_CurvePolygon
  FOR ST_CurvePolygon
  RETURN SELF.ST_SRID(ansrid).ST_ExteriorRing(acurve).
    ST_InteriorRings(CAST(ARRAY[] AS
      ST_Curve ARRAY[ST_MaxGeometryArrayElements]))

```

```

CREATE CONSTRUCTOR METHOD ST_CurvePolygon
  (acurve ST_Curve,
   acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
RETURNS ST_CurvePolygon
FOR ST_CurvePolygon
RETURN SELF.ST_SRID(ansrid).ST_ExteriorRing(acurve) .
      ST_InteriorRings(acurvearray)

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

### Description

- 1) The method *ST\_CurvePolygon*(CHARACTER LARGE OBJECT) takes the following input parameter:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call type preserving SQL-invoked constructor method *ST\_CurvePolygon*(CHARACTER LARGE OBJECT):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_CurvePolygon* value. If *awkt* is not producible in the BNF for <curvopolygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_CurvePolygon* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The method *ST\_CurvePolygon*(CHARACTER LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call type preserving SQL-invoked constructor method *ST\_CurvePolygon*(CHARACTER LARGE OBJECT, INTEGER):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_CurvePolygon* value. If *awkt* is not producible in the BNF for <curvopolygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_CurvePolygon* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.
- 5) The method *ST\_CurvePolygon*(BINARY LARGE OBJECT) takes the following input parameter:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 6) For the null-call type preserving SQL-invoked constructor method *ST\_CurvePolygon*(BINARY LARGE OBJECT):
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_CurvePolygon* value. If *awkb* is not producible in the BNF for <curvopolygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_CurvePolygon* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).

- 7) The method *ST\_CurvePolygon*(*BINARY LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *BINARY LARGE OBJECT* value *awkb*,
  - an *INTEGER* value *ansrid*.
- 8) For the null-call type preserving SQL-invoked constructor method *ST\_CurvePolygon*(*BINARY LARGE OBJECT*, *INTEGER*):
- The parameter *awkb* is the well-known binary representation of an *ST\_CurvePolygon* value. If *awkb* is not producible in the BNF for <curvopolygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_CurvePolygon* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.
- 9) The method *ST\_CurvePolygon*(*ST\_Curve*) takes the following input parameters:
- an *ST\_Curve* value *acurve*.
- 10) The null-call type preserving SQL-invoked constructor method *ST\_CurvePolygon*(*ST\_Curve*) returns an *ST\_CurvePolygon* value with:
- The spatial reference system identifier set to 0 (zero).
  - Using the method *ST\_ExteriorRing*(*ST\_Curve*), the *ST\_PrivateExteriorRing* attribute set to *acurve*, the *ST\_PrivateDimension* attribute set to 2, and the *ST\_PrivateCoordinateDimension* attribute set to 2.
  - Using the method *ST\_InteriorRings*(*ST\_Curve ARRAY*), the *ST\_PrivateInteriorRings* attribute set to an empty *ST\_Curve ARRAY* value.
- 11) The method *ST\_CurvePolygon*(*ST\_Curve*, *ST\_Curve ARRAY*) takes the following input parameters:
- an *ST\_Curve* value *acurve*,
  - an *ST\_Curve ARRAY* value *acurvearray*.
- 12) The null-call type preserving SQL-invoked constructor method *ST\_CurvePolygon*(*ST\_Curve*, *ST\_Curve ARRAY*) returns an *ST\_CurvePolygon* value with:
- The spatial reference system identifier set to 0 (zero).
  - Using the method *ST\_ExteriorRing*(*ST\_Curve*), the *ST\_PrivateExteriorRing* attribute set to *acurve*, the *ST\_PrivateDimension* attribute set to 2, and the *ST\_PrivateCoordinateDimension* attribute set to 2.
  - Using the method *ST\_InteriorRings*(*ST\_Curve ARRAY*), the *ST\_PrivateInteriorRings* attribute set to *acurvearray*.
- 13) The method *ST\_CurvePolygon*(*ST\_Curve*, *INTEGER*) takes the following input parameters:
- an *ST\_Curve* value *acurve*,
  - an *INTEGER* value *ansrid*.
- 14) The null-call type preserving SQL-invoked constructor method *ST\_CurvePolygon*(*ST\_Curve*, *INTEGER*) returns an *ST\_CurvePolygon* value with:
- The spatial reference system identifier set to *ansrid*.
  - Using the method *ST\_ExteriorRing*(*ST\_Curve*), the *ST\_PrivateExteriorRing* attribute set to *acurve*, the *ST\_PrivateDimension* attribute set to 2, and the *ST\_PrivateCoordinateDimension* attribute set to 2.
  - Using the method *ST\_InteriorRings*(*ST\_Curve ARRAY*), the *ST\_PrivateInteriorRings* attribute set to an empty *ST\_Curve ARRAY* value.

15) The method *ST\_CurvePolygon*(*ST\_Curve*, *ST\_Curve* ARRAY, *INTEGER*) takes the following input parameters:

- a) an *ST\_Curve* value *acurve*,
- b) an *ST\_Curve* ARRAY value *acurvearray*,
- c) an *INTEGER* value *ansrid*.

16) The null-call type preserving SQL-invoked constructor method *ST\_CurvePolygon*(*ST\_Curve*, *ST\_Curve* ARRAY, *INTEGER*) returns an *ST\_CurvePolygon* value with:

- a) The spatial reference system identifier set to *ansrid*.
- b) Using the method *ST\_ExteriorRing*(*ST\_Curve*), the *ST\_PrivateExteriorRing* attribute set to *acurve*, the *ST\_PrivateDimension* attribute set to 2, and the *ST\_PrivateCoordinateDimension* attribute set to 2.
- c) Using the method *ST\_InteriorRings*(*ST\_Curve* ARRAY), the *ST\_PrivateInteriorRings* attribute set to *acurvearray*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 8.2.3 ST\_ExteriorRing Methods

#### Purpose

Observe and mutate the ST\_PrivateExteriorRing attribute of an ST\_CurvePolygon value.

#### Definition

```

CREATE METHOD ST_ExteriorRing()
  RETURNS ST_Curve
  FOR ST_CurvePolygon
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      SELF.ST_PrivateExteriorRing
  END

CREATE METHOD ST_ExteriorRing
  (acurve ST_Curve)
  RETURNS ST_CurvePolygon
  FOR ST_CurvePolygon
  BEGIN
    DECLARE acounter INTEGER;

    IF acurve IS NULL THEN
      SIGNAL SQLSTATE '2FF03'
        SET MESSAGE_TEXT = 'null argument';
    END IF;
    -- If SELF is the null value, then return the null value.
    IF SELF IS NULL THEN
      RETURN CAST (NULL AS ST_CurvePolygon);
    END IF;
    -- Check that there are no mixed spatial reference
    -- systems between SELF and acurve.
    IF SELF.ST_SRID() <> acurve.ST_SRID() THEN
      SIGNAL SQLSTATE '2FF10'
        SET MESSAGE_TEXT = 'mixed spatial reference systems';
    END IF;
    -- If acurve is not a ring, then raise an exception.
    IF acurve.ST_IsRing() = 0 THEN
      SIGNAL SQLSTATE '2FF02'
        SET MESSAGE_TEXT = 'invalid argument';
    END IF;
    -- For all interior rings
    SET acounter = 1;
    WHILE acounter <= CARDINALITY(SELF.ST_InteriorRings()) DO
      -- If the current interior ring as a polygon is not within
      -- acurve as a polygon, then raise an exception
      IF SELF.ST_InteriorRings()[acounter].ST_Within(
        SELF.ST_CurvePolygon(acurve, SELF.ST_SRID())) = 0 THEN
        SIGNAL SQLSTATE '2FF02'
          SET MESSAGE_TEXT = 'invalid argument';
      END IF;
      -- If the current interior ring intersects acurve
      -- with a dimension greater than 0 (zero), then
      -- raise an exception.
      IF SELF.ST_InteriorRings()[acounter].ST_Intersection(acurve).
        ST_Dimension() > 0 THEN
        SIGNAL SQLSTATE '2FF02'
          SET MESSAGE_TEXT = 'invalid argument';
      END IF;
    END WHILE;
  END

```

```

        END IF;
        SET acounter = acounter + 1;
    END WHILE;
    -- Return an ST_CurvePolygon value with the ST_PrivateExteriorRing
    -- attribute set to acurve.
    RETURN
        SELF.ST_PrivateDimension(2) .
        ST_PrivateCoordinateDimension(2) .
        ST_PrivateExteriorRing(acurve);
END

```

### Description

- 1) The method *ST\_ExteriorRing()* has no input parameters.
- 2) For the null-call method *ST\_ExteriorRing()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the *ST\_PrivateExteriorRing* attribute of SELF.
- 3) The method *ST\_ExteriorRing(ST\_Curve)* takes the following input parameters:
  - a) an *ST\_Curve* value *acurve*.
- 4) For the type preserving method *ST\_ExteriorRing(ST\_Curve)*:
 

Case:

  - a) If *acurve* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
  - b) If SELF is the null value, then return the null value.
  - c) If the spatial reference system of SELF is not equal to the spatial reference system of *acurve*, then an exception condition is raised: *SQL/MM Spatial exception – mixed spatial reference systems*.
  - d) If *acurve* is not a ring, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
  - e) If any two rings in *acurve* and the interior rings of SELF spatially intersect with dimension of the result greater than 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
  - f) If any ring in *acurvearray* is not spatially within an *ST\_CurvePolygon* value formed from the exterior ring of SELF, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
  - g) Otherwise, return an *ST\_CurvePolygon* value with:
    - i) The dimension set to 2.
    - ii) The coordinate dimension set to 2.
    - iii) The *ST\_PrivateExteriorRing* attribute set to *acurve*.

## 8.2.4 ST\_InteriorRings Methods

### Purpose

Observe and mutate the ST\_PrivateInteriorRings attribute of an ST\_CurvePolygon value.

### Definition

```

CREATE METHOD ST_InteriorRings()
  RETURNS ST_Curve ARRAY[ST_MaxGeometryArrayElements]
  FOR ST_CurvePolygon
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      SELF.ST_PrivateInteriorRings
  END

CREATE METHOD ST_InteriorRings
  (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_CurvePolygon
  FOR ST_CurvePolygon
  BEGIN
    DECLARE acounter INTEGER;
    DECLARE bcounter INTEGER;

    IF SELF.ST_ExteriorRing() IS NULL THEN
      SIGNAL SQLSTATE '2FF07'
        SET MESSAGE_TEXT = 'null exterior ring';
    END IF;
    -- If acurvearray is the null value or contains null elements,
    -- then raise an exception.
    CALL ST_CheckNulls(acurvearray);
    -- If SELF is the null value, then return the null value.
    IF SELF IS NULL THEN
      RETURN SELF;
    END IF;
    -- Check that there are no mixed spatial reference
    -- systems between SELF and acurvearray.
    IF SELF.ST_SRID() <> ST_CheckSRID(acurvearray) THEN
      SIGNAL SQLSTATE '2FF10'
        SET MESSAGE_TEXT = 'mixed spatial reference systems';
    END IF;
    -- If any ST_Curve value is not a ring, then
    -- raise an exception.
    SET acounter = 1;
    WHILE acounter <= CARDINALITY(acurvearray) DO
      IF acurvearray[acounter].ST_IsRing() = 0 THEN
        SIGNAL SQLSTATE '2FF02'
          SET MESSAGE_TEXT = 'invalid argument';
      END IF;
      SET acounter = acounter + 1;
    END WHILE;
    -- For all rings in acurvearray
    SET acounter = 1;
    WHILE acounter <= CARDINALITY(acurvearray) DO
      -- If the current interior ring as a polygon not within
      -- the exterior ring as a polygon, then raise an exception
      IF acurvearray[acounter].ST_Within(
        SELF.ST_CurvePolygon(SELF.ST_ExteriorRing(),
          SELF.ST_SRID())) = 0 THEN

```

```

        SIGNAL SQLSTATE '2FF02'
          SET MESSAGE_TEXT = 'invalid argument';
      END IF;
      -- If the current interior ring intersects the exterior
      -- ring with a dimension greater than zero, then
      -- raise an exception.
      IF acurvearray[acounter].ST_Intersection(
        SELF.ST_ExteriorRing()).ST_Dimension() > 0 THEN
        SIGNAL SQLSTATE '2FF02'
          SET MESSAGE_TEXT = 'invalid argument';
      END IF;
      SET acounter = acounter + 1;
    END WHILE;
    SET acounter = 1;
    -- For each ring pair in acurvearray
    WHILE acounter <= CARDINALITY(acurvearray)-1 DO
      SET bcounter = acounter+1;
      WHILE bcounter <= CARDINALITY(acurvearray) DO
        -- If the current interior ring pair overlap, then
        -- raise an exception.
        IF SELF.ST_CurvePolygon(acurvearray[acounter],
          SELF.ST_SRID()).ST_Overlaps(
          SELF.ST_CurvePolygon(acurvearray[bcounter],
            SELF.ST_SRID())) = 1 THEN
          SIGNAL SQLSTATE '2FF02'
            SET MESSAGE_TEXT = 'invalid argument';
        END IF;
        -- If the current interior ring pair intersect
        -- with a dimension greater than zero, then
        -- raise an exception.
        IF acurvearray[acounter].ST_Intersection(
          acurvearray[bcounter]).ST_Dimension() > 0 THEN
          SIGNAL SQLSTATE '2FF02'
            SET MESSAGE_TEXT = 'invalid argument';
        END IF;
        SET bcounter = bcounter + 1;
      END WHILE;
      SET acounter = acounter + 1;
    END WHILE;
    -- Return an ST_CurvePolygon value with the ST_PrivateInteriorRings
    -- attribute set to acurvearray.
    RETURN SELF.ST_PrivateInteriorRings(acurvearray);
  END

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.

### Description

- 1) The method *ST\_InteriorRings()* has no input parameters.
- 2) For the null-call method *ST\_InteriorRings()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the *ST\_PrivateInteriorRings* attribute of SELF.
- 3) The method *ST\_InteriorRings(ST\_Curve ARRAY)* takes the following input parameters:
  - a) an *ST\_Curve* ARRAY value *acurvearray*.

4) For the type preserving method *ST\_InteriorRings()*:

Case:

- a) If *SELF.ST\_ExteriorRing()* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null exterior ring*.
- b) Call the procedure *ST\_CheckNulls(ST\_Geometry ARRAY)* to check if *acurvearray* is the null value or contains null elements.
- c) If *SELF* is the null value, then return the null value.
- d) If the spatial reference system of *SELF* is not equal to *ST\_CheckSRID(acurvearray)*, then an exception condition is raised: *SQL/MM Spatial exception – mixed spatial reference systems*.
- e) If any *ST\_Curve* value in *acurvearray* is not a ring, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
- f) If any rings in *acurvearray* and the exterior ring of *SELF* spatially intersect with dimension of the result greater than 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
- g) If any ring in *acurvearray* is not spatially within an *ST\_CurvePolygon* value formed from the exterior ring of *SELF*, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
- h) If any two rings in *acurvearray*, formed into *ST\_CurvePolygon* values with no interior rings spatially overlap, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
- i) Otherwise, return an *ST\_CurvePolygon* value with the *ST\_PrivateInteriorRings* attribute set to *acurvearray*.

### 8.2.5 ST\_NumInteriorRing Method

#### Purpose

Return the cardinality of the ST\_PrivateInteriorRings attribute of an ST\_CurvePolygon value.

#### Definition

```
CREATE METHOD ST_NumInteriorRing()  
  RETURNS INTEGER  
  FOR ST_CurvePolygon  
  RETURN  
  CASE  
    WHEN SELF.ST_IsEmpty() = 1 THEN  
      NULL  
    ELSE  
      CARDINALITY(SELF.ST_PrivateInteriorRings)  
  END
```

#### Description

- 1) The method *ST\_NumInteriorRing()* has no input parameters.
- 2) For the null-call method *ST\_NumInteriorRing()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the cardinality of the *ST\_PrivateInteriorRings* attribute.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 8.2.6 ST\_InteriorRingN Method

#### Purpose

Return the specified element in the ST\_PrivateInteriorRings attribute of an ST\_CurvePolygon value.

#### Definition

```
CREATE METHOD ST_InteriorRingN
  (aposition INTEGER)
  RETURNS ST_Curve
  FOR ST_CurvePolygon
  BEGIN
    IF SELF.ST_IsEmpty() = 1 THEN
      RETURN CAST (NULL AS ST_Curve);
    END IF;
    IF aposition < 1 OR
       aposition > CARDINALITY(SELF.ST_PrivateInteriorRings) THEN
      BEGIN
        SIGNAL SQLSTATE '01F01'
          SET MESSAGE_TEXT = 'invalid position';
        RETURN CAST (NULL AS ST_Curve);
      END;
    END IF;
    RETURN SELF.ST_PrivateInteriorRings[aposition];
  END
```

#### Description

- 1) The method *ST\_InteriorRingN(INTEGER)* takes the following input parameters:
  - a) an INTEGER value *aposition*.
- 2) For the null-call method *ST\_InteriorRingN(INTEGER)*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) If *aposition* is less than one or greater than the cardinality of the *ST\_PrivateInteriorRings* attribute, then:
    - i) A completion condition is raised: *SQL/MM Spatial warning – invalid position*.
    - ii) Return the null value.
  - c) Otherwise, return an *ST\_Curve* value at element *aposition* in the *ST\_PrivateInteriorRings* attribute of SELF.

### 8.2.7 ST\_CurvePolyToPoly Method

#### Purpose

Return the ST\_Polygon approximation of an ST\_CurvePolygon value.

#### Definition

```
CREATE METHOD ST_CurvePolyToPoly()  
  RETURNS ST_Polygon  
  FOR ST_CurvePolygon  
  BEGIN  
    --  
    -- See Description  
    --  
  END
```

#### Description

- 1) The method *ST\_CurvePolyToPoly()* has no input parameters.
- 2) For the null-call method *ST\_CurvePolyToPoly()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the implementation-defined *ST\_Polygon* value approximation of the *ST\_CurvePolygon* value.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 8.2.8 ST\_CPolyFromText Functions

### Purpose

Return a specified ST\_CurvePolygon value.

### Definition

```
CREATE FUNCTION ST_CPolyFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_CurvePolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt) AS ST_CurvePolygon)

CREATE FUNCTION ST_CPolyFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_CurvePolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt, ansrid) AS ST_CurvePolygon)
```

### 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.

### Description

- 1) The function *ST\_CPolyFromText*(CHARACTER LARGE OBJECT) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call function *ST\_CPolyFromText*(CHARACTER LARGE OBJECT):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_CurvePolygon* value. If *awkt* is not producible in the BNF for <curvopolygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_CurvePolygon* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_CPolyFromText*(CHARACTER LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_CPolyFromText*(CHARACTER LARGE OBJECT, INTEGER):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_CurvePolygon* value. If *awkt* is not producible in the BNF for <curvopolygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_CurvePolygon* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.

## 8.2.9 ST\_CPolyFromWKB Functions

### Purpose

Return a specified ST\_CurvePolygon value.

### Definition

```
CREATE FUNCTION ST_CPolyFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_CurvePolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_CurvePolygon)

CREATE FUNCTION ST_CPolyFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_CurvePolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_CurvePolygon)
```

### Definitional Rules

- 1) *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.

### Description

- 1) The function *ST\_CPolyFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 2) For the null-call function *ST\_CPolyFromWKB(BINARY LARGE OBJECT)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_CurvePolygon* value. If *awkb* is not producible in the BNF for <curvепolygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_CurvePolygon* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_CPolyFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_CPolyFromWKB(BINARY LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_CurvePolygon* value. If *awkb* is not producible in the BNF for <curvепolygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_CurvePolygon* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.

## 8.3 ST\_Polygon Type and Routines

### 8.3.1 ST\_Polygon Type

#### Purpose

The ST\_Polygon type is a subtype of the ST\_CurvePolygon type and represents a planar surface whose boundary is defined by linear rings.

#### Definition

```

CREATE TYPE ST_Polygon
  UNDER ST_CurvePolygon
  INSTANTIABLE
  NOT FINAL

CONSTRUCTOR METHOD ST_Polygon
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_Polygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Polygon
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_Polygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Polygon
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_Polygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Polygon
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_Polygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Polygon
  (alinestring ST_LineString)
  RETURNS ST_Polygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

CONSTRUCTOR METHOD ST_Polygon
  (alinesring ST_LineString,
   alinesringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements])
RETURNS ST_Polygon
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Polygon
  (alinesring ST_LineString,
   ansrid INTEGER)
RETURNS ST_Polygon
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Polygon
  (alinesring ST_LineString,
   alinesringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
RETURNS ST_Polygon
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_ExteriorRing()
  RETURNS ST_LineString,

OVERRIDING METHOD ST_ExteriorRing
  (acurve ST_Curve)
  RETURNS ST_Polygon,

OVERRIDING METHOD ST_InteriorRings()
  RETURNS ST_LineString ARRAY[ST_MaxGeometryArrayElements],

OVERRIDING METHOD ST_InteriorRings
  (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_Polygon,

OVERRIDING METHOD ST_InteriorRingN
  (aposition INTEGER)
  RETURNS ST_LineString

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

**Description**

- 1) The *ST\_Polygon* type provides for public use:
  - a) a method *ST\_Polygon*(*CHARACTER LARGE OBJECT*),
  - b) a method *ST\_Polygon*(*CHARACTER LARGE OBJECT, INTEGER*),
  - c) a method *ST\_Polygon*(*BINARY LARGE OBJECT*),
  - d) a method *ST\_Polygon*(*BINARY LARGE OBJECT, INTEGER*),
  - e) a method *ST\_Polygon*(*ST\_LineString*),
  - f) a method *ST\_Polygon*(*ST\_LineString, ST\_LineString ARRAY*),
  - g) a method *ST\_Polygon*(*ST\_LineString, INTEGER*),
  - h) a method *ST\_Polygon*(*ST\_LineString, ST\_LineString ARRAY, INTEGER*),
  - i) an overriding method *ST\_ExteriorRing*() ,
  - j) an overriding method *ST\_ExteriorRing*(*ST\_Curve*),
  - k) an overriding method *ST\_InteriorRings*() ,
  - l) an overriding method *ST\_InteriorRings*(*ST\_Curve ARRAY*),
  - m) an overriding method *ST\_InteriorRingN*(*INTEGER*),
  - n) a function *ST\_PolyFromText*(*CHARACTER LARGE OBJECT*),
  - o) a function *ST\_PolyFromText*(*CHARACTER LARGE OBJECT, INTEGER*),
  - p) a function *ST\_PolyFromWKB*(*BINARY LARGE OBJECT*),
  - q) a function *ST\_PolyFromWKB*(*BINARY LARGE OBJECT, INTEGER*),
  - r) a function *ST\_PolyFromGML*((*CHARACTER LARGE OBJECT*),
  - s) a function *ST\_PolyFromGML*((*CHARACTER LARGE OBJECT, INTEGER*),
  - t) a function *ST\_BdPolyFromText*(*CHARACTER LARGE OBJECT*),
  - u) a function *ST\_BdPolyFromText*(*CHARACTER LARGE OBJECT, INTEGER*),
  - v) a function *ST\_BdPolyFromWKB*(*BINARY LARGE OBJECT*),
  - w) a function *ST\_BdPolyFromWKB*(*BINARY LARGE OBJECT, INTEGER*).
- 2) The *ST\_PrivateExteriorRing* attribute is an *ST\_LineString* value that is a linear ring.
- 3) The *ST\_PrivateInteriorRings* attribute is a collection of *ST\_LineString* values. Each *ST\_LineString* value in the collection is a linear ring.
- 4) The linear ring in the *ST\_PrivateExteriorRing* attribute and the linear rings in the *ST\_PrivateInteriorRings* attribute represent the boundary of the *ST\_Polygon* value.
- 5) An *ST\_Polygon* value returned by the constructor function corresponds to the empty set.

### 8.3.2 ST\_Polygon Methods

#### Purpose

Return an ST\_Polygon value constructed from either the well-known text representation or the well-known binary representation of an ST\_Polygon value, or the specified ST\_LineString values.

#### Definition

```

CREATE CONSTRUCTOR METHOD ST_Polygon
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_Polygon
  FOR ST_Polygon
  RETURN ST_PolyFromText(awkt)

CREATE CONSTRUCTOR METHOD ST_Polygon
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_Polygon
  FOR ST_Polygon
  RETURN ST_PolyFromText(awkt, ansrid)

CREATE CONSTRUCTOR METHOD ST_Polygon
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_Polygon
  FOR ST_Polygon
  RETURN ST_PolyFromWKB(awkb)

CREATE CONSTRUCTOR METHOD ST_Polygon
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_Polygon
  FOR ST_Polygon
  RETURN ST_PolyFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_Polygon
  (alinesring ST_LineString)
  RETURNS ST_Polygon
  FOR ST_Polygon
  RETURN SELF.ST_SRID(0).ST_ExteriorRing(alinesring).
    ST_InteriorRings(CAST(ARRAY[] AS
      ST_LineString ARRAY[ST_MaxGeometryArrayElements]))

CREATE CONSTRUCTOR METHOD ST_Polygon
  (alinesring ST_LineString,
   alinesringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_Polygon
  FOR ST_Polygon
  RETURN SELF.ST_SRID(0).ST_ExteriorRing(alinesring).
    ST_InteriorRings(alinesringarray)

CREATE CONSTRUCTOR METHOD ST_Polygon
  (alinesring ST_LineString,
   ansrid INTEGER)
  RETURNS ST_Polygon
  FOR ST_Polygon
  RETURN SELF.ST_SRID(ansrid).ST_ExteriorRing(alinesring).
    ST_InteriorRings(CAST(ARRAY[] AS
    ST_LineString ARRAY[ST_MaxGeometryArrayElements]))

```

```

CREATE CONSTRUCTOR METHOD ST_Polygon
  (alinesstring ST_LineString,
   alinesstringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
RETURNS ST_Polygon
FOR ST_Polygon
RETURN SELF.ST_SRID(ansrid).ST_ExteriorRing(alinesstring).
      ST_InteriorRings(alinesstringarray)

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

### Description

- 1) The method *ST\_Polygon(CHARACTER LARGE OBJECT)* takes the following input parameter:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call type preserving SQL-invoked constructor method *ST\_Polygon(CHARACTER LARGE OBJECT)*:
  - a) The parameter *awkt* is the well-known text representation of an *ST\_Polygon* value. If *awkt* is not producible in the BNF for <polygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_Polygon* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The method *ST\_Polygon(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call type preserving SQL-invoked constructor method *ST\_Polygon(CHARACTER LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkt* is the well-known text representation of an *ST\_Polygon* value. If *awkt* is not producible in the BNF for <polygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_Polygon* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.
- 5) The method *ST\_Polygon(BINARY LARGE OBJECT)* takes the following input parameter:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 6) For the null-call type preserving SQL-invoked constructor method *ST\_Polygon(BINARY LARGE OBJECT)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_Polygon* value. If *awkb* is not producible in the BNF for <polygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_Polygon* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).

- 7) The method *ST\_Polygon*(*BINARY LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *BINARY LARGE OBJECT* value *awkb*,
  - an *INTEGER* value *ansrid*.
- 8) For the null-call type preserving SQL-invoked constructor method *ST\_Polygon*(*BINARY LARGE OBJECT*, *INTEGER*):
- The parameter *awkb* is the well-known binary representation of an *ST\_Polygon* value. If *awkb* is not producible in the BNF for <polygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_Polygon* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.
- 9) The method *ST\_Polygon*(*ST\_LineString*) takes the following input parameters:
- an *ST\_LineString* value *alinestring*.
- 10) The null-call type preserving SQL-invoked constructor method *ST\_Polygon*(*ST\_LineString*) returns an *ST\_Polygon* value with:
- The spatial reference system identifier set to 0 (zero).
  - Using the method *ST\_ExteriorRing*(*ST\_Curve*), the *ST\_PrivateExteriorRing* attribute set to *alinestring*, the *ST\_PrivateDimension* attribute set to 2, and the *ST\_PrivateCoordinateDimension* attribute set to 2.
  - Using the method *ST\_InteriorRings*(*ST\_Curve ARRAY*), the *ST\_PrivateInteriorRings* attribute set to an empty *ST\_LineString ARRAY* value.
- 11) The method *ST\_Polygon*(*ST\_LineString*, *ST\_LineString ARRAY*) takes the following input parameters:
- an *ST\_LineString* value *alinestring*,
  - an *ST\_LineString ARRAY* value *alinestringarray*.
- 12) The null-call type preserving SQL-invoked constructor method *ST\_Polygon*(*ST\_LineString*, *ST\_LineString ARRAY*) returns an *ST\_Polygon* value with:
- The spatial reference system identifier set to 0 (zero).
  - Using the method *ST\_ExteriorRing*(*ST\_Curve*), the *ST\_PrivateExteriorRing* attribute set to *alinestring*, the *ST\_PrivateDimension* attribute set to 2, and the *ST\_PrivateCoordinateDimension* attribute set to 2.
  - Using the method *ST\_InteriorRings*(*ST\_Curve ARRAY*), the *ST\_PrivateInteriorRings* attribute set to *alinestringarray*.
- 13) The method *ST\_Polygon*(*ST\_LineString*, *INTEGER*) takes the following input parameters:
- an *ST\_LineString* value *alinestring*,
  - an *INTEGER* value *ansrid*.
- 14) The null-call type preserving SQL-invoked constructor method *ST\_Polygon*(*ST\_LineString*, *INTEGER*) returns an *ST\_Polygon* value with:
- The spatial reference system identifier set to *ansrid*.
  - Using the method *ST\_ExteriorRing*(*ST\_Curve*), the *ST\_PrivateExteriorRing* attribute set to *alinestring*, the *ST\_PrivateDimension* attribute set to 2, and the *ST\_PrivateCoordinateDimension* attribute set to 2.
  - Using the method *ST\_InteriorRings*(*ST\_Curve ARRAY*), the *ST\_PrivateInteriorRings* attribute set to an empty *ST\_LineString ARRAY* value.

- 15) The method *ST\_Polygon*(*ST\_LineString*, *ST\_LineString* ARRAY, *INTEGER*) takes the following input parameters:
- a) an *ST\_LineString* value *alinestring*,
  - b) an *ST\_LineString* ARRAY value *alinestringarray*,
  - c) an *INTEGER* value *ansrid*.
- 16) The null-call type preserving SQL-invoked constructor method *ST\_Polygon*(*ST\_LineString*, *ST\_LineString* ARRAY, *INTEGER*) returns an *ST\_Polygon* value with:
- a) The spatial reference system identifier set to *ansrid*.
  - b) Using the method *ST\_ExteriorRing*(*ST\_Curve*), the *ST\_PrivateExteriorRing* attribute set to *alinestring*, the *ST\_PrivateDimension* attribute set to 2, and the *ST\_PrivateCoordinateDimension* attribute set to 2.
  - c) Using the method *ST\_InteriorRings*(*ST\_Curve* ARRAY), the *ST\_PrivateInteriorRings* attribute set to *alinestringarray*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 8.3.3 ST\_ExteriorRing Methods

#### Purpose

Observe and mutate the ST\_PrivateExteriorRing attribute of an ST\_Polygon value.

#### Definition

```
CREATE METHOD ST_ExteriorRing()
  RETURNS ST_LineString
  FOR ST_Polygon
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      CAST(SELF.ST_PrivateExteriorRing AS ST_LineString)
  END

CREATE METHOD ST_ExteriorRing
  (acurve ST_Curve)
  RETURNS ST_Polygon
  FOR ST_Polygon
  BEGIN
    -- If acurve is not an ST_LineString, then raise an exception
    IF acurve IS NOT OF (ST_LineString) THEN
      SIGNAL SQLSTATE '2FF12'
      SET MESSAGE_TEXT = 'curve value is not a linestring value';
    END IF;
    RETURN (SELF AS ST_CurvePolygon).ST_ExteriorRing(acurve);
  END
```

#### Description

- 1) The method *ST\_ExteriorRing()* has no input parameters.
- 2) For the null-call method *ST\_ExteriorRing()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the *ST\_PrivateExteriorRing* attribute of SELF.
- 3) The method *ST\_ExteriorRing(ST\_Curve)* takes the following input parameters:
  - a) an *ST\_Curve* value *alinststring*.
- 4) For the type preserving method *ST\_ExteriorRing(ST\_Curve)*:
 

Case:

  - a) If *acurve* is not an *ST\_LineString* value, then an exception condition is raised: *SQL/MM Spatial exception – curve value is not a linestring value*.
  - b) Otherwise, return an *ST\_Polygon* value as a result of the value expression: *(SELF AS ST\_CurvePolygon).ST\_ExteriorRing(acurve)*

### 8.3.4 ST\_InteriorRings Methods

#### Purpose

Observe and mutate the *ST\_PrivateInteriorRings* attribute of an *ST\_Polygon* value.

#### Definition

```

CREATE METHOD ST_InteriorRings()
  RETURNS ST_LineString ARRAY[ST_MaxGeometryArrayElements]
  FOR ST_Polygon
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      CAST(SELF.ST_PrivateInteriorRings AS
        ST_LineString ARRAY[ST_MaxGeometryArrayElements])
  END

CREATE METHOD ST_InteriorRings
  (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_Polygon
  FOR ST_Polygon
  BEGIN
    DECLARE counter INTEGER;

    -- Check if curves are ST_LineString values
    SET counter = 1;
    WHILE counter <= CARDINALITY(acurvearray) DO
      -- If the current element is not an ST_LineString value, then
      -- raise an exception.
      IF acurvearray[counter] IS NOT OF (ST_LineString) THEN
        SIGNAL SQLSTATE '2FF08'
          SET MESSAGE_TEXT = 'element is not an ST_LineString type';
      END IF;
      SET counter = counter + 1;
    END WHILE;
    RETURN (SELF AS ST_CurvePolygon).ST_InteriorRings(acurvearray);
  END

```

#### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.

#### Description

- 1) The method *ST\_InteriorRings()* has no input parameters.
- 2) For the null-call method *ST\_InteriorRings()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the *ST\_PrivateInteriorRings* attribute of SELF.
- 3) The method *ST\_InteriorRings(ST\_Curve ARRAY)* takes the following input parameters:
  - a) an *ST\_Curve* ARRAY value *acurvearray*.

4) For the type preserving method *ST\_InteriorRings*(*ST\_Curve ARRAY*):

Case:

- a) If any element in *acurvearray* is not an *ST\_LineString* value, then an exception condition is raised: *SQL/MM Spatial exception – element is not an ST\_LineString type*.
- b) Otherwise, return an *ST\_Polygon* value as a result of the value expression: (SELF AS *ST\_CurvePolygon*).*ST\_InteriorRings*(*acurvearray*).

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 8.3.5 ST\_InteriorRingN Method

#### Purpose

Return the specified element in the ST\_PrivateInteriorRings attribute of an ST\_Polygon value.

#### Definition

```
CREATE METHOD ST_InteriorRingN
  (aposition INTEGER)
  RETURNS ST_LineString
  FOR ST_Polygon
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      TREAT((SELF AS ST_CurvePolygon).ST_InteriorRingN(aposition) AS
        ST_LineString)
  END
```

#### Description

- 1) The method *ST\_InteriorRingN(INTEGER)* takes the following input parameters:
  - a) an INTEGER value *aposition*.
- 2) For the null-call method *ST\_InteriorRingN(INTEGER)*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return an *ST\_LineString* value as a result of the value expression *TREAT((SELF AS ST\_CurvePolygon).ST\_InteriorRingN(aposition) AS ST\_LineString)*.

### 8.3.6 ST\_PolyFromText Functions

#### Purpose

Return a specified ST\_Polygon value.

#### Definition

```
CREATE FUNCTION ST_PolyFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_Polygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt) AS ST_Polygon)

CREATE FUNCTION ST_PolyFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_Polygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt, ansrid) AS ST_Polygon)
```

#### 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.

#### Description

- 1) The function *ST\_PolyFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call function *ST\_PolyFromText(CHARACTER LARGE OBJECT)*:
  - a) The parameter *awkt* is the well-known text representation of an *ST\_Polygon* value. If *awkt* is not producible in the BNF for <polygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_Polygon* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_PolyFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_PolyFromText(CHARACTER LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkt* is the well-known text representation of an *ST\_Polygon* value. If *awkt* is not producible in the BNF for <polygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_Polygon* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.

### 8.3.7 ST\_PolyFromWKB Functions

#### Purpose

Return a specified ST\_Polygon value.

#### Definition

```
CREATE FUNCTION ST_PolyFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_Polygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_Polygon)

CREATE FUNCTION ST_PolyFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_Polygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_Polygon)
```

#### Definitional Rules

- 1) *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.

#### Description

- 1) The function *ST\_PolyFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 2) For the null-call function *ST\_PolyFromWKB(BINARY LARGE OBJECT)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_Polygon* value. If *awkb* is not producible in the BNF for <polygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_Polygon* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_PolyFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_PolyFromWKB(BINARY LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_Polygon* value. If *awkb* is not producible in the BNF for <polygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_Polygon* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.

### 8.3.8 ST\_PolyFromGML Functions

#### Purpose

Return a specified ST\_Polygon value.

#### Definition

```
CREATE FUNCTION ST_PolyFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
  RETURNS ST_Polygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromGML(agml) AS ST_Polygon)

CREATE FUNCTION ST_PolyFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
   ansrid INTEGER)
  RETURNS ST_Polygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromGML(agml, ansrid) AS ST_Polygon)
```

#### Definitional Rules

- 1) *ST\_MaxGeometryAsGML* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the GML representation of an *ST\_Geometry* value.

#### Description

- 1) The function *ST\_PolyFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*.
- 2) For the null-call function *ST\_PolyFromGML(CHARACTER LARGE OBJECT)*:
  - a) If the parameter *agml* does not contain a Polygon XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_Polygon* value represented by *agml* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_PolyFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_PolyFromGML(CHARACTER LARGE OBJECT, INTEGER)*:
  - a) If the parameter *agml* does not contain a Polygon XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_Polygon* value represented by *agml* with the spatial reference system identifier set to *ansrid*.

### 8.3.9 ST\_BdPolyFromText Functions

#### Purpose

Return a specified ST\_Polygon value.

#### Definition

```

CREATE FUNCTION ST_BdPolyFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_Polygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN ST_BdPolyFromText(awkt, 0)

CREATE FUNCTION ST_BdPolyFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_Polygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  BEGIN
    --
    -- See Description
    --
  END

```

#### Description

- 1) The function *ST\_BdPolyFromText*(*CHARACTER LARGE OBJECT*) takes the following input parameters:
  - a) a *CHARACTER LARGE OBJECT* value *awkt*.
- 2) For the null-call function *ST\_BdPolyFromText*(*CHARACTER LARGE OBJECT*) returns an *ST\_Polygon* value as the result of the value expression: *ST\_BdPolyFromText*(*awkt*, 0).
- 3) The function *ST\_BdPolyFromText*(*CHARACTER LARGE OBJECT*, *INTEGER*) takes the following input parameters:
  - a) a *CHARACTER LARGE OBJECT* value *awkt*.
  - b) an *INTEGER* value *ansrid*.
- 4) For the null-call function *ST\_BdPolyFromText*(*CHARACTER LARGE OBJECT*, *INTEGER*):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_MultiLineString* value. If *awkt* is not producible in the BNF for <multilinestring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Use *ST\_MLineFromText*(*CHARACTER LARGE OBJECT*) to transform *awkt* to an *ST\_MultiLineString* value, *AMLS*.
  - c) If any *ST\_LineString* value in *AMLS* is not a linear ring, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
  - d) Using an implementation-dependent algorithm, an exterior linear ring, *ELR*, and an array of zero or more interior rings, *AILR*, are determined from the array of linear rings in *AMLS*.

- e) Return an *ST\_Polygon* value with:
  - i) The spatial reference system identifier set to *ansrid*.
  - ii) Using the method *ST\_ExteriorRing(ST\_LineString)*, the *ST\_PrivateExteriorRing* attribute set to *ELR*.
  - iii) Using the method *ST\_InteriorRings(ST\_LineString ARRAY)*, the *ST\_PrivateInteriorRings* attribute set to *AILR*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 8.3.10 ST\_BdPolyFromWKB Functions

#### Purpose

Return a specified ST\_Polygon value.

#### Definition

```
CREATE FUNCTION ST_BdPolyFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_Polygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN ST_BdPolyFromWKB(awkb, 0)

CREATE FUNCTION ST_BdPolyFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_Polygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  BEGIN
    --
    -- See Description
    --
  END
```

#### Description

- 1) The function *ST\_BdPolyFromWKB*(*BINARY LARGE OBJECT*) takes the following input parameters:
  - a) a *BINARY LARGE OBJECT* value *awkb*.
- 2) For the null-call function *ST\_BdPolyFromWKB*(*BINARY LARGE OBJECT*) returns an *ST\_Polygon* value as the result of the value expression: *ST\_BdPolyFromText*(*awkt*, 0).
- 3) The function *ST\_BdPolyFromWKB*(*BINARY LARGE OBJECT*, *INTEGER*) takes the following input parameters:
  - a) a *BINARY LARGE OBJECT* value *awkb*.
  - b) an *INTEGER* value *ansrid*.
- 4) For the null-call function *ST\_BdPolyFromWKB*(*BINARY LARGE OBJECT*, *INTEGER*):
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_MultiLineString* value. If *awkb* is not producible in the BNF for <multilineestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Use *ST\_MLineFromWKB*(*BINARY LARGE OBJECT*) to transform *awkb* to an *ST\_MultiLineString* value, *AMLS*.
  - c) If any *ST\_LineString* value in *AMLS* is not a linear ring, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
  - d) Using an implementation-dependent algorithm, an exterior linear ring, *ELR*, and an array of zero or more interior rings, *AILR*, are determined from the array of linear rings in *AMLS*.

- e) Return an *ST\_Polygon* value with:
  - i) The spatial reference system identifier set to *ansrid*.
  - ii) Using the method *ST\_ExteriorRing(ST\_LineString)*, the *ST\_PrivateExteriorRing* attribute set to *ELR*.
  - iii) Using the method *ST\_InteriorRings(ST\_LineString ARRAY)*, the *ST\_PrivateInteriorRings* attribute set to *AILR*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 9 Geometry Collection Types

### 9.1 ST\_GeomCollection Type and Routines

#### 9.1.1 ST\_GeomCollection Type

##### Purpose

The ST\_GeomCollection type is a subtype of ST\_Geometry and represents a collection of zero or more ST\_Geometry values.

##### Definition

```

CREATE TYPE ST_GeomCollection
  UNDER ST_Geometry
  AS (
    ST_PrivateGeometries ST_Geometry
    ARRAY[ST_MaxGeometryArrayElements] DEFAULT ARRAY[]
  )
  INSTANTIABLE
  NOT FINAL

  CONSTRUCTOR METHOD ST_GeomCollection
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_GeomCollection
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

  CONSTRUCTOR METHOD ST_GeomCollection
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
     ansrid INTEGER)
  RETURNS ST_GeomCollection
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

  CONSTRUCTOR METHOD ST_GeomCollection
    (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_GeomCollection
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

  CONSTRUCTOR METHOD ST_GeomCollection
    (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
     ansrid INTEGER)
  RETURNS ST_GeomCollection
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

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

CONSTRUCTOR METHOD ST_GeomCollection
  (ageometryarray ST_Geometry
   ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_GeomCollection
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

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

CONSTRUCTOR METHOD ST_GeomCollection
  (ageometryarray ST_Geometry
   ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
  RETURNS ST_GeomCollection
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_Geometries()
  RETURNS ST_Geometry ARRAY[ST_MaxGeometryArrayElements]
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_Geometries
  (ageometryarray ST_Geometry
   ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_GeomCollection
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

```

```

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

```

```

METHOD ST_GeometryN
  (aPosition INTEGER)
  RETURNS ST_Geometry
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.
- 4) The attribute *ST\_PrivateGeometries* is not for public use. There are no GRANT statements granting EXECUTE privilege on the observer or mutator method for *ST\_PrivateGeometries*.

### Description

- 1) The *ST\_GeomCollection* type provides for public use:
  - a) a method *ST\_GeomCollection*(CHARACTER LARGE OBJECT),
  - b) a method *ST\_GeomCollection*(CHARACTER LARGE OBJECT, INTEGER),
  - c) a method *ST\_GeomCollection*(BINARY LARGE OBJECT),
  - d) a method *ST\_GeomCollection*(BINARY LARGE OBJECT, INTEGER),
  - e) a method *ST\_GeomCollection*(*ST\_Geometry*),
  - f) a method *ST\_GeomCollection*(*ST\_Geometry* ARRAY),
  - g) a method *ST\_GeomCollection*(*ST\_Geometry*, INTEGER),
  - h) a method *ST\_GeomCollection*(*ST\_Geometry* ARRAY, INTEGER),
  - i) a method *ST\_Geometries*(),
  - j) a method *ST\_Geometries*(*ST\_Geometry* ARRAY),
  - k) a method *ST\_NumGeometries*(),
  - l) a method *ST\_GeometryN*(INTEGER),
  - m) a function *ST\_GeomCollFromTxt*(CHARACTER LARGE OBJECT),
  - n) a function *ST\_GeomCollFromTxt*(CHARACTER LARGE OBJECT, INTEGER),
  - o) a function *ST\_GeomCollFromWKB*(BINARY LARGE OBJECT),
  - p) a function *ST\_GeomCollFromWKB*(BINARY LARGE OBJECT, INTEGER),
  - q) a function *ST\_GeomCollFromGML*(CHARACTER LARGE OBJECT),
  - r) a function *ST\_GeomCollFromGML*(CHARACTER LARGE OBJECT, INTEGER),
- 2) The *ST\_PrivateGeometries* attribute contains the collection of *ST\_Geometry* values.
- 3) The *ST\_PrivateGeometries* attribute shall not be the null value. The elements in the *ST\_PrivateGeometries* attribute shall not be the null value.

- 4) The coordinate dimension of an *ST\_GeomCollection* value is 2.
- 5) The dimension of an *ST\_GeomCollection* value is the maximum dimension value of all the *ST\_Geometry* values in the *ST\_PrivateGeometries* attribute.
- 6) An *ST\_GeomCollection* value returned by the constructor function corresponds to the empty set.
- 7) An *ST\_GeomCollection* value with no elements in the *ST\_PrivateGeometries* attribute corresponds to the empty set.
- 8) Subtypes of *ST\_GeomCollection* may restrict membership based on dimension and may place other constraints such as the degree that the elements spatially intersect between *ST\_Geometry* values.
- 9) An *ST\_GeomCollection* value is well formed only if all of the *ST\_Geometry* values in *ST\_PrivateGeometries* attribute are well formed.
- 10) All the *ST\_Geometry* values in the *ST\_PrivateGeometries* attribute shall be in the same spatial reference system as the *ST\_GeomCollection* value.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 9.1.2 ST\_GeomCollection Methods

### Purpose

Return an ST\_GeomCollection value constructed from either the well-known text representation or the well-known binary representation of an ST\_GeomCollection value, or the specified ST\_Geometry values.

### Definition

```

CREATE CONSTRUCTOR METHOD ST_GeomCollection
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_GeomCollection
  FOR ST_GeomCollection
  RETURN ST_GeomCollFromTxt(awkt)

CREATE CONSTRUCTOR METHOD ST_GeomCollection
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_GeomCollection
  FOR ST_GeomCollection
  RETURN ST_GeomCollFromTxt(awkt, ansrid)

CREATE CONSTRUCTOR METHOD ST_GeomCollection
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_GeomCollection
  FOR ST_GeomCollection
  RETURN ST_GeomCollFromWKB(awkb)

CREATE CONSTRUCTOR METHOD ST_GeomCollection
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_GeomCollection
  FOR ST_GeomCollection
  RETURN ST_GeomCollFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_GeomCollection
  (ageometry ST_Geometry)
  RETURNS ST_GeomCollection
  FOR ST_GeomCollection
  RETURN SELF.ST_SRID(ageometry.ST_SRID()).
    ST_Geometries(ARRAY[ageometry])

CREATE CONSTRUCTOR METHOD ST_GeomCollection
  (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_GeomCollection
  FOR ST_GeomCollection
  RETURN SELF.ST_SRID(ST_CheckSRID(ageometryarray)).
    ST_Geometries(ageometryarray)

CREATE CONSTRUCTOR METHOD ST_GeomCollection
  (ageometry ST_Geometry,
   ansrid INTEGER)
  RETURNS ST_GeomCollection
  FOR ST_GeomCollection
  RETURN SELF.ST_SRID(ansrid).ST_Geometries(ARRAY[ageometry])

CREATE CONSTRUCTOR METHOD ST_GeomCollection
  (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
  RETURNS ST_GeomCollection
  FOR ST_GeomCollection
  RETURN SELF.ST_SRID(ansrid).ST_Geometries(ageometryarray)

```

**Definitional Rules**

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

**Description**

- 1) The method *ST\_GeomCollection*(CHARACTER LARGE OBJECT) takes the following input parameter:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call type preserving SQL-invoked constructor method *ST\_GeomCollection*(CHARACTER LARGE OBJECT):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_GeomCollection* value. If *awkt* is not producible in the BNF for <collection text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_GeomCollection* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The method *ST\_GeomCollection*(CHARACTER LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call type preserving SQL-invoked constructor method *ST\_GeomCollection*(CHARACTER LARGE OBJECT, INTEGER):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_GeomCollection* value. If *awkt* is not producible in the BNF for <collection text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_GeomCollection* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.
- 5) The method *ST\_GeomCollection*(BINARY LARGE OBJECT) takes the following input parameter:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 6) For the null-call type preserving SQL-invoked constructor method *ST\_GeomCollection*(BINARY LARGE OBJECT):
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_GeomCollection* value. If *awkb* is not producible in the BNF for <collection binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_GeomCollection* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 7) The method *ST\_GeomCollection*(BINARY LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.

- 8) For the null-call type preserving SQL-invoked constructor method *ST\_GeomCollection*(*BINARY LARGE OBJECT*, *INTEGER*):
- The parameter *awkb* is the well-known binary representation of an *ST\_GeomCollection* value. If *awkb* is not producible in the BNF for <collection binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_GeomCollection* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.
- 9) The method *ST\_GeomCollection*(*ST\_Geometry*) takes the following input parameters:
- an *ST\_Geometry* value *ageometry*.
- 10) The null-call type preserving SQL-invoked constructor method *ST\_GeomCollection*(*ST\_Geometry*) returns an *ST\_GeomCollection* value with:
- The spatial reference system identifier set to the spatial reference system identifier of *ageometry*.
  - Using the method *ST\_Geometries*(*ST\_Geometry* ARRAY), the *ST\_PrivateGeometries* attribute set to ARRAY[*ageometry*], the *ST\_PrivateDimension* attribute set to *ST\_MaxDimension*(ARRAY[*ageometry*]), and the *ST\_PrivateCoordinateDimension* attribute set to 2.
- 11) The method *ST\_GeomCollection*(*ST\_Geometry* ARRAY) takes the following input parameters:
- an *ST\_Geometry* ARRAY value *ageometryarray*.
- 12) The null-call type preserving SQL-invoked constructor method *ST\_GeomCollection*(*ST\_Geometry* ARRAY) returns an *ST\_GeomCollection* value with:
- The spatial reference system identifier set to the value expression: *ST\_CheckSRID*(*ageometryarray*).
  - Using the method *ST\_Geometries*(*ST\_Geometry* ARRAY), the *ST\_PrivateGeometries* attribute set to *ageometryarray*, the *ST\_PrivateDimension* attribute set to *ST\_MaxDimension*(ARRAY[*ageometry*]), and the *ST\_PrivateCoordinateDimension* attribute set to 2.
- 13) The method *ST\_GeomCollection*(*ST\_Geometry*, *INTEGER*) takes the following input parameters:
- an *ST\_Geometry* value *ageometry*,
  - an *INTEGER* value *ansrid*.
- 14) The null-call type preserving SQL-invoked constructor method *ST\_GeomCollection*(*ST\_Geometry*, *INTEGER*) returns an *ST\_GeomCollection* value with:
- The spatial reference system identifier set to *ansrid*.
  - Using the method *ST\_Geometries*(*ST\_Geometry* ARRAY), the *ST\_PrivateGeometries* attribute set to ARRAY[*ageometry*], the *ST\_PrivateDimension* attribute set to *ST\_MaxDimension*(ARRAY[*ageometry*]), and the *ST\_PrivateCoordinateDimension* attribute set to 2.
- 15) The method *ST\_GeomCollection*(*ST\_Geometry* ARRAY, *INTEGER*) takes the following input parameters:
- an *ST\_Geometry* ARRAY value *ageometryarray*,
  - an *INTEGER* value *ansrid*.
- 16) The null-call type preserving SQL-invoked constructor method *ST\_GeomCollection*(*ST\_Geometry* ARRAY, *INTEGER*) returns an *ST\_GeomCollection* value with:
- The spatial reference system identifier set to *ansrid*.
  - Using the method *ST\_Geometries*(*ST\_Geometry* ARRAY), the *ST\_PrivateGeometries* attribute set to *ageometryarray*, the *ST\_PrivateDimension* attribute set to *ST\_MaxDimension*(ARRAY[*ageometry*]), and the *ST\_PrivateCoordinateDimension* attribute set to 2.

### 9.1.3 ST\_Geometries Methods

#### Purpose

Observe and mutate the `ST_PrivateGeometries` attribute of an `ST_GeomCollection` value.

#### Definition

```
CREATE METHOD ST_Geometries ()
  RETURNS ST_Geometry ARRAY[ST_MaxGeometryArrayElements]
  FOR ST_GeomCollection
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      SELF.ST_PrivateGeometries
  END

CREATE METHOD ST_Geometries
  (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_GeomCollection
  FOR ST_GeomCollection
  BEGIN
    -- If ageometryarray is the null value or contains null elements,
    -- then raise an exception.
    CALL ST_CheckNulls(ageometryarray);
    -- If SELF is the null value, then return the null value.
    IF SELF IS NULL THEN
      RETURN CAST (NULL AS ST_GeomCollection);
    END IF;
    -- Check that there are no mixed spatial reference
    -- systems between SELF and ageometryarray.
    IF SELF.ST_SRID() <> ST_CheckSRID(ageometryarray) THEN
      SIGNAL SQLSTATE '2FF10'
      SET MESSAGE_TEXT = 'mixed spatial reference systems';
    END IF;
    RETURN
      SELF.ST_PrivateDimension(ST_MaxDimension(ageometryarray)).
      ST_PrivateCoordinateDimension(2).
      ST_PrivateGeometries(ageometryarray);
  END
```

#### Definitional Rules

- 1) `ST_MaxGeometryArrayElements` is the implementation-defined maximum cardinality of an array of `ST_Geometry` values.

#### Description

- 1) The method `ST_Geometries()` has no input parameters.
- 2) For the null-call method `ST_Geometries()`:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the value of the `ST_PrivateGeometries` attribute.
- 3) The method `ST_Geometries(ST_Geometry ARRAY)` takes the following input parameters:
  - a) an `ST_Geometry` ARRAY value `ageometryarray`.

- 4) For the type preserving method *ST\_Geometries(ST\_Geometry ARRAY)*:
- a) Call the procedure *ST\_CheckNulls(ST\_Geometry ARRAY)* to check if *ageometryarray* is the null value or contains null elements.
  - b) Case:
    - i) If SELF is the null value, then return the null value.
    - ii) If the spatial reference system of SELF is not equal to *ST\_CheckSRID(ageometryarray)*, then an exception condition is raised: *SQL/MM Spatial exception – mixed spatial reference systems*.
    - iii) Otherwise, return an *ST\_GeomCollection* value with:
      - 1) The dimension set to *ST\_MaxDimension(ageometryarray)*.
      - 2) The coordinate dimension set to 2.
      - 3) The *ST\_PrivateGeometries* attribute set to *ageometryarray*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

#### 9.1.4 ST\_NumGeometries Method

##### Purpose

Return the cardinality of the ST\_PrivateGeometries attribute of an ST\_GeomCollection value.

##### Definition

```
CREATE METHOD ST_NumGeometries()  
  RETURNS INTEGER  
  FOR ST_GeomCollection  
  RETURN  
  CASE  
    WHEN SELF.ST_IsEmpty() = 1 THEN  
      NULL  
    ELSE  
      CARDINALITY(SELF.ST_PrivateGeometries)  
  END
```

##### Description

- 1) The method *ST\_NumGeometries()* has no input parameters.
- 2) For the null-call method *ST\_NumGeometries()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the cardinality of the *ST\_PrivateGeometries* attribute.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 9.1.5 ST\_GeometryN Method

#### Purpose

Return the specified ST\_Geometry value in the ST\_PrivateGeometries attribute of an ST\_GeomCollection value.

#### Definition

```
CREATE METHOD ST_GeometryN
(aposition INTEGER)
RETURNS ST_Geometry
FOR ST_GeomCollection
BEGIN
  IF SELF.ST_IsEmpty() = 1 THEN
    RETURN CAST (NULL AS ST_Geometry);
  END IF;
  IF aposition < 1 OR
  aposition > CARDINALITY(SELF.ST_PrivateGeometries) THEN
    BEGIN
      SIGNAL SQLSTATE '01F01'
      SET MESSAGE_TEXT = 'invalid position';
      RETURN CAST (NULL AS ST_Geometry);
    END;
  END IF;
  RETURN SELF.ST_PrivateGeometries[aposition];
END
```

#### Description

- 1) The method *ST\_GeometryN(INTEGER)* takes the following input parameters:
  - a) an INTEGER value *aposition*.
- 2) For the null-call method *ST\_GeometryN(INTEGER)*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) If *aposition* is less than one or greater than the cardinality of the *ST\_PrivateGeometries* attribute, then:
    - i) A completion condition is raised: *SQL/MM Spatial warning – invalid position*.
    - ii) Return the null value.
  - c) Otherwise, return the element of the *ST\_PrivateGeometries* attribute at position *aposition*.

### 9.1.6 ST\_GeomCollFromTxt Functions

#### Purpose

Return a specified ST\_GeomCollection value.

#### Definition

```
CREATE FUNCTION ST_GeomCollFromTxt
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_GeomCollection
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt) AS ST_GeomCollection)

CREATE FUNCTION ST_GeomCollFromTxt
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_GeomCollection
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt, ansrid) AS ST_GeomCollection)
```

#### 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.

#### Description

- 1) The function *ST\_GeomCollFromTxt(CHARACTER LARGE OBJECT)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call function *ST\_GeomCollFromTxt(CHARACTER LARGE OBJECT)*:
  - a) The parameter *awkt* is the well-known text representation of an *ST\_GeomCollection* value. If *awkt* is not producible in the BNF for <geometrycollection text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_GeomCollection* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_GeomCollFromTxt(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_GeomCollFromTxt(CHARACTER LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkt* is the well-known text representation of an *ST\_GeomCollection* value. If *awkt* is not producible in the BNF for <geometrycollection text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_GeomCollection* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.

### 9.1.7 ST\_GeomCollFromWKB Functions

#### Purpose

Return a specified ST\_GeomCollection value.

#### Definition

```
CREATE FUNCTION ST_GeomCollFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_GeomCollection
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_GeomCollection)

CREATE FUNCTION ST_GeomCollFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_GeomCollection
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_GeomCollection)
```

#### Definitional Rules

- 1) *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.

#### Description

- 1) The function *ST\_GeomCollFromWKB*(BINARY LARGE OBJECT) takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 2) For the null-call function *ST\_GeomCollFromWKB*(BINARY LARGE OBJECT):
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_GeomCollection* value. If *awkb* is not producible in the BNF for <geometrycollection binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_GeomCollection* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_GeomCollFromWKB*(BINARY LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_GeomCollFromWKB*(BINARY LARGE OBJECT, INTEGER):
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_GeomCollection* value. If *awkb* is not producible in the BNF for <geometrycollection binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_GeomCollection* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.

### 9.1.8 ST\_GeomCollFromGML Functions

#### Purpose

Return a specified ST\_GeomCollection value.

#### Definition

```
CREATE FUNCTION ST_GeomCollFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
  RETURNS ST_GeomCollection
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromGML(agml) AS ST_GeomCollection)

CREATE FUNCTION ST_GeomCollFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
   ansrid INTEGER)
  RETURNS ST_GeomCollection
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromGML(agml, ansrid) AS ST_GeomCollection)
```

#### Definitional Rules

- 1) *ST\_MaxGeometryAsGML* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the GML representation of an *ST\_Geometry* value.

#### Description

- 1) The function *ST\_GeomCollFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*.
- 2) For the null-call function *ST\_GeomCollFromGML(CHARACTER LARGE OBJECT)*:
  - a) If the parameter *agml* does not contain a GeometryCollection XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_GeomCollection* value represented by *agml* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_GeomCollFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_GeomCollFromGML(CHARACTER LARGE OBJECT, INTEGER)*:
  - a) If the parameter *agml* does not contain a GeometryCollection XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_GeomCollection* value represented by *agml* with the spatial reference system identifier set to *ansrid*.

## 9.2 ST\_MultiPoint Type and Routines

### 9.2.1 ST\_MultiPoint Type

#### Purpose

The ST\_MultiPoint type is a 0-dimensional geometry and represents a collection of ST\_Point values.

#### Definition

```

CREATE TYPE ST_MultiPoint
  UNDER ST_GeomCollection
  INSTANTIABLE
  NOT FINAL

CONSTRUCTOR METHOD ST_MultiPoint
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiPoint
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiPoint
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_MultiPoint
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiPoint
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiPoint
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiPoint
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_MultiPoint
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiPoint
  (apointarray ST_Point
   ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiPoint
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

CONSTRUCTOR METHOD ST_MultiPoint
  (apointarray ST_Point
   ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
RETURNS ST_MultiPoint
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_Geometries()
  RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements],

OVERRIDING METHOD ST_Geometries
  (ageometryarray ST_Geometry
   ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiPoint

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

### Description

- 1) The *ST\_MultiPoint* type provides for public use:
  - a) a method *ST\_MultiPoint*(CHARACTER LARGE OBJECT),
  - b) a method *ST\_MultiPoint*(CHARACTER LARGE OBJECT, INTEGER),
  - c) a method *ST\_MultiPoint*(BINARY LARGE OBJECT),
  - d) a method *ST\_MultiPoint*(BINARY LARGE OBJECT, INTEGER),
  - e) a method *ST\_MultiPoint*(*ST\_Point* ARRAY),
  - f) a method *ST\_MultiPoint*(*ST\_Point* ARRAY, INTEGER),
  - g) an overriding method *ST\_Geometries*(),
  - h) an overriding method *ST\_Geometries*(*ST\_Geometry* ARRAY),
  - i) a function *ST\_MPointFromText*(CHARACTER LARGE OBJECT),
  - j) a function *ST\_MPointFromText*(CHARACTER LARGE OBJECT, INTEGER),
  - k) a function *ST\_MPointFromWKB*(BINARY LARGE OBJECT),
  - l) a function *ST\_MPointFromWKB*(BINARY LARGE OBJECT, INTEGER).
  - m) a function *ST\_MPointFromGML*(CHARACTER LARGE OBJECT),
  - n) a function *ST\_MPointFromGML*(CHARACTER LARGE OBJECT, INTEGER),
- 2) The dimension of an *ST\_MultiPoint* value is 0 (zero).
- 3) The elements of the *ST\_PrivateGeometries* attribute are restricted to *ST\_Point* values.
- 4) The *ST\_Point* values in the *ST\_PrivateGeometries* attribute are not connected or ordered.
- 5) If no two *ST\_Point* values in the *ST\_MultiPoint* value are equal, then the *ST\_MultiPoint* value is simple.

- 6) The boundary of an *ST\_MultiPoint* value is the empty set.
- 7) An *ST\_MultiPoint* value is well formed only if and only if all of the *ST\_Point* values in the *ST\_PrivateGeometries* attribute are well formed.
- 8) An *ST\_MultiPoint* value returned by the constructor function corresponds to the empty set.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 9.2.2 ST\_MultiPoint Methods

### Purpose

Return an *ST\_MultiPoint* value constructed from either the well-known text representation or the well-known binary representation of an *ST\_MultiPoint* value, or the specified *ST\_Point* values.

### Definition

```
CREATE CONSTRUCTOR METHOD ST_MultiPoint
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiPoint
  FOR ST_MultiPoint
  RETURN ST_MPointFromText(awkt)

CREATE CONSTRUCTOR METHOD ST_MultiPoint
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_MultiPoint
  FOR ST_MultiPoint
  RETURN ST_MPointFromText(awkt, ansrid)

CREATE CONSTRUCTOR METHOD ST_MultiPoint
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiPoint
  FOR ST_MultiPoint
  RETURN ST_MPointFromWKB(awkb)

CREATE CONSTRUCTOR METHOD ST_MultiPoint
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_MultiPoint
  FOR ST_MultiPoint
  RETURN ST_MPointFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_MultiPoint
  (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiPoint
  FOR ST_MultiPoint
  RETURN SELF.ST_SRID(0).ST_Geometries(apointarray)

CREATE CONSTRUCTOR METHOD ST_MultiPoint
  (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
  RETURNS ST_MultiPoint
  FOR ST_MultiPoint
  RETURN SELF.ST_SRID(ansrid).ST_Geometries(apointarray)
```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

### Description

- 1) The method *ST\_MultiPoint(CHARACTER LARGE OBJECT)* takes the following input parameter:
  - a) a CHARACTER LARGE OBJECT value *awkt*.

- 2) For the null-call type preserving SQL-invoked constructor method *ST\_MultiPoint*(*CHARACTER LARGE OBJECT*):
- The parameter *awkt* is the well-known text representation of an *ST\_MultiPoint* value. If *awkt* is not producible in the BNF for <multipoint text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - Return an *ST\_MultiPoint* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The method *ST\_MultiPoint*(*CHARACTER LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *CHARACTER LARGE OBJECT* value *awkt*,
  - an *INTEGER* value *ansrid*.
- 4) For the null-call type preserving SQL-invoked constructor method *ST\_MultiPoint*(*CHARACTER LARGE OBJECT*, *INTEGER*):
- The parameter *awkt* is the well-known text representation of an *ST\_MultiPoint* value. If *awkt* is not producible in the BNF for <multipoint text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - Return an *ST\_MultiPoint* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.
- 5) The method *ST\_MultiPoint*(*BINARY LARGE OBJECT*) takes the following input parameter:
- a *BINARY LARGE OBJECT* value *awkb*.
- 6) For the null-call type preserving SQL-invoked constructor method *ST\_MultiPoint*(*BINARY LARGE OBJECT*):
- The parameter *awkb* is the well-known binary representation of an *ST\_MultiPoint* value. If *awkb* is not producible in the BNF for <multipoint binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_MultiPoint* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 7) The method *ST\_MultiPoint*(*BINARY LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *BINARY LARGE OBJECT* value *awkb*,
  - an *INTEGER* value *ansrid*.
- 8) For the null-call type preserving SQL-invoked constructor method *ST\_MultiPoint*(*BINARY LARGE OBJECT*, *INTEGER*):
- The parameter *awkb* is the well-known binary representation of an *ST\_MultiPoint* value. If *awkb* is not producible in the BNF for <multipoint binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_MultiPoint* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.
- 9) The method *ST\_MultiPoint*(*ST\_Point ARRAY*) takes the following input parameters:
- an *ST\_Point ARRAY* value *apointarray*.
- 10) The null-call type preserving SQL-invoked constructor method *ST\_MultiPoint*(*ST\_Point ARRAY*) returns an *ST\_MultiPoint* value with:
- The spatial reference system identifier set to 0 (zero).

- b) Using the method *ST\_Geometries*(*ST\_Geometry* ARRAY), the *ST\_PrivateGeometries* attribute set to *apointarray*, the *ST\_PrivateDimension* attribute set to 0 (zero), and the *ST\_PrivateCoordinateDimension* attribute set to 2.
- 11) The method *ST\_MultiPoint*(*ST\_Point* ARRAY, *INTEGER*) takes the following input parameters:
- a) an *ST\_Point* ARRAY value *apointarray*,
  - b) an *INTEGER* value *ansrid*.
- 12) The null-call type preserving SQL-invoked constructor method *ST\_MultiPoint*(*ST\_Point* ARRAY, *INTEGER*) returns an *ST\_MultiPoint* value with:
- a) The spatial reference system identifier set to *ansrid*.
  - b) Using the method *ST\_Geometries*(*ST\_Geometry* ARRAY), the *ST\_PrivateGeometries* attribute set to *apointarray*, the *ST\_PrivateDimension* attribute set to 0 (zero), and the *ST\_PrivateCoordinateDimension* attribute set to 2.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 9.2.3 ST\_Geometries Methods

#### Purpose

Observe and mutate the *ST\_PrivateGeometries* attribute of an *ST\_MultiPoint* value.

#### Definition

```

CREATE METHOD ST_Geometries()
  RETURNS ST_Point ARRAY[ST_MaxGeometryArrayElements]
  FOR ST_MultiPoint
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      CAST(SELF.ST_PrivateGeometries AS
            ST_Point ARRAY[ST_MaxGeometryArrayElements])
  END

CREATE METHOD ST_Geometries_
(ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiPoint
  FOR ST_MultiPoint
  BEGIN
    DECLARE apointarray ST_Point
      ARRAY[ST_MaxGeometryArrayElements];

    -- Cast ageometryarray to an ST_Point ARRAY
    SET apointarray = CAST(ageometryarray AS
      ST_Point ARRAY[ST_MaxGeometryArrayElements]);
    -- If SELF is the null value, then return the null value.
    -- Otherwise, return an ST_MultiPoint value containing
    -- apointarray.
    RETURN
    CASE
      WHEN SELF IS NULL THEN
        NULL
      ELSE
        (SELF AS ST_GeomCollection).
          ST_Geometries(apointarray)
    END;
  END

```

#### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.

#### Description

- 1) The method *ST\_Geometries()* has no input parameters.
- 2) For the null-call method *ST\_Geometries()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the value of the *ST\_PrivateGeometries* attribute as an *ST\_Point* ARRAY.
- 3) The method *ST\_Geometries(ST\_Geometry ARRAY)* takes the following input parameters:
  - a) an *ST\_Geometry* ARRAY value *ageometryarray*.

- 4) For the type preserving method *ST\_Geometries(ST\_Geometry ARRAY)*:
- a) Let *APOINTARRAY* be the result of casting *ageometryarray* to an *ST\_Point ARRAY* value (implicitly using *ST\_ToPointAry(ST\_Geometry ARRAY)*).
  - b) Case:
    - i) If *SELF* is the null value, then return the null value.
    - ii) Otherwise, using the method *ST\_Geometries(ST\_Geometry ARRAY)* for type *ST\_GeomCollection*, return an *ST\_MultiPoint* value with:
      - 1) The dimension set to 0 (zero).
      - 2) The coordinate dimension set to 2.
      - 3) The *ST\_PrivateGeometries* attribute set to *APOINTARRAY*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 9.2.4 ST\_MPointFromText Functions

### Purpose

Return a specified ST\_MultiPoint value.

### Definition

```
CREATE FUNCTION ST_MPointFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiPoint
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt) AS ST_MultiPoint)

CREATE FUNCTION ST_MPointFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_MultiPoint
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt, ansrid) AS ST_MultiPoint)
```

### 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.

### Description

- 1) The function *ST\_MPointFromText*(CHARACTER LARGE OBJECT) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call function *ST\_MPointFromText*(CHARACTER LARGE OBJECT):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_MultiPoint* value. If *awkt* is not producible in the BNF for <multipoint text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_MultiPoint* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_MPointFromText*(CHARACTER LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_MPointFromText*(CHARACTER LARGE OBJECT, INTEGER):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_MultiPoint* value. If *awkt* is not producible in the BNF for <multipoint text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_MultiPoint* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.

### 9.2.5 ST\_MPointFromWKB Functions

#### Purpose

Return a specified ST\_MultiPoint value.

#### Definition

```
CREATE FUNCTION ST_MPointFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiPoint
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_MultiPoint)

CREATE FUNCTION ST_MPointFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_MultiPoint
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_MultiPoint)
```

#### Definitional Rules

- 1) *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.

#### Description

- 1) The function *ST\_MPointFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 2) For the null-call function *ST\_MPointFromWKB(BINARY LARGE OBJECT)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_MultiPoint* value. If *awkb* is not producible in the BNF for <multipoint binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_MultiPoint* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_MPointFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_MPointFromWKB(BINARY LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_MultiPoint* value. If *awkb* is not producible in the BNF for <multipoint binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_MultiPoint* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.

## 9.2.6 ST\_MPointFromGML Functions

### Purpose

Return a specified ST\_MultiPoint value.

### Definition

```
CREATE FUNCTION ST_MPointFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
  RETURNS ST_MultiPoint
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromGML(agml) AS ST_MultiPoint)

CREATE FUNCTION ST_MPointFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
   ansrid INTEGER)
  RETURNS ST_MultiPoint
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromGML(agml, ansrid) AS ST_MultiPoint)
```

### Definitional Rules

- 1) *ST\_MaxGeometryAsGML* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the GML representation of an *ST\_Geometry* value.

### Description

- 1) The function *ST\_MPointFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*.
- 2) For the null-call function *ST\_MPointFromGML(CHARACTER LARGE OBJECT)*:
  - a) If the parameter *agml* does not contain a MultiPoint XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_MultiPoint* value represented by *agml* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_MPointFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_MPointFromGML(CHARACTER LARGE OBJECT, INTEGER)*:
  - a) If the parameter *agml* does not contain a MultiPoint XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_MultiPoint* value represented by *agml* with the spatial reference system identifier set to *ansrid*.

### 9.3 ST\_MultiCurve Type and Routines

#### 9.3.1 ST\_MultiCurve Type

##### Purpose

The ST\_MultiCurve type is a 1-dimensional geometry and represents a collection of ST\_Curve.

##### Definition

```

CREATE TYPE ST_MultiCurve
  UNDER ST_GeomCollection
  INSTANTIABLE
  NOT FINAL

CONSTRUCTOR METHOD ST_MultiCurve
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiCurve
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiCurve
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_MultiCurve
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiCurve
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiCurve
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiCurve
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_MultiCurve
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiCurve
  (acurvearray ST_Curve
   ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiCurve
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

CONSTRUCTOR METHOD ST_MultiCurve
  (acurvearray ST_Curve
   ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
RETURNS ST_MultiCurve
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

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

METHOD ST_Length()
  RETURNS DOUBLE PRECISION
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

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

OVERRIDING METHOD ST_Geometries()
  RETURNS ST_Curve ARRAY[ST_MaxGeometryArrayElements],

OVERRIDING METHOD ST_Geometries
  (ageometryarray ST_Geometry
   ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiCurve

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

### Description

- 1) The *ST\_MultiCurve* type provides for public use:
  - a) a method *ST\_MultiCurve*(CHARACTER LARGE OBJECT),
  - b) a method *ST\_MultiCurve*(CHARACTER LARGE OBJECT, INTEGER),
  - c) a method *ST\_MultiCurve*(BINARY LARGE OBJECT),
  - d) a method *ST\_MultiCurve*(BINARY LARGE OBJECT, INTEGER),
  - e) a method *ST\_MultiCurve*(*ST\_Curve* ARRAY),

- f) a method *ST\_MultiCurve*(*ST\_Curve* ARRAY, INTEGER),
  - g) a method *ST\_IsClosed*(),
  - h) a method *ST\_Length*(),
  - i) a method *ST\_Length*(CHARACTER VARYING),
  - j) an overriding method *ST\_Geometries*(),
  - k) an overriding method *ST\_Geometries*(*ST\_Geometry* ARRAY),
  - l) a function *ST\_MCurveFromText*(CHARACTER LARGE OBJECT),
  - m) a function *ST\_MCurveFromText*(CHARACTER LARGE OBJECT, INTEGER),
  - n) a function *ST\_MCurveFromWKB*(BINARY LARGE OBJECT),
  - o) a function *ST\_MCurveFromWKB*(BINARY LARGE OBJECT, INTEGER).
- 2) The dimension of an *ST\_MultiCurve* value is 1 (one).
  - 3) The elements of an *ST\_MultiCurve* value are *ST\_Curve* values.
  - 4) If all of the elements in the *ST\_PrivateGeometries* attribute are simple and any two elements only spatially intersect at the boundaries of both elements, then an *ST\_MultiCurve* is simple.
  - 5) The boundary of an *ST\_MultiCurve* value is obtained by applying the mod 2 union rule: an *ST\_Point* value is in the boundary of an *ST\_MultiCurve* if it is in the boundaries of an odd number of elements of the *ST\_MultiCurve*.
  - 6) An *ST\_MultiCurve* value is closed if all of its elements are closed. The boundary of a closed *ST\_MultiCurve* is the empty set.
  - 7) An *ST\_MultiCurve* value is defined as topologically closed.
  - 8) An *ST\_MultiCurve* value is well formed only if all of the *ST\_Curve* values in the *ST\_PrivateGeometries* attribute are well formed.
  - 9) An *ST\_MultiCurve* value returned by the constructor function corresponds to the empty set.

### 9.3.2 ST\_MultiCurve Methods

Return an ST\_MultiCurve value constructed from either the well-known text representation or the well-known binary representation of an ST\_MultiCurve value, or the specified ST\_Curve values.

#### Definition

```

CREATE CONSTRUCTOR METHOD ST_MultiCurve
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiCurve
  FOR ST_MultiCurve
  RETURN ST_MCurveFromText(awkt)

CREATE CONSTRUCTOR METHOD ST_MultiCurve
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_MultiCurve
  FOR ST_MultiCurve
  RETURN ST_MCurveFromText(awkt, ansrid)

CREATE CONSTRUCTOR METHOD ST_MultiCurve
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiCurve
  FOR ST_MultiCurve
  RETURN ST_MCurveFromWKB(awkb)

CREATE CONSTRUCTOR METHOD ST_MultiCurve
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_MultiCurve
  FOR ST_MultiCurve
  RETURN ST_MCurveFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_MultiCurve
  (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiCurve
  FOR ST_MultiCurve
  RETURN SELF.ST_SRID(0).ST_Geometries(acurvearray)

CREATE CONSTRUCTOR METHOD ST_MultiCurve
  (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
  RETURNS ST_MultiCurve
  FOR ST_MultiCurve
  RETURN SELF.ST_SRID(ansrid).ST_Geometries(acurvearray)

```

#### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

#### Description

- 1) The method *ST\_MultiCurve*(*CHARACTER LARGE OBJECT*) takes the following input parameter:
  - a) a CHARACTER LARGE OBJECT value *awkt*.

- 2) For the null-call type preserving SQL-invoked constructor method *ST\_MultiCurve*(*CHARACTER LARGE OBJECT*):
- The parameter *awkt* is the well-known text representation of an *ST\_MultiCurve* value. If *awkt* is not producible in the BNF for <multicurve text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - Return an *ST\_MultiCurve* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The method *ST\_MultiCurve*(*CHARACTER LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *CHARACTER LARGE OBJECT* value *awkt*,
  - an *INTEGER* value *ansrid*.
- 4) For the null-call type preserving SQL-invoked constructor method *ST\_MultiCurve*(*CHARACTER LARGE OBJECT*, *INTEGER*):
- The parameter *awkt* is the well-known text representation of an *ST\_MultiCurve* value. If *awkt* is not producible in the BNF for <multicurve text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - Return an *ST\_MultiCurve* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.
- 5) The method *ST\_MultiCurve*(*BINARY LARGE OBJECT*) takes the following input parameter:
- a *BINARY LARGE OBJECT* value *awkb*.
- 6) For the null-call type preserving SQL-invoked constructor method *ST\_MultiCurve*(*BINARY LARGE OBJECT*):
- The parameter *awkb* is the well-known binary representation of an *ST\_MultiCurve* value. If *awkb* is not producible in the BNF for <multicurve binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_MultiCurve* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 7) The method *ST\_MultiCurve*(*BINARY LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *BINARY LARGE OBJECT* value *awkb*,
  - an *INTEGER* value *ansrid*.
- 8) For the null-call type preserving SQL-invoked constructor method *ST\_MultiCurve*(*BINARY LARGE OBJECT*, *INTEGER*):
- The parameter *awkb* is the well-known binary representation of an *ST\_MultiCurve* value. If *awkb* is not producible in the BNF for <multicurve binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_MultiCurve* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.
- 9) The method *ST\_MultiCurve*(*ST\_Curve ARRAY*) takes the following input parameters:
- a *ST\_Curve ARRAY* value *acurvearray*.
- 10) The null-call type preserving SQL-invoked constructor method *ST\_MultiCurve*(*ST\_Curve ARRAY*) returns an *ST\_MultiCurve* value with:
- The spatial reference system identification set to 0 (zero).

- b) Using the method *ST\_Geometries*(*ST\_Geometry* ARRAY), the attribute *ST\_PrivateGeometries* set to *acurvearray*, the *ST\_PrivateDimension* attribute set to 1 (one), and the *ST\_PrivateCoordinateDimension* attribute set to 2.
- 11) The method *ST\_MultiCurve*(*ST\_Curve* ARRAY, *INTEGER*) takes the following input parameters:
- a) an *ST\_Curve* ARRAY value *acurvearray*,
  - b) an *INTEGER* value *ansrid*.
- 12) The null-call type preserving SQL-invoked constructor method *ST\_MultiCurve*(*ST\_Curve* ARRAY, *INTEGER*) returns an *ST\_MultiCurve* value with:
- a) The spatial reference system identification set to *ansrid*.
  - b) Using the method *ST\_Geometries*(*ST\_Geometry* ARRAY), the attribute *ST\_PrivateGeometries* set to *acurvearray*, the *ST\_PrivateDimension* attribute set to 1 (one), and the *ST\_PrivateCoordinateDimension* attribute set to 2.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 9.3.3 ST\_IsClosed Method

#### Purpose

Test if an ST\_MultiCurve value is closed.

#### Definition

```
CREATE METHOD ST_IsClosed()  
  RETURNS INTEGER  
  FOR ST_MultiCurve  
  RETURN  
  CASE  
    WHEN SELF.ST_IsEmpty = 1 THEN  
      0  
    ELSE  
      SELF.ST_Boundary().ST_IsEmpty()  
  END
```

#### Description

- 1) The method *ST\_IsClosed()* has no input parameters.
- 2) The null-call method *ST\_IsClosed()* returns:

Case:

- a) If SELF is the empty set, then 0 (zero).
- b) If the boundary of the *ST\_MultiCurve* value is the empty set, then 1 (one).
- c) Otherwise, 0 (zero).

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 9.3.4 ST\_Length Methods

#### Purpose

Return the length measurement of an ST\_MultiCurve value.

#### Definition

```

CREATE METHOD ST_Length()
  RETURNS DOUBLE PRECISION
  FOR ST_MultiCurve
  BEGIN
    DECLARE length DOUBLE PRECISION;
    DECLARE counter INTEGER;

    IF SELF.ST_IsEmpty() = 1 THEN
      RETURN CAST (NULL AS DOUBLE PRECISION);
    END IF;
    SET length = 0.0;
    SET counter = 1;
    WHILE counter <= SELF.ST_NumGeometries() DO
      SET length = length + SELF.ST_GeometryN(counter).ST_Length();
      SET counter = counter + 1;
    END WHILE;
    RETURN length;
  END

CREATE METHOD ST_Length
  (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
  RETURNS DOUBLE PRECISION
  FOR ST_MultiCurve
  BEGIN
    DECLARE length DOUBLE PRECISION;
    DECLARE counter INTEGER;

    IF SELF.ST_IsEmpty() = 1 THEN
      RETURN CAST (NULL AS DOUBLE PRECISION);
    END IF;
    SET length = 0.0;
    SET counter = 1;
    WHILE counter <= SELF.ST_NumGeometries() DO
      SET length = length +
        SELF.ST_GeometryN(counter).ST_Length(aunit);
      SET counter = counter + 1;
    END WHILE;
    RETURN length;
  END

```

#### Definitional Rules

- 1) *ST\_MaxUnitNameLength* is the implementation-defined maximum length for the character representation of a unit indication.

#### Description

- 1) The method *ST\_Length()* has no input parameters.
- 2) For the null-call method *ST\_Length()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the sum of the *ST\_Length()* values of each element in the *ST\_PrivateGeometries* attribute of SELF.

3) Case:

- a) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST\_Length()* is in the linear unit of measure identified by <linear unit>.
- b) Otherwise, the value returned by *ST\_Length()* is in an implementation-defined unit of measure.

4) The method *ST\_Length(CHARACTER VARYING)* takes the following input parameter:

- a) a CHARACTER VARYING value *ainit*.

5) For the null-call method *ST\_Length(CHARACTER VARYING)*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the sum of the *ST\_Length(ainit)* values of each element in the *ST\_PrivateGeometries* attribute of SELF.

6) The value returned by *ST\_Length(CHARACTER VARYING)* is in the units indicated by *ainit*.

7) The values for *ainit* shall be a supported <unit name>.

8) The value for *ainit* is a supported <unit name> if and only if the value of *ainit* is equal to the value of the UNIT\_NAME column of one of the rows where the value of the UNIT\_TYPE column is equal to 'LINEAR' in the ST\_UNITS\_OF\_MEASURE view.

9) If the unit specified by *ainit* is not supported by the implementation to compute the sum of the *ST\_Length(ainit)* values of each element in *ST\_PrivateGeometries* attribute of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 9.3.5 ST\_Geometries Methods

#### Purpose

Observe and mutate the `ST_PrivateGeometries` attribute of an `ST_MultiCurve` value.

#### Definition

```

CREATE METHOD ST_Geometries()
  RETURNS ST_Curve ARRAY[ST_MaxGeometryArrayElements]
  FOR ST_MultiCurve
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      CAST(SELF.ST_PrivateGeometries AS ST_Curve
        ARRAY[ST_MaxGeometryArrayElements])
  END

CREATE METHOD ST_Geometries_
(ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiCurve
  FOR ST_MultiCurve
  BEGIN
    DECLARE acurvearray ST_Curve
      ARRAY[ST_MaxGeometryArrayElements];

    -- Cast ageometryarray to an ST_Curve ARRAY
    SET acurvearray = CAST(ageometryarray AS
      ST_Curve ARRAY[ST_MaxGeometryArrayElements]);
    -- If SELF is the null value, then return the null value. Otherwise,
    -- return an ST_MultiCurve value containing acurvearray.
    RETURN
    CASE
      WHEN SELF IS NULL THEN
        NULL
      ELSE
        (SELF AS ST_GeomCollection).
          ST_Geometries(acurvearray)
    END;
  END

```

#### Definitional Rules

- 1) `ST_MaxGeometryArrayElements` is the implementation-defined maximum cardinality of an array of `ST_Geometry` values.

#### Description

- 1) The method `ST_Geometries()` has no input parameters.
- 2) For the null-call method `ST_Geometries()`:
 

Case:

  - a) If `SELF` is an empty set, then return the null value.
  - b) Otherwise, return the value of the `ST_PrivateGeometries` attribute as an `ST_Curve ARRAY`.
- 3) The method `ST_Geometries(ST_Geometry ARRAY)` takes the following input parameters:
  - a) an `ST_Geometry ARRAY` value `ageometryarray`.
- 4) For the type preserving method `ST_Geometries(ST_Geometry ARRAY)`:
  - a) Let `ACURVEARRAY` be the result of casting `ageometryarray` to an `ST_Curve ARRAY` value (implicitly using `ST_ToCurveAry(ST_Geometry ARRAY)`).

b) Case:

- i) If SELF is the null value, then return the null value.
- ii) Otherwise, return an *ST\_MultiCurve* value with:
  - 1) The dimension set to 1 (one).
  - 2) Using the method *ST\_Geometries(ST\_Geometry ARRAY)* for type *ST\_GeomCollection*, the *ST\_PrivateGeometries* attribute set to *ACURVEARRAY*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 9.3.6 ST\_MCurveFromText Functions

#### Purpose

Return a specified ST\_MultiCurve value.

#### Definition

```
CREATE FUNCTION ST_MCurveFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiCurve
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt) AS ST_MultiCurve)

CREATE FUNCTION ST_MCurveFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_MultiCurve
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt, ansrid) AS ST_MultiCurve)
```

#### 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.

#### Description

- 1) The function *ST\_MCurveFromText*(CHARACTER LARGE OBJECT) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call function *ST\_MCurveFromText*(CHARACTER LARGE OBJECT):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_MultiCurve* value. If *awkt* is not producible in the BNF for <multicurve text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_MultiCurve* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_MCurveFromText*(CHARACTER LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_MCurveFromText*(CHARACTER LARGE OBJECT, INTEGER):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_MultiCurve* value. If *awkt* is not producible in the BNF for <multicurve text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_MultiCurve* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.

### 9.3.7 ST\_MCurveFromWKB Functions

#### Purpose

Return a specified ST\_MultiCurve value.

#### Definition

```
CREATE FUNCTION ST_MCurveFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiCurve
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_MultiCurve)

CREATE FUNCTION ST_MCurveFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_MultiCurve
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_MultiCurve)
```

#### Definitional Rules

- 1) *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.

#### Description

- 1) The function *ST\_MCurveFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 2) For the null-call function *ST\_MCurveFromWKB(BINARY LARGE OBJECT)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_MultiCurve* value. If *awkb* is not producible in the BNF for <multicurve binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_MultiCurve* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_MCurveFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_MCurveFromWKB(BINARY LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_MultiCurve* value. If *awkb* is not producible in the BNF for <multicurve binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_MultiCurve* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.

## 9.4 ST\_MultiLineString Type and Routines

### 9.4.1 ST\_MultiLineString Type

#### Purpose

The ST\_MultiLineString type is a subtype of the ST\_MultiCurve and represents a collection of ST\_LineString values.

#### Definition

```
CREATE TYPE ST_MultiLineString
  UNDER ST_MultiCurve
  INSTANTIABLE
  NOT FINAL

  CONSTRUCTOR METHOD ST_MultiLineString
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiLineString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

  CONSTRUCTOR METHOD ST_MultiLineString
    (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
     ansrid INTEGER)
  RETURNS ST_MultiLineString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

  CONSTRUCTOR METHOD ST_MultiLineString
    (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiLineString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

  CONSTRUCTOR METHOD ST_MultiLineString
    (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
     ansrid INTEGER)
  RETURNS ST_MultiLineString
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,
```

```

CONSTRUCTOR METHOD ST_MultiLineString
  (alinesringarray ST_LineString
   ARRAY[ST_MaxGeometryArrayElements])
RETURNS ST_MultiLineString
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiLineString
  (alinesringarray ST_LineString
   ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
RETURNS ST_MultiLineString
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_Geometries()
  RETURNS ST_LineString ARRAY[ST_MaxGeometryArrayElements],

OVERRIDING METHOD ST_Geometries
  (ageometryarray ST_Geometry
   ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiLineString

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

### Description

- 1) The *ST\_MultiLineString* type provides for public use:
  - a) a method *ST\_MultiLineString*(CHARACTER LARGE OBJECT),
  - b) a method *ST\_MultiLineString*(CHARACTER LARGE OBJECT, INTEGER),
  - c) a method *ST\_MultiLineString*(BINARY LARGE OBJECT),
  - d) a method *ST\_MultiLineString*(BINARY LARGE OBJECT, INTEGER),
  - e) a method *ST\_MultiLineString*(*ST\_LineString* ARRAY),
  - f) a method *ST\_MultiLineString*(*ST\_LineString* ARRAY, INTEGER),
  - g) an overriding method *ST\_Geometries*(),
  - h) an overriding method *ST\_Geometries*(*ST\_Geometry* ARRAY),
  - i) a function *ST\_MLineFromText*(CHARACTER LARGE OBJECT),
  - j) a function *ST\_MLineFromText*(CHARACTER LARGE OBJECT, INTEGER),
  - k) a function *ST\_MLineFromWKB*(BINARY LARGE OBJECT),
  - l) a function *ST\_MLineFromWKB*(BINARY LARGE OBJECT, INTEGER).
  - m) a function *ST\_MLineFromGML*(CHARACTER LARGE OBJECT),

- n) a function *ST\_MLineFromGML*(*CHARACTER LARGE OBJECT*, *INTEGER*),
- 2) The elements of the *ST\_PrivateGeometries* attribute are restricted to *ST\_LineString* values.
  - 3) An *ST\_MultiLineString* value is well formed only if and only if all of the *ST\_LineString* values in the *ST\_PrivateGeometries* attribute are well formed.
  - 4) An *ST\_MultiLineString* value returned by the constructor function corresponds to the empty set.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 9.4.2 ST\_MultiLineString Methods

### Purpose

Return an ST\_MultiLineString value constructed from either the well-known text representation or the well-known binary representation of an ST\_MultiLineString value, or the specified ST\_LineString values.

### Definition

```
CREATE CONSTRUCTOR METHOD ST_MultiLineString
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiLineString
  FOR ST_MultiLineString
  RETURN ST_MLineFromText(awkt)

CREATE CONSTRUCTOR METHOD ST_MultiLineString
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_MultiLineString
  FOR ST_MultiLineString
  RETURN ST_MLineFromText(awkt, ansrid)

CREATE CONSTRUCTOR METHOD ST_MultiLineString
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiLineString
  FOR ST_MultiLineString
  RETURN ST_MLineFromWKB(awkb)

CREATE CONSTRUCTOR METHOD ST_MultiLineString
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_MultiLineString
  FOR ST_MultiLineString
  RETURN ST_MLineFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_MultiLineString
  (alinesringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiLineString
  FOR ST_MultiLineString
  RETURN SELF.ST_SRID(0).ST_Geometries(alinesringarray)

CREATE CONSTRUCTOR METHOD ST_MultiLineString
  (alinesringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
  RETURNS ST_MultiLineString
  FOR ST_MultiLineString
  RETURN SELF.ST_SRID(ansrid).ST_Geometries(alinesringarray)
```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

### Description

- 1) The method *ST\_MultiLineString(CHARACTER LARGE OBJECT)* takes the following input parameter:
  - a) a CHARACTER LARGE OBJECT value *awkt*.

- 2) For the null-call type preserving SQL-invoked constructor method *ST\_MultiLineString*(*CHARACTER LARGE OBJECT*):
- The parameter *awkt* is the well-known text representation of an *ST\_MultiLineString* value. If *awkt* is not producible in the BNF for <multilinestring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - Return an *ST\_MultiLineString* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The method *ST\_MultiLineString*(*CHARACTER LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *CHARACTER LARGE OBJECT* value *awkt*,
  - an *INTEGER* value *ansrid*.
- 4) For the null-call type preserving SQL-invoked constructor method *ST\_MultiLineString*(*CHARACTER LARGE OBJECT*, *INTEGER*):
- The parameter *awkt* is the well-known text representation of an *ST\_MultiLineString* value. If *awkt* is not producible in the BNF for <multilinestring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - Return an *ST\_MultiLineString* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.
- 5) The method *ST\_MultiLineString*(*BINARY LARGE OBJECT*) takes the following input parameter:
- a *BINARY LARGE OBJECT* value *awkb*.
- 6) For the null-call type preserving SQL-invoked constructor method *ST\_MultiLineString*(*BINARY LARGE OBJECT*):
- The parameter *awkb* is the well-known binary representation of an *ST\_MultiLineString* value. If *awkb* is not producible in the BNF for <multilinestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_MultiLineString* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 7) The method *ST\_MultiLineString*(*BINARY LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *BINARY LARGE OBJECT* value *awkb*,
  - an *INTEGER* value *ansrid*.
- 8) For the null-call type preserving SQL-invoked constructor method *ST\_MultiLineString*(*BINARY LARGE OBJECT*, *INTEGER*):
- The parameter *awkb* is the well-known binary representation of an *ST\_MultiLineString* value. If *awkb* is not producible in the BNF for <multilinestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_MultiLineString* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.
- 9) The method *ST\_MultiLineString*(*ST\_LineString ARRAY*) takes the following input parameters:
- an *ST\_LineString ARRAY* value *alinesringarray*.
- 10) The null-call type preserving SQL-invoked constructor method *ST\_MultiLineString*(*ST\_LineString ARRAY*) returns an *ST\_MultiLineString* value with:
- The spatial reference system identifier set to 0 (zero).

- b) Using the method *ST\_Geometries(ST\_Geometry ARRAY)*, the *ST\_PrivateGeometries* attribute set to *alinesstringarray*, the *ST\_PrivateDimension* attribute set to 1 (one), and the *ST\_PrivateCoordinateDimension* attribute set to 2.
- 11) The method *ST\_MultiLineString(ST\_LineString ARRAY, INTEGER)* takes the following input parameters:
- a) an *ST\_LineString* ARRAY value *alinesstringarray*,
  - b) an INTEGER value *ansrid*.
- 12) The null-call type preserving SQL-invoked constructor method *ST\_MultiLineString(ST\_LineString ARRAY, INTEGER)* returns an *ST\_MultiLineString* value with:
- a) The spatial reference system identifier set to *ansrid*.
  - b) Using the method *ST\_Geometries(ST\_Geometry ARRAY)*, the *ST\_PrivateGeometries* attribute set to *alinesstringarray*, the *ST\_PrivateDimension* attribute set to 1 (one), and the *ST\_PrivateCoordinateDimension* attribute set to 2.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 9.4.3 ST\_Geometries Methods

#### Purpose

Observe and mutate the *ST\_PrivateGeometries* attribute of an *ST\_MultiLineString* value.

#### Definition

```

CREATE METHOD ST_Geometries()
  RETURNS ST_LineString ARRAY[ST_MaxGeometryArrayElements]
  FOR ST_MultiLineString
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      CAST(SELF.ST_PrivateGeometries AS
        ST_LineString ARRAY[ST_MaxGeometryArrayElements])
  END

CREATE METHOD ST_Geometries
  (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiLineString
  FOR ST_MultiLineString
  BEGIN
    DECLARE alinestringarray ST_LineString
      ARRAY[ST_MaxGeometryArrayElements];

    -- Cast ageometryarray to an ST_LineString ARRAY
    SET alinestringarray = CAST(ageometryarray AS
      ST_LineString ARRAY[ST_MaxGeometryArrayElements]);
    -- If SELF is the null value, then return the null value. Otherwise,
    -- return an ST_MultiLineString value containing alinestringarray.
    RETURN
    CASE
      WHEN SELF IS NULL THEN
        NULL
      ELSE
        (SELF AS ST_MultiCurve).
          ST_Geometries(alinestringarray)
    END;
  END

```

#### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.

#### Description

- 1) The method *ST\_Geometries()* has no input parameters.
- 2) For the null-call method *ST\_Geometries()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the value of the *ST\_PrivateGeometries* attribute as an *ST\_LineString ARRAY*.
- 3) The method *ST\_Geometries(ST\_Geometry ARRAY)* takes the following input parameters:
  - a) an *ST\_Geometry ARRAY* value *ageometryarray*.
- 4) For the type preserving method *ST\_Geometries(ST\_Geometry ARRAY)*:
  - a) Let *ALINESTRINGARRAY* be the result of casting *ageometryarray* to an *ST\_LineString ARRAY* value (implicitly using *ST\_ToLineStringAry(ST\_Geometry ARRAY)*).

b) Case:

- i) If SELF is the null value, then return the null value.
- ii) Otherwise, return an *ST\_MultiLineString* value with:
  - 1) The dimension set to 1 (one).
  - 2) Using the method *ST\_Geometries(ST\_Geometry ARRAY)* for type *ST\_MultiCurve*, the *ST\_PrivateGeometries* attribute set to *ALINESTRINGARRAY*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

#### 9.4.4 ST\_MLineFromText Functions

##### Purpose

Return a specified ST\_MultiLineString value.

##### Definition

```
CREATE FUNCTION ST_MLineFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiLineString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt) AS ST_MultiLineString)

CREATE FUNCTION ST_MLineFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_MultiLineString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt, ansrid) AS ST_MultiLineString)
```

##### 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.

##### Description

- 1) The function *ST\_MLineFromText*(CHARACTER LARGE OBJECT) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call function *ST\_MLineFromText*(CHARACTER LARGE OBJECT):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_MultiLineString* value. If *awkt* is not producible in the BNF for <multilinestring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_MultiLineString* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_MLineFromText*(CHARACTER LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_MLineFromText*(CHARACTER LARGE OBJECT, INTEGER):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_MultiLineString* value. If *awkt* is not producible in the BNF for <multilinestring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_MultiLineString* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.

### 9.4.5 ST\_MLineFromWKB Functions

#### Purpose

Return a specified ST\_MultiLineString value.

#### Definition

```
CREATE FUNCTION ST_MLineFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiLineString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_MultiLineString)

CREATE FUNCTION ST_MLineFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_MultiLineString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_MultiLineString)
```

#### Definitional Rules

- 1) *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.

#### Description

- 1) The function *ST\_MLineFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 2) For the null-call function *ST\_MLineFromWKB(BINARY LARGE OBJECT)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_MultiLineString* value. If *awkb* is not producible in the BNF for <multilinestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_MultiLineString* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_MLineFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_MLineFromWKB(BINARY LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_MultiLineString* value. If *awkb* is not producible in the BNF for <multilinestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_MultiLineString* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.

### 9.4.6 ST\_MLineFromGML Functions

#### Purpose

Return a specified ST\_MultiLineString value.

#### Definition

```
CREATE FUNCTION ST_MLineFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
  RETURNS ST_MultiLineString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromGML(agml) AS ST_MultiLineString)

CREATE FUNCTION ST_MLineFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
   ansrid INTEGER)
  RETURNS ST_MultiLineString
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromGML(agml, ansrid) AS ST_MultiLineString)
```

#### Definitional Rules

- 1) *ST\_MaxGeometryAsGML* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the GML representation of an *ST\_Geometry* value.

#### Description

- 1) The function *ST\_MLineFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*.
- 2) For the null-call function *ST\_MLineFromGML(CHARACTER LARGE OBJECT)*:
  - a) If the parameter *agml* does not contain a MultiLineString XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_MultiLineString* value represented by *agml* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_MLineFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_MLineFromGML(CHARACTER LARGE OBJECT, INTEGER)*:
  - a) If the parameter *agml* does not contain a MultiLineString XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_MultiLineString* value represented by *agml* with the spatial reference system identifier set to *ansrid*.

## 9.5 ST\_MultiSurface Type and Routines

### 9.5.1 ST\_MultiSurface Type

#### Purpose

The ST\_MultiSurface type is a 2-dimensional geometry and represents a collection of ST\_Surface values.

#### Definition

```
CREATE TYPE ST_MultiSurface
  UNDER ST_GeomCollection
  INSTANTIABLE
  NOT FINAL

CONSTRUCTOR METHOD ST_MultiSurface
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiSurface
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiSurface
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_MultiSurface
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiSurface
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiSurface
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiSurface
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_MultiSurface
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiSurface
  (asurfacearray ST_Surface
   ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiSurface
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,
```

```

CONSTRUCTOR METHOD ST_MultiSurface
  (asurfacearray ST_Surface
   ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
RETURNS ST_MultiSurface
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

METHOD ST_Area()
  RETURNS DOUBLE PRECISION
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

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

METHOD ST_Perimeter()
  RETURNS DOUBLE PRECISION
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

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

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

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

OVERRIDING METHOD ST_Geometries()
  RETURNS ST_Surface ARRAY[ST_MaxGeometryArrayElements],

```

```

OVERRIDING METHOD ST_Geometries
  (ageometryarray ST_Geometry
   ARRAY[ST_MaxGeometryArrayElements])
RETURNS ST_MultiSurface

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.
- 4) *ST\_MaxUnitNameLength* is the implementation-defined maximum length for the character representation of a unit indication.

### Description

- 1) The *ST\_MultiSurface* type provides for public use:
  - a) a method *ST\_MultiSurface*(CHARACTER LARGE OBJECT),
  - b) a method *ST\_MultiSurface*(CHARACTER LARGE OBJECT, INTEGER),
  - c) a method *ST\_MultiSurface*(BINARY LARGE OBJECT),
  - d) a method *ST\_MultiSurface*(BINARY LARGE OBJECT, INTEGER),
  - e) a method *ST\_MultiSurface*(*ST\_Surface* ARRAY),
  - f) a method *ST\_MultiSurface*(*ST\_Surface* ARRAY, INTEGER),
  - g) a method *ST\_Area*(),
  - h) a method *ST\_Area*(CHARACTER VARYING),
  - i) a method *ST\_Perimeter*(),
  - j) a method *ST\_Perimeter*(CHARACTER VARYING),
  - k) a method *ST\_Centroid*(),
  - l) a method *ST\_PointOnSurface*()
  - m) an overriding method *ST\_Geometries*(),
  - n) an overriding method *ST\_Geometries*(*ST\_Geometry* ARRAY),
  - o) a function *ST\_MSurfaceFromTxt*(CHARACTER LARGE OBJECT),
  - p) a function *ST\_MSurfaceFromTxt*(CHARACTER LARGE OBJECT, INTEGER),
  - q) a function *ST\_MSurfaceFromWKB*(BINARY LARGE OBJECT),
  - r) a function *ST\_MSurfaceFromWKB*(BINARY LARGE OBJECT, INTEGER).
- 2) The dimension of an *ST\_MultiSurface* value is 2.
- 3) The interiors of any two *ST\_Surface* values in an *ST\_MultiSurface* shall not spatially intersect. The boundaries of any two elements in the *ST\_MultiSurface* shall, at most, intersect at a finite number of points.
- 4) An *ST\_MultiSurface* value is well formed only if all of the *ST\_Surface* values in the *ST\_PrivateGeometries* attribute are well formed.
- 5) An *ST\_MultiSurface* value returned by the constructor function corresponds to the empty set.

### 9.5.2 ST\_MultiSurface Methods

Return an ST\_MultiSurface value constructed from either the well-known text representation or the well-known binary representation of an ST\_MultiSurface value, or the specified ST\_Surface values.

#### Definition

```

CREATE CONSTRUCTOR METHOD ST_MultiSurface
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiSurface
  FOR ST_MultiSurface
  RETURN ST_MSurfaceFromTxt(awkt)

CREATE CONSTRUCTOR METHOD ST_MultiSurface
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_MultiSurface
  FOR ST_MultiSurface
  RETURN ST_MSurfaceFromTxt(awkt, ansrid)

CREATE CONSTRUCTOR METHOD ST_MultiSurface
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiSurface
  FOR ST_MultiSurface
  RETURN ST_MSurfaceFromWKB(awkb)

CREATE CONSTRUCTOR METHOD ST_MultiSurface
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_MultiSurface
  FOR ST_MultiSurface
  RETURN ST_MSurfaceFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_MultiSurface
  (asurfacearray ST_Surface ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiSurface
  FOR ST_MultiSurface
  RETURN SELF.ST_SRID(0).ST_Geometries(asurfacearray)

CREATE CONSTRUCTOR METHOD ST_MultiSurface
  (asurfacearray ST_Surface ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
  RETURNS ST_MultiSurface
  FOR ST_MultiSurface
  RETURN SELF.ST_SRID(ansrid).ST_Geometries(asurfacearray)

```

#### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

#### Description

- 1) The method *ST\_MultiSurface*(*CHARACTER LARGE OBJECT*) takes the following input parameter:
  - a) a CHARACTER LARGE OBJECT value *awkt*.

- 2) For the null-call type preserving SQL-invoked constructor method *ST\_MultiSurface*(*CHARACTER LARGE OBJECT*):
- The parameter *awkt* is the well-known text representation of an *ST\_MultiSurface* value. If *awkt* is not producible in the BNF for <multisurface text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - Return an *ST\_MultiSurface* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The method *ST\_MultiSurface*(*CHARACTER LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *CHARACTER LARGE OBJECT* value *awkt*,
  - an *INTEGER* value *ansrid*.
- 4) For the null-call type preserving SQL-invoked constructor method *ST\_MultiSurface*(*CHARACTER LARGE OBJECT*, *INTEGER*):
- The parameter *awkt* is the well-known text representation of an *ST\_MultiSurface* value. If *awkt* is not producible in the BNF for <multisurface text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - Return an *ST\_MultiSurface* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.
- 5) The method *ST\_MultiSurface*(*BINARY LARGE OBJECT*) takes the following input parameter:
- a *BINARY LARGE OBJECT* value *awkb*.
- 6) For the null-call type preserving SQL-invoked constructor method *ST\_MultiSurface*(*BINARY LARGE OBJECT*):
- The parameter *awkb* is the well-known binary representation of an *ST\_MultiSurface* value. If *awkb* is not producible in the BNF for <multisurface binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_MultiSurface* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 7) The method *ST\_MultiSurface*(*BINARY LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *BINARY LARGE OBJECT* value *awkb*,
  - an *INTEGER* value *ansrid*.
- 8) For the null-call type preserving SQL-invoked constructor method *ST\_MultiSurface*(*BINARY LARGE OBJECT*, *INTEGER*):
- The parameter *awkb* is the well-known binary representation of an *ST\_MultiSurface* value. If *awkb* is not producible in the BNF for <multisurface binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_MultiSurface* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.
- 9) The method *ST\_MultiSurface*(*ST\_Surface ARRAY*) takes the following input parameters:
- an *ST\_Surface ARRAY* value *asurfacearray*.
- 10) The null-call type preserving SQL-invoked constructor method *ST\_MultiSurface*(*ST\_Surface ARRAY*) returns an *ST\_MultiSurface* value with:
- The spatial reference system identification set to 0 (zero),

- b) Using the method *ST\_Geometries(ST\_Geometry ARRAY)*, the attribute *ST\_PrivateGeometries* set to *asurfacearray*, the *ST\_PrivateDimension* attribute set to 2, and the *ST\_PrivateCoordinateDimension* attribute set to 2.
- 11) The method *ST\_MultiSurface(ST\_Surface ARRAY, INTEGER)* takes the following input parameters:
- a) an *ST\_Surface ARRAY* value *asurfacearray*,
  - b) an *INTEGER* value *ansrid*.
- 12) The null-call type preserving SQL-invoked constructor method *ST\_MultiSurface(ST\_Surface ARRAY, INTEGER)* returns an *ST\_MultiSurface* value with:
- a) The spatial reference system identification set to *ansrid*.
  - b) Using the method *ST\_Geometries(ST\_Geometry ARRAY)*, the attribute *ST\_PrivateGeometries* set to *asurfacearray*, the *ST\_PrivateDimension* attribute set to 2, and the *ST\_PrivateCoordinateDimension* attribute set to 2.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 9.5.3 ST\_Area Methods

#### Purpose

Return the area measurement of an *ST\_MultiSurface* value.

#### Definition

```

CREATE METHOD ST_Area()
  RETURNS DOUBLE PRECISION
  FOR ST_MultiSurface
  BEGIN
    DECLARE area DOUBLE PRECISION;
    DECLARE counter INTEGER;

    IF SELF.ST_IsEmpty() = 1 THEN
      RETURN CAST (NULL AS DOUBLE PRECISION);
    END IF;
    SET area = 0.0;
    SET counter = 1;
    WHILE counter <= SELF.ST_NumGeometries() DO
      SET area = area + SELF.ST_GeometryN(counter).ST_Area();
      SET counter = counter + 1;
    END WHILE;
    RETURN area;
  END

CREATE METHOD ST_Area
  (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
  RETURNS DOUBLE PRECISION
  FOR ST_MultiSurface
  BEGIN
    DECLARE area DOUBLE PRECISION;
    DECLARE counter INTEGER;

    IF SELF.ST_IsEmpty() = 1 THEN
      RETURN CAST (NULL AS DOUBLE PRECISION);
    END IF;
    SET area = 0.0;
    SET counter = 1;
    WHILE counter <= SELF.ST_NumGeometries() DO
      SET area = area + SELF.ST_GeometryN(counter).ST_Area(aunit);
      SET counter = counter + 1;
    END WHILE;
    RETURN area;
  END

```

#### Definitional Rules

- 1) *ST\_MaxUnitNameLength* is the implementation-defined maximum length for the character representation of a unit indication.

#### Description

- 1) The method *ST\_Area()* has no input parameters.
- 2) For the null-call method *ST\_Area()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the sum of the *ST\_Area* values of the elements in the *ST\_PrivateGeometries* attribute of SELF.

- 3) Case:
- a) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST\_Area()* is in the linear unit of measure identified by <linear unit> squared.
  - b) Otherwise, the value returned by *ST\_Area()* is in an implementation-defined unit of measure.
- 4) The method *ST\_Area(CHARACTER VARYING)* takes the following input parameter:
- a) a CHARACTER VARYING value *unit*.
- 5) For the null-call method *ST\_Area(CHARACTER VARYING)*:
- Case:
- a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the sum of the *ST\_Area(unit)* values of each element in the *ST\_PrivateGeometries* attribute of SELF.
- 6) The value returned by *ST\_Area(CHARACTER VARYING)* is in the units indicated by *unit*.
- 7) The values for *unit* shall be a supported <unit name>.
- 8) The value for *unit* is a supported <unit name> if and only if the value of *unit* is equal to the value of the UNIT\_NAME column of one of the rows where the value of the UNIT\_TYPE column is equal to 'LINEAR' in the ST\_UNITS\_OF\_MEASURE view.
- 9) If the unit specified by *unit* is not supported by the implementation to compute sum of the *ST\_Area(unit)* values of each element in the *ST\_PrivateGeometries* attribute of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

#### 9.5.4 ST\_Perimeter Methods

##### Purpose

Return the length measurement of the boundary of an ST\_MultiSurface value.

##### Definition

```

CREATE METHOD ST_Perimeter()
  RETURNS DOUBLE PRECISION
  FOR ST_MultiSurface
  BEGIN
    DECLARE perimeter DOUBLE PRECISION;
    DECLARE counter INTEGER;

    IF SELF.ST_IsEmpty() = 1 THEN
      RETURN CAST (NULL AS DOUBLE PRECISION);
    END IF;
    SET perimeter = 0.0;
    SET counter = 1;
    WHILE counter <= SELF.ST_NumGeometries() DO
      SET perimeter = perimeter +
        SELF.ST_GeometryN(counter).ST_Perimeter();
      SET counter = counter + 1;
    END WHILE;
    RETURN perimeter;
  END

CREATE METHOD ST_Perimeter
  (aunit CHARACTER VARYING(ST_MaxUnitNameLength))
  RETURNS DOUBLE PRECISION
  FOR ST_MultiSurface
  BEGIN
    DECLARE perimeter DOUBLE PRECISION;
    DECLARE counter INTEGER;

    IF SELF.ST_IsEmpty() = 1 THEN
      RETURN CAST (NULL AS DOUBLE PRECISION);
    END IF;
    SET perimeter = 0.0;
    SET counter = 1;
    WHILE counter <= SELF.ST_NumGeometries() DO
      SET perimeter = perimeter +
        SELF.ST_GeometryN(counter).ST_Perimeter(aunit);
      SET counter = counter + 1;
    END WHILE;
    RETURN perimeter;
  END

```

##### Description

- 1) The method *ST\_Perimeter()* has no input parameters.
- 2) For the null-call method *ST\_Perimeter()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the sum of the *ST\_Perimeter* value of the elements in the *ST\_PrivateGeometries* attribute of SELF.

- 3) Case:
- a) If the spatial reference system of SELF defines a <linear unit>, then the value returned by *ST\_Perimeter()* is in the linear unit of measure identified by <linear unit> squared.
  - b) Otherwise, the value returned by *ST\_Perimeter()* is in an implementation-defined unit of measure.
- 4) The method *ST\_Perimeter(CHARACTER VARYING)* takes the following input parameter:
- a) a CHARACTER VARYING value *unit*.
- 5) For the null-call method *ST\_Perimeter(CHARACTER VARYING)*:
- Case:
- a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the sum of the *ST\_Perimeter(unit)* values of each element in the *ST\_PrivateGeometries* attribute of SELF.
- 6) The value returned by *ST\_Perimeter(CHARACTER VARYING)* is in the units indicated by *unit*.
- 7) The values for *unit* shall be a supported <unit name>.
- 8) The value for *unit* is a supported <unit name> if and only if the value of *unit* is equal to the value of the UNIT\_NAME column of one of the rows where the value of the UNIT\_TYPE column is equal to 'LINEAR' in the ST\_UNITS\_OF\_MEASURE view.
- 9) If the unit specified by *unit* is not supported by the implementation to compute sum of the *ST\_Perimeter(unit)* values of each element in the *ST\_PrivateGeometries* attribute of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

### 9.5.5 ST\_Centroid Method

#### Purpose

Return the mathematical centroid of the *ST\_MultiSurface* value.

#### Definition

```
CREATE METHOD ST_Centroid()
  RETURNS ST_Point
  FOR ST_MultiSurface
  BEGIN
    --
    -- See Description
    --
  END
```

#### Description

- 1) The method *ST\_Centroid()* has no input parameters.
- 2) For the null-call method *ST\_Centroid()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return the mathematical centroid of the *ST\_MultiSurface* value. The result is not guaranteed to spatially intersect an *ST\_Surface* value in the *ST\_PrivateGeometries* attribute of an *ST\_MultiSurface* value.

The spatial reference system identifier of the returned *ST\_Point* value is equal to the spatial reference system identifier of SELF.

### 9.5.6 ST\_PointOnSurface Method

#### Purpose

Return an *ST\_Point* value guaranteed to spatially intersect an *ST\_Surface* value in the *ST\_PrivateGeometries* attribute of an *ST\_MultiSurface* value.

#### Definition

```
CREATE METHOD ST_PointOnSurface()
  RETURNS ST_Point
  FOR ST_MultiSurface
  BEGIN
    --
    -- See Description
    --
  END
```

#### Description

- 1) The method *ST\_PointOnSurface()* has no input parameters.
- 2) For the null-call method *ST\_PointOnSurface()*:

Case:

- a) If SELF is an empty set, then return the null value.
- b) Otherwise, return an *ST\_Point* value guaranteed to spatially intersect an element in the collection of the *ST\_MultiSurface* value.

The spatial reference system identifier of the returned *ST\_Point* value is equal to the spatial reference system identifier of SELF.

### 9.5.7 ST\_Geometries Methods

#### Purpose

Observe and mutate the `ST_PrivateGeometries` attribute of an `ST_MultiSurface` value.

#### Definition

```

CREATE METHOD ST_Geometries ()
  RETURNS ST_Surface ARRAY[ST_MaxGeometryArrayElements]
  FOR ST_MultiSurface
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      CAST(SELF.ST_PrivateGeometries AS
        ST_Surface ARRAY[ST_MaxGeometryArrayElements])
  END

CREATE METHOD ST_Geometries
  (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiSurface
  FOR ST_MultiSurface
  BEGIN
    DECLARE acounter INTEGER;
    DECLARE bcounter INTEGER;
    DECLARE asurfacearray ST_Surface ARRAY[ST_MaxGeometryArrayElements];

    -- Cast ageometryarray to an ST_Surface ARRAY
    SET asurfacearray = CAST(ageometryarray AS
      ST_Surface ARRAY[ST_MaxGeometryArrayElements]);
    -- If any two surfaces intersect with the dimension of the result
    -- greater than 0 (zero), then raise an exception.
    SET acounter = 1;
    WHILE acounter <= CARDINALITY(asurfacearray)-1 DO
      SET bcounter = acounter+1;
      WHILE bcounter <= CARDINALITY(asurfacearray) DO
        IF asurfacearray[acounter].ST_Intersection(
          asurfacearray[bcounter]).ST_Dimension() > 0 THEN
          SIGNAL SQLSTATE '2FF02'
            SET MESSAGE_TEXT = 'invalid argument';
        END IF;
        SET bcounter = bcounter + 1;
      END WHILE;
      SET acounter = acounter + 1;
    END WHILE;
    -- If SELF is the null value, then return the null value. Otherwise,
    -- return an ST_MultiSurface value containing asurfacearray.
    RETURN
    CASE
      WHEN SELF IS NULL THEN
        NULL
      ELSE
        (SELF AS ST_GeomCollection).
          ST_Geometries(asurfacearray)
    END;
  END
END

```

#### Definitional Rules

- 1) `ST_MaxGeometryArrayElements` is the implementation-defined maximum cardinality of an array of `ST_Geometry` values.

**Description**

- 1) The method *ST\_Geometries()* has no input parameters.
- 2) For the null-call method *ST\_Geometries()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the value of the *ST\_PrivateGeometries* attribute as an *ST\_Surface ARRAY*.
- 3) The method *ST\_Geometries(ST\_Geometry ARRAY)* takes the following input parameters:
  - a) an *ST\_Geometry ARRAY* value *ageometryarray*.
- 4) For the type preserving method *ST\_Geometries(ST\_Geometry ARRAY)*:
  - a) Let *ASURFACEARRAY* be the result of casting *ageometryarray* to an *ST\_Surface ARRAY* value (implicitly using *ST\_ToSurfaceAry(ST\_Geometry ARRAY)*).
  - b) Case:
    - i) If any two elements of *ASURFACEARRAY* intersect with more than a finite number of points, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
    - ii) If SELF is the null value, then return the null value.
    - iii) Otherwise, return an *ST\_MultiSurface* value with:
      - 1) The dimension set to 2.
      - 2) Using the method *ST\_Geometries(ST\_Geometry ARRAY)* for type *ST\_GeomCollection*, the *ST\_PrivateGeometries* attribute set to *ASURFACEARRAY*.

### 9.5.8 ST\_MSurfaceFromTxt Functions

#### Purpose

Return a specified ST\_MultiSurface value.

#### Definition

```
CREATE FUNCTION ST_MSurfaceFromTxt
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiSurface
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt) AS ST_MultiSurface)

CREATE FUNCTION ST_MSurfaceFromTxt
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_MultiSurface
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt, ansrid) AS ST_MultiSurface)
```

#### 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.

#### Description

- 1) The function *ST\_MSurfaceFromTxt(CHARACTER LARGE OBJECT)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call function *ST\_MSurfaceFromTxt(CHARACTER LARGE OBJECT)*:
  - a) The parameter *awkt* is the well-known text representation of an *ST\_MultiSurface* value. If *awkt* is not producible in the BNF for <multisurface text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_MultiSurface* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_MSurfaceFromTxt(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_MSurfaceFromTxt(CHARACTER LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkt* is the well-known text representation of an *ST\_MultiSurface* value. If *awkt* is not producible in the BNF for <multisurface text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_MultiSurface* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.

### 9.5.9 ST\_MSurfaceFromWKB Functions

#### Purpose

Return a specified ST\_MultiSurface value.

#### Definition

```
CREATE FUNCTION ST_MSurfaceFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiSurface
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_MultiSurface)

CREATE FUNCTION ST_MSurfaceFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_MultiSurface
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_MultiSurface)
```

#### Definitional Rules

- 1) *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.

#### Description

- 1) The function *ST\_MSurfaceFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 2) For the null-call function *ST\_MSurfaceFromWKB(BINARY LARGE OBJECT)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_MultiSurface* value. If *awkb* is not producible in the BNF for <multisurface binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_MultiSurface* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_MSurfaceFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_MSurfaceFromWKB(BINARY LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_MultiSurface* value. If *awkb* is not producible in the BNF for <multisurface binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_MultiSurface* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.

## 9.6 ST\_MultiPolygon Type and Routines

### 9.6.1 ST\_MultiPolygon Type

#### Purpose

The ST\_MultiPolygon type is a subtype of the ST\_MultiSurface and represents a collection of ST\_Polygon values.

#### Definition

```

CREATE TYPE ST_MultiPolygon
  UNDER ST_MultiSurface
  INSTANTIABLE
  NOT FINAL

CONSTRUCTOR METHOD ST_MultiPolygon
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiPolygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiPolygon
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_MultiPolygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiPolygon
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiPolygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiPolygon
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_MultiPolygon
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

CONSTRUCTOR METHOD ST_MultiPolygon
  (apolygonarray ST_Polygon
   ARRAY[ST_MaxGeometryArrayElements])
RETURNS ST_MultiPolygon
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_MultiPolygon
  (apolygonarray ST_Polygon
   ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
RETURNS ST_MultiPolygon
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

OVERRIDING METHOD ST_Geometries()
  RETURNS ST_Polygon ARRAY[ST_MaxGeometryArrayElements],

OVERRIDING METHOD ST_Geometries
  (ageometryarray ST_Geometry
   ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiPolygon

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

### Description

- 1) The *ST\_MultiPolygon* type provides for public use:
  - a) a method *ST\_MultiPolygon*(CHARACTER LARGE OBJECT),
  - b) a method *ST\_MultiPolygon*(CHARACTER LARGE OBJECT, INTEGER),
  - c) a method *ST\_MultiPolygon*(BINARY LARGE OBJECT),
  - d) a method *ST\_MultiPolygon*(BINARY LARGE OBJECT, INTEGER),
  - e) a method *ST\_MultiPolygon*(ST\_Polygon ARRAY),
  - f) a method *ST\_MultiPolygon*(ST\_Polygon ARRAY, INTEGER),
  - g) an overriding method *ST\_Geometries*(),
  - h) an overriding method *ST\_Geometries*(ST\_Geometry ARRAY),
  - i) a function *ST\_MPolyFromText*(CHARACTER LARGE OBJECT),
  - j) a function *ST\_MPolyFromText*(CHARACTER LARGE OBJECT, INTEGER),
  - k) a function *ST\_MPolyFromWKB*(BINARY LARGE OBJECT),
  - l) a function *ST\_MPolyFromWKB*(BINARY LARGE OBJECT, INTEGER),
  - m) a function *ST\_MPolyFromGML*(CHARACTER LARGE OBJECT),

- n) a function  $ST\_MPolyFromGML(CHARACTER\ LARGE\ OBJECT, INTEGER)$ ,
- o) a function  $ST\_BdMPolyFromText(CHARACTER\ LARGE\ OBJECT)$ ,
- p) a function  $ST\_BdMPolyFromText(CHARACTER\ LARGE\ OBJECT, INTEGER)$ ,
- q) a function  $ST\_BdMPolyFromWKB(BINARY\ LARGE\ OBJECT)$ ,
- r) a function  $ST\_BdMPolyFromWKB(BINARY\ LARGE\ OBJECT, INTEGER)$ .
- 2) The elements of the  $ST\_PrivateGeometries$  attribute are restricted to  $ST\_Polygon$  values.
- 3) The interiors of any two  $ST\_Polygon$  values that are elements of the  $ST\_PrivateGeometries$  attribute shall not spatially intersect.
- $$\forall m \in ST\_MultiPolygon, \forall p_i, p_j \in m.ST\_Geometries(), i \neq j, Interior(p_i) \cap Interior(p_j) = \emptyset$$
- 4) The boundaries of any two  $ST\_Polygon$  values that are elements of the  $ST\_PrivateGeometries$  attribute may only intersect at only a finite number of points.
- $$\forall m \in ST\_MultiPolygon, \forall p_i, p_j \in m.ST\_Geometries(),$$
- $$\forall c_i \in Boundary(p_i), c_j \in Boundary(p_j) \quad c_i \cap c_j = \{ p_1, \dots, p_k \mid p_i \in ST\_Point, 1 \leq i \leq k \}$$
- 5) An  $ST\_MultiPolygon$  value is defined as topologically closed.
- 6) An  $ST\_MultiPolygon$  is a topologically closed point set.
- 7) An  $ST\_MultiPolygon$  shall not have cut lines, spikes or punctures.
- $$\forall m \in ST\_MultiPolygon, m = Closure(Interior(m))$$
- 8) An  $ST\_MultiPolygon$  value is simple.
- 9) The interior of an  $ST\_MultiPolygon$  with more than one  $ST\_Polygon$  value is not a connected point set. The number of connected components of the interior of an  $ST\_MultiPolygon$  is equal to the cardinality of the  $ST\_PrivateGeometries$  attribute.
- 10) The boundary of an  $ST\_MultiPolygon$  is a set of linear rings corresponding to the boundaries of the  $ST\_Polygon$  values of the  $ST\_PrivateGeometries$ . Each linear ring in the boundary of the  $ST\_MultiPolygon$  is in the boundary of exactly one  $ST\_Polygon$  in the  $ST\_PrivateGeometries$  attribute. Every linear ring in the boundary of an  $ST\_Polygon$  in the  $ST\_PrivateGeometries$  attribute is in the boundary of the  $ST\_MultiPolygon$ .
- 11) An  $ST\_MultiPolygon$  value is well formed only if and only if all of the  $ST\_Polygon$  values in the  $ST\_PrivateGeometries$  attribute are well formed.
- 12) An  $ST\_MultiPolygon$  value returned by the constructor function corresponds to the empty set.

## 9.6.2 ST\_MultiPolygon Methods

### Purpose

Return an ST\_MultiPolygon value constructed from either the well-known text representation or the well-known binary representation of an ST\_MultiPolygon value, or the specified ST\_Polygon values.

### Definition

```

CREATE CONSTRUCTOR METHOD ST_MultiPolygon
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiPolygon
  FOR ST_MultiPolygon
  RETURN ST_MPolyFromText(awkt)

CREATE CONSTRUCTOR METHOD ST_MultiPolygon
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_MultiPolygon
  FOR ST_MultiPolygon
  RETURN ST_MPolyFromText(awkt, ansrid)

CREATE CONSTRUCTOR METHOD ST_MultiPolygon
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiPolygon
  FOR ST_MultiPolygon
  RETURN ST_MPolyFromWKB(awkb)

CREATE CONSTRUCTOR METHOD ST_MultiPolygon
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_MultiPolygon
  FOR ST_MultiPolygon
  RETURN ST_MPolyFromWKB(awkb, ansrid)

CREATE CONSTRUCTOR METHOD ST_MultiPolygon
  (apolygonarray ST_Polygon ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiPolygon
  FOR ST_MultiPolygon
  RETURN SELF.ST_SRID(0).ST_Geometries(apolygonarray)

CREATE CONSTRUCTOR METHOD ST_MultiPolygon
  (apolygonarray ST_Polygon ARRAY[ST_MaxGeometryArrayElements],
   ansrid INTEGER)
  RETURNS ST_MultiPolygon
  FOR ST_MultiPolygon
  RETURN SELF.ST_SRID(ansrid).ST_Geometries(apolygonarray)

```

### Definitional Rules

- 1) *ST\_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST\_Geometry* values.
- 2) *ST\_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST\_Geometry* value.
- 3) *ST\_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_Geometry* value.

### Description

- 1) The method *ST\_MultiPolygon(CHARACTER LARGE OBJECT)* takes the following input parameter:
  - a) a CHARACTER LARGE OBJECT value *awkt*.

- 2) For the null-call type preserving SQL-invoked constructor method *ST\_MultiPolygon*(*CHARACTER LARGE OBJECT*):
- The parameter *awkt* is the well-known text representation of an *ST\_MultiPolygon* value. If *awkt* is not producible in the BNF for <multipolygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - Return an *ST\_MultiPolygon* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The method *ST\_MultiPolygon*(*CHARACTER LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *CHARACTER LARGE OBJECT* value *awkt*,
  - an *INTEGER* value *ansrid*.
- 4) For the null-call type preserving SQL-invoked constructor method *ST\_MultiPolygon*(*CHARACTER LARGE OBJECT*, *INTEGER*):
- The parameter *awkt* is the well-known text representation of an *ST\_MultiPolygon* value. If *awkt* is not producible in the BNF for <multipolygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - Return an *ST\_MultiPolygon* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.
- 5) The method *ST\_MultiPolygon*(*BINARY LARGE OBJECT*) takes the following input parameter:
- a *BINARY LARGE OBJECT* value *awkb*.
- 6) For the null-call type preserving SQL-invoked constructor method *ST\_MultiPolygon*(*BINARY LARGE OBJECT*):
- The parameter *awkb* is the well-known binary representation of an *ST\_MultiPolygon* value. If *awkb* is not producible in the BNF for <multipolygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_MultiPolygon* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 7) The method *ST\_MultiPolygon*(*BINARY LARGE OBJECT*, *INTEGER*) takes the following input parameters:
- a *BINARY LARGE OBJECT* value *awkb*,
  - an *INTEGER* value *ansrid*.
- 8) For the null-call type preserving SQL-invoked constructor method *ST\_MultiPolygon*(*BINARY LARGE OBJECT*, *INTEGER*):
- The parameter *awkb* is the well-known binary representation of an *ST\_MultiPolygon* value. If *awkb* is not producible in the BNF for <multipolygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - Return an *ST\_MultiPolygon* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.
- 9) The method *ST\_MultiPolygon*(*ST\_Polygon ARRAY*) takes the following input parameters:
- an *ST\_Polygon* value *apolygonarray*.
- 10) The null-call type preserving SQL-invoked constructor method *ST\_MultiPolygon*(*ST\_Polygon ARRAY*) returns an *ST\_MultiPolygon* value with:
- The spatial reference system identifier set to 0 (zero).

- b) Using the method *ST\_Geometries(ST\_Geometry ARRAY)*, the *ST\_PrivateGeometries* attribute set to *apolygonarray*, the *ST\_PrivateDimension* attribute set to 2, and the *ST\_PrivateCoordinateDimension* attribute set to 2.
- 11) The method *ST\_MultiPolygon(ST\_Polygon ARRAY, INTEGER)* takes the following input parameters:
- a) an *ST\_Polygon ARRAY* value *apolygonarray*,
  - b) an *INTEGER* value *ansrid*.
- 12) The null-call type preserving SQL-invoked constructor method *ST\_MultiPolygon(ST\_Polygon ARRAY, INTEGER)* returns an *ST\_MultiPolygon* value with:
- a) The spatial reference system identifier set to *ansrid*.
  - b) Using the method *ST\_Geometries(ST\_Geometry ARRAY)*, the *ST\_PrivateGeometries* attribute set to *apolygonarray*, the *ST\_PrivateDimension* attribute set to 2, and the *ST\_PrivateCoordinateDimension* attribute set to 2.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 9.6.3 ST\_Geometries Methods

#### Purpose

Observe and mutate the `ST_PrivateGeometries` attribute of an `ST_MultiPolygon` value.

#### Definition

```

CREATE METHOD ST_Geometries ()
  RETURNS ST_Polygon ARRAY[ST_MaxGeometryArrayElements]
  FOR ST_MultiPolygon
  RETURN
  CASE
    WHEN SELF.ST_IsEmpty() = 1 THEN
      NULL
    ELSE
      CAST(SELF.ST_PrivateGeometries AS
        ST_Polygon ARRAY[ST_MaxGeometryArrayElements])
  END

CREATE METHOD ST_Geometries
  (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
  RETURNS ST_MultiPolygon
  FOR ST_MultiPolygon
  BEGIN
    DECLARE acounter INTEGER;
    DECLARE bcounter INTEGER;
    DECLARE apolygonarray ST_Polygon ARRAY[ST_MaxGeometryArrayElements];

    -- Cast ageometryarray to an ST_Polygon ARRAY
    SET apolygonarray = CAST(ageometryarray AS
      ST_Polygon ARRAY[ST_MaxGeometryArrayElements]);
    -- If any two polygons intersect with the dimension of the result
    -- greater than 0 (zero), then raise an exception.
    SET acounter = 1;
    WHILE acounter <= CARDINALITY(apolygonarray)-1 DO
      SET bcounter = acounter+1;
      WHILE bcounter <= CARDINALITY(apolygonarray) DO
        IF apolygonarray[acounter].ST_Intersection(
          apolygonarray[bcounter]).ST_Dimension() > 0 THEN
          SIGNAL SQLSTATE '2FF02'
            SET MESSAGE_TEXT = 'invalid argument';
        END IF;
        SET bcounter = bcounter + 1;
      END WHILE;
      SET acounter = acounter + 1;
    END WHILE;
    -- If SELF is the null value, then return the null value. Otherwise,
    -- return an ST_MultiPolygon value containing apolygonarray.
    RETURN
    CASE
      WHEN SELF IS NULL THEN
        NULL
      ELSE
        (SELF AS ST_MultiSurface).
          ST_Geometries(apolygonarray)
    END;
  END
END

```

#### Definitional Rules

- 1) `ST_MaxGeometryArrayElements` is the implementation-defined maximum cardinality of an array of `ST_Geometry` values.

**Description**

- 1) The method *ST\_Geometries()* has no input parameters.
- 2) For the null-call method *ST\_Geometries()*:
 

Case:

  - a) If SELF is an empty set, then return the null value.
  - b) Otherwise, return the value of the *ST\_PrivateGeometries* attribute as an *ST\_Polygon ARRAY*.
- 3) The method *ST\_Geometries(ST\_Geometry ARRAY)* takes the following input parameters:
  - a) an *ST\_Geometry ARRAY* value *ageometryarray*.
- 4) For the type preserving method *ST\_Geometries(ST\_Geometry ARRAY)*:
  - a) Let *APOLYGONARRAY* be the result of casting *ageometryarray* to an *ST\_Polygon ARRAY* value (implicitly using *ST\_ToPolygonAry(ST\_Geometry ARRAY)*).
  - b) Case:
    - i) If any two elements of *APOLYGONARRAY* intersect with more than a finite number of points, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
    - ii) If SELF is the null value, then return the null value.
    - iii) Otherwise, return an *ST\_MultiPolygon* value with:
      - 1) The dimension set to 2.
      - 2) Using the method *ST\_Geometries(ST\_Geometry ARRAY)* for type *ST\_MultiSurface*, the *ST\_PrivateGeometries* attribute set to *APOLYGONARRAY*.

#### 9.6.4 ST\_MPolyFromText Functions

##### Purpose

Return a specified ST\_MultiPolygon value.

##### Definition

```
CREATE FUNCTION ST_MPolyFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiPolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt) AS ST_MultiPolygon)

CREATE FUNCTION ST_MPolyFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_MultiPolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromText(awkt, ansrid) AS ST_MultiPolygon)
```

##### 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.

##### Description

- 1) The function *ST\_MPolyFromText*(CHARACTER LARGE OBJECT) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call function *ST\_MPolyFromText*(CHARACTER LARGE OBJECT):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_MultiPolygon* value. If *awkt* is not producible in the BNF for <multipolygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_MultiPolygon* value represented by *awkt* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_MPolyFromText*(CHARACTER LARGE OBJECT, INTEGER) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_MPolyFromText*(CHARACTER LARGE OBJECT, INTEGER):
  - a) The parameter *awkt* is the well-known text representation of an *ST\_MultiPolygon* value. If *awkt* is not producible in the BNF for <multipolygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Return an *ST\_MultiPolygon* value represented by *awkt* with the spatial reference system identifier set to *ansrid*.

### 9.6.5 ST\_MPolyFromWKB Functions

#### Purpose

Return a specified ST\_MultiPolygon value.

#### Definition

```
CREATE FUNCTION ST_MPolyFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiPolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_MultiPolygon)

CREATE FUNCTION ST_MPolyFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_MultiPolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromWKB(awkb, ansrid) AS ST_MultiPolygon)
```

#### Definitional Rules

- 1) *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.

#### Description

- 1) The function *ST\_MPolyFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 2) For the null-call function *ST\_MPolyFromWKB(BINARY LARGE OBJECT)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_MultiPolygon* value. If *awkb* is not producible in the BNF for <multipolygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_MultiPolygon* value represented by *awkb* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_MPolyFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_MPolyFromWKB(BINARY LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_MultiPolygon* value. If *awkb* is not producible in the BNF for <multipolygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Return an *ST\_MultiPolygon* value represented by *awkb* with the spatial reference system identifier set to *ansrid*.

### 9.6.6 ST\_MPolyFromGML Functions

#### Purpose

Return a specified ST\_MultiPolygon value.

#### Definition

```
CREATE FUNCTION ST_MPolyFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML))
  RETURNS ST_MultiPolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromGML(agml) AS ST_MultiPolygon)

CREATE FUNCTION ST_MPolyFromGML
  (agml CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML),
   ansrid INTEGER)
  RETURNS ST_MultiPolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN TREAT(ST_GeomFromGML(agml, ansrid) AS ST_MultiPolygon)
```

#### Definitional Rules

- 1) *ST\_MaxGeometryAsGML* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the GML representation of an *ST\_Geometry* value.

#### Description

- 1) The function *ST\_MPolyFromGML(CHARACTER LARGE OBJECT)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*.
- 2) For the null-call function *ST\_MPolyFromGML(CHARACTER LARGE OBJECT)*:
  - a) If the parameter *agml* does not contain a MultiPolygon XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_MultiPolygon* value represented by *agml* with the spatial reference system identifier set to 0 (zero).
- 3) The function *ST\_MPolyFromGML(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *agml*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_MPolyFromGML(CHARACTER LARGE OBJECT, INTEGER)*:
  - a) If the parameter *agml* does not contain a MultiPolygon XML element in the GML representation, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid GML representation*.
  - b) Return an *ST\_MultiPolygon* value represented by *agml* with the spatial reference system identifier set to *ansrid*.

### 9.6.7 ST\_BdMPolyFromText Functions

#### Purpose

Return a specified ST\_MultiPolygon value.

#### Definition

```

CREATE FUNCTION ST_BdMPolyFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText))
  RETURNS ST_MultiPolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN ST_BdMPolyFromText(awkt, 0)

CREATE FUNCTION ST_BdMPolyFromText
  (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
   ansrid INTEGER)
  RETURNS ST_MultiPolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  BEGIN
    --
    -- See Description
    --
  END

```

#### Description

- 1) The function *ST\_BdMPolyFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) For the null-call function *ST\_BdMPolyFromText(CHARACTER LARGE OBJECT)* returns an *ST\_MultiPolygon* value as the result of the value expression: *ST\_BdMPolyFromText(awkt, 0)*.
- 3) The function *ST\_BdMPolyFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_BdMPolyFromText(CHARACTER LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkt* is the well-known text representation of an *ST\_MultiLineString* value. If *awkt* is not producible in the BNF for <multilinestring text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
  - b) Use *ST\_MLineFromText(CHARACTER LARGE OBJECT)* to transform *awkt* to an *ST\_MultiLineString* value, *AMLS*.
  - c) If any *ST\_LineString* value in *AMLS* is not a linear ring, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
  - d) Using an implementation-dependent algorithm, an array of *ST\_Polygon* values, *APA*, is determined from the array of linear rings in *AMLS*.
  - e) Return an *ST\_MultiPolygon* value with:
    - i) The spatial reference system identifier set to *ansrid*.

- ii) Using the method *ST\_Geometries*(*ST\_Geometry ARRAY*), the *ST\_PrivateGeometries* attribute set to *APA*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 9.6.8 ST\_BdMPolyFromWKB Functions

#### Purpose

Return a specified ST\_MultiPolygon value.

#### Definition

```
CREATE FUNCTION ST_BdMPolyFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary))
  RETURNS ST_MultiPolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  RETURN ST_BdMPolyFromWKB(awkb, 0)

CREATE FUNCTION ST_BdMPolyFromWKB
  (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
   ansrid INTEGER)
  RETURNS ST_MultiPolygon
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  BEGIN
    --
    -- See Description
    --
  END
```

#### Description

- 1) The function *ST\_BdMPolyFromWKB(BINARY LARGE OBJECT)* takes the following input parameters:
  - a) a BINARY LARGE OBJECT value *awkb*.
- 2) For the null-call function *ST\_BdMPolyFromWKB(BINARY LARGE OBJECT)* returns an *ST\_MultiPolygon* value as the result of the value expression: *ST\_BdMPolyFromWKB(awkb, 0)*.
- 3) The function *ST\_BdMPolyFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
  - a) an BINARY LARGE OBJECT value *awkb*,
  - b) an INTEGER value *ansrid*.
- 4) For the null-call function *ST\_BdMPolyFromWKB(BINARY LARGE OBJECT, INTEGER)*:
  - a) The parameter *awkb* is the well-known binary representation of an *ST\_MultiLineString* value. If *awkb* is not producible in the BNF for <multilinestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
  - b) Use *ST\_MLineFromWKB(BINARY LARGE OBJECT)* to transform *awkb* to an *ST\_MultiLineString* value, *AMLS*.
  - c) If any *ST\_LineString* value in *AMLS* is not a linear ring, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
  - d) Using an implementation-dependent algorithm, an array of *ST\_Polygon* values, *APA*, is determined from the array of linear rings in *AMLS*.
  - e) Return an *ST\_MultiPolygon* value with:
    - i) The spatial reference system identifier set to *ansrid*.

- ii) Using the method *ST\_Geometries(ST\_Geometry ARRAY)*, the *ST\_PrivateGeometries* attribute set to *APA*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 10 Spatial Reference System Type

### 10.1 ST\_SpatialRefSys Type and Routines

#### 10.1.1 ST\_SpatialRefSys Type

##### Purpose

The ST\_SpatialRefSys type encapsulates all aspects of spatial reference systems.

##### Definition

```

CREATE TYPE ST_SpatialRefSys
  AS (
    --
    -- See Description
    --
  )
  INSTANTIABLE
  NOT FINAL

  CONSTRUCTOR METHOD ST_SpatialRefSys
    (awkt CHARACTER LARGE OBJECT(ST_MaxSRSAstext))
  RETURNS ST_SpatialRefSys
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

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

  METHOD ST_AsWKTSRS()
  RETURNS CHARACTER LARGE OBJECT(ST_MaxSRSAstext)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

  METHOD ST_WKTSRSToSQL
    (awkt CHARACTER LARGE OBJECT(ST_MaxSRSAstext))
  RETURNS ST_SpatialRefSys
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

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

```

```
METHOD ST_Equals  
  (ansrs ST_SpatialRefSys)  
RETURNS INTEGER  
LANGUAGE SQL  
DETERMINISTIC  
CONTAINS SQL  
RETURNS NULL ON NULL INPUT
```

#### Definitional Rules

- 1) *ST\_MaxSRSAstext* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_SpatialRefSys* value.

#### Description

- 1) The *ST\_SpatialRefSys* type provides for public use:
  - a) a method *ST\_SpatialRefSys*(CHARACTER LARGE OBJECT),
  - b) a method *ST\_SpatialRefSys*(INTEGER),
  - c) a method *ST\_AsWKTSRS*(),
  - d) a method *ST\_WKTSRSToSQL*(CHARACTER LARGE OBJECT),
  - e) a method *ST\_SRID*(),
  - f) a method *ST\_Equals*(*ST\_SpatialRefSys*),
  - g) an ordering function *ST\_OrderingEquals*(*ST\_SpatialRefSys*, *ST\_SpatialRefSys*),
  - h) an SQL Transform group *ST\_WellKnownText*.
- 2) The attribute definitions in the *ST\_SpatialRefSys* type are implementation-dependent.

NOTE 11 Implementations should refer to ISO 19111 as a model to follow for the implementation-dependent attribute definitions in the *ST\_SpatialRefSys* type.

### 10.1.2 ST\_SpatialRefSys Methods

#### Purpose

Return a specified ST\_SpatialRefSys value.

#### Definition

```

CREATE CONSTRUCTOR METHOD ST_SpatialRefSys
  (awkt CHARACTER LARGE OBJECT(ST_MaxSRSAsText))
  RETURNS ST_SpatialRefSys
  FOR ST_SpatialRefSys
  BEGIN
    --
    -- See Description
    --
  END

CREATE CONSTRUCTOR METHOD ST_SpatialRefSys
  (ansrid INTEGER)
  RETURNS ST_SpatialRefSys
  FOR ST_SpatialRefSys
  BEGIN
    --
    -- See Description
    --
  END

```

#### Definitional Rules

- 1) *ST\_MaxSRSAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_SpatialRefSys* value.

#### Description

- 1) The method *ST\_SpatialRefSys(CHARACTER LARGE OBJECT)* takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT value *awkt*.
- 2) The parameter *awkt* is the well-known text representation of an *ST\_SpatialRefSys* value. If *awkt* is not producible in the BNF for <spatial reference system>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
- 3) The null-call type preserving SQL-invoked constructor method *ST\_SpatialRefSys(CHARACTER LARGE OBJECT)* returns an *ST\_SpatialRefSys* value representing the spatial reference system defined by *awkt*.
- 4) The method *ST\_SpatialRefSys(INTEGER)* takes the following input parameters:
  - a) an INTEGER value *ansrid*.
- 5) The null-call type preserving SQL-invoked constructor method *ST\_SpatialRefSys(INTEGER)* returns an *ST\_SpatialRefSys* value representing the spatial reference system defined by the spatial reference system identifier, *ansrid*.

### 10.1.3 ST\_AsWKTSRS Method

#### Purpose

Return the well-known text representation of a spatial reference system.

#### Definition

```
CREATE METHOD ST_AsWKTSRS ()
  RETURNS CHARACTER LARGE OBJECT (ST_MaxSRSAstext)
  FOR ST_SpatialRefSys
  BEGIN
    --
    -- See Description
    --
  END
```

#### Definitional Rules

- 1) *ST\_MaxSRSAstext* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_SpatialRefSys* value.

#### Description

- 1) The method *ST\_AsWKTSRS()* has no input parameters.
- 2) The null-call method *ST\_AsWKTSRS()* returns a CHARACTER LARGE OBJECT value containing the well-known text representation of SELF. Values shall be produced in the BNF for <spatial reference system>.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

**10.1.4 ST\_WKTSRSToSQL Method****Purpose**

Return a specified ST\_SpatialRefSys value.

**Definition**

```
CREATE METHOD ST_WKTSRSToSQL
  (awkt CHARACTER LARGE OBJECT(ST_MaxSRsAsText))
  RETURNS ST_SpatialRefSys
  FOR ST_SpatialRefSys
  RETURN SELF.ST_SpatialRefSys(awkt)
```

**Definitional Rules**

- 1) *ST\_MaxSRsAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_SpatialRefSys*.

**Description**

- 1) The method *ST\_WKTSRSToSQL*(CHARACTER LARGE OBJECT) takes the following input parameters:
  - a) a CHARACTER LARGE OBJECT *awkt*.
- 2) The parameter *awkt* is the well-known text representation of an *ST\_SpatialRefSys* value. If *awkt* is not producible in the BNF for <spatial reference system>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
- 3) The null-call method *ST\_WKTSRSToSQL*(CHARACTER LARGE OBJECT) returns an *ST\_SpatialRefSys* value represented by *awkt*.

### 10.1.5 ST\_SRID Method

#### Purpose

Return a spatial reference system identifier of an ST\_SpatialRefSys value.

#### Definition

```
CREATE METHOD ST_SRID()  
  RETURNS INTEGER  
  FOR ST_SpatialRefSys  
  BEGIN  
    --  
    -- See Description  
    --  
  END
```

#### Description

- 1) The method *ST\_SRID()* has no input parameters.
- 2) The null-call method *ST\_SRID()* returns an INTEGER value representing a unique identifier. This unique identifier is called the *spatial reference system identifier*. A spatial reference system identifier that is equal to 0 (zero) is implementation-defined. A spatial reference system identifier that is not equal to 0 (zero) is implementation-dependent.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

**10.1.6 ST\_Equals Method****Purpose**

Test if two *ST\_SpatialRefSys* values are equal.

**Definition**

```
CREATE METHOD ST_Equals
  (ansrs ST_SpatialRefSys)
  RETURNS INTEGER
  FOR ST_SpatialRefSys
  BEGIN
    --
    -- See Description
    --
  END
```

**Description**

- 1) The method *ST\_Equals(ST\_SpatialRefSys)* takes the following input parameters:
  - a) an *ST\_SpatialRefSys* value *ansrs*.
- 2) For the null-call method *ST\_Equals(ST\_SpatialRefSys)*:
 

Case:

  - a) If SELF is equal to *ansrs*, then return 1 (one).
  - b) Otherwise, return 0 (zero).
- 3) The method *ST\_Equals(ST\_SpatialRefSys)* is implementation-defined.

**10.1.7 ST\_OrderingEquals Function****Purpose**

Test if two *ST\_SpatialRefSys* values are equal.

**Definition**

```
CREATE FUNCTION ST_OrderingEquals
  (ansrs ST_SpatialRefSys,
   othersrs ST_SpatialRefSys)
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  STATIC DISPATCH
  RETURN
  CASE
    WHEN ansrs.ST_Equals(othersrs) = 1 THEN
      0
    ELSE
      1
  END

CREATE ORDERING FOR ST_SpatialRefSys
  EQUALS ONLY BY RELATIVE
  WITH FUNCTION ST_OrderingEquals(ST_SpatialRefSys, ST_SpatialRefSys)
```

**Description**

- 1) The function *ST\_OrderingEquals(ST\_SpatialRefSys, ST\_SpatialRefSys)* takes the following input parameters:
  - a) an *ST\_SpatialRefSys* value *ansrs*,
  - b) an *ST\_SpatialRefSys* value *othersrs*.
- 2) For the null-call function *ST\_OrderingEquals(ST\_SpatialRefSys, ST\_SpatialRefSys)*:
 

Case:

  - a) If the value expression *ansrs.ST\_Equals(othersrs)* is 1 (one), then return 0 (zero).
  - b) Otherwise, return 1 (one).
- 3) Use the function *ST\_OrderingEquals(ST\_SpatialRefSys, ST\_SpatialRefSys)* to define ordering for the *ST\_SpatialRefSys* type.

### 10.1.8 ST\_WellKnownText SQL Transform Group

#### Purpose

Define SQL transform functions for the ST\_SpatialRefSys type.

#### Definition

```
CREATE TRANSFORM FOR ST_SpatialRefSys
  ST_WellKnownText
  (TO SQL WITH METHOD ST_WKTSRSToSQL
   (CHARACTER LARGE OBJECT(ST_MaxSRsAsText)),
   FROM SQL WITH METHOD ST_AsWKTSRS())
```

#### Definitional Rules

- 1) *ST\_MaxSRsAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST\_SpatialRefSys*.

#### Description

- 1) Use the method *ST\_WKTSRSToSQL(CHARACTER LARGE OBJECT)* and the method *ST\_AsWKTSRS()* to define the transform group *ST\_WellKnownText*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

**10.1.9 <spatial reference system>****Purpose**

This subclause contains the definition of <spatial reference system>.

**Description**

- 1) The well-known text representation of an *ST\_SpatialRefSys* value is defined by the following BNF for <spatial reference system>.

```

<spatial reference system> ::=
  <projected cs>
  | <geographic cs>
  | <geocentric cs>

<projected cs> ::=
  PROJCS <left delimiter>
    <double quote> <name> <double quote> <comma>
    <geographic cs> <comma>
    <projection> <comma>
    { <parameter> <comma> }...
    <linear unit>
    <right delimiter>

<geographic cs> ::=
  GEOGCS <left delimiter>
    <double quote> <name> <double quote> <comma>
    <datum> <comma>
    <prime meridian> <comma>
    <angular unit>
    [ <linear unit> ]
    <right delimiter>

<geocentric cs> ::=
  GEOCCS <left delimiter>
    <double quote> <name> <double quote> <comma>
    <datum> <comma>
    <prime meridian> <comma>
    <linear unit>
    <right delimiter>

<projection> ::=
  PROJECTION <left delimiter>
    <double quote> <projection name> <double quote>
    <right delimiter>

<parameter> ::=
  PARAMETER <left delimiter>
    <double quote> <parameter name> <double quote> <comma>
    <value>
    <right delimiter>

<value> ::= <number>

<datum> ::=
  DATUM <left delimiter>
    <double quote> <datum name> <double quote> <comma>
    <spheroid>
    <right delimiter>

```

```

<spheroid> ::=
    SPHEROID <left delimiter>
        <double quote> <spheroid name> <double quote> <comma>
        <semi-major axis> <comma>
        <inverse flattening>
        <right delimiter>

<semi-major axis> ::= <number>

<inverse flattening> ::= <number>

<prime meridian> ::=
    PRIMEM <left delimiter>
        <double quote> <prime meridian name> <double quote> <comma>
        <longitude>
        <right delimiter>

<longitude> ::= <number>

<angular unit> ::= <unit>

<linear unit> ::= <unit>

<unit> ::=
    UNIT <left delimiter>
        <double quote> <unit name> <double quote> <comma>
        <conversion factor>
        <right delimiter>

<conversion factor> ::= <number>

<datum name> ::= <letters>

<parameter name> ::= <letters>

<prime meridian name> ::= <letters>

<projection name> ::= <letters>

<spheroid name> ::= <letters>

<unit name> ::= <letters>

<name> ::= <letters>

<letters> ::= <letter>...

<letter> ::=
    <simple Latin letter>
    | <digit>
    | <special>

<special> ::=
    <left paren>
    | <right paren>
    | <minus>
    | <underscore>
    | <period>
    | <quote>
    | <space>

<number> ::=
    <exact numeric literal>
    | <approximate numeric literal>

<left delimiter> ::=
    <left paren>
    | <left bracket>

```

<right delimiter> ::=  
     <right paren>  
     | <right bracket>

<exact numeric literal> ::=  
     !! See Subclause 5.3, "<literal>", in ISO/IEC 9075-2

<approximate numeric literal> ::=  
     !! See Subclause 5.3, "<literal>", in ISO/IEC 9075-2

<simple Latin letter> ::=  
     !! See Subclause 5.1, "<SQL terminal character>", in  
     ISO/IEC 9075-2

<digit> ::=  
     !! See Subclause 5.1, "<SQL terminal character>", in  
     ISO/IEC 9075-2

<double quote> ::=  
     !! See Subclause 5.1, "<SQL terminal character>", in  
     ISO/IEC 9075-2

<comma> ::=  
     !! See Subclause 5.1, "<SQL terminal character>", in  
     ISO/IEC 9075-2

<left bracket> ::=  
     !! See Subclause 5.1, "<SQL terminal character>", in  
     ISO/IEC 9075-2

<right bracket> ::=  
     !! See Subclause 5.1, "<SQL terminal character>", in  
     ISO/IEC 9075-2

<left paren> ::=  
     !! See Subclause 5.1, "<SQL terminal character>", in  
     ISO/IEC 9075-2

<right paren> ::=  
     !! See Subclause 5.1, "<SQL terminal character>", in  
     ISO/IEC 9075-2

<minus> ::=  
     !! See Subclause 5.1, "<SQL terminal character>", in  
     ISO/IEC 9075-2

<underscore> ::=  
     !! See Subclause 5.1, "<SQL terminal character>", in  
     ISO/IEC 9075-2

<period> ::=  
     !! See Subclause 5.1, "<SQL terminal character>", in  
     ISO/IEC 9075-2

<quote> ::=  
     !! See Subclause 5.1, "<SQL terminal character>", in  
     ISO/IEC 9075-2

<space> ::=  
     !! See Subclause 5.1, "<SQL terminal character>", in  
     ISO/IEC 9075-2

a) Case:

- i) If <left paren> is used as a <left delimiter>, then <right paren> shall be used as the corresponding <right delimiter>.
- ii) If <left bracket> is used as a <left delimiter>, then <right bracket> shall be used as the corresponding <right delimiter>.

- b) A <spatial reference system> is a geographic (latitude-longitude), a projected (X, Y), or a geocentric (X, Y, Z) coordinate system.
- The coordinate system is composed of several items. Each item has a keyword in upper case followed by the defining, comma-delimited, parameters of the item in brackets. Some items are composed of other items in a nested structure.
- c) The list of keywords are DATUM, GEOCCS, GEOGCS, PROJCS, PARAMETER, PRIMEM, PROJECTION, SPHEROID, and UNIT.
- d) <projected cs> is a projected coordinate system. <projection name> is an implementation-defined name of a parameter.
- e) <geographic cs> is a geographic coordinate system.
- f) <geocentric cs> is a geocentric coordinate system.
- g) <name> is the name of the coordinate system.
- h) <projection> is the projection system of a <projected cs>.
- i) <parameter> is a parameter of the <projection> to define the <projected cs>. <parameter name> is an implementation-defined name of a parameter.
- j) <datum> is the horizontal datum used by the <geographic cs> or the <geocentric cs>. <datum name> is an implementation-defined name of a datum.
- k) <spheroid> is the spheroid of a datum defined by a semi-major axis, <semi-major axis>, and an inverse flattening, <inverse flattening>. <spheroid>s are implementation-defined.
- l) <prime meridian> is the longitude used by the <geographic cs> or the <geocentric cs>. The <semi-major axis> is greater than 0 (zero) and it is measured in meters. <prime meridian>s are implementation-defined.
- m) <angular unit> specifies an angular unit and <linear unit> defines a linear unit. <conversion factor> specifies the number of meters for a linear unit or the number of radians for an angular unit per unit. <conversion factor> is greater than zero. <angular unit>s and <linear unit>s are implementation-defined.

Blank page

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 11 Angle and Direction Types

### 11.1 ST\_Angle Type and Routines

#### 11.1.1 ST\_Angle Type

##### Purpose

The ST\_Angle type is used to express circular measurement or to measure the degree of separation of two intersecting lines.

##### Definition

```

CREATE TYPE ST_Angle
AS (
    ST_PrivateRadians DOUBLE PRECISION DEFAULT NULL
)
INSTANTIABLE
NOT FINAL

CONSTRUCTOR METHOD ST_Angle
    (angle DOUBLE PRECISION)
    RETURNS ST_Angle
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Angle
    (units CHARACTER(1),
     angle DOUBLE PRECISION)
    RETURNS ST_Angle
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Angle
    (degrees INTEGER, minutes DOUBLE PRECISION)
    RETURNS ST_Angle
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Angle
    (degrees INTEGER, minutes INTEGER, seconds DOUBLE PRECISION)
    RETURNS ST_Angle
    SELF AS RESULT
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

```

```

CONSTRUCTOR METHOD ST_Angle
  (atpoint ST_Point, _apoint ST_Point, anotherpoint ST_Point)
  RETURNS ST_Angle
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Angle
  (adirection ST_Direction,
   anotherdirection ST_Direction)
  RETURNS ST_Angle
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Angle
  (aline ST_LineString,
   anotherline ST_LineString)
  RETURNS ST_Angle
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Angle
  (awkt CHARACTER VARYING(ST_MaxAngleAsText))
  RETURNS ST_Angle
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_Radians()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_Radians
  (radians DOUBLE PRECISION)
  RETURNS ST_Angle
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD ST_Degrees()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

METHOD ST\_Degrees  
 (degrees DOUBLE PRECISION)  
 RETURNS ST\_Angle  
 SELF AS RESULT  
 LANGUAGE SQL  
 DETERMINISTIC  
 CONTAINS SQL  
 CALLED ON NULL INPUT,

METHOD ST\_DegreeComponent()  
 RETURNS INTEGER  
 LANGUAGE SQL  
 DETERMINISTIC  
 CONTAINS SQL  
 RETURNS NULL ON NULL INPUT,

METHOD ST\_MinuteComponent()  
 RETURNS INTEGER  
 LANGUAGE SQL  
 DETERMINISTIC  
 CONTAINS SQL  
 RETURNS NULL ON NULL INPUT,

METHOD ST\_SecondComponent()  
 RETURNS DOUBLE PRECISION  
 LANGUAGE SQL  
 DETERMINISTIC  
 CONTAINS SQL  
 RETURNS NULL ON NULL INPUT,

METHOD ST\_String()  
 RETURNS CHARACTER VARYING(*ST\_MaxAngleString*)  
 LANGUAGE SQL  
 DETERMINISTIC  
 CONTAINS SQL  
 RETURNS NULL ON NULL INPUT,

METHOD ST\_String  
 (numdecdigits INTEGER)  
 RETURNS CHARACTER VARYING(*ST\_MaxAngleString*)  
 LANGUAGE SQL  
 DETERMINISTIC  
 CONTAINS SQL  
 RETURNS NULL ON NULL INPUT,

METHOD ST\_String  
 (dms CHARACTER VARYING(*ST\_MaxAngleString*))  
 RETURNS ST\_Angle  
 SELF AS RESULT  
 LANGUAGE SQL  
 DETERMINISTIC  
 CONTAINS SQL  
 CALLED ON NULL INPUT,

METHOD ST\_Gradians()  
 RETURNS DOUBLE PRECISION  
 LANGUAGE SQL  
 DETERMINISTIC  
 CONTAINS SQL  
 RETURNS NULL ON NULL INPUT,

```
METHOD ST_Gradians
  (gradians DOUBLE PRECISION)
  RETURNS ST_Angle
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,
```

```
METHOD ST_Add
  (anangle ST_Angle)
  RETURNS ST_Angle
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,
```

```
METHOD ST_Subtract
  (anangle ST_Angle)
  RETURNS ST_Angle
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,
```

```
METHOD ST_Multiply
  (afactor DOUBLE PRECISION)
  RETURNS ST_Angle
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,
```

```
METHOD ST_Divide
  (adivisor DOUBLE PRECISION)
  RETURNS ST_Angle
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,
```

```
METHOD ST_AsText()
  RETURNS CHARACTER VARYING(ST_MaxAngleAsText)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
```

### Definitional Rules

- 1) *ST\_MaxAngleString* is the implementation-defined maximum cardinality of the CHARACTER VARYING used for the character string representation of the *ST\_PrivateRadians* attribute of an *ST\_Angle* value.
- 2) *ST\_MaxAngleAsText* is the implementation-defined maximum cardinality of the CHARACTER VARYING used for the well-known text representation of an *ST\_Angle* value.
- 3) The attribute *ST\_PrivateRadians* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST\_PrivateRadians*.

**Description**

- 1) The *ST\_Angle* type provides for public use:
  - a) a constructor method *ST\_Angle(DOUBLE PRECISION)*,
  - b) a constructor method *ST\_Angle(CHARACTER, DOUBLE PRECISION)*,
  - c) a constructor method *ST\_Angle(INTEGER, DOUBLE PRECISION)*,
  - d) a constructor method *ST\_Angle(INTEGER, INTEGER, DOUBLE PRECISION)*,
  - e) a constructor method *ST\_Angle(ST\_Point, ST\_Point, ST\_Point)*,
  - f) a constructor method *ST\_Angle(ST\_Direction, ST\_Direction)*,
  - g) a constructor method *ST\_Angle(ST\_LineString, ST\_LineString)*,
  - h) a constructor method *ST\_Angle(CHARACTER VARYING)*,
  - i) a method *ST\_Radians()*,
  - j) a method *ST\_Radians(DOUBLE PRECISION)*,
  - k) a method *ST\_Degrees()*,
  - l) a method *ST\_Degrees(DOUBLE PRECISION)*,
  - m) a method *ST\_DegreeComponent()*,
  - n) a method *ST\_MinuteComponent()*,
  - o) a method *ST\_SecondComponent()*,
  - p) a method *ST\_String()*,
  - q) a method *ST\_String(INTEGER)*,
  - r) a method *ST\_String(CHARACTER VARYING)*,
  - s) a method *ST\_Gradians()*,
  - t) a method *ST\_Gradians(DOUBLE PRECISION)*,
  - u) a method *ST\_Add(ST\_Angle)*,
  - v) a method *ST\_Subtract(ST\_Angle)*,
  - w) a method *ST\_Multiply(DOUBLE PRECISION)*,
  - x) a method *ST\_Divide(DOUBLE PRECISION)*,
  - y) a method *ST\_AsText()*,
  - z) an ordering function *ST\_OrderingCompare(ST\_Angle, ST\_Angle)*,
  - aa) an SQL Transform group *ST\_WellKnownText*,
  - ab) an SQL Transform group *ST\_WellKnownBinary*.
- 2) The attribute *ST\_PrivateRadians* contains the measurement of the angle expressed in radians.
- 3) The direction of an angle is not specified.

### 11.1.2 ST\_Angle Methods

#### Purpose

Return a specified ST\_Angle value.

#### Definition

```

CREATE CONSTRUCTOR METHOD ST_Angle
  (angle DOUBLE PRECISION)
  RETURNS ST_Angle
  FOR ST_Angle
  BEGIN
    --
    -- See Description
    --
  END

CREATE CONSTRUCTOR METHOD ST_Angle
  (units CHARACTER(1),
   angle DOUBLE PRECISION)
  RETURNS ST_Angle
  FOR ST_Angle
  BEGIN
    --
    -- See Description
    --
  END

CREATE CONSTRUCTOR METHOD ST_Angle
  (degrees INTEGER,
   minutes DOUBLE PRECISION)
  RETURNS ST_Angle
  FOR ST_Angle
  BEGIN
    --
    -- See Description
    --
  END

CREATE CONSTRUCTOR METHOD ST_Angle
  (degrees INTEGER,
   minutes INTEGER,
   seconds DOUBLE PRECISION)
  RETURNS ST_Angle
  FOR ST_Angle
  BEGIN
    --
    -- See Description
    --
  END

```

```

CREATE CONSTRUCTOR METHOD ST_Angle
(atpoint ST_Point,
 apoint ST_Point,
 anotherpoint ST_Point)
RETURNS ST_Angle
FOR ST_Angle
BEGIN
  -- check atpoint
  IF atpoint.ST_IsEmpty() = 1 THEN
    SIGNAL SQLSTATE '2FF17'
      SET MESSAGE_TEXT = 'empty point value';
  END IF;
  IF atpoint.ST_IsValid() = 0 THEN
    SIGNAL SQLSTATE '2FF18'
      SET MESSAGE_TEXT = 'point value not well formed';
  END IF;

  -- check apoint
  IF apoint.ST_IsEmpty() = 1 THEN
    SIGNAL SQLSTATE '2FF17'
      SET MESSAGE_TEXT = 'empty point value';
  END IF;
  IF apoint.ST_IsValid() = 0 THEN
    SIGNAL SQLSTATE '2FF18'
      SET MESSAGE_TEXT = 'point value not well formed';
  END IF;

  -- check anotherpoint
  IF anotherpoint.ST_IsEmpty() = 1 THEN
    SIGNAL SQLSTATE '2FF17'
      SET MESSAGE_TEXT = 'empty point value';
  END IF;
  IF anotherpoint.ST_IsValid() = 0 THEN
    SIGNAL SQLSTATE '2FF18'
      SET MESSAGE_TEXT = 'point value not well formed';
  END IF;

  -- check if points are coincident
  IF atpoint.ST_Equals(apoint) = 1 THEN
    -- points are co-located so direction is not defined
    SIGNAL SQLSTATE '2FF19'
      SET MESSAGE_TEXT = 'points are equal';
  END IF;

  IF atpoint.ST_Equals(anotherpoint) = 1 THEN
    -- points are co-located so direction is not defined
    SIGNAL SQLSTATE '2FF19'
      SET MESSAGE_TEXT = 'points are equal';
  END IF;
  --
  -- See Description
  --
END

```

```

CREATE CONSTRUCTOR METHOD ST_Angle
  (adirection ST_Direction,
   anotherdirection ST_Direction)
RETURNS ST_Angle
FOR ST_Angle
BEGIN
  --
  -- See Description
  --
END

CREATE CONSTRUCTOR METHOD ST_Angle
  (aline ST_LineString,
   anotherline ST_LineString)
RETURNS ST_Angle
FOR ST_Angle
BEGIN
  -- check if aline is valid
  IF aline.ST_NumPoints() <> 2 THEN
    -- not a line
    SIGNAL SQLSTATE '2FF20'
      SET MESSAGE_TEXT = 'linestring is not a line';
  ELSEIF aline.ST_IsClosed() = 1 THEN
    -- not a line
    SIGNAL SQLSTATE '2FF21'
      SET MESSAGE_TEXT = 'degenerate line has no direction';
  END IF;

  -- check if anotherline is valid
  IF anotherline.ST_NumPoints() <> 2 THEN
    -- not a line
    SIGNAL SQLSTATE '2FF20'
      SET MESSAGE_TEXT = 'linestring is not a line';
  ELSEIF anotherline.ST_IsClosed() = 1 THEN
    -- not a line
    SIGNAL SQLSTATE '2FF21'
      SET MESSAGE_TEXT = 'degenerate line has no direction';
  END IF;
  --
  -- See Description
  --
END

CREATE CONSTRUCTOR METHOD ST_Angle
  (awkt CHARACTER VARYING(ST_MaxAngleAsText))
RETURNS ST_Angle
FOR ST_Angle
BEGIN
  --
  -- See Description
  --
END

```

**Description**

- 1) The method *ST\_Angle*(*DOUBLE PRECISION*) takes the following input parameters:
  - a) a *DOUBLE PRECISION* value *angle*.
- 2) The null-call type preserving SQL-invoked constructor method *ST\_Angle*(*DOUBLE PRECISION*) returns an *ST\_Angle* value with the attribute *ST\_PrivateRadians* set to *angle*.

- 3) The method *ST\_Angle*(*CHARACTER*, *DOUBLE PRECISION*) takes the following input parameters:
- a *CHARACTER* value *units*,
  - a *DOUBLE PRECISION* value *angle*.
- 4) Let *IND* be the *CHARACTER* value specified by *units*.
- 5) For the null-call type preserving SQL-invoked constructor method *ST\_Angle*(*CHARACTER*, *DOUBLE PRECISION*) return an *ST\_Angle* value with:
- Case:
- If *IND* is equal to 'R', then the attribute *ST\_PrivateRadians* set to *angle*.
  - If *IND* is equal to 'D', then the attribute *ST\_PrivateRadians* set to  $(\pi * \textit{angle}) / 180$
  - If *IND* is equal to 'G', then the attribute *ST\_PrivateRadians* set to  $(\pi * \textit{angle}) / 200$ .
- 6) The method *ST\_Angle*(*INTEGER*, *DOUBLE PRECISION*) takes the following input parameters:
- an *INTEGER* value *degrees*,
  - a *DOUBLE PRECISION* value *minutes*.
- 7) Let *D* be the *INTEGER* value specified by *degrees*, and *M* the *DOUBLE PRECISION* value specified by *minutes*.
- 8) The *DOUBLE PRECISION* value *M* shall be in the range  $0 \leq M < 60$ .
- 9) For the null-call type preserving SQL-invoked constructor method *ST\_Angle*(*INTEGER*, *DOUBLE PRECISION*) return an *ST\_Angle* value with:
- Case:
- If  $D \geq 0$  (zero), then the attribute *ST\_PrivateRadians* set to the angle measurement represented by the combination of *degrees* and *minutes*, expressed in radians, equal to:  

$$(\pi * (D + M / 60)) / 180$$
  - Otherwise, the attribute *ST\_PrivateRadians* set to the angle measurement represented by the combination of *degrees* and *minutes*, expressed in radians, equal to:  

$$(\pi * (D - M / 60)) / 180$$
- 10) The method *ST\_Angle*(*INTEGER*, *INTEGER*, *DOUBLE PRECISION*) takes the following input parameters:
- an *INTEGER* value *degrees*,
  - an *INTEGER* value *minutes*,
  - a *DOUBLE PRECISION* value *seconds*.
- 11) Let *D* be the *INTEGER* value specified by *degrees*, *M* the *INTEGER* value specified by *minutes*, and *S* the *DOUBLE PRECISION* value specified by *seconds*.
- 12) The *INTEGER* value *M* shall be in the range  $0 \leq M < 60$ .
- 13) The *DOUBLE PRECISION* value *S* shall be in the range  $0 \leq S < 60$
- 14) For the null-call type preserving SQL-invoked constructor method *ST\_Angle*(*INTEGER*, *INTEGER*, *DOUBLE PRECISION*) return an *ST\_Angle* value with:
- Case:
- If  $D \geq 0$  (zero), then the attribute *ST\_PrivateRadians* set to the angle measurement represented by the combination of *degrees*, *minutes*, and *seconds*, expressed in radians, equal to:  

$$(\pi * (D + M / 60 + S / 3600)) / 180$$
  - Otherwise, the attribute *ST\_PrivateRadians* set to the angle measurement represented by the combination of *degrees*, *minutes*, and *seconds*, expressed in radians, equal to:  

$$(\pi * (D - M / 60 - S / 3600)) / 180$$

- 15) The method *ST\_Angle(ST\_Point, ST\_Point, ST\_Point)* takes the following input parameters:
- an *ST\_Point* value *atpoint*,
  - an *ST\_Point* value *apoint*,
  - an *ST\_Point* value *anotherpoint*.
- 16) Let *D1* be the direction from *atpoint* to *apoint* and let *D2* be the direction from *atpoint* to *anotherpoint*.
- 17) For the null-call type preserving SQL-invoked constructor method *ST\_Angle(ST\_Point, ST\_Point, ST\_Point)*:
- Case:
- If *atpoint* is the empty set, then an exception condition is raised: *SQL/MM Spatial exception – empty point value*.
  - If *atpoint* is not well formed, then an exception condition is raised: *SQL/MM Spatial exception – point value not well formed*.
  - If *apoint* is the empty set, then an exception condition is raised: *SQL/MM Spatial exception – empty point value*.
  - If *apoint* is not well formed, then an exception condition is raised: *SQL/MM Spatial exception – point value not well formed*.
  - If *anotherpoint* is the empty set, then an exception condition is raised: *SQL/MM Spatial exception – empty point value*.
  - If *anotherpoint* is not well formed, then an exception condition is raised: *SQL/MM Spatial exception – point value not well formed*.
  - If *atpoint* equals *apoint*, then an exception condition is raised: *SQL/MM Spatial exception – points are equal*.
  - If *atpoint* equals *anotherpoint*, then an exception condition is raised: *SQL/MM Spatial exception – points are equal*.
  - Otherwise, return an *ST\_Angle* value with the attribute *ST\_PrivateRadians* set to the lesser of:
    - the clockwise angle measured in radians from *D1* to *D2*, or
    - the clockwise angle measured in radians from *D2* to *D1*.
- 18) The method *ST\_Angle(ST\_Direction, ST\_Direction)* takes the following input parameters:
- an *ST\_Direction* value *adirection*,
  - an *ST\_Direction* value *anotherdirection*.
- 19) For the null-call type preserving SQL-invoked constructor method *ST\_Angle(ST\_Direction, ST\_Direction)* return an *ST\_Angle* value with the attribute *ST\_PrivateRadians* set to the lesser of:
- the clockwise angle measured in radians from *adirection* to *anotherdirection*, or
  - the clockwise angle measured in radians from *anotherdirection* to *adirection*.
- 20) The method *ST\_Angle(ST\_LineString, ST\_LineString)* takes the following input parameters:
- an *ST\_LineString* value *aline*,
  - an *ST\_LineString* value *anotherline*,
- 21) Let *P1* be the point value returned by *aline.ST\_StartPoint()* and *P2* be the point value returned by *aline.ST\_EndPoint()*. Let *D1* be the direction from *P1* to *P2*.
- 22) Let *P3* be the point value returned by *anotherline.ST\_StartPoint()* and *P4* be the point value returned by *anotherline.ST\_EndPoint()*. Then let *D2* be the direction from *P3* to *P4*.

- 23) For the null-call type preserving SQL-invoked constructor method *ST\_Angle(ST\_LineString, ST\_LineString)*:
- Case:
- If *aline* is not a line, then an exception condition is raised: *SQL/MM Spatial exception – linestring is not a line*.
  - If *aline* is closed, then an exception condition is raised: *SQL/MM Spatial exception – degenerate line has no direction*.
  - If *anotherline* is not a line, then an exception condition is raised: *SQL/MM Spatial exception – linestring is not a line*.
  - If *anotherline* is closed, then an exception condition is raised: *SQL/MM Spatial exception – degenerate line has no direction*.
  - Otherwise, return an *ST\_Angle* value with the attribute *ST\_PrivateRadians* set to the lesser of:
    - the clockwise angle measured in radians from *D1* to *D2*, or
    - the clockwise angle measured in radians from *D2* to *D1*
- 24) The method *ST\_Angle(CHARACTER VARYING)* takes the following input parameters:
- a CHARACTER VARYING value *awkt*.
- 25) The well-known text representation of an *ST\_Angle* value is defined by the following BNF for <angle text representation>:
- ```

<angle text representation> ::=
    ANGLE <angle text>

<angle text> ::=
    DEGREES <left paren> <degrees> <right paren>
  | GRADIANS <left paren> <gradians> <right paren>
  | RADIANS <left paren> <radians> <right paren>

<degrees> ::=
    <number>

<gradians> ::=
    <number>

<radians> ::=
    <number>

```
- <angle text representation> is the well-known text representation for an *ST\_Angle* value that is produced by <angle text>.
  - <angle text> produces the *ST\_Angle* value from <degrees>, <gradians>, or <radians>
  - Case:
    - Let *DEGREES* be the DOUBLE PRECISION value specified by <degrees>. <degrees> produces an *ST\_Angle* value as the result of the value expression: *NEW ST\_Angle('D', DEGREES)*.
    - Let *GRADIANS* be the DOUBLE PRECISION value specified by <gradians>. <gradians> produces an *ST\_Angle* value as the result of the value expression: *NEW ST\_Angle('G', GRADIANS)*.
    - Let *RADIANS* be the DOUBLE PRECISION value specified by <radians>. <radians> produces an *ST\_Angle* value as the result of the value expression: *NEW ST\_Angle('R', RADIANS)*.
- 26) The parameter *awkt* is the well-known text representation of an *ST\_Angle* value. If *awkt* is not producible in the BNF for <angle text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

- 27) The null-call type-preserving SQL-invoked constructor method *ST\_Angle*(*CHARACTER VARYING*) returns an *ST\_Angle* value represented by *awkt*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 11.1.3 ST\_Radians Methods

#### Purpose

Observe and mutate the radians attribute of an ST\_Angle value.

#### Definition

```
CREATE METHOD ST_Radians()
  RETURNS DOUBLE PRECISION
  FOR ST_Angle
  RETURN SELF.ST_PrivateRadians

CREATE METHOD ST_Radians
  (radians DOUBLE PRECISION)
  RETURNS ST_Angle
  FOR ST_Angle
  BEGIN
    IF radians IS NULL THEN
      SIGNAL SQLSTATE '2FF03'
        SET MESSAGE_TEXT = 'null argument';
    ELSE
      RETURN
        CASE
          WHEN SELF IS NULL THEN
            NULL
          ELSE
            SELF.ST_PrivateRadians(radians)
        END;
    END IF;
  END
```

#### Description

- 1) The method *ST\_Radians()* has no input parameters.
- 2) The null-call method *ST\_Radians()* returns the value of the *ST\_PrivateRadians* attribute.
- 3) The method *ST\_Radians(DOUBLE PRECISION)* takes the following input parameters:
  - a) a DOUBLE PRECISION value *radians*.
- 4) For the type preserving method *ST\_Radians(DOUBLE PRECISION)*:
 

Case:

  - a) If *radians* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
  - b) If SELF is the null value, then return the null value.
  - c) Otherwise, return the value expression: *SELF.ST\_PrivateRadians(radians)*.

#### 11.1.4 ST\_Degrees Methods

##### Purpose

Observe and mutate the radians attribute of an ST\_Angle value using decimal degrees.

##### Definition

```
CREATE METHOD ST_Degrees()
  RETURNS DOUBLE PRECISION
  FOR ST_Angle
  BEGIN
    --
    -- See Description
    --
  END

CREATE METHOD ST_Degrees
  (degrees DOUBLE PRECISION)
  RETURNS ST_Angle
  FOR ST_Angle
  BEGIN
    IF degrees IS NULL THEN
      SIGNAL SQLSTATE '2FF03'
        SET MESSAGE_TEXT = 'null argument';
    ELSE
      RETURN
        CASE
          WHEN SELF IS NULL THEN
            NULL
          ELSE
            SELF.ST_PrivateRadians(
              (ST_ApproximatePi * degrees)/180)
        END;
    END IF;
  END
```

##### Definitional Rules

- 1) *ST\_ApproximatePi* is the implementation-defined meta-variable representing pi.

##### Description

- 1) The method *ST\_Degrees()* has no input parameters.
- 2) The null-call method *ST\_Degrees()* returns the value of the *ST\_PrivateRadians* attribute converted to decimal degrees.
- 3) The method *ST\_Degrees (DOUBLE PRECISION)* takes the following input parameters:
  - a) a DOUBLE PRECISION value *degrees*.
- 4) For the type preserving method *ST\_Degrees(DOUBLE PRECISION)*:
 

Case:

  - a) If *degrees* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
  - b) If SELF is the null value, then return the null value.
  - c) Otherwise, return the value expression: *SELF.ST\_PrivateRadians((ST\_ApproximatePi \* degrees) / 180)*.

### 11.1.5 ST\_DegreeComponent Method

#### Purpose

Observe the INTEGER degrees part of the degrees, minutes, and seconds representation of the radians attribute of an ST\_Angle value.

#### Definition

```
CREATE METHOD ST_DegreeComponent ()
  RETURNS INTEGER
  FOR ST_Angle
  BEGIN
    --
    -- See Description
    --
  END
```

#### Description

- 1) The method *ST\_DegreeComponent()* has no input parameters.
- 2) The null-call method *ST\_DegreeComponent()* returns the INTEGER degree part of the degrees, minutes, and seconds representation of the value of the *ST\_PrivateRadians* attribute.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 11.1.6 ST\_MinuteComponent Method

#### Purpose

Observe the INTEGER minutes part of the degrees, minutes, and seconds representation of the radians attribute of an ST\_Angle value.

#### Definition

```
CREATE METHOD ST_MinuteComponent()  
  RETURNS INTEGER  
  FOR ST_Angle  
  BEGIN  
    --  
    -- See Description  
    --  
  END
```

#### Description

- 1) The method *ST\_MinuteComponent()* has no input parameters.
- 2) The null-call method *ST\_MinuteComponent()* returns the INTEGER minutes part of the degrees, minutes, and seconds representation of the value of the *ST\_PrivateRadians* attribute.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 11.1.7 ST\_SecondComponent Method

#### Purpose

Observe the DOUBLE PRECISION seconds part of the degrees, minutes, and seconds representation of the radians attribute of an ST\_Angle value.

#### Definition

```
CREATE METHOD ST_SecondComponent ()
  RETURNS DOUBLE PRECISION
  FOR ST_Angle
  BEGIN
    --
    -- See Description
    --
  END
```

#### Description

- 1) The method *ST\_SecondComponent()* has no input parameters.
- 2) The null-call method *ST\_SecondComponent()* returns the DOUBLE PRECISION seconds part of the degrees, minutes, and seconds representation of the value of the *ST\_PrivateRadians* attribute.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 11.1.8 ST\_String Methods

#### Purpose

Observe and mutate the radians attribute of an *ST\_Angle* value using a space separated character string of degrees, minutes, and seconds.

#### Definition

```

CREATE METHOD ST_String()
  RETURNS CHARACTER VARYING(ST_MaxAngleString)
  FOR ST_Angle
  BEGIN
    --
    -- See Description
    --
  END

CREATE METHOD ST_String
  (numdecdigits INTEGER)
  RETURNS CHARACTER VARYING(ST_MaxAngleString)
  FOR ST_Angle
  BEGIN
    --
    -- See Description
    --
  END

CREATE METHOD ST_String
  (dms CHARACTER VARYING(ST_MaxAngleString))
  RETURNS ST_Angle
  FOR ST_Angle
  BEGIN
    DECLARE degrees DOUBLE PRECISION;

    IF dms IS NULL THEN
      SIGNAL SQLSTATE '2FF03'
        SET MESSAGE_TEXT = 'null argument';
    ELSE
      RETURN
        CASE
          WHEN SELF IS NULL THEN
            NULL
          ELSE
            -- SET degrees = !! See Description
            --
            SELF.ST_PrivateRadians(
              (ST_ApproximatePi * degrees)/180)
          END;
        END IF;
    END
  
```

#### Definitional Rules

- 1) *ST\_ApproximatePi* is the implementation-defined meta-variable representing pi.
- 2) *ST\_MaxAngleString* is the implementation-defined maximum cardinality of the CHARACTER VARYING used for the character string representation of the *ST\_PrivateRadians* attribute of an *ST\_Angle* value.

#### Description

- 1) The method *ST\_String()* has no input parameters.

- 2) The null-call method *ST\_String()* returns the value of the *ST\_PrivateRadians* attribute expressed as a space separated string of degrees, minutes, and seconds. The value of seconds shall be rounded or truncated to 2 decimal digits. The choice of whether to truncate or round is implementation-defined.
- 3) The method *ST\_String(INTEGER)* takes the following input parameters:
- an INTEGER value *numdecdigits*.
- 4) The null-call method *ST\_String(INTEGER)* returns the value of the *ST\_PrivateRadians* attribute expressed as a space separated string of degrees, minutes, and seconds. The value of seconds shall be rounded or truncated to the number of decimal places indicated by the lesser of
- Case:
- numdecdigits*
  - ST\_MaxAngleString* minus (7 plus the number of digits needed to express the degrees part).
- The choice of whether to truncate or round is implementation-defined.
- 5) The maximum measure value of *numdecdigits* is implementation-defined.
- 6) The method *ST\_String(CHARACTER VARYING)* takes the following input parameters:
- a CHARACTER VARYING value *dms*.
- 7) For the type preserving method *ST\_String(CHARACTER VARYING)*:
- Case:
- If *dms* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
  - If SELF is the null value, then return the null value.
  - Otherwise:
    - Let *DEGREES* equal the DOUBLE PRECISION value obtained by converting the degrees, minutes, and seconds representation of an angle represented by *dms*, into an equivalent decimal degrees representation.
    - Return the value expression:  $SELF.ST\_PrivateRadians((ST\_ApproximatePi * DEGREES) / 180)$ .

### 11.1.9 ST\_Gradians Methods

#### Purpose

Observe and mutate the radians attribute of an ST\_Angle value using gradians.

#### Definition

```
CREATE METHOD ST_Gradians()
  RETURNS DOUBLE PRECISION
  FOR ST_Angle
  BEGIN
    --
    -- See Description
    --
  END

CREATE METHOD ST_Gradians
  (gradians DOUBLE PRECISION)
  RETURNS ST_Angle
  FOR ST_Angle
  BEGIN
    IF gradians IS NULL THEN
      SIGNAL SQLSTATE '2FF03'
        SET MESSAGE_TEXT = 'null argument';
    ELSE
      RETURN
        CASE
          WHEN SELF IS NULL THEN
            NULL
          ELSE
            SELF.ST_PrivateRadians(
              (ST_ApproximatePi * gradians) / 200)
        END;
    END IF;
  END
```

#### Definitional Rules

- 1) *ST\_ApproximatePi* is the implementation-defined meta-variable representing pi.

#### Description

- 1) The method *ST\_Gradians()* has no input parameters.
- 2) The null-call method *ST\_Gradians()* returns the value of the *ST\_PrivateRadians* attribute converted to gradians.
- 3) The method *ST\_Gradians(DOUBLE PRECISION)* takes the following input parameters:
  - a) a DOUBLE PRECISION value *gradians*.
- 4) For the type preserving method *ST\_Gradians(DOUBLE PRECISION)*:
 

Case:

  - a) If *gradians* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
  - b) If SELF is the null value, then return the null value.
  - c) Otherwise, return the value expression: *SELF.ST\_PrivateRadians((ST\_ApproximatePi \* gradians) / 200)*.

### 11.1.10 ST\_Add Method

#### Purpose

Add the value of an angle to the ST\_Angle value.

#### Definition

```
CREATE METHOD ST_Add
  (anangle ST_Angle)
  RETURNS ST_Angle
  FOR ST_Angle
  BEGIN
    --
    -- See Description
    --
  END
```

#### Description

- 1) The method *ST\_Add(ST\_Angle)* takes the following input parameters:
  - a) an *ST\_Angle* value *anangle*.
- 2) The null-call type preserving method *ST\_Add(ST\_Angle)* returns the value of SELF with the SELF.ST\_PrivateRadians attribute value set to its original value plus the value of *anangle.ST\_PrivateRadians*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 11.1.11 ST\_Subtract Method

#### Purpose

Subtract the value of an angle from the ST\_Angle value.

#### Definition

```
CREATE METHOD ST_Subtract
  (anangle ST_Angle)
  RETURNS ST_Angle
  FOR ST_Angle
  BEGIN
    --
    -- See Description
    --
  END
```

#### Description

- 1) The method *ST\_Subtract(ST\_Angle)* takes the following input parameters:
  - a) an *ST\_Angle* value *anangle*.
- 2) The null-call type preserving method *ST\_Subtract(ST\_Angle)* returns the value of SELF with the SELF.*ST\_PrivateRadians* attribute value set to its original value minus the value of *anangle.ST\_PrivateRadians*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 11.1.12 ST\_Multiply Method

#### Purpose

Multiply the ST\_Angle value by a numeric value.

#### Definition

```
CREATE METHOD ST_Multiply
  (afactor DOUBLE PRECISION)
  RETURNS ST_Angle
  FOR ST_Angle
  BEGIN
    --
    -- See Description
    --
  END
```

#### Description

- 1) The method *ST\_Multiply(DOUBLE PRECISION)* takes the following input parameters:
  - a) a DOUBLE PRECISION value *afactor*.
- 2) The null-call type preserving method *ST\_Multiply(ST\_Angle)* returns the value of SELF with the SELF.ST\_PrivateRadians attribute value set to its original value multiplied by *afactor*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

**11.1.13 ST\_Divide Method****Purpose**

Divide the ST\_Angle value by a non-zero, numeric value.

**Definition**

```
CREATE METHOD ST_Divide
  (advisor DOUBLE PRECISION)
  RETURNS ST_Angle
  FOR ST_Angle
  BEGIN
    IF advisor = 0 THEN
      SIGNAL SQLSTATE '2FF13'
      SET MESSAGE_TEXT = 'attempted division by zero';
    END IF;
    --
    -- See Description
    --
  END
```

**Description**

- 1) The method *ST\_Divide(DOUBLE PRECISION)* takes the following input parameters:
  - a) a DOUBLE PRECISION value *advisor*.
- 2) For the null-call type preserving method *ST\_Divide(ST\_Angle)*:

Case:

- a) If *advisor* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – attempted division by zero*.
- b) Otherwise, return the value of SELF with the SELF.ST\_PrivateRadians attribute value set to its original value divided by *advisor*.

#### 11.1.14 ST\_AsText Method

##### Purpose

Return the well-known text representation of an *ST\_Angle* value.

##### Definition

```
CREATE METHOD ST_AsText()  
  RETURNS CHARACTER VARYING(ST_MaxAngleAsText)  
  FOR ST_Angle  
  BEGIN  
    --  
    -- See Description  
    --  
  END
```

##### Definitional Rules

- 1) *ST\_MaxAngleAsText* is the implementation-defined maximum cardinality of the CHARACTER VARYING used for the well-known text representation of an *ST\_Angle* value.

##### Description

- 1) The method *ST\_AsText()* has no input parameters.
- 2) The null-call method *ST\_AsText()* returns a CHARACTER VARYING value containing the well-known text representation of SELF. Values shall be produced in the BNF for <angle text representation>.

**11.1.15 ST\_Angle Ordering Definition****Purpose**

Define ordering for ST\_Angle.

**Definition**

```

CREATE FUNCTION ST_OrderingCompare
  (anangle ST_Angle,
   anotherangle ST_Angle)
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT
  STATIC DISPATCH
  RETURN
  CASE
    WHEN anangle.ST_PrivateRadians <
          anotherangle.ST_PrivateRadians THEN
      -1
    WHEN anangle.ST_PrivateRadians >
          anotherangle.ST_PrivateRadians THEN
      1
    ELSE
      0
  END

CREATE ORDERING FOR ST_Angle
  ORDER FULL BY RELATIVE
  WITH FUNCTION ST_OrderingCompare(ST_Angle, ST_Angle)

```

**Description**

- 1) The function *ST\_OrderingCompare(ST\_Angle, ST\_Angle)* takes the following input parameters:
  - a) an *ST\_Angle* value *anangle*,
  - b) an *ST\_Angle* value *anotherangle*.
- 2) For the null-call function *ST\_OrderingCompare(ST\_Angle, ST\_Angle)*:
 

Case:

  - a) If *anangle.ST\_PrivateRadians < anotherangle.ST\_PrivateRadians*, then return -1,
  - b) If *anangle.ST\_PrivateRadians > anotherangle.ST\_PrivateRadians*, then return 1 (one),
  - c) Otherwise, return 0 (zero).
- 3) Use the function *ST\_OrderingCompare(ST\_Angle, ST\_Angle)* to define ordering for the *ST\_Angle* type.

### 11.1.16 SQL Transform Functions

#### Purpose

Define SQL transform functions for the *ST\_Angle* type.

#### Definition

```
CREATE TRANSFORM FOR ST_Angle
  ST_WellKnownText
    (TO SQL WITH METHOD ST_Angle(CCHARACTER VARYING(ST_MaxAngleAsText)),
     FROM SQL WITH METHOD ST_AsText())
  ST_WellKnownBinary
    (TO SQL WITH METHOD ST_Angle(DOUBLE PRECISION),
     FROM SQL WITH METHOD ST_Radians())
```

#### Definitional Rules

- 1) *ST\_MaxAngleAsText* is the implementation-defined maximum cardinality of the *CHARACTER VARYING* used for the well-known text representation of an *ST\_Angle* value.

#### Description

- 1) Use the method *ST\_Angle(CHARACTER VARYING)* and the method *ST\_AsText()* to define the transform group *ST\_WellKnownText*.
- 2) Use the method *ST\_Angle(DOUBLE PRECISION)* and the method *ST\_Radians()* to define the transform group *ST\_WellKnownBinary*.

## 11.2 ST\_Direction Type and Routines

### 11.2.1 ST\_Direction Type

#### Purpose

The ST\_Direction type is used to express direction, either as an azimuth or bearing.

#### Definition

```
CREATE TYPE ST_Direction
AS (
    ST_PrivateAngleNAzimuth ST_Angle DEFAULT NULL
)
INSTANTIABLE
NOT FINAL

CONSTRUCTOR METHOD ST_Direction
(direction DOUBLE PRECISION)
RETURNS ST_Direction
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Direction
(northsouth CHARACTER(1),
bearingangle ST_Angle,
eastwest CHARACTER(1))
RETURNS ST_Direction
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Direction
(northsouth CHARACTER(1),
azimuthangle ST_Angle)
RETURNS ST_Direction
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Direction
(frompoint ST_Point,
topoint ST_Point)
RETURNS ST_Direction
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURNS NULL ON NULL INPUT,
```

```

CONSTRUCTOR METHOD ST_Direction
  (aline ST_LineString)
  RETURNS ST_Direction
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Direction
  (awkt CHARACTER VARYING(ST_MaxDirectionAsText))
  RETURNS ST_Direction
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_Radians()
  RETURNS DOUBLE PRECISION
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_AngleNAzimuth()
  RETURNS ST_Angle
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_AngleNAzimuth
  (nazimuthangle ST_Angle)
  RETURNS ST_Direction
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  CALLED ON NULL INPUT,

METHOD ST_Astext()
  RETURNS CHARACTER VARYING(ST_MaxDirectionAsText)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

METHOD ST_RadianBearing
  (numdecdigits INTEGER)
  RETURNS CHARACTER VARYING(ST_MaxDirectionString)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

METHOD ST_DegreesBearing
  (numdecdigits INTEGER)
  RETURNS CHARACTER VARYING(ST_MaxDirectionString)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

METHOD ST_DMSBearing
  (numdecdigits INTEGER)
  RETURNS CHARACTER VARYING(ST_MaxDirectionString)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

METHOD ST_RadianNAzimuth
  (numdecdigits INTEGER)
  RETURNS CHARACTER VARYING(ST_MaxDirectionString)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

METHOD ST_DegreesNAzimuth
  (numdecdigits INTEGER)
  RETURNS CHARACTER VARYING(ST_MaxDirectionString)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

METHOD ST_DMSNAzimuth
  (numdecdigits INTEGER)
  RETURNS CHARACTER VARYING(ST_MaxDirectionString)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

METHOD ST_RadianSAzimuth
  (numdecdigits INTEGER)
  RETURNS CHARACTER VARYING(ST_MaxDirectionString)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

METHOD ST_DegreesSAzimuth
  (numdecdigits INTEGER)
  RETURNS CHARACTER VARYING(ST_MaxDirectionString)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

METHOD ST_DMSSAzimuth
  (numdecdigits INTEGER)
  RETURNS CHARACTER VARYING(ST_MaxDirectionString)
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

METHOD ST_AddAngle
  (anangle ST_Angle)
  RETURNS ST_Direction
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT,

```

```

METHOD ST_SubtractAngle
  (anangle ST_Angle)
  RETURNS ST_Direction
  SELF AS RESULT
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  RETURNS NULL ON NULL INPUT

```

### Definitional Rules

- 1) *ST\_MaxDirectionString* is the implementation-defined maximum cardinality of the CHARACTER VARYING used for the character string representation of an *ST\_Direction* value.
- 2) *ST\_MaxDirectionAsText* is the implementation-defined maximum cardinality of the CHARACTER VARYING used for the well-known text representation of an *ST\_Direction* value.
- 3) The attribute *ST\_PrivateAngleNAzimuth* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST\_PrivateAngleNAzimuth*.

### Description

- 1) The *ST\_Direction* type provides for public use:
  - a) a constructor method *ST\_Direction(DOUBLE PRECISION)*,
  - b) a constructor method *ST\_Direction(CHARACTER, ST\_Angle, CHARACTER)*,
  - c) a constructor method *ST\_Direction(CHARACTER, ST\_Angle)*,
  - d) a constructor method *ST\_Direction(ST\_Point, ST\_Point)*,
  - e) a constructor method *ST\_Direction(ST\_LineString)*,
  - f) a constructor method *ST\_Direction(CHARACTER VARYING)*,
  - g) a method *ST\_Radians()*,
  - h) a method *ST\_AngleNAzimuth()*,
  - i) a method *ST\_AngleNAzimuth(ST\_Angle)*,
  - j) a method *ST\_AsText()*,
  - k) a method *ST\_RadianBearing(INTEGER)*,
  - l) a method *ST\_DegreesBearing(INTEGER)*,
  - m) a method *ST\_DMSBearing(INTEGER)*,
  - n) a method *ST\_RadianNAzimuth(INTEGER)*,
  - o) a method *ST\_DegreesNAzimuth(INTEGER)*,

- p) a method *ST\_DMSNAzimuth*(*INTEGER*),
  - q) a method *ST\_RadianSAzimuth*(*INTEGER*),
  - r) a method *ST\_DegreesSAzimuth*(*INTEGER*),
  - s) a method *ST\_DMSSAzimuth*(*INTEGER*),
  - t) a method *ST\_AddAngle*(*ST\_Angle*),
  - u) a method *ST\_SubtractAngle*(*ST\_Angle*),
  - v) an ordering function *ST\_OrderingCompare*(*ST\_Direction*, *ST\_Direction*),
  - w) an SQL Transform group *ST\_WellKnownText*,
  - x) an SQL Transform group *ST\_WellKnownBinary*.
- 2) The attribute *ST\_PrivateAngleNAzimuth* contains the angle measured clockwise from True North.
- 3) The value of the angle in *ST\_PrivateAngleNAzimuth* shall be greater than or equal to 0 (zero) and less than 360 degrees (or 2π radians or 400 gradians). Methods mutating the value of an *ST\_Direction* (for example, *ST\_AddAngle*) shall convert the resultant to be within this range.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 11.2.2 ST\_Direction Methods

### Purpose

Return a specified ST\_Direction value.

### Definition

```

CREATE CONSTRUCTOR METHOD ST_Direction
  (direction DOUBLE PRECISION)
  RETURNS ST_Direction
  FOR ST_Direction
  BEGIN
    --
    -- See Description
    --
  END

CREATE CONSTRUCTOR METHOD ST_Direction
  (northsouth CHARACTER(1),
   bearingangle ST_Angle,
   eastwest CHARACTER(1))
  RETURNS ST_Direction
  FOR ST_Direction
  BEGIN
    --
    -- See Description
    --
  END

CREATE CONSTRUCTOR METHOD ST_Direction
  (northsouth CHARACTER(1),
   azimuthangle ST_Angle)
  RETURNS ST_Direction
  FOR ST_Direction
  BEGIN
    --
    -- See Description
    --
  END

CREATE CONSTRUCTOR METHOD ST_Direction
  (frompoint ST_Point,
   topoint ST_Point)
  RETURNS ST_Direction
  FOR ST_Direction
  BEGIN
    -- check frompoint
    IF frompoint.ST_IsEmpty() = 1 THEN
      SIGNAL SQLSTATE '2FF17'
        SET MESSAGE_TEXT = 'empty point value';
    END IF;
    IF frompoint.ST_IsValid() = 0 THEN
      SIGNAL SQLSTATE '2FF18'
        SET MESSAGE_TEXT = 'point value not well formed';
    END IF;

    -- check topoint
    IF topoint.ST_IsEmpty() = 1 THEN
      SIGNAL SQLSTATE '2FF17'
        SET MESSAGE_TEXT = 'empty point value';
    END IF;
    IF topoint.ST_IsValid() = 0 THEN

```

```

        SIGNAL SQLSTATE '2FF18'
        SET MESSAGE_TEXT = 'point value not well formed';
    END IF;

    -- check if points are coincident
    IF frompoint.ST_Equals(topoint) = 1 THEN
        -- points are co-located so direction is not defined
        SIGNAL SQLSTATE '2FF19'
        SET MESSAGE_TEXT = 'points are equal';
    END IF;
    --
    -- See Description
    --
END

CREATE CONSTRUCTOR METHOD ST_Direction
    (aline ST_LineString)
    RETURNS ST_Direction
    FOR ST_Direction
    BEGIN
        -- check if aline is valid
        IF aline.ST_NumPoints() <> 2 THEN
            -- not a line
            SIGNAL SQLSTATE '2FF20'
            SET MESSAGE_TEXT = 'linestring is not a line';
        ELSEIF aline.ST_IsClosed() = 1 THEN
            -- not a line
            SIGNAL SQLSTATE '2FF21'
            SET MESSAGE_TEXT = 'degenerate line has no direction';
        END IF;
        --
        -- See Description
        --
    END

END

CREATE CONSTRUCTOR METHOD ST_Direction
    (awkt CHARACTER VARYING(ST_MaxDirectionAsText))
    RETURNS ST_Direction
    FOR ST_Direction
    BEGIN
        --
        -- See Description
        --
    END

END

```

**Description**

- 1) The method *ST\_Direction*(*DOUBLE PRECISION*) takes the following input parameters:
  - a) a *DOUBLE PRECISION* value *direction*, measured in radians.

- 2) For the null-call type preserving SQL-invoked constructor method *ST\_Direction(DOUBLE PRECISION)* returns:
- Case:
- If *direction* is less than 0 (zero) radians or *direction* is greater than or equal to  $(2 \cdot \pi)$  radians, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
  - Otherwise, return an *ST\_Direction* value with the attribute *ST\_PrivateAngleNAzimuth* set to *NEW ST\_Angle(direction)*.
- 3) The method *ST\_Direction(CHARACTER, ST\_Angle, CHARACTER)* takes the following input parameters:
- a CHARACTER value *northsouth*,
  - an *ST\_Angle* value *bearingangle*,
  - a CHARACTER value *eastwest*.
- 4) For the null-call type preserving SQL-invoked constructor method *ST\_Direction(CHARACTER, ST\_Angle, CHARACTER)*:
- If *northsouth* is not 'N' (for North) or 'S' (for South), then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
  - If *bearingangle.ST\_Radians()* is less than 0 (zero) radians or *bearingangle.ST\_Radians()* is greater than  $(\pi/2)$  radians, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
  - If *eastwest* is not 'E' (for East) or 'W' (for West), then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
  - Otherwise, return an *ST\_Direction* value with the attribute *ST\_PrivateAngleNAzimuth* set to the *ST\_Angle* value which, when measured clockwise from True North, specifies a direction equivalent to the bearing specified by *northsouth*, *bearingangle*, and *eastwest*.
- 5) The method *ST\_Direction(CHARACTER, ST\_Angle)* takes the following input parameters:
- a CHARACTER value *northsouth*,
  - an *ST\_Angle* value *azimuthangle*.
- 6) Let *NS* be the CHARACTER value specified by *northsouth*.
- 7) For the null-call type preserving SQL-invoked constructor method *ST\_Direction(CHARACTER, ST\_Angle)*:
- Case:
- If *NS* is not 'N' (for North) or 'S' (for South), then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
  - If *azimuthangle.ST\_Radians()* is less than 0 (zero) radians or *azimuthangle.ST\_Radians()* is greater than or equal to  $(2 \cdot \pi)$  radians, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
  - Otherwise, returns an *ST\_Direction* value with:
 

Case:

    - If *NS* is equal to 'N', then the attribute *ST\_PrivateAngleNAzimuth* set to *azimuthangle*.
    - If *NS* is equal to 'S' and  $0 \text{ (zero)} \leq \text{azimuthangle.ST\_Degrees()} < 180$ , then the attribute *ST\_PrivateAngleNAzimuth* set to *azimuthangle.ST\_Add(NEW ST\_Angle('D', 180))*.
    - Otherwise, the attribute *ST\_PrivateAngleNAzimuth* set to *azimuthangle.ST\_Subtract(NEW ST\_Angle('D', 180))*.
- 8) The method *ST\_Direction(ST\_Point, ST\_Point)* takes the following input parameters:
- an *ST\_Point* value *frompoint*,
  - an *ST\_Point* value *topoint*.

9) For the null-call type preserving SQL-invoked constructor method *ST\_Direction*(*ST\_Point*, *ST\_Point*):

Case:

- a) If *frompoint* is the empty set, then an exception condition is raised: *SQL/MM Spatial exception – empty point value*.
- b) If *frompoint* is not well formed, then an exception condition is raised: *SQL/MM Spatial exception – point value not well formed*.
- c) If *topoint* is the empty set, then an exception condition is raised: *SQL/MM Spatial exception – empty point value*.
- d) If *topoint* is not well formed, then an exception condition is raised: *SQL/MM Spatial exception – point value not well formed*.
- e) If *frompoint* equals *topoint*, then an exception condition is raised: *SQL/MM Spatial exception – points are equal*.
- f) Otherwise, return an *ST\_Direction* value with:
  - i) the attribute *ST\_PrivateAngleNAzimuth* set to the *ST\_Angle* value which, when measured clockwise from True North, specifies the North azimuth direction from the *frompoint* to the *topoint*.

10) The method *ST\_Direction*(*ST\_LineString*) takes the following input parameters:

- a) an *ST\_LineString* value *aline*.

11) Let *P1* be the point value returned by *aline.ST\_StartPoint()* and *P2* be the point value returned by *aline.ST\_EndPoint()*.

12) For the null-call type preserving SQL-invoked constructor method *ST\_Direction*(*ST\_LineString*):

Case:

- a) If *aline* is not a line, then an exception condition is raised: *SQL/MM Spatial exception – linestring is not a line*.
- b) If *aline* is closed, then an exception condition is raised: *SQL/MM Spatial exception – degenerate line has no direction*.
- c) Otherwise, return an *ST\_Direction* value with:
  - i) the attribute *ST\_PrivateAngleNAzimuth* set to the *ST\_Angle* value which, when measured clockwise from True North, specifies the North azimuth direction from *P1* to *P2*.

13) The method *ST\_Direction*(*CHARACTER VARYING*) takes the following input parameters:

- a) a *CHARACTER VARYING* value *awkt*

14) The well-known text representation of an *ST\_Direction* value is defined by the following BNF for <direction text representation>:

```

<direction text representation> ::=
    DIRECTION <nazimuth text>

<nazimuth text> ::=
    <left paren> N <radians> <right paren>

<radians> ::=
    <number>
    
```

- a) <direction text representation> is the well-known text representation for an *ST\_Direction* value that is produced by <nazimuth text>.
- b) <nazimuth text> produces the *ST\_Direction* value from <radians>.
- c) Let *RADIANS* be the *DOUBLE PRECISION* value specified by <radians>. Then <radians> produces an *ST\_Direction* value as the result of the value expression: *NEW ST\_Direction('N', NEW ST\_Angle('R', RADIANS))*.

- 15) The parameter *awkt* is the well-known text representation of an *ST\_Direction* value. If *awkt* is not producible in the BNF for <direction text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
- 16) The null-call type preserving SQL-invoked constructor method *ST\_Direction*(*CHARACTER VARYING*) returns an *ST\_Direction* value represented by *awkt*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 11.2.3 ST\_Radians Method

#### Purpose

Observe the ST\_Direction value as a DOUBLE PRECISION value in radians, representing clockwise rotation from True North.

#### Definition

```
CREATE METHOD ST_Radians()  
  RETURNS DOUBLE PRECISION  
  FOR ST_Direction  
  RETURN SELF.ST_PrivateAngleNAzimuth.ST_Radians()
```

#### Description

- 1) The method *ST\_Radians()* has no input parameters.
- 2) The null-call method *ST\_Radians()* returns the value of the *ST\_PrivateRadians* attribute of the value of the *ST\_PrivateAngleNAzimuth* attribute of the *ST\_Direction* value..

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

## 11.2.4 ST\_AngleNAzimuth Methods

### Purpose

Observe and mutate the North azimuth angle attribute of an ST\_Direction value.

### Definition

```
CREATE METHOD ST_AngleNAzimuth()
  RETURNS ST_Angle
  FOR ST_Direction
  RETURN SELF.ST_PrivateAngleNAzimuth

CREATE METHOD ST_AngleNAzimuth
  (nazimuthangle ST_Angle)
  RETURNS ST_Direction
  FOR ST_Direction
  BEGIN
    IF nazimuthangle IS NULL THEN
      SIGNAL SQLSTATE '2FF03'
        SET MESSAGE_TEXT = 'null argument';
    ELSE
      RETURN
        CASE
          WHEN SELF IS NULL THEN
            NULL
          ELSE
            SELF.ST_PrivateAngleNAzimuth(nazimuthangle)
        END;
    END IF;
  END
```

### Description

- 1) The method *ST\_AngleNAzimuth()* has no input parameters.
- 2) The null-call method *ST\_AngleNAzimuth()* returns the value of the *ST\_PrivateAngleNAzimuth* attribute.
- 3) The method *ST\_AngleNAzimuth(ST\_Angle)* takes the following input parameters:
  - a) an *ST\_Angle* value *nazimuthangle*.
- 4) For the type preserving method *ST\_AngleNAzimuth(ST\_Angle)*:
 

Case:

  - a) If *nazimuthangle* is the null value, then an exception condition is raised: *SQL/MM Spatial exception – null argument*.
  - b) If SELF is the null value, then return the null value.
  - c) If *nazimuthangle.ST\_Radians()* is less than 0 (zero) radians or *nazimuthangle.ST\_Radians()* is greater than or equal to (2\*pi) radians , then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
  - d) Otherwise, return the value expression: *SELF.ST\_PrivateAngleNAzimuth(nazimuthangle)*.

### 11.2.5 ST\_AsText Method

#### Purpose

Return the well-known text representation of an ST\_Direction value.

#### Definition

```
CREATE METHOD ST_AsText()
  RETURNS CHARACTER VARYING(ST_MaxDirectionAsText)
  FOR ST_Direction
  BEGIN
    --
    -- See Description
    --
  END
```

#### Definitional Rules

- 1) *ST\_MaxDirectionAsText* is the implementation-defined maximum cardinality of the CHARACTER VARYING used for the well-known text representation of an *ST\_Direction* value.

#### Description

- 1) The method *ST\_AsText()* has no input parameters.
- 2) The null-call method *ST\_AsText()* returns a CHARACTER VARYING value containing the well-known text representation of SELF. Values shall be produced in the BNF for <direction text representation>.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 11.2.6 ST\_RadianBearing Method

#### Purpose

Observe the *ST\_Direction* value as a bearing with its angle part expressed in radians.

#### Definition

```
CREATE METHOD ST_RadianBearing
  (numdecdigits INTEGER)
  RETURNS CHARACTER VARYING(ST_MaxDirectionString)
  FOR ST_Direction
  BEGIN
    --
    -- See Description
    --
  END
```

#### Description

- 1) The method *ST\_RadianBearing(INTEGER)* takes the following input parameters:
  - a) an INTEGER value *numdecdigits*.
- 2) If *numdecdigits* is less than 0 (zero) or *numdecdigits* is greater than *ST\_MaxDirectionString* minus 6, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
- 3) Let *NAZ* be the *ST\_Angle* value equal to *SELF.ST\_PrivateAngleNAZimuth*.
- 4) Case:
  - a) If  $0 \leq \text{NAZ.ST\_Degrees}() \leq 90$ , then:
    - i) Let *NS* be the CHARACTER value equal to 'N',
    - ii) Let *A* be the *ST\_Angle* value equal to *NAZ*, and
    - iii) Let *EW* be the CHARACTER value equal to 'E'.
  - b) If  $90 < \text{NAZ.ST\_Degrees}() \leq 180$ , then:
    - i) Let *NS* be the CHARACTER value equal to 'S',
    - ii) Let *A* be the *ST\_Angle* value equal to *NEW ST\_Angle('D',180).ST\_Subtract(NAZ)*, and
    - iii) Let *EW* be the CHARACTER value equal to 'E'.
  - c) If  $180 < \text{NAZ.ST\_Degrees}() < 270$ , then:
    - i) Let *NS* be the CHARACTER value equal to 'S',
    - ii) Let *A* be the *ST\_Angle* value equal to *NAZ.ST\_Subtract(NEW ST\_Angle('D',180))*, and
    - iii) Let *EW* be the CHARACTER value equal to 'W'.
  - d) If  $270 \leq \text{NAZ.ST\_Degrees}() < 360$ , then:
    - i) Let *NS* be the CHARACTER value equal to 'N',
    - ii) Let *A* be the *ST\_Angle* value equal to *NEW ST\_Angle('D',360).ST\_Subtract(NAZ)*, and
    - iii) Let *EW* be the CHARACTER value equal to 'W'.
- 5) The null-call method *ST\_RadianBearing(INTEGER)* returns the value of the *ST\_Direction* as a bearing measured in radians and expressed as a character string concatenated from:
  - a) the *NS* CHARACTER value,
  - b) a space CHARACTER value,
  - c) the *A.ST\_Radians()* DOUBLE PRECISION value rounded or truncated to the number of decimal places indicated by *numdecdigits* and then expressed as a CHARACTER VARYING value. The choice of whether to truncate or round is implementation-defined.

- d) a space CHARACTER value,
  - e) the *EW* CHARACTER value.
- 6) The maximum measure value of *numdecdigits* is implementation-defined.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 13249-3:2003

### 11.2.7 ST\_DegreesBearing Method

#### Purpose

Observe the *ST\_Direction* value as a bearing with its angle part expressed in decimal degrees.

#### Definition

```
CREATE METHOD ST_DegreesBearing
  (numdecdigits INTEGER)
  RETURNS CHARACTER VARYING(ST_MaxDirectionString)
  FOR ST_Direction
  BEGIN
    --
    -- See Description
    --
  END
```

#### Description

- 1) The method *ST\_DegreesBearing(INTEGER)* takes the following input parameters:
  - a) an INTEGER value *numdecdigits*.
- 2) If *numdecdigits* is less than 0 (zero) or *numdecdigits* is greater than *ST\_MaxDirectionString* minus 7, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
- 3) Let *NAZ* be the *ST\_Angle* value equal to *SELF.ST\_PrivateAngleNAZimuth*.
- 4) Case:
  - a) If  $0 \leq NAZ.ST\_Degrees() \leq 90$ , then:
    - i) Let *NS* be the CHARACTER value equal to 'N',
    - ii) Let *A* be the *ST\_Angle* value equal to *NAZ*, and
    - iii) Let *EW* be the CHARACTER value equal to 'E'.
  - b) If  $90 < NAZ.ST\_Degrees() \leq 180$ , then:
    - i) Let *NS* be the CHARACTER value equal to 'S',
    - ii) Let *A* be the *ST\_Angle* value equal to *NEW ST\_Angle('D',180).ST\_Subtract(NAZ)*, and
    - iii) Let *EW* be the CHARACTER value equal to 'E'.
  - c) If  $180 < NAZ.ST\_Degrees() < 270$ , then:
    - i) Let *NS* be the CHARACTER value equal to 'S',
    - ii) Let *A* be the *ST\_Angle* value equal to *NAZ.ST\_Subtract(NEW ST\_Angle('D',180))*, and
    - iii) Let *EW* be the CHARACTER value equal to 'W'.
  - d) If  $270 \leq NAZ.ST\_Degrees() < 360$ , then:
    - i) Let *NS* be the CHARACTER value equal to 'N',
    - ii) Let *A* be the *ST\_Angle* value equal to *NEW ST\_Angle('D',360).ST\_Subtract(NAZ)*, and
    - iii) Let *EW* be the CHARACTER value equal to 'W'.
- 5) The null-call method *ST\_DegreesBearing(INTEGER)* returns the value of the *ST\_Direction* as a bearing measured in degrees and expressed as a character string concatenated from:
  - a) the *NS* CHARACTER value,
  - b) a space CHARACTER value,
  - c) the *A.ST\_Degrees()* DOUBLE PRECISION value rounded or truncated to the number of decimal places indicated by *numdecdigits* and then expressed as a CHARACTER VARYING value. The choice of whether to truncate or round is implementation-defined.