INTERNATIONAL STANDARD

ISO/IEC 13249-3

Second edition
2003-10-15

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

Part 3: **Spatial**

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

Partie 3: Spatial

Reference number
ISO/IEC 13249-3:2003(E)

Contents Page

Figures

Tables

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

2 Normative references

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

2.1 ISO/IEC JTC 1 standards

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

2.2 ISO standards

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

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

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

2.3 IEC standards

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

2.4 Other international standards

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

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

Blank page

3 Terms and definitions, notations and conventions

3.1 Terms and definitions

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

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

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

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

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

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

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

3.1.2.4
angle
the inclination to each other of two intersecting lines (which may be measured by the arc of a circle intercepted between the two lines forming the angle, the center of the circle being the point of intersection). The value of an angle can be expressed in degrees, in degrees, minutes, and seconds, in radians, or in gradians

3.1.2.5
azimuth
a representation of a geographic heading that is given by a rotation measured clockwise either from True North (North azimuth) or True South (South azimuth), having a value greater than or equal to 0 (zero) and less than 360 degrees (or 2pi 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 negative-ness 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,
dist(g_1, g_2) = Min ({ distance between p and q | $p \in g_1 \wedge q \in g_2$ })

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: $a \cap b \neq \varnothing$

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 2pi 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: $(a — b) \cup (b — a) = \varnothing$.

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 that 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
$\{ x \mid \ldots \}$	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) $\Leftrightarrow$ 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) $\Leftrightarrow$ 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) $\Leftrightarrow$ 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) $\Leftrightarrow$ Closure(a — b) $\cup$ 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_WKTToSQL: 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 ($\varnothing$). A dimensionally extended nine intersection matrix (DE-9IM) is specified in Table 2 — DE-9IM.

Table 2 — DE-9IM

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

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

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

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

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

Case:

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

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

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

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

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

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

The pattern matrix can be represented as a <character string literal> with cardinality 9 representing the DE-9IM in row major order. See Subclause 5.2, "<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:

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

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:

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

   Expressed in terms of the DE-9IM:

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

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

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

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

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

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

   Case:

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

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

   Expressed in terms of the DE-9IM:

   Case:

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

   b) Otherwise:

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

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

   Expressed in terms of the DE-9IM:

   Case:

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

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

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

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

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

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

   Expressed in terms of the DE-9IM:

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

   b) Otherwise, the null value

Expressed in terms of the DE-9IM:

Case:

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

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

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

a.ST_Overlaps(b) $\Leftrightarrow$
((Interior(a) $\cap$ Interior(b)).ST_Dimension() = 1 (one)) $\wedge$
(Interior(a) $\cap$ Exterior(b) $\neq \emptyset$) $\wedge$ (Exterior(a) $\cap$ Interior(b) $\neq \emptyset$) $\Leftrightarrow$
a.ST_Relate(b, '1*T***T**')

c) Otherwise, the null value

4.1.3 ST_Point

The ST_Point type is a subtype of ST_Geometry. The ST_Point type is instantiable. An ST_Point value is a 0-dimensional geometry and represents a single location in two-dimensional coordinate space (R^2). An ST_Point has an x coordinate value and a y coordinate value. The boundary of an ST_Point value is the empty set. ST_Point values are simple.

4.1.3.1 Methods on ST_Point

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

4.1.3.2 Functions on ST_Point

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

4.1.4 ST_Curve

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

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

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

4.1.4.1 Methods on ST_Curve

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

4.1.5 ST_LineString

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

4.1.5.1 Methods on ST_LineString

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

4.1.5.2 Functions on ST_LineString

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

4.1.6 ST_CircularString

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

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

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

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

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

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

4.1.6.1 Methods on ST_CircularString

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

4.1.6.2 Functions on ST_CircularString

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

4.1.7 ST_CompoundCurve

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

4.1.7.1 Methods on ST_CompoundCurve

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

4.1.7.2 Functions on ST_CompoundCurve

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

4.1.8 ST_Surface

The ST_Surface type is a subtype of ST_Geometry. The ST_Surface type is not instantiable. An ST_Surface value is a 2-dimensional geometry defined as a simple surface consisting of a single patch whose boundary is specified by one exterior ring and zero or more interior rings. Simple surfaces in three-dimensional coordinate space are isomorphic to planar surfaces. Polyhedral surfaces are formed by stitching together simple surfaces along their boundaries, Polyhedral surfaces in three-dimensional coordinate space may not be planar.

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

4.1.8.1 Methods on ST_Surface

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

4.1.9 ST_CurvePolygon

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

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

ST_CurvePolygon values are simple.

4.1.9.1 Methods on ST_CurvePolygon

1) ST_CurvePolygon: returns an ST_CurvePolygon value constructed from either:
   a) the well-known text representation or the well-known binary representation of an ST_CurvePolygon value;
   b) the specified ST_Curve values.
2) ST_ExteriorRing: observes and mutates the exterior ring of an ST_CurvePolygon value.
3) ST_InteriorRings: observes and mutates the collection of interior rings of an ST_CurvePolygon value.
4) ST_NumInteriorRing: returns the cardinality of the collection of interior rings of an ST_CurvePolygon value.
5) ST_InteriorRingN: returns the specified element in the collection of interior rings of an ST_CurvePolygon value.
6) ST_CurvePolyToPoly: returns an ST_Polygon value approximating the ST_CurvePolygon value.

4.1.9.2 Functions on ST_CurvePolygon

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

4.1.10 ST_Polygon

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

4.1.10.1 Methods on ST_Polygon

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

4.1.10.2 Functions on ST_Polygon

1) ST_PolyFromText: returns an ST_Polygon value, which is transformed from a CHARACTER LARGE OBJECT that represents the well-known text representation of an ST_Polygon.
2) ST_PolyFromWKB: returns an ST_Polygon value, which is transformed from a BINARY LARGE OBJECT that represents the well-known binary representation of an ST_Polygon.
3) ST_PolyFromGML: transforms a CHARACTER LARGE OBJECT containing a GML representation to an ST_Polygon value.
4) ST_BdPolyFromText: returns an ST_Polygon value, which is built from a well-known text representation of an ST_MultiLineString.
5) ST_BdPolyFromWKB: returns an ST_Polygon value, which is built from a well-known binary representation of an ST_MultiLineString.

4.1.11 ST_GeomCollection

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

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

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

4.1.11.1 Methods on ST_GeomCollection

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

4.1.11.2 Functions on ST_GeomCollection

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

4.1.12 ST_MultiPoint

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

4.1.12.1 Methods on ST_MultiPoint

1) ST_MultiPoint returns an ST_MultiPoint value constructed from either:

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

   b) the specified ST_Point values.

4.1.12.2 Functions on ST_MultiPoint

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

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

3) ST_MPointFromGML: transforms a CHARACTER LARGE OBJECT containing a GML representation to an ST_MultiPoint value.

4.1.13 ST_MultiCurve

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

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

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

4.1.13.1 Methods on ST_MultiCurve

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

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

   b) the specified ST_Curve values.

2) ST_IsClosed: tests if an ST_MultiCurve value is closed.

3) ST_Length: returns the length of an ST_MultiCurve value.

4.1.13.2 Functions on ST_MultiCurve

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

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

4.1.14 ST_MultiLineString

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

4.1.14.1 Methods on ST_MultiLineString

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

4.1.14.2 Functions on ST_MultiLineString

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

4.1.15 ST_MultiSurface

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

4.1.15.1 Methods on ST_MultiSurface

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

4.1.15.2 Functions on ST_MultiSurface

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

4.1.16 ST_MultiPolygon

The ST_MultiPolygon type is a subtype of ST_MultiSurface. The ST_MultiPolygon type is instantiable. The elements of an ST_MultiPolygon value are restricted to ST_Polygon values. The interiors of distinct elements of an ST_MultiPolygon do not intersect. The interiors of two ST_Polygon values that are elements of an ST_MultiPolygon shall not intersect. The boundaries of any two ST_Polygon values that are elements of an ST_MultiPolygon shall not cross and may touch at only a finite number of points. An ST_MultiPolygon value is defined to be topologically closed.

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

ST_MultiPolygon values are simple.

4.1.16.1 Methods on ST_MultiPolygon

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

4.1.16.2 Functions on ST_MultiPolygon

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

4.2 Spatial Reference System Type

4.2.1 ST_SpatialRefSys

The ST_SpatialRefSys type encapsulates all aspects of spatial reference systems.

4.2.1.1 Methods on ST_SpatialRefSys

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

4.2.1.2 Ordering on ST_SpatialRefSys

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

4.2.1.3 SQL Transforms on ST_SpatialRefSys

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

4.3 Angle and Direction Types

The following types are supported: ST_Angle and ST_Direction.

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

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

4.3.1 ST_Angle

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

4.3.1.1 Methods on ST_Angle

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

4.3.1.2 Ordering on ST_Angle

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

4.3.1.3 SQL Transforms on ST_Angle

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

4.3.2 ST_Direction

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

4.3.2.1 Methods on ST_Direction

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

4.3.2.2 Ordering on ST_Direction

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

4.3.2.3 SQL Transforms on ST_Direction

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

4.4 Support Routines

4.4.1 ST_Geometry ARRAY Support Routines

4.4.1.1 Support Functions

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

4.4.1.2 Supporting Procedures

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

4.4.1.3 Supporting Cast Functions

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

4.5 Tables with columns using geometry types

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

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

4.6 The Spatial Information Schema

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

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

Blank page

5 Geometry Types

5.1 ST_Geometry Type and Routines

5.1.1 ST_Geometry Type

Purpose

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

Definition

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

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

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

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

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

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

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

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

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


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

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.

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 subclauses 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.

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.

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.

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 *aunit*.
8) The values for *aunit* shall be a supported <unit name>.
9) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.
10) If the unit specified by *aunit* 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*.

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: Closure(SELF $\cap$ *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.

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: Closure(SELF ∪ *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: Closure(Closure(SELF — *ageometry*) ∪ Closure(*ageometry* — 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	*ST_Point*
C	*ST_Curve*
S	*ST_Surface*
MP	*ST_MultiPoint*
MC	*ST_MultiCurve*
MS	*ST_MultiSurface*
GC	*ST_GeomCollection*

Table 4 — Return Type Sets

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

Table 5 — Return Type Matrix for the ST_Intersection Method

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

Table 6 — Return Type Matrix for the ST_Union Method

a ∪ b								
b / a	∅	P	C	S	MP	MC	MS	GC
∅	R00	R01	R02	R03	R04	R05	R06	R07
P	R01	R20	R15	R22	R04	R17	R19	R07
C	R02	R15	R16	R21	R15	R16	R18	R07
S	R03	R22	R21	R23	R22	R21	R23	R07
MP	R04	R04	R15	R22	R04	R17	R19	R07
MC	R05	R17	R16	R21	R17	R16	R18	R07
MS	R06	R19	R18	R23	R19	R18	R23	R07
GC	R07	R07	R07	R07	R07	R07	R07	R07

Table 7 — Return Type Matrix for the ST_Difference Method

a — b								
b / a	∅	P	C	S	MP	MC	MS	GC
∅	R00	R00	R00	R00	R00	R00	R00	R00
P	R01	R09	R09	R09	R09	R09	R09	R09
C	R02	R02	R08	R08	R02	R08	R08	R08
S	R03	R03	R03	R14	R14	R14	R14	R14
MP	R04	R13	R13	R13	R13	R13	R13	R13
MC	R05	R05	R08	R08	R05	R08	R08	R08
MS	R06	R06	R06	R14	R06	R05	R06	R14
GC	R07	R12	R12	R12	R12	R12	R12	R12

Table 8 — Return Type Matrix for the ST_SymDifference Method

(a — b) ∪ (b — a)								
b / a	∅	P	C	S	MP	MC	MS	GC
∅	R00	R01	R02	R03	R04	R05	R06	R07
P	R01	R10	R15	R22	R10	R17	R19	R12
C	R02	R15	R08	R21	R15	R08	R18	R12
S	R03	R22	R21	R14	R22	R21	R14	R12
MP	R04	R10	R15	R22	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	R12	R12	R12	R12	R12	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 *aunit*.

7) The values for *aunit* shall be a supported <unit name>.

8) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.

9) If the unit specified by *aunit* is not supported by the implementation to compute the distance between SELF and ageometry, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

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()*.

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	(Interior(*SELF*) ∩ Interior(*ageometry*)).*ST_Dimension()*
2	(Interior(*SELF*) ∩ Boundary(*ageometry*)).*ST_Dimension()*
3	(Interior(*SELF*) ∩ Exterior(*ageometry*)).*ST_Dimension()*
4	(Boundary(*SELF*) ∩ Interior(*ageometry*)).*ST_Dimension()*
5	(Boundary(*SELF*) ∩ Boundary(*ageometry*)).*ST_Dimension()*
6	(Boundary(*SELF*) ∩ Exterior(*ageometry*)).*ST_Dimension()*
7	(Exterior(*SELF*) ∩ Interior(*ageometry*)).*ST_Dimension()*
8	(Exterior(*SELF*) ∩ Boundary(*ageometry*)).*ST_Dimension()*
9	(Exterior(*SELF*) ∩ Exterior(*ageometry*)).*ST_Dimension()*

See Subclause 4.1.2.1, "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****').

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
```

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**F***')
```

Description

1) The method *ST_Within(ST_Geometry)* takes the following input parameters:
   a) an *ST_Geometry* value *ageometry*.
2) The null-call method *ST_Within(ST_Geometry)* returns the result of the value expression: SELF.*ST_Relate*(*ageometry*, 'T*F**F***').

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).

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()*:
    a) If SELF is of type *ST_GeomCollection*, 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_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()*:
    a) If SELF is of type *ST_MultiPoint*, 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_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()*:
    a) If SELF is of type *ST_MultiCurve*, then return SELF.
    b) If *SELF* is not an empty set value, then an exception condition is raised: *SQL/MM Spatial exception – not an empty set*.
    c) 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()*:
    a) If SELF is of type *ST_MultiLineString*, 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_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()*:
    a) If SELF is of type *ST_MultiSurface*, 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_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.

5.1.30 ST_WKTToSQL Method

Purpose

Return a specified ST_Geometry value.

Definition

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

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum cardinality of the CHARACTER LARGE OBJECT used for the well-known text representation of an *ST_Geometry* value.

Description

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

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>.

5.1.32 ST_WKBToSQL Method

Purpose

Return a specified ST_Geometry value.

Definition

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

Definitional Rules

1) *ST_MaxGeometryAsBinary* is the implementation-defined maximum cardinality of the BINARY LARGE OBJECT used for the well-known binary representation of an *ST_Geometry* value.

Description

1) The method *ST_WKBToSQL(BINARY LARGE OBJECT)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *awkb*.

2) The parameter *awkb* is the well-known binary representation of an *ST_Geometry* value. If *awkb* is not producible in the BNF for <well-known binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

3) The null-call method *ST_WKBToSQL(BINARY LARGE OBJECT)* returns an *ST_Geometry* value represented by *awkb*.

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>.

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*.

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.

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*.

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*.

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_WKTToSQL
         (CHARACTER LARGE OBJECT(ST_MaxGeometryAsText)),
       FROM SQL WITH METHOD ST_AsText())
   ST_WellKnownBinary
      (TO SQL WITH METHOD ST_WKBToSQL
         (BINARY LARGE OBJECT(ST_MaxGeometryAsBinary)),
       FROM SQL WITH METHOD ST_AsBinary())
   ST_GML
      (TO SQL WITH METHOD ST_GMLToSQL
         (CHARACTER LARGE OBJECT(ST_MaxGeometryAsGML)),
       FROM SQL WITH METHOD ST_AsGML())
```

Definitional Rules

1) *ST_MaxGeometryAsText* is the implementation-defined maximum 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_WKTToSQL(CHARACTER LARGE OBJECT)* and the method *ST_AsText()* to define the transform group *ST_WellKnownText*.

2) Use the method *ST_WKBToSQL(BINARY LARGE OBJECT)* and the method ST_AsBinary*()* to define the transform group *ST_WellKnownBinary*.

3) Use the method *ST_GMLToSQL(CHARACTER LARGE OBJECT)* and the method *ST_AsGML()* to define the transform group *ST_GML*.

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> ::=
     <curvepolygon text representation>

<curvepolygon text representation> ::=
     CURVEPOLYGON <curvepolygon 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 <curvepolygon text representation>.

h) <curvepolygon text representation> is the well-known text representation for an *ST_CurvePolygon* value.

   Case:

   i) If <curvepolygon text representation> immediately contains a <curvepolygon text body>, then <curvepolygon text representation> produces an *ST_CurvePolygon* value specified by the immediately contained <curvepolygon text body>.

   ii) Otherwise, <curvepolygon text representation> produces an *ST_Polygon* value specified by the immediately contained <polygon text representation>.

i) <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*. 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> <wkbpoint> [ <wkbpoint binary> ]

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

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

<circularstring binary representation> ::=
     <byte order> <wkbcircularstring> [ <num> <wkbpoint 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> <wkblinearring 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 <curvepolygon binary representation>.

h) Case:

   i) If <curvepolygon binary representation> immediately contains a <num>, then <curvepolygon binary representation> produces an *ST_CurvePolygon*. Let *ACA* be an *ST_Curve* ARRAY value with cardinality of <num> that contains the *ST_Curve* values specified by the immediately contained <wkbring binary>s.

      Case:

      1) If the cardinality of *ACA* is 0 (zero), then <curvepolygon binary representation> produces an empty set of type *ST_CurvePolygon*.

      2) If the cardinality of *ACA* is 1 (one), then let *AER* be the element of *ACA*. <curvepolygon binary representation> produces an *ST_CurvePolygon* value as the result of the value expression: NEW *ST_CurvePolygon*(*AER*).

      3) Otherwise, let *AER* be the first element in *ACA* and let *AIR* be the sublist of *ACA* containing the other elements of *ACA*. <curvepolygon binary representation> produces an *ST_CurvePolygon* value as the result of the value expression: NEW *ST_CurvePolygon*(*AER, AIR*).

   ii) If <curvepolygon binary representation> immediately contains a <polygon binary representation>, then <curvepolygon binary representation> produces an *ST_Polygon* value specified by the immediately contained <polygon binary representation>.

   iii) Otherwise, <curvepolygon binary representation> produces an empty set of type *ST_CurvePolygon*.

i) Case:

   i) If <polygon binary representation> immediately contains <num>, then <polygon binary representation> is the well-known binary representation for an *ST_Polygon* value. Let *ALSA* be an *ST_LineString* ARRAY value with cardinality of <num> that contains the *ST_LineString* values specified by the immediately contained <wkblinearring binary>s.

      Case:

      1) If the cardinality of *ALSA* is 0 (zero), then <polygon binary representation> produces an empty set of type *ST_Polygon*.

      2) If the cardinality of *ALSA* is 1 (one), then let *ALS* be the element of *ALSA*. <polygon binary representation> produces an *ST_Polygon* value as the result of the value expression: NEW *ST_Polygon*(*ALS*).

      3) Otherwise, let *AER* be the first element in *ALSA* and let *AIR* be the sublist of *ALSA* containing the other elements of *ALSA*. <polygon binary representation> produces an *ST_Polygon* value as the result of the value expression: NEW *ST_Polygon*(*AER, AIR*).

   ii) Otherwise, <polygon binary representation> produces an empty set of type *ST_Polygon*.

j) Case:

   i) If <collection binary representation> immediately contains a <multipoint binary representation>, then <collection binary representation> produces an *ST_MultiPoint* value specified by the immediately contained <multipoint binary representation>.

   ii) If <collection binary representation> immediately contains a <multicurve binary representation>, then <collection binary representation> produces an *ST_MultiCurve* value specified by the immediately contained <multicurve binary representation>.

   iii) If <collection binary representation> immediately contains a <multisurface binary representation>, then <collection binary representation> produces an *ST_MultiSurface* value specified by the immediately contained <multisurface binary representation>.

   iv) Otherwise, <collection binary representation> produces an *ST_GeomCollection* value specified by the immediately contained <geometrycollection binary representation>.

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) <wkblinearring binary> produces an *ST_Point* ARRAY value with cardinality of <num> that contains the *ST_Point* values specified by the immediately contained <wkbpoint binary>s.

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
<wkbpoint>	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> s 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

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
```

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*.

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:

   Domain = $[a, b] = \{ x \in \mathrm{R} \mid a <= x <= b \}$ under a mapping $f:[a, b] \rightarrow \mathrm{R}^2$.
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 $\in$ *ST_Curve*, [a, b] = c.Domain,

   c.*ST_IsSimple()* $\Leftrightarrow$
   $(\forall x_1, x_2 \in (a, b]\ \ x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2)) \wedge (\forall x_1, x_2 \in [a, b)\ x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2))$
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 *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.

9) If the unit specified by *aunit* is not supported by the implementation to compute the length of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

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.

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.

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).

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.

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)*:
   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 method *ST_LineString(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_LineString(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*.
5) The method *ST_LineString(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_LineString(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).
7) The method *ST_LineString(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
8) For the null-call type preserving SQL-invoked constructor method *ST_LineString(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*.
9) The method *ST_LineString(ST_Point ARRAY)* takes the following input parameters:
   a) an *ST_Point* ARRAY value *apointarray*.
10) The null-call type preserving SQL-invoked constructor method *ST_LineString(ST_Point ARRAY)* returns an *ST_LineString* value with:
    a) The spatial reference system identifier set to 0 (zero).

b) Using the method *ST_Points(ST_Point ARRAY),* the *ST_PrivatePoints* attribute set to *apointarray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

11) The method *ST_LineString(ST_Point ARRAY, INTEGER)* takes the following input parameters:

a) an *ST_Point* ARRAY value *apointarray*,

b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_LineString(ST_Point ARRAY, INTEGER)* returns an *ST_LineString* value with:

a) The spatial reference system identifier set to *ansrid*.

b) Using the method *ST_Points(ST_Point ARRAY),* the *ST_PrivatePoints* attribute set to *apointarray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

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*.

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.

7.2.5 ST_PointN Method

Purpose

Return the specified element in the ST_PrivatePoints attribute of an ST_LineString value.

Definition

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

Description

1) The method *ST_PointN(INTEGER)* takes the following input parameters:

   a) an INTEGER value *aposition*.

2) For the null-call method *ST_PointN(INTEGER)*:

   Case:

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

   b) If *aposition* is less than 1 (one) or greater than the cardinality of the *ST_PrivatePoints* attribute, then:

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

      ii) Return the null value.

   c) Otherwise, return an *ST_Point* value at element *aposition* in the *ST_PrivatePoints* attribute of SELF.

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].

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)*:

   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 method *ST_CircularString(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_CircularString(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*.

5) The method *ST_CircularString(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_CircularString(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 <circularstring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) 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) a BINARY LARGE OBJECT value *awkb*,

   b) an INTEGER value *ansrid*.

8) For the null-call type preserving SQL-invoked constructor method *ST_CircularString(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 <circularstring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) 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:

   a) an *ST_Point* ARRAY value *apointarray*

10) The null-call type preserving SQL-invoked constructor method *ST_CircularString(ST_Point ARRAY)* returns an *ST_CircularString* value with:

   a) The spatial reference system identifier set to 0 (zero).

b) Using the method *ST_Points(ST_Point ARRAY),* the attribute *ST_PrivatePoints* array set to *apointarray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

11) The method *ST_CircularString(ST_Point ARRAY, INTEGER)* takes the following input parameters:

a) an *ST_Point* ARRAY value *apointarray*,

b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_CircularString(ST_Point ARRAY, INTEGER)* returns an *ST_CircularString* value with:

a) The spatial reference system identifier set to *ansrid*.

b) Using the method *ST_Points(ST_Point ARRAY),* the attribute *ST_PrivatePoints* array set to *apointarray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

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*.

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.

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()*].

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 <circularstring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
   b) 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 <circularstring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
   b) 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),
       ansrid INTEGER)
      RETURNS ST_CompoundCurve
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

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

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

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[acounter] IS OF (ST_CompoundCurve) THEN
            SIGNAL SQLSTATE '2FF02'
               SET MESSAGE_TEXT = 'invalid argument';
         END IF;
         SET counter = counter + 1;
      END WHILE;
      -- If the start point of any curve is not coincident with the end
      -- point of the previous curve, then raise an exception
      SET counter = 2;
      WHILE counter <= CARDINALITY(acurvearray) DO
            IF acurvearray[counter].ST_StartPoint() <>
               acurvearray[counter-1].ST_EndPoint() THEN
            SIGNAL SQLSTATE '2FF11'
               SET MESSAGE_TEXT = 'non-contiguous curves';
         END IF;
         SET counter = counter + 1;
      END WHILE;
      -- If SELF is the null value, then return the null value. Otherwise,
      -- return an ST_CompoundCurve value with the ST_PrivateCurves
```

```
         -- 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.

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].

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

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.

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 *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.

9) If the unit specified by *aunit* is not supported by the implementation to compute the area of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

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 *aunit* shall be a supported <unit name>.
8) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.
9) If the unit specified by *aunit* is not supported by the implementation to compute the length of the boundary of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

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.

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),
       ansrid INTEGER)
      RETURNS ST_CurvePolygon
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

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

   CONSTRUCTOR METHOD ST_CurvePolygon
      (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
       ansrid INTEGER)
      RETURNS ST_CurvePolygon
      SELF AS RESULT
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,
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 <curvepolygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) 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 <curvepolygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.

   b) 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 <curvepolygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) 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) a BINARY LARGE OBJECT value *awkb*,

   b) an INTEGER value *ansrid*.

8) For the null-call type preserving SQL-invoked constructor method *ST_CurvePolygon(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 <curvepolygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.

   b) 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:

   b) an *ST_Curve* value *acurve*.

10) The null-call type preserving SQL-invoked constructor method *ST_CurvePolygon(ST_Curve)* returns an *ST_CurvePolygon* value with:

   a) The spatial reference system identifier set to 0 (zero).

   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 an empty *ST_Curve* ARRAY value.

11) The method *ST_CurvePolygon(ST_Curve, ST_Curve ARRAY)* takes the following input parameters:

   a) an *ST_Curve* value *acurve*,

   b) an *ST_Curve* ARRAY value *acurvearray*.

12) The null-call type preserving SQL-invoked constructor method *ST_CurvePolygon(ST_Curve, ST_Curve ARRAY)* returns an *ST_CurvePolygon* value with:

   a) The spatial reference system identifier set to 0 (zero).

   b) Using the method *ST_ExteriorRing(ST_Curve),* the *ST_PrivateExteriorRing* attribute set to *acurve*, the *ST_PrivateDimension* attribute set to 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*.

13) The method *ST_CurvePolygon(ST_Curve, INTEGER)* takes the following input parameters:

   a) an *ST_Curve* value *acurve*,

   b) 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:

   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 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*.

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;
            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.

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.

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 <curvepolygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
   b) 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 <curvepolygon text representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known text representation*.
   b) 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 <curvepolygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
   b) 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 <curvepolygon binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
   b) 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
   (alinestring ST_LineString,
    alinestringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_Polygon
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Polygon
   (alinestring ST_LineString,
    ansrid INTEGER)
   RETURNS ST_Polygon
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

CONSTRUCTOR METHOD ST_Polygon
   (alinestring ST_LineString,
    alinestringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements],
    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
    (alinestring ST_LineString)
   RETURNS ST_Polygon
   FOR ST_Polygon
   RETURN SELF.ST_SRID(0).ST_ExteriorRing(alinestring).
      ST_InteriorRings(CAST(ARRAY[] AS
         ST_LineString ARRAY[ST_MaxGeometryArrayElements]))

CREATE CONSTRUCTOR METHOD ST_Polygon
    (alinestring ST_LineString,
     alinestringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_Polygon
   FOR ST_Polygon
   RETURN SELF.ST_SRID(0).ST_ExteriorRing(alinestring).
      ST_InteriorRings(alinestringarray)

CREATE CONSTRUCTOR METHOD ST_Polygon
    (alinestring ST_LineString,
     ansrid INTEGER)
   RETURNS ST_Polygon
   FOR ST_Polygon
   RETURN SELF.ST_SRID(ansrid).ST_ExteriorRing(alinestring).
      ST_InteriorRings(CAST(ARRAY[] AS
         ST_LineString ARRAY[ST_MaxGeometryArrayElements]))
```

```
CREATE CONSTRUCTOR METHOD ST_Polygon
    (alinestring ST_LineString,
     alinestringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements],
     ansrid INTEGER)
    RETURNS ST_Polygon
    FOR ST_Polygon
    RETURN SELF.ST_SRID(ansrid).ST_ExteriorRing(alinestring).
        ST_InteriorRings(alinestringarray)
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum 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) a BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
8) For the null-call type preserving SQL-invoked constructor method *ST_Polygon(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*.
9) The method *ST_Polygon(ST_LineString)* takes the following input parameters:
   b) an *ST_LineString* value *alinestring*.
10) The null-call type preserving SQL-invoked constructor method *ST_Polygon(ST_LineString)* returns an *ST_Polygon* value with:
   a) The spatial reference system identifier set to 0 (zero).
   b) Using the method *ST_ExteriorRing(ST_Curve),* the *ST_PrivateExteriorRing* attribute set to *alinestring*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to 2.
   c) Using the method *ST_InteriorRings(ST_Curve ARRAY),* the *ST_PrivateInteriorRings* attribute set to an empty *ST_LineString* ARRAY value.
11) The method *ST_Polygon(ST_LineString, ST_LineString ARRAY)* takes the following input parameters:
   a) an *ST_LineString* value *alinestring*,
   b) an *ST_LineString* ARRAY value *alinestringarray*.
12) The null-call type preserving SQL-invoked constructor method *ST_Polygon(ST_LineString, ST_LineString ARRAY)* returns an *ST_Polygon* value with:
   a) The spatial reference system identifier set to 0 (zero).
   b) Using the method *ST_ExteriorRing(ST_Curve),* the *ST_PrivateExteriorRing* attribute set to *alinestring*, the *ST_PrivateDimension* attribute set to 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*.
13) The method *ST_Polygon(ST_LineString, INTEGER)* takes the following input parameters:
   a) an *ST_LineString* value *alinestring*,
   b) 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:
   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 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*.

8.3.3 ST_ExteriorRing Methods

Purpose

Observe and mutate the ST_PrivateExteriorRing attribute of an ST_Polygon value.

Definition

```
CREATE METHOD ST_ExteriorRing()
   RETURNS ST_LineString
   FOR ST_Polygon
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CAST(SELF.ST_PrivateExteriorRing AS ST_LineString)
      END

CREATE METHOD ST_ExteriorRing
   (acurve ST_Curve)
   RETURNS ST_Polygon
   FOR ST_Polygon
   BEGIN
      -- If acurve is not an ST_LineString, then raise an exception
      IF acurve IS NOT OF (ST_LineString) THEN
         SIGNAL SQLSTATE '2FF12'
            SET MESSAGE_TEXT = 'curve value is not a linestring value';
      END IF;
      RETURN (SELF AS ST_CurvePolygon).ST_ExteriorRing(acurve);
   END
```

Description

1) The method *ST_ExteriorRing()* has no input parameters.

2) For the null-call method *ST_ExteriorRing()*:

   Case:

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

   b) Otherwise, return the *ST_PrivateExteriorRing* attribute of SELF.

3) The method *ST_ExteriorRing(ST_Curve)* takes the following input parameters:

   a) an *ST_Curve* value *alinestring*.

4) For the type preserving method *ST_ExteriorRing(ST_Curve)*:

   Case:

   a) If *acurve* is not an ST_LineString value, then an exception condition is raised: *SQL/MM Spatial exception – curve value is not a linestring value*.

   b) Otherwise, return an *ST_Polygon* value as a result of the value expression: (SELF AS *ST_CurvePolygon*).*ST_ExteriorRing(acurve)*

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).*

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*.

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 <multilinestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
   b) Use *ST_MLineFromWKB(BINARY LARGE OBJECT)* to transform *awkb* to an *ST_MultiLineString* value, *AMLS*.
   c) If any *ST_LineString* value in *AMLS* is not a linear ring, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
   d) Using an implementation-dependent algorithm, an exterior linear ring, *ELR*, and an array of zero or more interior rings, *AILR*, are determined from the array of linear rings in *AMLS*.

e) Return an *ST_Polygon* value with:

   i) The spatial reference system identifier set to *ansrid*.

   ii) Using the method *ST_ExteriorRing(ST_LineString),* the *ST_PrivateExteriorRing* attribute set to *ELR*.

   iii) Using the method *ST_InteriorRings(ST_LineString ARRAY),* the *ST_PrivateInteriorRings* attribute set to *AILR*.

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.

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)*:

   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 *ansrid*.

9) The method *ST_GeomCollection(ST_Geometry)* takes the following input parameters:

   a) an *ST_Geometry* value *ageometry*.

10) The null-call type preserving SQL-invoked constructor method *ST_GeomCollection(ST_Geometry)* returns an *ST_GeomCollection* value with:

   a) The spatial reference system identifier set to the spatial reference system identifier of *ageometry*.

   b) Using the method *ST_Geometries(ST_Geometry ARRAY),* the *ST_PrivateGeometries* attribute set to ARRAY[*ageometry*], the *ST_PrivateDimension* attribute set to *ST_MaxDimension*(ARRAY[*ageometry*]), and the *ST_PrivateCoordinateDimension* attribute set to 2.

11) The method *ST_GeomCollection(ST_Geometry ARRAY)* takes the following input parameters:

   a) 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:

   a) The spatial reference system identifier set to the value expression: *ST_CheckSRID*(*ageometryarray*).

   b) Using the method *ST_Geometries(ST_Geometry ARRAY),* the *ST_PrivateGeometries* attribute set to *ageometryarray*, the *ST_PrivateDimension* attribute set to *ST_MaxDimension*(ARRAY[*ageometry*]), and the *ST_PrivateCoordinateDimension* attribute set to 2.

13) The method *ST_GeomCollection(ST_Geometry, INTEGER)* takes the following input parameters:

   a) an *ST_Geometry* value *ageometry*,

   b) 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:

   a) The spatial reference system identifier set to *ansrid*.

   b) Using the method *ST_Geometries(ST_Geometry ARRAY),* the *ST_PrivateGeometries* attribute set to ARRAY[*ageometry*], the *ST_PrivateDimension* attribute set to *ST_MaxDimension*(ARRAY[*ageometry*]), and the *ST_PrivateCoordinateDimension* attribute set to 2.

15) The method *ST_GeomCollection(ST_Geometry ARRAY, INTEGER)* takes the following input parameters:

   a) an *ST_Geometry* ARRAY value *ageometryarray*,

   b) an INTEGER value *ansrid*.

16) The null-call type preserving SQL-invoked constructor method *ST_GeomCollection(ST_Geometry ARRAY, INTEGER)* returns an *ST_GeomCollection* value with:

   a) The spatial reference system identifier set to *ansrid*.

   b) Using the method *ST_Geometries(ST_Geometry ARRAY),* the *ST_PrivateGeometries* attribute set to *ageometryarray*, the *ST_PrivateDimension* attribute set to *ST_MaxDimension*(ARRAY[*ageometry*]), and the *ST_PrivateCoordinateDimension* attribute set to 2.

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*.

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.

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.

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)*:

   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 method *ST_MultiPoint(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_MultiPoint(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*.

5) The method *ST_MultiPoint(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_MultiPoint(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).

7) The method *ST_MultiPoint(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *awkb*,

   b) an INTEGER value *ansrid*.

8) For the null-call type preserving SQL-invoked constructor method *ST_MultiPoint(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) The method *ST_MultiPoint(ST_Point ARRAY)* takes the following input parameters:

   a) an *ST_Point* ARRAY value *apointarray*.

10) The null-call type preserving SQL-invoked constructor method *ST_MultiPoint(ST_Point ARRAY)* returns an *ST_MultiPoint* value with:

   a) The spatial reference system identifier set to 0 (zero).

b) Using the method *ST_Geometries(ST_Geometry ARRAY),* the *ST_PrivateGeometries* attribute set to *apointarray*, the *ST_PrivateDimension* attribute set to 0 (zero), and the *ST_PrivateCoordinateDimension* attribute set to 2.

11) The method *ST_MultiPoint(ST_Point ARRAY, INTEGER)* takes the following input parameters:

  a) an *ST_Point* ARRAY value *apointarray*,

  b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_MultiPoint(ST_Point ARRAY, INTEGER)* returns an *ST_MultiPoint* value with:

  a) The spatial reference system identifier set to *ansrid*.

  b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the *ST_PrivateGeometries* attribute set to *apointarray*, the *ST_PrivateDimension* attribute set to 0 (zero), and the *ST_PrivateCoordinateDimension* attribute set to 2.

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*.

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)*:

   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 method *ST_MultiCurve(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_MultiCurve(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*.

5) The method *ST_MultiCurve(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_MultiCurve(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).

7) The method *ST_MultiCurve(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *awkb*,

   b) an INTEGER value *ansrid*.

8) For the null-call type preserving SQL-invoked constructor method *ST_MultiCurve(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) The method *ST_MultiCurve(ST_Curve ARRAY)* takes the following input parameters:

   a) an *ST_Curve* ARRAY value *acurvearray*.

10) The null-call type preserving SQL-invoked constructor method *ST_MultiCurve(ST_Curve ARRAY)* returns an *ST_MultiCurve* value with:

   a) The spatial reference system identification set to 0 (zero).

   b) Using the method *ST_Geometries(ST_Geometry ARRAY),* the attribute *ST_PrivateGeometries* set to *acurvearray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

11) The method *ST_MultiCurve(ST_Curve ARRAY, INTEGER)* takes the following input parameters:

   a) an *ST_Curve* ARRAY value *acurvearray*,

   b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_MultiCurve(ST_Curve ARRAY, INTEGER)* returns an *ST_MultiCurve* value with:

   a) The spatial reference system identification set to *ansrid*.

   b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the attribute *ST_PrivateGeometries* set to *acurvearray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

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).

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 *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 sum of the *ST_Length(aunit)* 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 *aunit*.
7) The values for *aunit* shall be a supported <unit name>.
8) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.
9) If the unit specified by *aunit* is not supported by the implementation to compute the sum of the *ST_Length(aunit)* values of each element in *ST_PrivateGeometries* attribute of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

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*.

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
   (alinestringarray ST_LineString
      ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiLineString
   SELF AS RESULT
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT,

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

OVERRIDING METHOD ST_Geometries()
   RETURNS ST_LineString ARRAY[ST_MaxGeometryArrayElements],

OVERRIDING METHOD ST_Geometries
   (ageometryarray ST_Geometry
      ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiLineString
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum 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.

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
    (alinestringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiLineString
   FOR ST_MultiLineString
   RETURN SELF.ST_SRID(0).ST_Geometries(alinestringarray)

CREATE CONSTRUCTOR METHOD ST_MultiLineString
    (alinestringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements],
     ansrid INTEGER)
   RETURNS ST_MultiLineString
   FOR ST_MultiLineString
   RETURN SELF.ST_SRID(ansrid).ST_Geometries(alinestringarray)
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

2) *ST_MaxGeometryAsBinary* is the implementation-defined maximum 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)*:

   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 method *ST_MultiLineString(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_MultiLineString(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*.

5) The method *ST_MultiLineString(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_MultiLineString(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).

7) The method *ST_MultiLineString(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *awkb*,

   b) an INTEGER value *ansrid*.

8) For the null-call type preserving SQL-invoked constructor method *ST_MultiLineString(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) The method *ST_MultiLineString(ST_LineString ARRAY)* takes the following input parameters:

   a) an *ST_LineString* ARRAY value *alinestringarray*.

10) The null-call type preserving SQL-invoked constructor method *ST_MultiLineString(ST_LineString ARRAY)* returns an *ST_MultiLineString* value with:

   a) The spatial reference system identifier set to 0 (zero).

   b) Using the method *ST_Geometries(ST_Geometry ARRAY),* the *ST_PrivateGeometries* attribute set to *alinestringarray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

11) The method *ST_MultiLineString(ST_LineString ARRAY, INTEGER)* takes the following input parameters:

   a) an *ST_LineString* ARRAY value *alinestringarray*,

   b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_MultiLineString(ST_LineString ARRAY, INTEGER)* returns an *ST_MultiLineString* value with:

   a) The spatial reference system identifier set to *ansrid*.

   b) Using the method *ST_Geometries(ST_Geometry ARRAY),* the *ST_PrivateGeometries* attribute set to *alinestringarray*, the *ST_PrivateDimension* attribute set to 1 (one), and the *ST_PrivateCoordinateDimension* attribute set to 2.

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*.

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)*:

   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 method *ST_MultiSurface(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_MultiSurface(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*.

5) The method *ST_MultiSurface(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_MultiSurface(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).

7) The method *ST_MultiSurface(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *awkb*,

   b) an INTEGER value *ansrid*.

8) For the null-call type preserving SQL-invoked constructor method *ST_MultiSurface(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) The method *ST_MultiSurface(ST_Surface ARRAY)* takes the following input parameters:

   a) an *ST_Surface* ARRAY value *asurfacearray*.

10) The null-call type preserving SQL-invoked constructor method *ST_MultiSurface(ST_Surface ARRAY)* returns an *ST_MultiSurface* value with:

    a) The spatial reference system identification set to 0 (zero),

b) Using the method *ST_Geometries(ST_Geometry ARRAY),* the attribute *ST_PrivateGeometries* set to *asurfacearray*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to 2.

11) The method *ST_MultiSurface(ST_Surface ARRAY, INTEGER)* takes the following input parameters:

a) an *ST_Surface* ARRAY value *asurfacearray*,

b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_MultiSurface(ST_Surface ARRAY, INTEGER)* returns an *ST_MultiSurface* value with:

a) The spatial reference system identification set to *ansrid*.

b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the attribute *ST_PrivateGeometries* set to *asurfacearray*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to 2.

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 *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 sum of the *ST_Area(aunit)* 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 *aunit*.
7) The values for *aunit* shall be a supported <unit name>.
8) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.
9) If the unit specified by *aunit* is not supported by the implementation to compute sum of the *ST_Area(aunit)* values of each element in the *ST_PrivateGeometries* attribute of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

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 *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 sum of the *ST_Perimeter(aunit)* 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 *aunit*.

7) The values for *aunit* shall be a supported <unit name>.

8) The value for *aunit* is a supported <unit name> if and only if the value of *aunit* is equal to the value of the UNIT_NAME column of one of the rows where the value of the UNIT_TYPE column is equal to 'LINEAR' in the ST_UNITS_OF_MEASURE view.

9) If the unit specified by *aunit* is not supported by the implementation to compute sum of the *ST_Perimeter(aunit)* values of each element in the *ST_PrivateGeometries* attribute of SELF, then an exception condition is raised: *SQL/MM Spatial exception – unsupported unit specified*.

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
```

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) = $\varnothing$

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) $c_i \cap c_j = \{ p_1, ..., p_k \mid p_i \in$ *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)*:

   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 method *ST_MultiPolygon(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_MultiPolygon(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*.

5) The method *ST_MultiPolygon(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_MultiPolygon(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).

7) The method *ST_MultiPolygon(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a BINARY LARGE OBJECT value *awkb*,

   b) an INTEGER value *ansrid*.

8) For the null-call type preserving SQL-invoked constructor method *ST_MultiPolygon(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) The method *ST_MultiPolygon(ST_Polygon ARRAY)* takes the following input parameters:

   a) an *ST_Polygon* value *apolygonarray*.

10) The null-call type preserving SQL-invoked constructor method *ST_MultiPolygon(ST_Polygon ARRAY)* returns an *ST_MultiPolygon* value with:

   a) The spatial reference system identifier set to 0 (zero).

b) Using the method *ST_Geometries(ST_Geometry ARRAY),* the *ST_PrivateGeometries* attribute set to *apolygonarray*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to 2.

11) The method *ST_MultiPolygon(ST_Polygon ARRAY, INTEGER)* takes the following input parameters:

a) an *ST_Polygon* ARRAY value *apolygonarray*,

b) an INTEGER value *ansrid*.

12) The null-call type preserving SQL-invoked constructor method *ST_MultiPolygon(ST_Polygon ARRAY, INTEGER)* returns an *ST_MultiPolygon* value with:

a) The spatial reference system identifier set to *ansrid*.

b) Using the method *ST_Geometries(ST_Geometry ARRAY),* the *ST_PrivateGeometries* attribute set to *apolygonarray*, the *ST_PrivateDimension* attribute set to 2, and the *ST_PrivateCoordinateDimension* attribute set to 2.

9.6.3 ST_Geometries Methods

Purpose

Observe and mutate the ST_PrivateGeometries attribute of an ST_MultiPolygon value.

Definition

```
CREATE METHOD ST_Geometries()
   RETURNS ST_Polygon ARRAY[ST_MaxGeometryArrayElements]
   FOR ST_MultiPolygon
   RETURN
      CASE
         WHEN SELF.ST_IsEmpty() = 1 THEN
            NULL
         ELSE
            CAST(SELF.ST_PrivateGeometries AS
               ST_Polygon ARRAY[ST_MaxGeometryArrayElements])
      END

CREATE METHOD ST_Geometries
   (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiPolygon
   FOR ST_MultiPolygon
   BEGIN
      DECLARE acounter INTEGER;
      DECLARE bcounter INTEGER;
      DECLARE apolygonarray ST_Polygon ARRAY[ST_MaxGeometryArrayElements];

      -- Cast ageometryarray to an ST_Polygon ARRAY
      SET apolygonarray = CAST(ageometryarray AS
         ST_Polygon ARRAY[ST_MaxGeometryArrayElements]);
      -- If any two polygons intersect with the dimension of the result
      -- greater than 0 (zero), then raise an exception.
      SET acounter = 1;
      WHILE acounter <= CARDINALITY(apolygonarray)-1 DO
         SET bcounter = acounter+1;
         WHILE bcounter <= CARDINALITY(apolygonarray) DO
            IF apolygonarray[acounter].ST_Intersection(
               apolygonarray[bcounter]).ST_Dimension() > 0 THEN
               SIGNAL SQLSTATE '2FF02'
                  SET MESSAGE_TEXT = 'invalid argument';
            END IF;
            SET bcounter = bcounter + 1;
         END WHILE;
         SET acounter = acounter + 1;
      END WHILE;
      -- If SELF is the null value, then return the null value. Otherwise,
      -- return an ST_MultiPolygon value containing apolygonarray.
      RETURN
         CASE
            WHEN SELF IS NULL THEN
               NULL
            ELSE
               (SELF AS ST_MultiSurface).
                  ST_Geometries(apolygonarray)
         END;
   END
```

Definitional Rules

1) *ST_MaxGeometryArrayElements* is the implementation-defined maximum cardinality of an array of *ST_Geometry* values.

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*.

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*(*awkt*, 0).
3) The function *ST_BdMPolyFromWKB(BINARY LARGE OBJECT, INTEGER)* takes the following input parameters:
   a) an BINARY LARGE OBJECT value *awkb*,
   b) an INTEGER value *ansrid*.
4) For the null-call function *ST_BdMPolyFromWKB(BINARY LARGE OBJECT, INTEGER)*:
   a) The parameter *awkb* is the well-known binary representation of an *ST_MultiLineString* value. If *awkb* is not producible in the BNF for <multilinestring binary representation>, then it is implementation-defined whether or not the following exception condition is raised: *SQL/MM Spatial Exception – invalid well-known binary representation*.
   b) Use *ST_MLineFromWKB(BINARY LARGE OBJECT)* to transform *awkb* to an *ST_MultiLineString* value, *AMLS*.
   c) If any *ST_LineString* value in *AMLS* is not a linear ring, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.
   d) Using an implementation-dependent algorithm, an array of *ST_Polygon* values, *APA*, is determined from the array of linear rings in *AMLS*.
   e) Return an *ST_MultiPolygon* value with:
      i) The spatial reference system identifier set to *ansrid*.

ii) Using the method *ST_Geometries(ST_Geometry ARRAY),* the *ST_PrivateGeometries* attribute set to *APA*.

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>.

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.

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,
     anothersrs ST_SpatialRefSys)
   RETURNS INTEGER
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN
      CASE
         WHEN ansrs.ST_Equals(anothersrs) = 1 THEN
            0
         ELSE
            1
      END

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

Description

1) The function *ST_OrderingEquals(ST_SpatialRefSys, ST_SpatialRefSys)* takes the following input parameters:

   a) an *ST_SpatialRefSys* value *ansrs*,

   b) an *ST_SpatialRefSys* value *anothersrs*.

2) For the null-call function *ST_OrderingEquals(ST_SpatialRefSys, ST_SpatialRefSys)*:

   Case:

   a) If the value expression *ansrs.ST_Equals(anothersrs)* is 1 (one), then return 0 (zero).

   b) Otherwise, return 1 (one).

3) Use the function *ST_OrderingEquals(ST_SpatialRefSys, ST_SpatialRefSys)* to define ordering for the *ST_SpatialRefSys* type.

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*.

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

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) a CHARACTER value *units*,
   b) a DOUBLE PRECISION value *angle*.
4) Let *IND* be the CHARACTER value specified by *units*.
5) For the null-call type preserving SQL-invoked constructor method *ST_Angle(CHARACTER, DOUBLE PRECISION)* return an *ST_Angle* value with:

   Case:
   a) If *IND* is equal to 'R', then the attribute *ST_PrivateRadians* set to *angle*.
   b) If *IND* is equal to 'D', then the attribute *ST_PrivateRadians* set to (pi * *angle*) / 180
   c) If *IND* is equal to 'G', then the attribute *ST_PrivateRadians* set to (pi * *angle*) / 200.
6) The method *ST_Angle(INTEGER, DOUBLE PRECISION)* takes the following input parameters:
   a) an INTEGER value *degrees*,
   b) a DOUBLE PRECISION value *minutes*.
7) Let *D* be the INTEGER value specified by *degrees*, and *M* the DOUBLE PRECISION value specified by *minutes*.
8) The DOUBLE PRECISION value *M* shall be in the range 0 <= *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:
   a) If *D* >= 0 (zero), then the attribute *ST_PrivateRadians* set to the angle measurement represented by the combination of *degrees* and *minutes,* expressed in radians, equal to:

      (pi * (*D* + *M* / 60)) / 180
   b) Otherwise, the attribute *ST_PrivateRadians* set to the angle measurement represented by the combination of *degrees* and *minutes*, expressed in radians, equal to:

      (pi * (*D* - *M* / 60)) / 180
10) The method *ST_Angle(INTEGER, INTEGER, DOUBLE PRECISION)* takes the following input parameters:
    a) an INTEGER value *degrees*,
    b) an INTEGER value *minutes*,
    c) a DOUBLE PRECISION value *seconds*.
11) Let *D* be the INTEGER value specified by *degrees*, *M* the INTEGER value specified by *minutes*, and *S* the DOUBLE PRECISION value specified by *seconds*.
12) The INTEGER value *M* shall be in the range 0 <= *M* < 60.
13) The DOUBLE PRECISION value *S* shall be in the range 0 <= *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:
    a) If *D* >= 0 (zero), then the attribute *ST_PrivateRadians* set to the angle measurement represented by the combination of *degrees*, *minutes*, and *seconds*, expressed in radians, equal to:

       (pi * (*D* + *M* / 60 + *S* / 3600)) / 180
    b) Otherwise, the attribute *ST_PrivateRadians* set to the angle measurement represented by the combination of *degrees*, *minutes*, and *seconds*, expressed in radians, equal to:

       (pi * (*D* - *M* / 60 - *S* / 3600)) / 180

15) The method *ST_Angle(ST_Point, ST_Point, ST_Point)* takes the following input parameters:
    a) an *ST_Point* value *atpoint*,
    b) an *ST_Point* value *apoint*,
    c) an *ST_Point* value *anotherpoint*.
16) Let *D1* be the direction from *atpoint* to *apoint* and let *D2* be the direction from *atpoint* to *anotherpoint*.
17) For the null-call type preserving SQL-invoked constructor method *ST_Angle(ST_Point, ST_Point, ST_Point)*:

    Case:

    a) If *atpoint* is the empty set, then an exception condition is raised: *SQL/MM Spatial exception – empty point value*.
    b) If *atpoint* is not well formed, then an exception condition is raised: *SQL/MM Spatial exception – point value not well formed*.
    c) If *apoint* is the empty set, then an exception condition is raised: *SQL/MM Spatial exception – empty point value*.
    d) If *apoint* is not well formed, then an exception condition is raised: *SQL/MM Spatial exception – point value not well formed*.
    e) If *anotherpoint* is the empty set, then an exception condition is raised: *SQL/MM Spatial exception – empty point value*.
    f) If *anotherpoint* is not well formed, then an exception condition is raised: *SQL/MM Spatial exception – point value not well formed*.
    g) If *atpoint* equals *apoint*, then an exception condition is raised: *SQL/MM Spatial exception – points are equal*.
    h) If *atpoint* equals *anotherpoint*, then an exception condition is raised: *SQL/MM Spatial exception – points are equal*.
    i) Otherwise, return an ST_Angle value with the attribute ST_PrivateRadians set to the lesser of:
        i) the clockwise angle measured in radians from *D1* to *D2*, or
        ii) the clockwise angle measured in radians from *D2* to *D1*.
18) The method *ST_Angle(ST_Direction, ST_Direction)* takes the following input parameters:
    a) an *ST_Direction* value *adirection*,
    b) an *ST_Direction* value *anotherdirection*.
19) For the null-call type preserving SQL-invoked constructor method *ST_Angle(ST_Direction, ST_Direction)* return an *ST_Angle* value with the attribute *ST_PrivateRadians* set to the lesser of:
    a) the clockwise angle measured in radians from *adirection* to *anotherdirection*, or
    b) the clockwise angle measured in radians from *anotherdirection* to *adirection*.
20) The method *ST_Angle(ST_LineString, ST_LineString)* takes the following input parameters:
    a) an *ST_LineString* value *aline*,
    b) an *ST_LineString* value *anotherline*,
21) Let *P1* be the point value returned by *aline.ST_StartPoint()* and *P2* be the point value returned by *aline.ST_EndPoint()*. Let *D1* be the direction from *P1* to *P2*.
22) Let *P3* be the point value returned by *anotherline.ST_StartPoint()* and *P4* be the point value returned by *anotherline.ST_EndPoint()* Then let *D2* be the direction from *P3* to *P4*.

23) For the null-call type preserving SQL-invoked constructor method *ST_Angle(ST_LineString, ST_LineString)*:

    Case:

    a) If *aline* is not a line, then an exception condition is raised: *SQL/MM Spatial exception – linestring is not a line*.

    b) If *aline* is closed, then an exception condition is raised: *SQL/MM Spatial exception – degenerate line has no direction*.

    c) If *anotherline* is not a line, then an exception condition is raised: *SQL/MM Spatial exception – linestring is not a line*.

    d) If *anotherline* is closed, then an exception condition is raised: *SQL/MM Spatial exception – degenerate line has no direction*.

    e) Otherwise, return an *ST_Angle* value with the attribute *ST_PrivateRadians* set to the lesser of:

        i) the clockwise angle measured in radians from *D1* to *D2*, or

        ii) the clockwise angle measured in radians from *D2* to *D1*

24) The method *ST_Angle(CHARACTER VARYING)* takes the following input parameters:

    a) a CHARACTER VARYING value *awkt*.

25) The well-known 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>
```

    a) <angle text representation> is the well-known text representation for an *ST_Angle* value that is produced by <angle text>.

    b) <angle text> produces the *ST_Angle* value from <degrees>, <gradians>, or <radians>

    c) Case:

        i) Let *DEGREES* be the DOUBLE PRECISION value specified by <degrees>. <degrees> produces an *ST_Angle* value as the result of the value expression: NEW *ST_Angle*('D', *DEGREES*).

        ii) Let *GRADIANS* be the DOUBLE PRECISION value specified by <gradians>. <gradians> produces an *ST_Angle* value as the result of the value expression: NEW *ST_Angle*('G', *GRADIANS*).

        iii) Let *RADIANS* be the DOUBLE PRECISION value specified by <radians>. <radians> produces an *ST_Angle* value as the result of the value expression: NEW *ST_Angle*('R', *RADIANS*).

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*.

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.

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.

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.

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:
   a) 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:
   a) *numdecdigits*
   b) *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) a CHARACTER VARYING value *dms*.
7) For the type preserving method *ST_String(CHARACTER VARYING)*:

   Case:
   a) If *dms* 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:
      i) 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.
      ii) 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*.

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*.

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*.

11.1.13 ST_Divide Method

Purpose

Divide the ST_Angle value by a non-zero, numeric value.

Definition

```
CREATE METHOD ST_Divide
   (adivisor DOUBLE PRECISION)
   RETURNS ST_Angle
   FOR ST_Angle
   BEGIN
      IF adivisor = 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 *adivisor*.

2) For the null-call type preserving method *ST_Divide(ST_Angle)*:

   Case:

   a) If *adivisor* 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 *adivisor*.

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(CHARACTER 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 2pi 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.

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

   CREATE CONSTRUCTOR METHOD ST_Direction
      (awkt CHARACTER VARYING(ST_MaxDirectionAsText))
      RETURNS ST_Direction
      FOR ST_Direction
      BEGIN
         --
         -- See Description
         --
      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:

   a) If *direction* is less than 0 (zero) radians or *direction* is greater than or equal to (2*pi) radians, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

   b) 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) a CHARACTER value *northsouth*,

   b) an *ST_Angle* value *bearingangle*,

   c) a CHARACTER value *eastwest*.

4) For the null-call type preserving SQL-invoked constructor method *ST_Direction(CHARACTER, ST_Angle, CHARACTER)*:

   a) If *northsouth* is not 'N' (for North) or 'S' (for South), then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

   b) 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*.

   c) If *eastwest* is not 'E' (for East) or 'W' (for West), then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

   d) 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) a CHARACTER value *northsouth*,

   b) 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:

   a) If *NS* is not 'N' (for North) or 'S' (for South), then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

   b) If *azimuthangle.ST_Radians()* is less than 0 (zero) radians or *azimuthangle.ST_Radians()* is greater than or equal to (2*pi) radians, then an exception condition is raised: *SQL/MM Spatial exception – invalid argument*.

   c) Otherwise, returns an *ST_Direction* value with:

      Case:

      i) If *NS* is equal to 'N', then the attribute *ST_PrivateAngleNAzimuth* set to *azimuthangle*.

      ii) If *NS* is equal to 'S' and 0 (zero) <= *azimuthangle.ST_Degrees()* < 180, then the attribute *ST_PrivateAngleNAzimuth* set to *azimuthangle.ST_Add*(NEW *ST_Angle('D',180))*.

      iii) 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:

   a) an *ST_Point* value *frompoint,*

   b) 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*.

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..

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>.

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 <= *NAZ.ST_Degrees()* <= 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()* <= 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 <= *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.

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 <= *NAZ.ST_Degrees()* <= 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()* <= 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 <= *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.