INTERNATIONAL STANDARD

ISO/IEC 13249-3

First edition
1999-12-01

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

Part 3: **Spatial**

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

Partie 3: Spatial

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

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents Page

Figures

Page

Tables

Page

Foreword

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

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

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

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

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

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

— *Part 1: Framework*

— *Part 2: Full text*

— *Part 3: Spatial*

— *Part 4: General purpose facilities*

— *Part 5: Still image*

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

Introduction

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

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

— Part 1: Framework

— Part 2: Full text

— Part 3: Spatial

— Part 4: General purpose facilities

— Part 5: Still image

The organization of this part of ISO/IEC 13249 is as follows:

1) Clause 1, "Scope", specifies the scope of this part of ISO/IEC 13249.

2) Clause 2, "Normative references", identifies additional standards that, through reference in this part of ISO/IEC 13249, constitute provisions of this part of ISO/IEC 13249.

3) Clause 3, "Definitions, notations, and conventions", defines the notations and conventions used in this part of ISO/IEC 13249.

4) Clause 4, "Concepts", presents concepts used in the definition of this part of ISO/IEC 13249.

5) Clause 5, "Geometry Types", defines the geometry supertype.

6) Clause 6, "Point Types", defines primitive 0-dimensional geometry types.

7) Clause 7, "Curve Types", defines primitive 1-dimensional geometry types.

8) Clause 8, "Surface Types", defines primitive 2-dimensional geometry types.

9) Clause 9, "Geometry Collection Types", defines the geometry collection types.

10) Clause 10, "Spatial Reference System Types", defines the user-defined type to manage spatial referencing systems.

11) Clause 11, "Support Routines", defines supporting functions and procedures used by this part of ISO/IEC 13249.

12) Clause 12, "Conformance", defines the criteria for conformance to this part of ISO/IEC 13249.

13) Clause 13, "Status Codes", defines the SQLSTATE codes used in this part of ISO/IEC 13249.

14) Annex A, "Implementation-defined elements", is an informative Annex. It lists those features for which the body of this part of ISO/IEC 13249 states that the syntax or meaning or effect on the database is partly or wholly implementation-defined, and describes the defining information that an implementor shall provide in each case.

15) Annex B, "Implementation-dependent elements", is an informative Annex. It lists those features for which the body of this part of ISO/IEC 13249 states explicitly that the meaning or effect on the database is implementation-dependent.

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

17) Bibliography is the last informative Annex. It is a list of selective reading relating to this part of ISO/IEC 13249.

In the text of this part of ISO/IEC 13249, Clauses begin a new odd-numbered page, and in Clause 5, "Geometry Types", through Clause 11, "Support Routines", subclauses begin a new page. Any resulting blank space is not significant.

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

Part 3: Spatial

1 Scope

This part of ISO/IEC 13249:

a) introduces the Spatial part of ISO/IEC 13249,

b) gives the references necessary for this part of ISO/IEC 13249,

c) defines notations and conventions specific to this part of ISO/IEC 13249,

d) defines concepts specific to this part of ISO/IEC 13249,

e) defines spatial user-defined types and their associated routines.

The spatial user-defined types defined in this part adhere to the following:

— A spatial user-defined type is generic to spatial data handling. It addresses the need to store, manage and retrieve information based on aspects of spatial data such as geometry, location, and topology.

— A spatial user-defined type does not redefine the database language SQL directly or in combination with another spatial data type.

Implementations of this part of ISO/IEC 13249 may exist in environments that also support geographic information, decision support, data mining, and data warehousing systems.

Application areas addressed by implementations of this part of ISO/IEC 13249 include, but are not restricted to, automated mapping, desktop mapping, facilities management, geoengineering, graphics, multimedia, and resource management applications.

Blank page

2 Normative references

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

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

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

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

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

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

Blank page

3 Definitions, notations, and conventions

3.1 Definitions

3.1.1 Definitions provided in Part 1

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

3.1.2 Definitions provided in Part 3

For the purposes of this part of ISO/IEC 13249, the following definitions apply.

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

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

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

3.1.2.4
boundary of a curve
the empty set if the curve is closed, otherwise the set containing the start and end points of the curve

3.1.2.5
boundary of a point
the empty set

3.1.2.6
boundary of a surface
the set of curves that delineate the edge of the surface, including interior and exterior rings

3.1.2.7
closed curve
a curve value such that its start point is equal to its end point

3.1.2.8
closure
a topological function to cause an open point-set to include its boundary making the point-set topologically closed

3.1.2.9
dimension
geometric dimension

3.1.2.10
geometry
the shape and geographic location of a feature

3.1.2.11
intersection
two geometries intersect if their point set representations, a and b intersect: $a \cap b \neq \varnothing$

3.1.2.12
linear ring
a linestring value that is closed and simple

3.1.2.13
linestring
a curve with linear interpolation between control points

3.1.2.14
non-closed curve
a curve value such that its start point is not equal to its end point

3.1.2.15
polygon
a surface that uses linear rings to define its boundary

3.1.2.16
point set
the representation of a geometry as a finite set or infinite set of points

NOTE Mathematical set intersection (∩), set union (∪) and set difference (—) operation work on point sets.

3.1.2.17
ring
an curve value that is closed and simple

3.1.2.18
spatially equals
the test for *spatially equals* on geometry values has a similar definition as 'equals' for mathematical sets

NOTE 1 If the point sets of two geometry values are equal, then the geometries are spatially equal. Two point sets, *a* and *b*, are equal if: $(a - b) \cup (b - a) = \varnothing$.

NOTE 2 Since an infinite set of points cannot be tested, the internal representation of equal must test for equivalents between two, possibly quite different, representations. The test may be limited to the resolution of the spatial referencing system or the accuracy of the data. An implementation-defined tolerance may be provided such that two points are considered equal if the distance between the points is less that the tolerance.

3.1.2.19
topologically closed
a characteristic of a geometry type that every value includes its own boundary

3.1.3 Definitions taken from ISO/IEC 9075

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

a) immediately contained.

3.1.4 Definitions taken from ISO 15046-7

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

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

3.1.4.1
buffer
geometric object that contains all points whose distance from another geometric object is less than or equal to a given distance

3.1.4.2
computational topology, **algebraic topology**
topological concepts, structures and algebra that aid, enhance or define operations usually performed in computational geometry

3.1.4.3
connected
property of a geometric object implying that any two points on the object can be connected by a curve that remains within the object

3.1.4.4
convex hull of a geometric object
smallest convex set containing the geometric set associated with the geometric object

3.1.4.5
coordinate dimension
number of measurements or ordinates needed to describe a position in a coordinate system

3.1.4.6
coordinate
one of a set of *N* numbers designating the position of a point in N-dimensional space

3.1.4.7
coordinate system
set of (mathematical) rules for specifying how coordinates are to be assigned to points

3.1.4.8
coordinate reference system
coordinate system which is related to the earth by a datum

3.1.4.9
curve
bounded, connected 1-dimensional geometric primitive, representing the continuous image of a line, and therefore fully realizable as a 1-parameter set of points

3.1.4.10
end point
last point of a curve

3.1.4.11
geometric complex
set of disjoint geometric primitives such that the geometric boundary of each geometric primitive can be represented as the union of other geometric primitives within the geometric complex

3.1.4.12
geometric dimension
largest number *n* such that each direct position in a geometric set is associated with a subset that has the direct position in its interior and is similar (isomorphic) to R^n, Euclidean *n*-space

3.1.4.13
geometric object
geometric primitive, collection of geometric primitives, or geometric complex treated as a single entity

3.1.4.14
geometric primitive
object representing a single, connected, homogeneous element of geometry

3.1.4.15
homomorphism
relationship between two objects (two complexes) such that there is a structure preserving function from one to the other

3.1.4.16
interior (of a geometric object)
set of all points that are on the geometric object but which are not on its geometric boundary

3.1.4.17
isomorphism
relationship between two objects (two complexes) such that there is a one to one, structure preserving function from one onto the other

3.1.4.18
point
0-dimensional geometric primitive, representing a position, but not having extent

3.1.4.19
start point
first point of a curve

3.1.4.20
surface
bounded, connected 2-dimensional geometric primitive, representing the continuous image of region of a plane, and therefore fully realizable locally as a 2-parameter set of points

3.1.5 Definitions taken from ISO 15046-11

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

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

3.1.5.1
datum
any quantity or set of quantities that may serve as a reference or basis for the calculation of other quantities

3.1.5.2
ellipsoid
figure formed by the rotation of an ellipse about an axis

3.1.5.3
flattening
the ratio of the difference between the semi-major and semi-minor axis of an ellipsoid to the semi-major axis of an ellipsoid

3.1.5.4
geodetic coordinate system, ellipsoidal coordinate system
coordinate system in which position is specified by latitude, longitude and (in the three-dimensional case) ellipsoidal height

3.1.5.5
meridian
intersection between an ellipsoid and a plane containing the semi-minor axis of the ellipsoid

3.1.5.6
prime meridian, zero meridian
meridian from which the longitudes of other meridians are quantified

3.1.5.7
projected coordinate system
two-dimensional Cartesian coordinate system resulting from a map projection

3.1.5.8
semi-major axis
half of the longest diameter of an ellipsoid

3.1.5.9
semi-minor axis
half of the shortest diameter of an ellipsoid

3.1.5.10
unit
defined quantity in which dimensioned parameters are expressed

3.2 Notations

3.2.1 Notations provided in Part 1

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

3.2.2 Notations provided in Part 3

This part of ISO/IEC 13249 uses the prefix 'ST_' for user-defined type, attribute and SQL-invoked routine names.

This part of ISO/IEC 13249 uses the prefix 'ST_Private' for names of certain attributes. The use of 'ST_Private' indicates that the attribute is not for public use.

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

Table 1 — Symbols

Symbols	Meaning
∅	empty set
∩	Intersection
∪	Union
—	Difference
∈	is a member of
∉	is not a member of
⊂	is a proper subset of
⊆	is a subset of
⇔	if and only if
⇒	Implies
∀	for all
{ x | ... }	set of all *x* such that ...
∧	And
∨	Or
¬	Not

3.3 Conventions

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

4 Concepts

4.1 Geometry Types

The following geometry types are supported: ST_Geometry, ST_Point, ST_Curve, ST_LineString, ST_CircularString, ST_CompoundCurve, ST_Surface, ST_CurvePolygon, ST_Polygon, ST_GeomCollection, ST_MultiPoint, ST_MultiCurve, ST_MultiLineString, ST_MultiSurface, and ST_MultiPolygon. The geometry type hierarchy is visually described in Annex C, "ST_Geometry Type Hierarchy".

ST_Geometry, ST_Curve, and ST_Surface are not instantiable types. No implicitly defined constructor functions are defined for these types.

ST_Point, ST_LineString, ST_CircularString, ST_CompoundCurve, ST_CurvePolygon, ST_Polygon, ST_GeomCollection, ST_MultiPoint, ST_MultiCurve, ST_MultiLineString, ST_MultiSurface, and ST_MultiPolygon are instantiable and have implicitly defined constructor functions.

Any geometry type can be used as the type for a column. Declaring a column to be of a particular type implies that any instance of the type or of any of its subtypes can be stored in the column.

4.1.1 ST_Geometry

The ST_Geometry type is the root type of all geometry types. The ST_Geometry type is not instantiable. The instantiable subtypes of the ST_Geometry type are 0-dimensional geometry, 1-dimensional geometry, and 2-dimensional geometry types that exist in two-dimensional coordinate space (R^2)

NOTE 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 sub-type of *A*, an implementation may introduce another type *T* between types *A* and *B*.

4.1.1.1 Methods on ST_Geometry

1) ST_Dimension: returns the dimension of an ST_Geometry value. The dimension of an ST_Geometry value is less than or equal to the coordinate dimension.

2) ST_CoordDim: returns the coordinate dimension of an ST_Geometry value. The coordinate dimension shall be the same as the coordinate dimension of the spatial reference system for the ST_Geometry value.

3) ST_GeometryType: returns the type of the ST_Geometry value as a CHARACTER VARYING value.

4) ST_SRID: observes and mutates the spatial reference system identifier of an ST_Geometry value.

5) ST_Transform: returns the ST_Geometry value in the specified spatial reference system.

6) ST_IsEmpty: tests if an ST_Geometry value corresponds to the empty set.

7) ST_IsSimple: tests if an ST_Geometry value has no anomalous geometric point, such as self intersection or self tangency. Subtypes of ST_Geometry will define the specific conditions that cause a value to be classified as simple.

8) ST_Boundary: returns the boundary of an ST_Geometry value.

9) ST_Envelope: returns the bounding rectangle of an ST_Geometry value.

10) ST_ConvexHull: returns the convex hull of an ST_Geometry value.

11) ST_Buffer: returns the ST_Geometry value that represents all points whose distance from an ST_Geometry value is less than or equal to a specified distance.

12) ST_Intersection: returns the ST_Geometry value that represents the point set intersection of two ST_Geometry values.

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

    a.ST_Intersection(*b*) ⇔ Closure(*a* ∩ *b*)

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

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

    a.ST_Union(*b*) ⇔ Closure(*a* ∪ *b*)

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

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

    a.ST_Difference(*b*) ⇔ Closure(*a* — *b*)

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

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

    a.ST_SymDifference(*b*) ⇔ Closure(*a* — *b*) ∪ Closure(*b* — *a*) ⇔
    a.ST_Difference(*b*).ST_Union(*b*.ST_Difference(*a*))

16) ST_Distance: returns the minimum shortest distance between any two points in the two ST_Geometry values

17) ST_WKTToSQL: returns the ST_Geometry value for the specified well-known text representation.

18) ST_AsText: returns the well-known text representation for the specified ST_Geometry value.

19) ST_WKBToSQL: returns the ST_Geometry value for the specified well-known binary representation.

20) ST_AsBinary: returns the well-known binary representation for the specified ST_Geometry value.

4.1.1.2 Functions on ST_Geometry

1) ST_GeomFromText: transforms a CHARACTER LARGE OBJECT to an ST_Geometry value.

2) ST_GeomFromWKB: transforms a BINARY LARGE OBJECT to an ST_Geometry value.

4.1.1.3 Ordering on ST_Geometry

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

4.1.1.4 SQL Transforms on ST_Geometry

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

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

4.1.2 Spatial Relationships using ST_Geometry

The spatial relationships are methods that are used to test for the existence of a specified topological spatial relationship between two ST_Geometry values. The basic approach to comparing two ST_Geometry values is to make pair-wise tests of the intersections between the interiors, boundaries and exteriors of the two ST_Geometry values and to classify the relationship between the two ST_Geometry values based on the entries in the resulting intersection matrix.

The concepts of interior, boundary and exterior are well defined in general topology. These concepts can be applied in defining spatial relationships between 2-dimensional geometry values in two-dimensional coordinate space (R^2). In order to apply the concepts of interior, boundary and exterior to 0-dimensional geometry and 1-dimensional geometry values in R^2, a combinatorial topology approach must be applied. This approach is based on the accepted definitions of the boundaries, interiors and exteriors for simplicial complexes and yields the following results.

The boundary of an ST_Geometry value is a set of ST_Geometry values of the next lower dimension. The boundary of an ST_Point value or an ST_MultiPoint value is the empty set. The boundary of a non-closed ST_Curve consists of the start and end ST_Point values; the boundary of a closed ST_Curve value is empty. The boundary of an ST_MultiCurve consists of those ST_Point values that are in the boundaries of an odd number of its element ST_Curve values. The boundary of an ST_Polygon value consists of its set of linear rings. The boundary of an ST_MultiPolygon value consists of the set of linear rings of its ST_Polygon values. The boundary of an arbitrary collection of geometries whose interiors are disjoint consists of geometries drawn from the boundaries of the element geometries by application of the mod 2 union rule.

The domain of ST_Geometry values considered consists of those values that are topologically closed. The interior of an ST_Geometry value consists of those points that are left when the boundary points are removed. The exterior of an ST_Geometry value consists of points not in the interior or boundary.

4.1.2.1 The Dimensionally Extended 9 Intersection Model

Given an ST_Geometry value *g*, let Interior(*g*), Boundary(*g*) and Exterior(*g*) represent the interior, boundary and exterior of *g,* respectively. The intersection of any two of Interior(*a*), Boundary(*a*) and Exterior(*a*) can result in a set of ST_Geometry values, *x*, of mixed dimension. For example, the intersection of the boundaries of two ST_Polygon values may consist of an ST_Point value and an ST_LineString value. Let *x*.ST_Dimension return the maximum dimension (-1, 0, 1, or 2) of *x*, with a value of -1 corresponding to dimension of the empty set (∅). 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:

if *p* = T, then *x*.ST_Dimension ∈ { 0, 1, 2 }, i.e. *x* ≠∅

if *p* = F, then *x*.ST_Dimension = -1, i.e. *x* = ∅

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

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

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

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

The pattern matrix can be represented as an <character string literal> with cardinality 9 representing the DE-9IM in row major order. See Subclause 5.2, "<token> and <separator>" in ISO/IEC 9075-2 for the definition of <character string literal>.

4.1.2.2 Common Spatial Relationships based on the DE-9IM

The ST_Relate method based on the pattern matrix enables a large number of spatial relationships to be tested and fine-tuned to the particular relationship being tested. However it is a low level building block and does not have a corresponding natural language equivalent. For this reason, commonly used spatial relationships have been specified: ST_Disjoint, ST_Intersects, ST_Touches, ST_Crosses, ST_Within, ST_Contains and ST_Overlaps. These spatial relationships have the following properties:

1) They are mutually exclusive.

2) They provide a complete covering of all topological cases.

3) They apply to spatial relationships between two geometries of either the same or different dimension.

4) Each predicate can be expressed in terms of a corresponding set of DE-9IM patterns.

5) Any realizable DE-9IM can be expressed as a Boolean expression over the 7 predicates, given the ST_Boundary method on the ST_Geometry type and the ST_StartPoint and ST_EndPoint method on the ST_Curve type.

4.1.2.3 Spatial Methods using ST_Geometry

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

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

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

b) Otherwise:

a.ST_Crosses(*b*) ⇔
((Interior(*a*) ∩ Interior(*b*)).ST_Dimension <
max(Interior(*a*).ST_Dimension, Interior(*b*).ST_Dimension)) ∧ (*a* ∩ *b* ≠ *a*) ∧ (*a* ∩ *b* ≠ *b*)

Expressed in terms of the DE-9IM:

Case:

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

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

a.ST_Crosses(*b*) ⇔
(Interior(*a*) ∩ Interior(*b*) ≠ ∅) ∧ (Interior(*a*) ∩ Exterior(*b*) ≠ ∅) ⇔
a.ST_Relate(*b*, 'T*T******')

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

a.ST_Crosses(*b*) ⇔
(Interior(*a*) ∩ Interior(*b*) ≠ ∅) ∧ (Interior(*b*) ∩ Exterior(*a*) ≠ ∅) ⇔
b.ST_Relate(*a*, 'T*T******')

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

a.ST_Crosses(*b*) ⇔
(Interior(*a*) ∩ Interior(*b*)).ST_Dimension = 0 (zero) ⇔
a.ST_Relate(*b*, '0********')

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*) ⇔ (*a* ∩ *b* = *a*) ∧ (Interior(*a*) ∩ Exterior(*b*) ≠ ∅)

Expressed in terms of the DE-9IM:

a.ST_Within(*b*) ⇔
(Interior(*a*) ∩ Interior(*b*) ≠ ∅) ∧
(Interior(*a*) ∩ Exterior(*b*) = ∅) ∧ (Boundary(*a*) ∩ Exterior(*b*) = ∅)) ⇔
a.ST_Relate(*b*, 'T*****FF*')

8) ST_Contains: tests if an ST_Geometry value spatially contains another ST_Geometry value.

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

a.ST_Contains(*b*) ⇔ *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*) ⇔
(Interior(*a*).ST_Dimension = Interior(*b*).ST_Dimension =
(Interior(*a*) ∩ Interior(*b*)).ST_Dimension) ∧ (*a* ∩ *b* ≠ *a*) ∧ (*a* ∩ *b* ≠ *b*)

b) Otherwise: the null value

Expressed in terms of the DE-9IM:

Case:

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

a.ST_Overlaps(*b*) ⇔
(Interior(*a*) ∩ Interior(*b*) ≠ ∅) ∧
(Interior(*a*) ∩ Exterior(*b*) ≠ ∅) ∧
(Exterior(*a*) ∩ Interior(*b*) ≠ ∅) ⇔
a.ST_Relate(*b*, 'T*T***T**')

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

a.ST_Overlaps(*b*) ⇔
((Interior(*a*) ∩ Interior(*b*)).ST_Dimension = 1 (one)) ∧
(Interior(*a*) ∩ Exterior(*b*) ≠ ∅) ∧ (Exterior(*a*) ∩ Interior(*b*) ≠ ∅) ⇔
a.ST_Relate(*b*, '1*T***T**')

c) Otherwise: the null value

4.1.3 Columns of type ST_Geometry

4.1.3.1 GEOMETRY_COLUMNS Table (or View)

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

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

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

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

a) The values F_TABLE_CATALOG, F_TABLE_SCHEMA, F_TABLE_NAME, and F_GEOMETRY_COLUMN are the catalog name, unqualified schema name, qualified identifier, and column name, respectively, of a column identified by a <column reference> (See Subclause 5.4, "Names and identifiers" in ISO/IEC 9075-2). The <column reference> identifies a geometry column in a feature table.

b) COORD_DIMENSION is the number of coordinates used in the ST_Geometry values, usually corresponding to the number of dimensions in the spatial reference system.

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

4.1.3.2 Tables with column of type ST_Geometry

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

The following is the general form of such a table:

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

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

4.1.4 ST_Point

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

4.1.4.1 Methods on ST_Point

1) ST_Point: returns the specified ST_Point value.

2) ST_X: observes and mutates the x coordinate value of an ST_Point value.

3) ST_Y: observes and mutates the y coordinate value of an ST_Point value.

4) ST_ExplicitPoint: returns the coordinate values as a DOUBLE PRECSION ARRAY value.

4.1.4.2 Functions on ST_Point

1) ST_PointFromText: transforms a CHARACTER LARGE OBJECT to an ST_Point value.

2) ST_PointFromWKB: transforms a BINARY LARGE OBJECT to an ST_Point value.

4.1.5 ST_Curve

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

Topologically an ST_Curve value is a 1-dimensional geometry that is the homomorphic image of a real, closed, interval. An ST_Curve value is simple if it does not pass through the same point twice.

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

4.1.5.1 Methods on ST_Curve

1) ST_Length: returns the length of an ST_Curve value.

2) ST_StartPoint: returns the ST_Point value that is the start point of an ST_Curve value.

3) ST_EndPoint: returns the ST_Point value that is the end point of an ST_Curve value.

4) ST_IsClosed: tests if an ST_Curve value is closed.

5) ST_IsRing: tests if an ST_Curve value is a ring.

6) ST_CurveToLine: returns the ST_LineString value approximating the ST_Curve value.

4.1.6 ST_LineString

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

4.1.6.1 Methods on ST_LineString

1) ST_LineString: returns the specified ST_LineString value.

2) ST_Points: observes and mutates the ST_Point collection in the ST_LineString value.

3) ST_NumPoints: returns the cardinality of the ST_Point collection in the ST_LineString value.

4) ST_PointN: returns the specified element in the ST_Point collection in the ST_LineString value.

4.1.6.2 Functions on ST_LineString

1) ST_LineFromText: transforms a CHARACTER LARGE OBJECT to an ST_LineString value.

2) ST_LineFromWKB: transforms a BINARY LARGE OBJECT to an ST_LineString value.

4.1.7 ST_CircularString

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

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

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

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

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

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

4.1.7.1 Methods on ST_CircularString

1) ST_CircularString: returns the specified ST_CircularString value.

2) ST_Points: observes and mutates the ST_Point collection in the ST_CircularString value.

3) ST_NumPoints: returns the cardinality of the ST_Point collection in the ST_CircularString value.

4) ST_PointN: returns the specified element in the ST_Point collection in the ST_CircularString value.

5) ST_MidPointRep: returns the array of points which identify an ST_CircularString value including start, mid, and end points for each curve segment.

4.1.7.2 Functions on ST_CircularString

1) ST_CircularFromText: transforms a CHARACTER LARGE OBJECT to an ST_CircularString value.

2) ST_CircularFromWKB: transforms a BINARY LARGE OBJECT to an ST_CircularString value.

4.1.8 ST_CompoundCurve

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

4.1.8.1 Methods on ST_CompoundCurve

1) ST_CompoundCurve: returns the specified ST_CompoundCurve value.

2) ST_Curves: observes and mutates the ST_Curve collection in the ST_CompoundCurve value.

3) ST_NumCurves: returns the cardinality of the ST_Curve collection in the ST_CompoundCurve value.

4) ST_CurveN: returns the specified element in the ST_Curve collection in the ST_CompoundCurve value.

4.1.8.2 Functions on ST_CompoundCurve

1) ST_CompoundFromText: transforms a CHARACTER LARGE OBJECT to an ST_CompoundCurve value.

2) ST_CompoundFromWKB: transforms a BINARY LARGE OBJECT to an ST_CompoundCurve value.

4.1.9 ST_Surface

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

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

4.1.9.1 Methods on ST_Surface

1) ST_Area: returns the area of an ST_Surface value.

2) ST_Perimeter: returns the length of the perimeter of an ST_Surface value.

3) ST_Centroid: returns the ST_Point value that is the mathematical centroid of the ST_Surface value.

4) ST_PointOnSurface: returns then ST_Point value that is guaranteed to be on the ST_Surface value.

4.1.10 ST_CurvePolygon

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

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

ST_CurvePolygon values are simple.

4.1.10.1 Methods on ST_CurvePolygon

1) ST_CurvePolygon: returns the specified ST_CurvePolygon value.

2) ST_ExteriorRing: observes and mutates the exterior ring of an ST_CurvePolygon value.

3) ST_InteriorRings: observes and mutates the collection of interior rings of an ST_CurvePolygon value.

4) ST_NumInteriorRing: returns the cardinality of the collection of interior rings of an ST_CurvePolygon value.

5) ST_InteriorRingN: returns the specified element in the collection of interior rings of an ST_CurvePolygon value.

6) ST_CurvePolyToPoly: returns an ST_Polygon value approximating the ST_CurvePolygon value.

4.1.10.2 Functions on ST_CurvePolygon

1) ST_CPolyFromText: transforms a CHARACTER LARGE OBJECT to an ST_CurvePolygon value.

2) ST_CPolyFromWKB: transforms a BINARY LARGE OBJECT to an ST_CurvePolygon value.

4.1.11 ST_Polygon

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

4.1.11.1 Methods on ST_Polygon

1) ST_Polygon: returns the specified ST_Polygon value.

4.1.11.2 Functions on ST_Polygon

1) ST_PolyFromText: transforms a CHARACTER LARGE OBJECT to an ST_Polygon value.

2) ST_PolyFromWKB: transforms a BINARY LARGE OBJECT to an ST_Polygon value.

3) ST_BdPolyFromText: builds an ST_Polygon value from a well-known text representation of an ST_MultiLineString.

4) ST_BdPolyFromWKB: builds an ST_Polygon value from a well-known binary representation of an ST_MultiLineString.

4.1.12 ST_GeomCollection

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

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

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

4.1.12.1 Methods on ST_GeomCollection

1) ST_GeomCollection: returns the specified ST_GeomCollection value.

2) ST_Geometries: observes and mutates the ST_Geometry collection in the ST_GeomCollection value.

3) ST_NumGeometries: returns the cardinality of the ST_Geometry collection in the ST_GeomCollection value.

4) ST_GeometryN: returns the specified element in the ST_Geometry collection in the ST_GeomCollection value.

4.1.12.2 Functions on ST_GeomCollection

1) ST_GeomCollFromTxt: transforms a CHARACTER LARGE OBJECT to an ST_GeomCollection value.

2) ST_GeomCollFromWKB: transforms a BINARY LARGE OBJECT to an ST_GeomCollection value.

4.1.13 ST_MultiPoint

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

4.1.13.1 Methods on ST_MultiPoint

1) ST_MultiPoint returns the specified ST_MultiPoint value.

4.1.13.2 Functions on ST_MultiPoint

1) ST_MPointFromText: transforms a CHARACTER LARGE OBJECT to an ST_MultiPoint value.

2) ST_MPointFromWKB: transforms a BINARY LARGE OBJECT to an ST_MultiPoint value.

4.1.14 ST_MultiCurve

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

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

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

4.1.14.1 Methods on ST_MultiCurve

1) ST_MultiCurve: returns the specified ST_MultiCurve value.

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

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

4.1.14.2 Functions on ST_MultiCurve

1) ST_MCurveFromText: transforms a CHARACTER LARGE OBJECT to an ST_MultiCurve value.

2) ST_MCurveFromWKB: transforms a BINARY LARGE OBJECT to an ST_MultiCurve value.

4.1.15 ST_MultiLineString

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

4.1.15.1 Methods on ST_MultiLineString

1) ST_MultiLineString: returns the specified ST_MultiLineString value.

4.1.15.2 Functions on ST_MultiLineString

1) ST_MLineFromText: transforms a CHARACTER LARGE OBJECT to an ST_MultiLineString value.

2) ST_MLineFromWKB: transforms a BINARY LARGE OBJECT to an ST_MultiLineString value.

4.1.16 ST_MultiSurface

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

4.1.16.1 Methods on ST_MultiSurface

1) ST_MultiSurface: returns the specified ST_MultiSurface value.

2) ST_Area: returns the area of an ST_MultiSurface value.

3) ST_Perimeter: returns the length of the perimeter of an ST_MultiSurface value.

4) ST_Centroid: returns the ST_Point value that is the mathematical centroid of the ST_MultiSurface value.

5) ST_PointOnSurface: returns the ST_Point value that is guaranteed to be on the ST_MultiSurface value.

4.1.16.2 Functions on ST_MultiSurface

1) ST_MSurfaceFromText: transforms a CHARACTER LARGE OBJECT to an ST_MultiSurface value.

2) ST_MSurfaceFromWKB: transforms a BINARY LARGE OBJECT to an ST_MultiSurface value.

4.1.17 ST_MultiPolygon

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

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

ST_MultiPolygon values are simple.

4.1.17.1 Methods on ST_MultiPolygon

1) ST_MultiPolygon: returns the specified ST_MultiPolygon value.

4.1.17.2 Functions on ST_MultiPolygon

1) ST_MPolyFromText: transforms a CHARACTER LARGE OBJECT to an ST_MultiPolygon value.

2) ST_MPolyFromWKB: transforms a BINARY LARGE OBJECT to an ST_MultiPolygon value.

3) ST_BdMPolyFromText: builds an ST_MultiPolygon value from a well-known text representation of an ST_MultiLineString.

4) ST_BdMPolyFromWKB: builds an ST_MultiPolygon value from a well-known binary representation of an ST_MultiLineString.

4.2 Spatial Reference System Type

4.2.1 ST_SpatialRefSys

The ST_SpatialRefSys type encapsulates all aspects of spatial reference systems.

4.2.1.1 Methods on ST_SpatialRefSys

1) ST_SpatialRefSys: returns the specified ST_SpatialRefSys value.

2) ST_AsWKTSRS: returns the well-known text representation of a spatial reference system for the specified ST_SpatialRefSys value.

3) ST_WKTSRSToSQL: returns the ST_SpatialRefSys value of a well-known text representation of a spatial reference system.

4) ST_SRID: returns the integer identifier of an ST_SpatialRefSys value.

5) ST_Equals: tests if two ST_SpatialRefSys values are equal.

4.2.1.2 Ordering on ST_SpatialRefSys

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

4.2.1.3 SQL Transforms on ST_SpatialRefSys

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

4.2.1.4 SPATIAL_REF_SYS Table (or View)

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

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

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

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

a) SRID is the spatial reference system identifier.

b) AUTH_NAME is the name of the standard or standards body that is being cited for this reference system.

c) AUTH_SRID is the identifier of the spatial reference system as defined by the Authority cited in AUTH_NAME.

d) SRTEXT is the <spatial reference system> of the spatial reference system.

4.3 Support Routines

4.3.1 ST_Geometry ARRAY Support Routines

4.3.1.1 Support Functions

1) ST_MinDimension: returns the minimum dimension value in an ST_Geometry ARRAY value.

2) ST_MaxDimension: returns the maximum dimension value in an ST_Geometry ARRAY value.

3) ST_IsMixedDim: tests if an ST_Geometry ARRAY value has ST_Geometry values with different dimensions.

4) ST_CheckSRID: if the elements in the ST_Geometry ARRAY value have mixed spatial reference systems, then raises an exception. Otherwise, returns the spatial reference system identifier of the ST_Geometry elements.

4.3.1.2 Supporting Procedures

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

2) ST_CheckConsecDups: raises an exception if an ST_Geometry ARRAY value has consecutive duplicate values.

4.3.1.3 Supporting Cast Functions

1) ST_ToPointAry Cast: casts an ST_Geometry ARRAY value to an ST_Point ARRAY value.

2) ST_ToCurveAry Cast: casts an ST_Geometry ARRAY value to an ST_Curve ARRAY value.

3) ST_ToLineStringAry Cast: casts an ST_Geometry ARRAY value to an ST_LineString ARRAY value.

4) ST_ToCircularAry Cast: casts an ST_Geometry ARRAY value to an ST_CircularString ARRAY value.

5) ST_ToCompoundAry Cast: casts an ST_Geometry ARRAY value to an ST_CompoundCurve ARRAY value.

6) ST_ToSurfaceAry Cast: casts an ST_Geometry ARRAY value to an ST_Surface ARRAY value.

7) ST_ToCurvePolyAry Cast: casts an ST_Geometry ARRAY value to an ST_CurvePolygon ARRAY value.

8) ST_ToPolygonAry Cast: casts an ST_Geometry ARRAY value to an ST_Polygon ARRAY value.

Blank page

5 Geometry Types

5.1 ST_Geometry Type and Routines

5.1.1 ST_Geometry Type

Purpose

The ST_Geometry type is the root type for geometry types. All subtypes have position specified in their attributes.

Definition

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

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

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

   METHOD ST_GeometryType()
      RETURNS CHARACTER VARYING(128)
      LANGUAGE SQL
      DETERMINISTIC
      CONTAINS SQL
      RETURNS NULL ON NULL INPUT,

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Definitional Rules

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

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

3) The attribute *ST_PrivateDimension* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST_PrivateDimension*.

4) The attribute *ST_PrivateCoordinateDimension* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST_PrivateCoordinateDimension*.

Description

1) The *ST_Geometry* type provides for public use:

   a) a method *ST_Dimension()*,

   b) a method *ST_CoordDim()*,

   c) a method *ST_GeometryType()*,

   d) a method *ST_SRID()*,

   e) a method *ST_SRID(INTEGER)*,

   f) a method *ST_Transform(INTEGER)*,

   g) a method *ST_IsEmpty()*,

   h) a method *ST_IsSimple()*,

   i) a method *ST_IsValid()*,

   j) a method *ST_Boundary()*,

   k) a method *ST_Envelope()*,

   l) a method *ST_ConvexHull()*,

   m) a method *ST_Buffer(DOUBLE PRECISION)*,

   n) a method *ST_Intersection(ST_Geometry)*,

o) a method *ST_Union(ST_Geometry)*,

p) a method *ST_Difference(ST_Geometry)*,

q) a method *ST_SymDifference(ST_Geometry)*,

r) a method *ST_Distance(ST_Geometry)*,

s) a method *ST_Equals(ST_Geometry)*,

t) a method *ST_Relate(ST_Geometry, CHARACTER)*,

u) a method *ST_Disjoint(ST_Geometry)*,

v) a method *ST_Intersects(ST_Geometry)*,

w) a method *ST_Touches(ST_Geometry)*,

x) a method *ST_Crosses(ST_Geometry)*,

y) a method *ST_Within(ST_Geometry)*,

z) a method *ST_Contains(ST_Geometry)*,

aa) a method *ST_Overlaps(ST_Geometry)*,

bb) a method *ST_WKTToSQL(CHARACTER LARGE OBJECT)*,

cc) a method *ST_AsText()*,

dd) a method *ST_WKBToSQL(BINARY LARGE OBJECT)*,

ee) a method ST_AsBinary*()*,

ff) a function *ST_GeomFromText(CHARACTER LARGE OBJECT)*,

gg) a function *ST_GeomFromText(CHARACTER LARGE OBJECT, INTEGER)*,

hh) a function *ST_GeomFromWKB(BINARY LARGE OBJECT)*,

ii) a function *ST_GeomFromWKB(BINARY LARGE OBJECT, INTEGER)*,

jj) an ordering function *ST_OrderingEquals*(*ST_Geometry, ST_Geometry)*,

kk) an SQL Transform group *ST_WellKnownText*,

ll) an SQL Transform group *ST_WellKnownBinary*.

2) The *ST_PrivateDimension* attribute contains the dimension of the *ST_Geometry* value:

Case:

a) If the *ST_Geometry* value corresponds to the empty set, then the dimension is -1.

b) If the *ST_Geometry* value is 0-dimensional geometry, then the dimension is 0 (zero).

c) If the *ST_Geometry* value is 1-dimensional geometry, then the dimension is 1 (one).

d) If the *ST_Geometry* value is 2-dimensional geometry, then the dimension is 2.

3) The *ST_PrivateCoordinateDimension* attribute contains the coordinate dimension of the *ST_Geometry* value.

4) The *ST_PrivateCoordinateDimension* attribute shall be 2 for *ST_Geometry* values in two-dimensional coordinate space (R^2).

5) The *ST_PrivateDimension* attribute shall be less than or equal to the *ST_PrivateCoordinateDimension* attribute.

6) The *ST_Geometry* type is the root type of the geometry hierarchy.

7) All instantiable *ST_Geometry* subtypes are defined so that simple values of the geometry type are topologically closed.

8) The coordinate dimension shall be the same as the coordinate dimension of the spatial reference system for the *ST_Geometry* value.

9) An *ST_Geometry* value has an associated spatial reference system specified by a spatial reference system identifier.

5.1.2 ST_Dimension Method

Purpose

Return the dimension of the ST_Geometry value.

Definition

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

Description

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

2) The null-call method *ST_Dimension()* returns the value of the *ST_PrivateDimension* attribute.

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(128)
   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
```

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
   --
   -- See Description
   --

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

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 *asrid*.

4) The parameter *asrid* 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 *asrid*.

5.1.6 ST_Transform Method

Purpose

Return an ST_Geometry value transformed to the specified spatial reference system.

Definition

```
CREATE METHOD ST_Transform(asrid INTEGER)
   RETURNS ST_Geometry
   FOR ST_Geometry
   --
   -- See Description
   --
```

Description

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

   a) an INTEGER value *asrid*.

2) The parameter *asrid* 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 *asrid*, then return SELF.

   b) 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 *asrid*. The value returned has the spatial reference system identifier equal to *asrid*.

5.1.7 ST_IsEmpty Method

Purpose

Tests if an ST_Geometry value corresponds to the empty set.

Definition

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

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 of each instantiable subtype of *ST_Geometry* includes the specific conditions that cause a value of that type to correspond to the empty set.

5.1.8 ST_IsSimple Method

Purpose

Tests 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
   --
   -- See Description
   --
```

Description

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

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

   Case:

   a) If the *ST_Geometry* value is simple, then return 1 (one).

   b) Otherwise, return 0 (zero).

3) The description of each instantiable subtype of *ST_Geometry* includes the specific conditions that cause a value of that type to be classified as simple.

5.1.9 ST_IsValid Method

Purpose

Tests if an ST_Geometry value is well formed.

Definition

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

Description

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

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

   Case:

   a) If the *ST_Geometry* value is well formed, then return 1 (one).

   b) Otherwise, return 0 (zero).

3) The description of each instantiable subtype of *ST_Geometry* includes the specific conditions that cause a value of that type to be classified as well formed.

5.1.10 ST_Boundary Method

Purpose

Returns the boundary of the ST_Geometry value.

Definition

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

Description

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

2) The null-call method *ST_Boundary()* returns 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
   --
   -- See Description
   --
```

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) 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.

   b) Let *ETOL* be an implementation-defined envelope tolerance. *ETOL* shall be greater than zero.

   c) If *MINX* is equal to *MAXX*, then set *MINX* to *MINX* - *ETOL* and set *MAXX* to *MAXX* + *ETOL*.

   d) If *MINY* is equal to *MAXY*, then set *MINY* to *MINY* - *ETOL* and set *MAXY* to *MAXY* + *ETOL*.

   e) The bounding rectangle is 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
   --
   -- See Description
   --
```

Description

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

2) The null-call method *ST_ConvexHull()* returns 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 Method

Purpose

Return a buffer around the ST_Geometry value.

Definition

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

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 the linear units of measure in the spatial reference system of SELF.

3) The null-call method *ST_Buffer(DOUBLE PRECISION)* returns 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.

5.1.14 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
   --
   -- See Description
   --
```

Description

1) The method *ST_Intersection(ST_Geometry)* takes the following input parameters:

   a) an *ST_Geometry* value *ageometry*.

2) The null-call type preserving method *ST_Intersection(ST_Geometry)* returns an *ST_Geometry* value that represents the point set intersection: Closure(SELF ∩ *ageometry*).

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
   --
   -- See Description
   --
```

Description

1) The method *ST_Union(ST_Geometry)* takes the following input parameters:

   a) an *ST_Geometry* value *ageometry*.

2) The null-call type preserving method *ST_Union(ST_Geometry)* returns an *ST_Geometry* value that represents the point set union: Closure(SELF ∪ *ageometry*).

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

Returns 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
   --
   -- See Description
   --
```

Description

1) The method *ST_Difference(ST_Geometry)* takes the following input parameters:

   a) an *ST_Geometry* value *ageometry*.

2) The null-call type preserving method *ST_Difference(ST_Geometry)* returns an *ST_Geometry* value that represents the point set difference: Closure(SELF — *ageometry*).

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

Returns 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 type preserving method *ST_SymDifference(ST_Geometry)* returns an *ST_Geometry* value that represents the point set symmetric difference: Closure(SELF — *ageometry*) ∪ Closure(*ageometry* — SELF).

3) The spatial reference system identifier of the returned *ST_Geometry* value is equal to the spatial reference system identifier of SELF.

5.1.18 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 result may be one of set of possible subtypes of *ST_Geometry*. The result 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 result set code for the two parameter types. The matrix for the *ST_Intersection* method is 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.

There is no order implied in results of type *ST_GeomCollection*.

Table 3 — Parameter Types

Parameter Type	
Code	Type
∅	empty set
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

Result Type Sets	
Code	Set of Types
R00	empty set
R01	*ST_Point*
R02	*ST_Curve*
R03	*ST_Surface*
R04	*ST_MultiPoint*,
R05	*ST_MultiCurve*
R06	*ST_MultiSurface*
R07	*ST_GeomCollection*
R08	empty set, *ST_Curve*, *ST_MultiCurve*
R09	empty set, *ST_Point*
R10	empty set, *ST_MultiPoint*
R11	empty set, *ST_Point*, *ST_Curve*, *ST_MultiPoint*, *ST_MultiCurve*, *ST_GeomCollection* of *ST_Point* and *ST_Curve* values
R12	empty set, *ST_Point*, *ST_Curve*, *ST_Surface*, *ST_MultiPoint*, *ST_MultiCurve*, *ST_MultiSurface*, *ST_GeomCollection*
R13	empty set, *ST_Point*, *ST_MultiPoint*
R14	empty set, *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	∅	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	∅	P	C	S	MP	MC	MS	GC
∅	∅	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

	∅	P	C	S	MP	MC	MS	GC
∅	∅	∅	∅	∅	∅	∅	∅	∅
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

∪ (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 Method

Purpose

Return the minimum shortest distance between any two points in the two ST_Geometry values.

Definition

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

Description

1) The method *ST_Distance(ST_Geometry)* takes the following input parameters:

   a) an *ST_Geometry* value *ageometry*.

2) The null-call method *ST_Distance(ST_Geometry)* returns the shortest distance between any two points in SELF and *ageometry* as calculated in the spatial referencing system of SELF. If SELF and *ageometry* spatially intersect, then the distance returned is 0 (zero). Otherwise, SELF and *ageometry* are spatially disjoint and the method *ST_Distance(ST_Geometry)* determines the nearest two points, one from SELF and one from *ageometry*, and calculates the implementation-defined distance between the points.

3) The returned value is measured in the linear units of measure in the spatial reference system of SELF.

5.1.20 ST_Equals Method

Purpose

Tests 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

Tests 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 cr INTEGER;

      -- If amatrix does not have cardinality of 9, then
      -- raise an exception.
      IF CARDINALITY(amatrix) <> 9 THEN
         SIGNAL SQLSTATE '2FF04'
            SET MESSAGE_TEXT 'invalid intersection matrix';
      END IF;
      -- 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 cr to the dimension of the current intersection
         CASE counter
            WHEN 1 THEN
               SET cr = (Interior(SELF) ∩ Interior(ageometry)).
                  ST_Dimension; -- See Description
            WHEN 2 THEN
               SET cr = (Interior(SELF) ∩ Boundary(ageometry)).
                  ST_Dimension; -- See Description
            WHEN 3 THEN
               SET cr = (Interior(SELF) ∩ Exterior(ageometry)).
                  ST_Dimension; -- See Description
            WHEN 4 THEN
               SET cr = (Boundary(SELF) ∩ Interior(ageometry)).
                  ST_Dimension; -- See Description
            WHEN 5 THEN
               SET cr = (Boundary(SELF) ∩ Boundary(ageometry)).
                  ST_Dimension; -- See Description
            WHEN 6 THEN
               SET cr = (Boundary(SELF) ∩ Exterior(ageometry)).
                  ST_Dimension; -- See Description
            WHEN 7 THEN
               SET cr = (Boundary(SELF) ∩ Interior(ageometry)).
                  ST_Dimension; -- See Description
```

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

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 the CARDINALITY of *amatrix* is not 9, then an exception condition is raised: *SQL/MM Spatial exception – invalid intersection matrix*.

   b) 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*.

c) 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 in 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 *CR* 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 *CR* is not in the set *SRI*, then set *RESULT* to 0 (zero).

v) Return *RESULT*.

5.1.22 ST_Disjoint Method

Purpose

Tests 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

Tests if an ST_Geometry value is 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) The null-call method *ST_Intersects(ST_Geometry)* returns the result of the value expression: CASE SELF.*ST_Disjoint*(*ageometry*) WHEN 1 THEN 0 WHEN 0 THEN 1 ELSE NULL END.

5.1.24 ST_Touches Method

Purpose

Tests if an ST_Geometry value is 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, return the result of the value expression: 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.

5.1.25 ST_Crosses Method

Purpose

Tests if an ST_Geometry value is 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 = 0 THEN
            ageometry.ST_Relate(SELF, '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******')
         WHEN SELF.ST_Dimension = 2 AND ageometry.ST_Dimension = 0 THEN
            ageometry.ST_Relate(SELF, 'T*T******')
         WHEN SELF.ST_Dimension = 2 AND ageometry.ST_Dimension = 1 THEN
            ageometry.ST_Relate(SELF, '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 0 (zero), then return the result of the value expression: *ageometry*.*ST_Relate*(SELF, 'T*T******').

   d) 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********').

   e) 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******').

   f) If the dimension of SELF is equal to 2 and the dimension of *ageometry* is equal to 0 (zero), then return the result of the value expression: *ageometry*.*ST_Relate*(SELF, 'T*T******').

g) If the dimension of SELF is equal to 2 and the dimension of *ageometry* is equal to 1 (one), then return the result of the value expression: *ageometry*.*ST_Relate*(SELF, 'T*T******').

h) Otherwise, return the null value.

5.1.26 ST_Within Method

Purpose

Tests 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*****FF*')
```

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*****FF*').

5.1.27 ST_Contains Method

Purpose

Tests if an ST_Geometry value is spatially contains 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

Tests if an ST_Geometry value is 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 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
   --
   -- See Description
   --
```

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 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 body>

<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 *ST_CircularString* value.

   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 *ST_CompoundCurve* value.

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 *ST_MultiPoint* value.

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 *ST_MultiCurve* value.

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 *ST_MultiLineString* value.

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 *ST_MultiSurface* value.

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 *ST_MultiPolygon* value.

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 *ST_GeomCollection* value.

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 *ST_LineString* value.

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 *ST_CurvePolygon* value.

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 *ST_Polygon* value.

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>.

bb) 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>.

cc) 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.

dd) 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.

ee) 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.

ff) 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.

gg) 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.

hh) 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.

ii) 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.

jj) 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.

kk) The list of keywords are CIRCULARSTRING, COMPOUNDCURVE, CURVEPOLYGON, EMPTY, GEOMETRYCOLLECTION, LINESTRING, MULTICURVE, MULTILINESTRING, MULTIPOINT, MULTIPOLYGON, MULTISURFACE, POINT, and POLYGON.

3) The parameter *awkt* is the well-known text representation of an *ST_Geometry* value and it must be producible in the BNF for <well-known text representation>.

4) The null-call method *ST_WKTToSQL(CHARACTER LARGE OBJECT)* returns an *ST_Geometry* value represented by *awkt.*

5.1.30 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
   --
   -- See Description
   --
```

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 must be produced in the BNF for <well-known text representation>.

5.1.31 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
   --
   -- See Description
   --
```

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 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> ::= !! See Description

<wkblinestring> ::= !! See Description

<wkbcircularstring> ::= !! See Description
```

```
<wkbcompoundcurve> ::= !! See Description

<wkbpolygon> ::= !! See Description

<wkbcurvepolygon> ::= !! See Description

<wkbmultipoint> ::= !! See Description

<wkbmultilinestring> ::= !! See Description

<wkbmulticurve> ::= !! See Description

<wkbmultisurface> ::= !! See Description

<wkbmultipolygon> ::= !! See Description

<wkbgeometrycollection> ::= !! See Description

<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) <point binary representation> is the well-known binary representation for an *ST_Point* value that is produced by <wkbpoint binary>.

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) <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*).

e) <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*).

f) <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*).

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 <curvepolygon binary>, 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 *ST_CurvePolygon* value.

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

i) <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:

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

ii) 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*).

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 binary representation> produces an *ST_Polygon* value as the result of the value expression: NEW *ST_Polygon*(*AER, AIR*).

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) <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*)

l) Case:

i) If <multicurve binary representation> immediately contains a <multicurve binary>, 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) Otherwise, <multicurve binary representation> produces an *ST_MultiLineString* value specified by the immediately contained <multilinestring binary representation>.

m) <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*).

n) Case:

   i) If <multisurface binary representation> immediately contains a <multisurface binary>, 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) Otherwise, <multisurface binary representation> produces an *ST_MultiPolygon* value specified by the immediately contained <multipolygon binary representation>.

o) <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*).

p) <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*).

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) <wkbpoint> is a <uint32> with the value 1 (one). <wkblinestring> is a <uint32> with the value 2. <wkbcircularstring> is a <uint32> with the value 1000001. <wkbcompoundcurve> is a <uint32> with the value 1000002. <wkbpolygon> is a <uint32> with the value 3. <wkbcurvepolygon> is a <uint32> with the value 1000003. <wkbmultipoint> is a <uint32> with the value 4. <wkbmulticurve> is a <uint32> with the value 1000004. <wkbmultilinestring> is a <uint32> with the value 5. <wkbmultisurface> is a <uint32> with the value 1000005. <wkbmultipolygon> is a <uint32> with the value 6. <wkbgeometrycollection> is a <uint32> with the value 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).

bb) <byte> is an 8 bit (1 (one) octet) data type that encodes an unsigned integer in the range [0, 255].

cc) <uint32> s a 32 bit (4 octets) data type that encodes an unsigned integer in the range [0, 4294967295].

dd) <double> is a 64 bit (8 octets) double precision data type that encodes a double precision format using the IEC 559:1989.

ee) <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.

3) The parameter *awkb* is the well-known binary representation of an *ST_Geometry* value and it must be producible in the BNF for <well-known binary representation>.

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

5.1.32 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
   --
   -- See Description
   --
```

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 must be produced in the BNF for <well-known binary representation>.

5.1.33 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
   STATIC DISPATCH
   --
   -- See Description
   --

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

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 and it must be producible in the BNF for <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 *asrid*.

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 and it must be producible in the BNF for <well-known text representation>.

   b) Return an *ST_Geometry* value represented by *awkt* with the spatial reference system identifier set to *asrid*.

5.1.34 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
   STATIC DISPATCH
   --
   -- See Description
   --

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

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 and it must be producible in the BNF for <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 *asrid*.

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 and it must be producible in the BNF for <well-known binary representation>.

   b) Return an *ST_Geometry* value represented by *awkb* with the spatial reference system identifier set to *asrid*.

5.1.35 ST_OrderingEquals Function

Purpose

Tests 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 ageometry.ST_Equals(anothergeometry)
         WHEN 1 THEN 0
         ELSE 1
      END

CREATE ORDERING FOR ST_Geometry
   EQUALS ONLY BY RELATIVE 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.

5.1.36 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())
```

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.

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*.

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

   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,

   METHOD ST_Point
      (xcoord DOUBLE PRECISION, ycoord DOUBLE PRECISION, asrid 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) The attribute *ST_PrivateX* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST_PrivateX*.

2) The attribute *ST_PrivateY* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST_PrivateY*.

Description

1) The *ST_Point* type provides for public use:

   a) a method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION)*,

   b) a method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION, INTEGER)*,

   c) a method *ST_X()*,

   d) a method *ST_X(DOUBLE PRECISION)*,

   e) a method *ST_Y()*,

   f) a method *ST_Y(DOUBLE PRECISION)*,

   g) a method *ST_ExplicitPoint()*,

   h) a function *ST_PointFromText(CHARACTER LARGE OBJECT)*,

   i) a function *ST_PointFromText(CHARACTER LARGE OBJECT, INTEGER)*,

   j) a function *ST_PointFromWKB(BINARY LARGE OBJECT)*,

   k) a function *ST_PointFromWKB(BINARY 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 implicitly defined constructor function corresponds to the empty set.

10) An *ST_Point* value is not well formed if either:

    a) *ST_PrivateX* is the null value and *ST_PrivateY* is not the null value, or

    b) *ST_PrivateY* is the null value and *ST_PrivateX* is not the null value.

6.1.2 ST_Point Methods

Purpose

Return a specified ST_Point value.

Definition

```
CREATE METHOD ST_Point
   (xcoord DOUBLE PRECISION, ycoord DOUBLE PRECISION)
   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(0).                       -- SRID = asrid,
      ST_X(xcoord).                     -- ST_X = xcoord,
      ST_Y(ycoord)                      -- ST_Y = ycoord

CREATE METHOD ST_Point
   (xcoord DOUBLE PRECISION, ycoord DOUBLE PRECISION, asrid 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(asrid).                   -- SRID = asrid,
      ST_X(xcoord).                     -- ST_X = xcoord,
      ST_Y(ycoord)                      -- ST_Y = ycoord
```

Description

1) 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*.

2) The null-call type preserving method *ST_Point(DOUBLE PRECISION, DOUBLE PRECISION)* 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 0 (zero).

   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*.

3) 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 *asrid*.

4) The null-call type preserving 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 *asrid*.

   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 parameter';
      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 parameter*.

   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 parameter';
      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 parameter*.

   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 ARRAY[SELF.ST_X, SELF.ST_Y]
```

Description

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

2) The null-call method *ST_ExplicitPoint()* returns 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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt) AS ST_Point)

CREATE FUNCTION ST_PointFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    asrid INTEGER)
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt, asrid) 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 and it must be producible in the BNF for <point 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 *asrid*.

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 and it must be producible in the BNF for <point text representation>.

   b) Return an *ST_Point* value represented by *awkt* with the spatial reference system identifier set to *asrid*.

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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_Point)

CREATE FUNCTION ST_PointFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    asrid INTEGER)
   RETURNS ST_Point
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb, asrid) 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 and it must be producible in the BNF for <point 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 *asrid*.

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 and it must be producible in the BNF for <point binary representation>.

   b) Return an *ST_Point* value represented by *awkb* with the spatial reference system identifier set to *asrid*.

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_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
```

Description

1) The *ST_Curve* type provides for public use:

   a) a method *ST_Length()*,

   b) a method *ST_StartPoint()*,

   c) a method *ST_EndPoint()*,

   d) a method *ST_IsClosed()*,

   e) a method *ST_IsRing()*,

   f) 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 $\in$ R | a <= x <= b } under a mapping f:[a, b] $\rightarrow R^2$.

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 Method

Purpose

Return the length measurement of an ST_Curve value.

Definition

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

Description

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

2) The null-call method *ST_Length()* returns the implementation-defined length of SELF as measured in its spatial reference system.

3) The returned value is in the linear units of measure of the spatial reference system of SELF.

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
   --
   -- See Description
   --
```

Description

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

2) The null-call method *ST_StartPoint()* returns 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
   --
   -- See Description
   --
```

Description

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

2) The null-call method *ST_EndPoint()* returns the *ST_Point* value that is the end point of SELF.

7.1.5 ST_IsClosed Method

Purpose

Tests if an ST_Curve value is closed.

Definition

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

Description

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

2) The null-call method *ST_IsClosed()* returns 1 (one) if the start point of SELF is equal to the end point of SELF; otherwise 0 (zero).

7.1.6 ST_IsRing Method

Purpose

Tests if an ST_Curve value is a ring.

Definition

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

Description

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

2) The null-call method *ST_IsRing()* returns 1 (one) if SELF is simple and SELF is closed; otherwise 0 (zero).

7.1.7 ST_CurveToLine Method

Purpose

Returns the ST_LineString value approximation of an ST_Curve value.

Definition

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

Description

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

2) The null-call method *ST_CurveToLine()* returns 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

   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,

   METHOD ST_LineString
      (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
       asrid 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) The attribute *ST_PrivatePoints* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST_PrivatePoints*.

Description

1) The *ST_LineString* type provides for public use:

   a) a method *ST_LineString(ST_Point ARRAY)*,

   b) a method *ST_LineString(ST_Point ARRAY, INTEGER)*,

   c) a method *ST_Points()*,

   d) a method *ST_Points(ST_Point ARRAY)*,

   e) a method *ST_NumPoints()*,

   f) a method *ST_PointN(INTEGER)*,

   g) an overriding method *ST_StartPoint()*,

   h) an overriding method *ST_EndPoint()*,

   i) a function *ST_LineFromText(CHARACTER LARGE OBJECT)*,

   j) a function *ST_LineFromText(CHARACTER LARGE OBJECT, INTEGER)*,

   k) a function *ST_LineFromWKB(BINARY LARGE OBJECT)*,

   l) a function *ST_LineFromWKB(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) 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 is called 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 implicitly defined 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 a specified ST_LineString value.

Definition

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

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

Definitional Rules

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

Description

1) The method *ST_LineString(ST_Point ARRAY)* takes the following input parameters:

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

2) The null-call type preserving 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*.

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

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

   b) an INTEGER value *asrid*.

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

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

   b) Using the method *ST_Points(ST_Point ARRAY),* the *ST_PrivatePoints* attribute set to *apointarray*.

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 SELF.ST_PrivatePoints

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 NULL;
      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) The null-call method *ST_Points()* returns 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 CARDINALITY(SELF.ST_PrivatePoints)
```

Description

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

2) The null-call method *ST_NumPoints()* returns 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_NumPoints = 0 THEN
         SIGNAL SQLSTATE '2FF06'
            SET MESSAGE_TEXT 'empty array';
      END IF;
      IF aposition < 1 OR
         aposition > SELF.ST_NumPoints THEN
         SIGNAL SQLSTATE '2FF01'
            SET MESSAGE_TEXT 'invalid position';
      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 the cardinality of the *ST_PrivatePoints* attribute is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – empty array*.

   b) If *aposition* is less than 1 (one) or greater than the cardinality of the *ST_PrivatePoints* attribute, then an exception condition is raised: *SQL/MM Spatial exception – invalid position*.

   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 SELF.ST_Points[1]
```

Description

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

2) The null-call method *ST_StartPoint()* returns 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 SELF.ST_Points[SELF.ST_NumPoints]
```

Description

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

2) The null-call method *ST_EndPoint()* returns 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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt) AS ST_LineString)

CREATE FUNCTION ST_LineFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    asrid INTEGER)
   RETURNS ST_LineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt, asrid) 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 and it must be producible in the BNF for <linestring 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 *asrid*.

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 and it must be producible in the BNF for <linestring text representation>.

   b) Return an *ST_LineString* value represented by *awkt* with the spatial reference system identifier set to *asrid*.

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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_LineString)

CREATE FUNCTION ST_LineFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    asrid INTEGER)
   RETURNS ST_LineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb, asrid) 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 and it must be producible in the BNF for <linestring 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 *asrid*.

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 and it must be producible in the BNF for <linestring binary representation>.

   b) Return an *ST_LineString* value represented by *awkb* with the spatial reference system identifier set to *asrid*.

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 []
   )
   INSTANTIABLE
   NOT FINAL

   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,

   METHOD ST_CircularString
      (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
       asrid 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
   DTERMINISTIC
   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) The attribute *ST_PrivatePoints* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST_PrivatePoints*.

Description

1) The *ST_CircularString* type provides for public use:

   a) a method *ST_CircularString(ST_Point ARRAY)*,

   b) a method *ST_CircularString(ST_Point ARRAY, INTEGER)*,

   c) a method *ST_Points()*,

   d) a method *ST_Points(ST_Point ARRAY)*,

   e) a method *ST_NumPoints()*,

   f) a method *ST_PointN(INTEGER)*,

   g) a method *ST_MidPointRep()*,

   h) an overriding method *ST_StartPoint()*,

   i) an overriding method *ST_EndPoint()*,

   j) a function *ST_CircularFromText(CHARACTER LARGE OBJECT)*,

k) a function *ST_CircularFromText(CHARACTER LARGE OBJECT, INTEGER)*,

l) a function *ST_CircularFromWKB(BINARY LARGE OBJECT)*,

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

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 implicitly defined 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 a specified ST_CircularString value.

Definition

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

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

Definitional Rules

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

Description

1) The method *ST_CircularString(ST_Point ARRAY)* takes the following input parameters:

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

2) The null call type preserving 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* .

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

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

   b) an INTEGER value *asrid*.

4) The null call type preserving method *ST_CircularString(ST_Point ARRAY, INTEGER)* returns an *ST_CircularString* value with:

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

   b) Using the method *ST_Points(ST_Point ARRAY),* the attribute *ST_PrivatePoints* array set to *apointarray*.

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 SELF.ST_PrivatePoints


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 NULL;
      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) The null-call method *ST_Points()* returns 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 CARDINALITY(SELF.ST_PrivatePoints)
```

Description

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

2) The null-call method *ST_NumPoints()* returns 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_NumPoints = 0 THEN
         SIGNAL SQLSTATE '2FF06'
            SET MESSAGE_TEXT 'empty array';
      END IF;
      IF aposition < 1 OR
         aposition > SELF.ST_NumPoints THEN
         SIGNAL SQLSTATE '2FF01'
            SET MESSAGE_TEXT 'invalid position';
      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 the cardinality of the attribute *ST_PrivatePoints* is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – empty array*.

   b) If *aposition* is less than 1 (one) or greater than the cardinality of the attribute *ST_PrivatePoints*, then an exception condition is raised: *SQL/MM Spatial exception – invalid position*.

   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
   --
   -- See Description
   --
```

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) The null-call method *ST_MidPointRep()* returns an *ST_Points* ARRAY such that:

   a) For the first circular arc segment of the curve, the points returned are the start, mid, and end points of the segment.

   b) 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 SELF.ST_Points[1]
```

Description

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

2) The null-call method *ST_StartPoint()* returns 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 SELF.ST_Points[SELF.ST_NumPoints]
```

Description

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

2) The null-call method *ST_EndPoint()* returns the result of the value expression: SELF.*ST_Points*[SELF.*ST_NumPoints*].

7.3.9 ST_CircularFromText Functions

Purpose

Return a specified ST_CircularString value.

Definition

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

CREATE FUNCTION ST_CircularFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    asrid INTEGER)
   RETURNS ST_CircularString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt, asrid) 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_CircularFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:

   a) a CHARACTER LARGE OBJECT value *awkt*.

2) For the null-call function *ST_CircularFromText(CHARACTER LARGE OBJECT)*:

   a) The parameter *awkt* is the well-known text representation of an *ST_CircularString* value and it must be producible in the BNF for <circularstring 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_CircularFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a CHARACTER LARGE OBJECT value *awkt*,

   b) an INTEGER value *asrid*.

4) For the null-call function *ST_CircularFromText(CHARACTER LARGE OBJECT, INTEGER)*:

   a) The parameter *awkt* is the well-known text representation of an *ST_CircularString* value and it must be producible in the BNF for <circularstring text representation>.

   b) Return an *ST_CircularString* value represented by *awkt* with the spatial reference system identifier set to *asrid*.

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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_CircularString)

CREATE FUNCTION ST_CircularFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    asrid INTEGER)
   RETURNS ST_CircularString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb, asrid) 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 and it must be producible in the BNF for <circularstring 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 *asrid*.

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 and it must be producible in the BNF for <circularstring binary representation>.

   b) Return an *ST_CircularString* value represented by *awkb* with the spatial reference system identifier set to *asrid*.

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 must 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

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

   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,

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

   METHOD ST_CompoundCurve
      (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements],
       asrid 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) The attribute *ST_PrivateCurves* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST_PrivateCurves*.

Description

1) The *ST_CompoundCurve* type provides for public use:

   a) a method *ST_CompoundCurve(ST_Curve)*,

   b) a method *ST_CompoundCurve(ST_Curve ARRAY)*,

   c) a method *ST_CompoundCurve(ST_Curve, INTEGER)*,

   d) a method *ST_CompoundCurve(ST_Curve ARRAY, INTEGER)*,

   e) a method *ST_Curves()*,

   f) a method *ST_Curves(ST_Curve ARRAY)*,

g) a method *ST_NumCurves()*,

h) a method *ST_CurveN(INTEGER)*,

i) an overriding method *ST_StartPoint()*,

j) an overriding method *ST_EndPoint()*,

k) a function *ST_CompoundFromText(CHARACTER LARGE OBJECT)*,

l) a function *ST_CompoundFromText(CHARACTER LARGE OBJECT, INTEGER)*,

m) a function *ST_CompoundFromWKB(BINARY LARGE OBJECT)*,

n) 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 and each *ST_Curve* value in the *ST_PrivateCurves* attribute, except the first, has a start point equal to the end point of the previous *ST_Curve* value, 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_Curve* value is the 2.

7) This type of *ST_CompoundCurve* 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 must 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 implicitly defined 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 a specified ST_CompoundCurve value.

Definition

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

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

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

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

Definitional Rules

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

Description

1) The method *ST_CompoundCurve(ST_Curve)* takes the following input parameters:

   b) an *ST_Curve* value *acurve*.

2) The null-call type preserving 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*].

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

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

4) The null-call type preserving 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*.

5) The method *ST_CompoundCurve(ST_Curve, INTEGER)* takes the following input parameters:

   a) an *ST_Curve* value *acurve*,

   b) an INTEGER value *asrid*.

6) The null-call type preserving method *ST_CompoundCurve(ST_Curve, INTEGER)* returns an *ST_CompoundCurve* value with:

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

   b) Using the method *ST_Curves(ST_Curve ARRAY)*, the *ST_PrivateCurves* attribute set to ARRAY[*acurve*].

7) The method *ST_CompoundCurve(ST_Curve ARRAY, INTEGER)* takes the following input parameters:

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

   b) an INTEGER value *asrid*.

8) The null-call type preserving method *ST_CompoundCurve(ST_Curve ARRAY, INTEGER)* returns an *ST_CompoundCurve* value with:

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

   b) Using the method *ST_Curves(ST_Curve ARRAY)*, the *ST_PrivateCurves* attribute set to *acurvearray*.

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 SELF.ST_PrivateCurves

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 NULL;
      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 parameter';
         ENDIF;
         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';
         ENDIF;
         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) The null-call method *ST_Curves()* returns 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 parameter*.

      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 CARDINALITY(SELF.ST_PrivateCurves)
```

Description

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

2) The null-call method *ST_NumCurves()* returns 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 CARDINALITY(SELF.ST_PrivateCurves) = 0 THEN
         SIGNAL SQLSTATE '2FF06'
            SET MESSAGE_TEXT 'empty array';
      END IF;
      IF aposition < 1 OR
         aposition > CARDINALITY(SELF.ST_PrivateCurves) THEN
         SIGNAL SQLSTATE '2FF01'
            SET MESSAGE_TEXT 'invalid position';
      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 the cardinality of the *ST_PrivateCurves* attribute is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – empty array*.

   b) If *aposition* is less than 1 (one) or greater than the cardinality of the *ST_PrivateCurves* attribute, then an exception condition is raised: *SQL/MM Spatial exception – invalid position*.

   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 SELF.ST_Curves[1].ST_Points[1]
```

Description

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

2) The null-call method *ST_StartPoint()* returns 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 SELF.ST_Curves[SELF.ST_NumCurves].
      ST_Points[SELF.ST_Curves[SELF.ST_NumCurves].ST_NumPoints]
```

Description

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

2) The null-call method *ST_EndPoint()* returns 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_CompoundFromText Functions

Purpose

Return a specified ST_CompoundCurve value.

Definition

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

CREATE FUNCTION ST_CompoundFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    asrid INTEGER)
   RETURNS ST_CompoundCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt, asrid) 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_CompoundFromText(CHARACTER LARGE OBJECT)* takes the following input parameters:

   a) a CHARACTER LARGE OBJECT value *awkt*.

2) For the null-call function *ST_CompoundFromText(CHARACTER LARGE OBJECT)*:

   a) The parameter *awkt* is the well-known text representation of an *ST_CompoundCurve* value and it must be producible in the BNF for <compoundcurve 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_CompoundFromText(CHARACTER LARGE OBJECT, INTEGER)* takes the following input parameters:

   a) a CHARACTER LARGE OBJECT value *awkt*,

   b) an INTEGER value *asrid*.

4) For the null-call function *ST_CompoundFromText(CHARACTER LARGE OBJECT, INTEGER)*:

   a) The parameter *awkt* is the well-known text representation of an *ST_CompoundCurve* value and it must be producible in the BNF for <compoundcurve text representation>.

   b) Return an *ST_CompoundCurve* value represented by *awkt* with the spatial reference system identifier set to *asrid*.

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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_CompoundCurve)

CREATE FUNCTION ST_CompoundFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    asrid INTEGER)
   RETURNS ST_CompoundCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb, asrid) 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 and it must be producible in the BNF for <compoundcurve 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 *asrid*.

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 and it must be producible in the BNF for <compoundcurve binary representation>.

   b) Return an *ST_CompoundCurve* value represented by *awkb* with the spatial reference system identifier set to *asrid*.

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_Perimeter()
      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
```

Description

1) The *ST_Surface* type provides for public use:

   a) a method *ST_Area()*,

   b) a method *ST_Perimeter()*,

   c) a method *ST_Centroid()*,

   d) 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 Method

Purpose

Return the area measurement of an ST_Surface value.

Definition

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

Description

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

2) The null-call method *ST_Area()* returns the implementation-defined area of SELF as measured in its spatial reference system.

3) The returned value is in the linear units of measure of the spatial reference system of SELF squared.

8.1.3 ST_Perimeter Method

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 SELF.ST_Boundary.ST_Length
```

Description

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

2) The null-call method *ST_Perimeter()* returns the implementation-defined length of the boundary of SELF as measured in its spatial reference system.

3) The returned value is in the linear units of measure in the spatial reference system of SELF.

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
   --
   -- See Description
   --
```

Description

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

2) The null-call method *ST_Centroid()* returns the mathematical centroid of the *ST_Surface* value. The result is not guaranteed to spatially intersect the *ST_Surface* value.

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
   --
   -- See Description
   --
```

Description

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

2) The null-call method *ST_PointOnSurface()* returns an *ST_Point* value guaranteed to spatially intersect the *ST_Surface* value.

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

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

   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,

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

```
METHOD ST_CurvePolygon
   (acurve ST_Curve,
    acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements],
    asrid 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) The attribute *ST_PrivateExteriorRing* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST_PrivateExteriorRing*.

3) The attribute *ST_PrivateInteriorRings* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST_PrivateInteriorRings*.

Description

1) The *ST_CurvePolygon* type provides for public use:

   a) a method *ST_CurvePolygon(ST_Curve)*,

   b) a method *ST_CurvePolygon(ST_Curve, ST_Curve ARRAY)*,

   c) a method *ST_CurvePolygon(ST_Curve, INTEGER)*,

   d) a method *ST_CurvePolygon(ST_Curve, ST_Curve ARRAY, INTEGER)*,

   e) a method *ST_ExteriorRing()*,

   f) a method *ST_ExteriorRing(ST_Curve)*,

   g) a method *ST_InteriorRings()*,

   h) a method *ST_InteriorRings(ST_Curve ARRAY)*,

   i) a method *ST_NumInteriorRing()*,

   j) a method *ST_InteriorRingN(INTEGER)*,

   k) a method *ST_CurvePolyToPoly()*,

   l) a function *ST_CPolyFromText(CHARACTER LARGE OBJECT)*,

   m) a function *ST_CPolyFromText(CHARACTER LARGE OBJECT, INTEGER)*,

   n) a function *ST_CPolyFromWKB(BINARY LARGE OBJECT)*,

   o) 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 may 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 closed point set.

15) An *ST_CurvePolygon* value returned by the implicitly defined 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 a specified ST_CurvePolygon value.

Definition

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

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

Definitional Rules

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

Description

1) The method *ST_CurvePolygon(ST_Curve)* takes the following input parameters:

   b) an *ST_Curve* value *acurve*.

2) The null-call type preserving 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*.

c) Using the method *ST_InteriorRings(ST_Curve ARRAY)*, the *ST_PrivateInteriorRings* attribute set to an empty *ST_Curve* ARRAY value.

3) 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*.

4) The null-call type preserving 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*.

c) Using the method *ST_InteriorRings(ST_Curve ARRAY)*, the *ST_PrivateInteriorRings* attribute set to *acurvearray*.

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

a) an *ST_Curve* value *acurve*,

b) an INTEGER value *asrid*.

6) The null-call type preserving method *ST_CurvePolygon(ST_Curve, INTEGER)* returns an *ST_CurvePolygon* value with:

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

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

c) Using the method *ST_InteriorRings(ST_Curve ARRAY)*, the *ST_PrivateInteriorRings* attribute set to an empty *ST_Curve* ARRAY value.

7) 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 *asrid*.

8) The null-call type preserving method *ST_CurvePolygon(ST_Curve, ST_Curve ARRAY, INTEGER)* returns an *ST_CurvePolygon* value with:

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

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

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 SELF.ST_PrivateExteriorRing

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

      IF acurve IS NULL THEN
         SIGNAL SQLSTATE '2FF03'
            SET MESSAGE_TEXT 'null parameter';
      END IF;
      -- If SELF is the null value, then return the null value.
      IF SELF IS NULL THEN
         RETURN NULL;
      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 parameter';
      ENDIF;
      -- 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 parameter';
         ENDIF;
         -- 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 parameter';
         ENDIF;
         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) The null-call method *ST_ExteriorRing()* returns 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 parameter*.

   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 parameter*.

   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 parameter*.

   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 parameter*.

   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 SELF.ST_PrivateInteriorRings

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
         SIGNAL SQLSTATE '2FF03'
            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 parameter';
         ENDIF;
         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 parameter';
         ENDIF;
```

```
            -- 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 parameter';
            ENDIF;
            SET acounter = acounter + 1;
         END WHILE;
         -- For each ring pair in acurvearray
         SET acounter = 1;
         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 parameter';
               ENDIF;
               -- 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 parameter';
               ENDIF;
               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_PrivateDimension(2).
            ST_PrivateCoordinateDimension(2).
            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) The null-call method *ST_InteriorRings()* returns 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 *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 parameter*.

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 parameter*.

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 parameter*.

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 parameter*.

i) Otherwise, return an *ST_CurvePolygon* value with

   i) The dimension set to 2.

   ii) The coordinate dimension set to 2.

   iii) 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 CARDINALITY(SELF.ST_PrivateInteriorRings)
```

Description

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

2) The null-call method *ST_NumInteriorRing()* returns 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 CARDINALITY(SELF.ST_PrivateInteriorRings) = 0 THEN
         SIGNAL SQLSTATE '2FF06'
            SET MESSAGE_TEXT 'empty array';
      END IF;
      IF aposition < 1 OR
         aposition > CARDINALITY(SELF.ST_PrivateInteriorRings) THEN
         SIGNAL SQLSTATE '2FF01'
            SET MESSAGE_TEXT 'invalid position';
      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 the cardinality of the *ST_PrivateInteriorRings* attribute is equal to 0 (zero), then an exception condition is raised: *SQL/MM Spatial exception – empty array*.

   b) If *aposition* is less than one or greater than the cardinality of the *ST_PrivateInteriorRings* attribute, then an exception condition is raised: *SQL/MM Spatial exception – invalid position*.

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

8.2.7 ST_CurvePolyToPoly Method

Purpose

Returns the ST_Polygon approximation of an ST_CurvePolygon value.

Definition

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

Description

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

2) The null-call method *ST_CurvePolyToPoly()* returns 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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt) AS ST_CurvePolygon)

CREATE FUNCTION ST_CPolyFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    asrid INTEGER)
   RETURNS ST_CurvePolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt, asrid) 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 and it must be producible in the BNF for <curvepolygon 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 *asrid*.

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 and it must be producible in the BNF for <curvepolygon text representation>.

   b) Return an *ST_CurvePolygon* value represented by *awkt* with the spatial reference system identifier set to *asrid*.

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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_CurvePolygon)

CREATE FUNCTION ST_CPolyFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    asrid INTEGER)
   RETURNS ST_CurvePolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb, asrid) 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 and it must be producible in the BNF for <curvepolygon 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 *asrid*.

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 and it must be producible in the BNF for <curvepolygon binary representation>.

   b) Return an *ST_CurvePolygon* value represented by *awkb* with the spatial reference system identifier set to *asrid*.

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

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

   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,

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

   METHOD ST_Polygon
      (alinestring ST_LineString,
       alinestringarray ST_LineString ARRAY[ST_MaxGeometryArrayElements],
       asrid 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.

Description

1) The *ST_Polygon* type provides for public use:

   a) a method *ST_Polygon(ST_LineString)*,

   b) a method *ST_Polygon(ST_LineString, ST_LineString ARRAY)*,

   c) a method *ST_Polygon(ST_LineString, INTEGER)*,

   d) a method *ST_Polygon(ST_LineString, ST_LineString ARRAY, INTEGER)*,

   e) an overriding method *ST_ExteriorRing()*,

   f) an overriding method *ST_ExteriorRing(ST_Curve)*,

   g) an overriding method *ST_InteriorRings()*,

   h) an overriding method *ST_InteriorRings(ST_Curve ARRAY)*,

   i) an overriding method *ST_InteriorRingN(INTEGER)*,

   j) a function *ST_PolyFromText(CHARACTER LARGE OBJECT)*,

   k) a function *ST_PolyFromText(CHARACTER LARGE OBJECT, INTEGER)*,

   l) a function *ST_PolyFromWKB(BINARY LARGE OBJECT)*,

   m) a function *ST_PolyFromWKB(BINARY LARGE OBJECT, INTEGER)*,

   n) a function *ST_BdPolyFromText(CHARACTER LARGE OBJECT)*,

   o) a function *ST_BdPolyFromText(CHARACTER LARGE OBJECT, INTEGER)*,

   p) a function *ST_BdPolyFromWKB(BINARY LARGE OBJECT)*,

   q) 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.

8.3.2 ST_Polygon Methods

Purpose

Return a specified ST_Polygon value.

Definition

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

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

Definitional Rules

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

Description

1) The method *ST_Polygon(ST_LineString)* takes the following input parameters:

   b) an *ST_LineString* value *alinestring*.

2) The null-call type preserving 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*.

   c) Using the method *ST_InteriorRings(ST_Curve ARRAY),* the *ST_PrivateInteriorRings* attribute set to an empty *ST_LineString* ARRAY value.

3) 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*.

4) The null-call type preserving 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*.

   c) Using the method *ST_InteriorRings(ST_Curve ARRAY),* the *ST_PrivateInteriorRings* attribute set to *alinestringarray*.

5) The method *ST_Polygon(ST_LineString, INTEGER)* takes the following input parameters:

   a) an *ST_LineString* value *alinestring*,

   b) an INTEGER value *asrid*.

6) The null-call type preserving method *ST_Polygon(ST_LineString, INTEGER)* returns an *ST_Polygon* value with:

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

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

   c) Using the method *ST_InteriorRings(ST_Curve ARRAY),* the *ST_PrivateInteriorRings* attribute set to an empty *ST_LineString* ARRAY value.

7) 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 *asrid*.

8) The null-call type preserving method *ST_Polygon(ST_LineString, ST_LineString ARRAY, INTEGER)* returns an *ST_Polygon* value with:

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

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

   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 CAST(SELF.ST_PrivateExteriorRing AS ST_LineString)

CREATE METHOD ST_ExteriorRing(acurve ST_Curve)
   RETURNS ST_Polygon
   FOR ST_Polygon
      -- 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 an linestring value';
      END IF;
      RETURN (SELF AS ST_CurvePolygon).ST_ExteriorRing(acurve);
   END
```

Description

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

2) The null-call method *ST_ExteriorRing()* returns 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 an 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 CAST(SELF.ST_PrivateInteriorRings AS
      ST_LineString ARRAY[ST_MaxGeometryArrayElements])

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';
         ENDIF;
         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) The null-call method *ST_InteriorRings()* returns 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 *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 TREAT((SELF AS ST_CurvePolygon).ST_InteriorRingN(aposition) AS
      ST_LineString)
```

Description

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

   a) an INTEGER value *aposition*.

2) The null-call method *ST_InteriorRingN(INTEGER)* returns 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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt) AS ST_Polygon)

CREATE FUNCTION ST_PolyFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    asrid INTEGER)
   RETURNS ST_Polygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt, asrid) 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 and it must be producible in the BNF for <polygon 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 *asrid*.

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 and it must be producible in the BNF for <polygon text representation>.

   b) Return an *ST_Polygon* value represented by *awkt* with the spatial reference system identifier set to *asrid*.

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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_Polygon)

CREATE FUNCTION ST_PolyFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    asrid INTEGER)
   RETURNS ST_Polygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb, asrid) 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 and it must be producible in the BNF for <polygon 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 *asrid*.

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 and it must be producible in the BNF for <polygon binary representation>.

   b) Return an *ST_Polygon* value represented by *awkb* with the spatial reference system identifier set to *asrid*.

8.3.8 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
   STATIC DISPATCH
   RETURN ST_BdPolyFromText(awkt, 0)

CREATE FUNCTION ST_BdPolyFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    asrid INTEGER)
   RETURNS ST_Polygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   --
   -- See Description
   --
```

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* 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 *asrid*.

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 and it must be producible in the BNF for <multilinestring 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 parameter*.

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) Returns an *ST_Polygon* value with:

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

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*.

8.3.9 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
   STATIC DISPATCH
   RETURN ST_BdPolyFromWKB(wkb, 0)

CREATE FUNCTION ST_BdPolyFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    asrid INTEGER)
   RETURNS ST_Polygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   --
   -- See Description
   --
```

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* 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 *asrid*.

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 and it must be producible in the BNF for <multilinestring 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 parameter*.

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) Returns an *ST_Polygon* value with:

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

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

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

   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,

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

```
METHOD ST_GeomCollection
   (ageometryarray ST_Geometry
      ARRAY[ST_MaxGeometryArrayElements],
    asrid 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) The attribute *ST_PrivateGeometries* is not for public use. There are no GRANT statements granting EXECUTE privilege to the observer or mutator method for *ST_PrivateGeometries*.

Description

1) The *ST_GeomCollection* type provides for public use:

   a) a method *ST_GeomCollection(ST_Geometry)*,

   b) a method *ST_GeomCollection(ST_Geometry ARRAY)*,

   c) a method *ST_GeomCollection(ST_Geometry, INTEGER)*,

d) a method *ST_GeomCollection(ST_Geometry ARRAY, INTEGER)*,

e) a method *ST_Geometries()*,

f) a method *ST_Geometries(ST_Geometry ARRAY)*,

g) a method *ST_NumGeometries()*,

h) a method *ST_GeometryN(INTEGER)*,

i) a function *ST_GeomCollFromTxt(CHARACTER LARGE OBJECT)*,

j) a function *ST_GeomCollFromTxt(CHARACTER LARGE OBJECT, INTEGER)*,

k) a function *ST_GeomCollFromWKB(BINARY LARGE OBJECT)*,

l) a function *ST_GeomCollFromWKB(BINARY 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 implicitly defined 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 a specified ST_GeomCollection value.

Definition

```
CREATE METHOD ST_GeomCollection
   (ageometry ST_Geometry)
   RETURNS ST_GeomCollection
   FOR ST_GeomCollection
   RETURN SELF.ST_SRID(ageometry.ST_SRID).
      ST_Geometries(ARRAY[ageometry])

CREATE 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 METHOD ST_GeomCollection
   (ageometry ST_Geometry,
    asrid INTEGER)
   RETURNS ST_GeomCollection
   FOR ST_GeomCollection
   RETURN SELF.ST_SRID(asrid).ST_Geometries(ARRAY[ageometry])

CREATE METHOD ST_GeomCollection
   (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements],
    asrid INTEGER)
   RETURNS ST_GeomCollection
   FOR ST_GeomCollection
   RETURN SELF.ST_SRID(asrid).ST_Geometries(ageometryarray)
```

Definitional Rules

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

Description

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

   a) an *ST_Geometry* value *ageometry*.

2) The null-call type preserving 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*].

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

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

4) The null-call type preserving 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*.

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

   a) an *ST_Geometry* value *ageometry*,

   b) an INTEGER value *asrid*.

6) The null-call type preserving method *ST_GeomCollection(ST_Geometry, INTEGER)* returns an *ST_GeomCollection* value with:

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

   b) Using the method *ST_Geometries(ST_Geometry ARRAY),* the *ST_PrivateGeometries* attribute set to ARRAY[*ageometry*].

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

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

   b) an INTEGER value *asrid*.

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

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

   b) Using the method *ST_Geometries(ST_Geometry ARRAY),* the *ST_PrivateGeometries* attribute set to *ageometryarray*.

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 SELF.ST_PrivateGeometries

CREATE METHOD ST_Geometries
   (ageometryarray ST_Geometry ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_GeomCollection
   FOR ST_GeomCollection
   BEGIN
      DECLARE counter INTEGER;

      -- 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 NULL;
      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) The null-call method *ST_Geometries()* returns 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 CARDINALITY(SELF.ST_PrivateGeometries)
```

Description

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

2) The null-call method *ST_NumGeometries()* returns 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
   RETURN SELF.ST_PrivateGeometries[aposition]
```

Description

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

   a) an INTEGER value a*position*.

2) The null-call method *ST_GeometryN(INTEGER)* returns 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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt) AS ST_GeomCollection)

CREATE FUNCTION ST_GeomCollFromTxt
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    asrid INTEGER)
   RETURNS ST_GeomCollection
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt, asrid) 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 and it must be producible in the BNF for <geometrycollection 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 *asrid*.

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 and it must be producible in the BNF for <geometrycollection text representation>.

   b) Return an *ST_GeomCollection* value represented by *awkt* with the spatial reference system identifier set to *asrid*.

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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_GeomCollection)

CREATE FUNCTION ST_GeomCollFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    asrid INTEGER)
   RETURNS ST_GeomCollection
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb, asrid) 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 and it must be producible in the BNF for <geometrycollection 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 *asrid*.

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 and it must be producible in the BNF for <geometrycollection binary representation>.

   b) Return an *ST_GeomCollection* value represented by *awkb* with the spatial reference system identifier set to *asrid*.

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

   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,

   METHOD ST_MultiPoint
      (apointarray ST_Point
         ARRAY[ST_MaxGeometryArrayElements],
       asrid 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.

Description

1) The *ST_MultiPoint* type provides for public use:

   a) a method *ST_MultiPoint(ST_Point ARRAY)*,

   b) a method *ST_MultiPoint(ST_Point ARRAY, INTEGER)*,

   c) an overriding method *ST_Geometries()*,

   d) an overriding method *ST_Geometries(ST_Geometry ARRAY)*,

e) a function *ST_MPointFromText(CHARACTER LARGE OBJECT)*,

f) a function *ST_MPointFromText(CHARACTER LARGE OBJECT, INTEGER)*,

g) a function *ST_MPointFromWKB(BINARY LARGE OBJECT)*,

h) a function *ST_MPointFromWKB(BINARY 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 all of *ST_Point* values in *ST_PrivateGeometries* attribute are well formed.

9.2.2 ST_MultiPoint Methods

Purpose

Return a specified ST_MultiPoint value.

Definition

```
CREATE METHOD ST_MultiPoint
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiPoint
   FOR ST_MultiPoint
   RETURN SELF.ST_SRID(0).ST_Geometries(apointarray)

CREATE METHOD ST_MultiPoint
   (apointarray ST_Point ARRAY[ST_MaxGeometryArrayElements],
    asrid INTEGER)
   RETURNS ST_MultiPoint
   FOR ST_MultiPoint
   RETURN SELF.ST_SRID(asrid).ST_Geometries(apointarray)
```

Definitional Rules

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

Description

1) The method *ST_MultiPoint(ST_Point ARRAY)* takes the following input parameters:

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

2) The null-call type preserving 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*.

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

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

   b) an INTEGER value *asrid*.

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

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

   b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the *ST_PrivateGeometries* attribute set to *apointarray*.

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 CAST(SELF.ST_PrivateGeometries AS
      ST_Point ARRAY[ST_MaxGeometryArrayElements])

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) The null-call method *ST_Geometries()* returns 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, return an *ST_MultiPoint* value with:

         1) The dimension set to 0 (zero).

         2) Using the method *ST_Geometries(ST_Geometry ARRAY)* for type *ST_GeomCollection,* 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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt) AS ST_MultiPoint)

CREATE FUNCTION ST_MPointFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    asrid INTEGER)
   RETURNS ST_MultiPoint
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt, asrid) 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 and it must be producible in the BNF for <multipoint 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 *asrid*.

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 and it must be producible in the BNF for <multipoint text representation>.

   b) Return an *ST_MultiPoint* value represented by *awkt* with the spatial reference system identifier set to *asrid*.

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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_MultiPoint)

CREATE FUNCTION ST_MPointFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    asrid INTEGER)
   RETURNS ST_MultiPoint
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb, asrid) 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 and it must be producible in the BNF for <multipoint 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 *asrid*.

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 and it must be producible in the BNF for <multipoint binary representation>.

   b) Return an *ST_MultiPoint* value represented by *awkb* with the spatial reference system identifier set to *asrid*.

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

   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,

   METHOD ST_MultiCurve
      (acurvearray ST_Curve
         ARRAY[ST_MaxGeometryArrayElements],
       asrid 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,

   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.

Description

1) The *ST_MultiCurve* type provides for public use:

   a) a method *ST_MultiCurve(ST_Curve ARRAY)*,

   b) a method *ST_MultiCurve(ST_Curve ARRAY, INTEGER)*,

   c) a method *ST_IsClosed()*,

   d) a method *ST_Length()*,

   e) an overriding method *ST_Geometries()*,

   f) an overriding method *ST_Geometries(ST_Geometry ARRAY)*,

   g) a function *ST_MCurveFromText(CHARACTER LARGE OBJECT)*,

   h) a function *ST_MCurveFromText(CHARACTER LARGE OBJECT, INTEGER)*,

   i) a function *ST_MCurveFromWKB(BINARY LARGE OBJECT)*,

   j) 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.

9.3.2 ST_MultiCurve Methods

Return a specified ST_MultiCurve value.

Definition

```
CREATE METHOD ST_MultiCurve
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiCurve
   FOR ST_MultiCurve
   RETURN SELF.ST_SRID(0).ST_Geometries(acurvearray)

CREATE METHOD ST_MultiCurve
   (acurvearray ST_Curve ARRAY[ST_MaxGeometryArrayElements],
    asrid INTEGER)
   RETURNS ST_MultiCurve
   FOR ST_MultiCurve
   RETURN SELF.ST_SRID(asrid).ST_Geometries(acurvearray)
```

Definitional Rules

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

Description

1) The method *ST_MultiCurve(ST_Curve ARRAY)* takes the following input parameters:

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

2) The null-call type preserving 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*.

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

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

   b) an INTEGER value *asrid*.

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

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

   b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the attribute *ST_PrivateGeometries* set to *acurvearray*.

9.3.3 ST_IsClosed Method

Purpose

Tests if an ST_MultiCurve value is closed.

Definition

```
CREATE METHOD ST_IsClosed()
   RETURNS INTEGER
   FOR ST_MultiCurve
   --
   -- See Description
   --
```

Description

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

2) The null-call method *ST_IsClosed()* returns:

   Case:

   a) If the *ST_MultiCurve* value is closed, then 1 (one).

   b) Otherwise, 0 (zero).

9.3.4 ST_Length Method

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;

      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
```

Description

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

2) The null-call method *ST_Length()* returns the sum of the *ST_Length* values of each element in the *ST_PrivateGeometries* attribute of SELF.

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 CAST(SELF.ST_PrivateGeometries AS ST_Curve
      ARRAY[ST_MaxGeometryArrayElements])

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) The null-call method *ST_Geometries()* returns 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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt) AS ST_MultiCurve)

CREATE FUNCTION ST_MCurveFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    asrid INTEGER)
   RETURNS ST_MultiCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt, asrid) 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 and it must be producible in the BNF for <multicurve 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 *asrid*.

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 and it must be producible in the BNF for <multicurve text representation>.

   b) Return an *ST_MultiCurve* value represented by *awkt* with the spatial reference system identifier set to *asrid*.

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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_MultiCurve)

CREATE FUNCTION ST_MCurveFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    asrid INTEGER)
   RETURNS ST_MultiCurve
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb, asrid) 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 and it must be producible in the BNF for <multicurve 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 *asrid*.

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 and it must be producible in the BNF for <multicurve binary representation>.

   b) Return an *ST_MultiCurve* value represented by *awkb* with the spatial reference system identifier set to *asrid*.

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

   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,

   METHOD ST_MultiLineString
      (alinestringarray ST_LineString
         ARRAY[ST_MaxGeometryArrayElements],
       asrid 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.

Description

1) The *ST_MultiLineString* type provides for public use:

   a) a method *ST_MultiLineString(ST_LineString ARRAY)*,

   b) a method *ST_MultiLineString(ST_LineString ARRAY, INTEGER)*,

   c) an overriding method *ST_Geometries()*,

d) an overriding method *ST_Geometries(ST_Geometry ARRAY)*,

e) a function *ST_MLineFromText(CHARACTER LARGE OBJECT)*,

f) a function *ST_MLineFromText(CHARACTER LARGE OBJECT, INTEGER)*,

g) a function *ST_MLineFromWKB(BINARY LARGE OBJECT)*,

h) a function *ST_MLineFromWKB(BINARY 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 all of *ST_LineString* values in *ST_PrivateGeometries* attribute are well formed.

9.4.2 ST_MultiLineString Methods

Purpose

Return a specified ST_MultiLineString value.

Definition

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

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

Definitional Rules

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

Description

1) The method *ST_MultiLineString(ST_LineString ARRAY)* takes the following input parameters:

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

2) The null-call type preserving 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*.

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

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

   b) an INTEGER value *asrid*.

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

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

   b) Using the method *ST_Geometries(ST_Geometry ARRAY)*, the *ST_PrivateGeometries* attribute set to *alinestringarray*.

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 CAST(SELF.ST_PrivateGeometries AS
         ST_LineString ARRAY[ST_MaxGeometryArrayElements])

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) The null-call method *ST_Geometries()* returns 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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt) AS ST_MultiLineString)

CREATE FUNCTION ST_MLineFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    asrid INTEGER)
   RETURNS ST_MultiLineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt, asrid) 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 and it must be producible in the BNF for <multilinestring 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 *asrid*.

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 and it must be producible in the BNF for <multilinestring text representation>.

   b) Return an *ST_MultiLineString* value represented by *awkt* with the spatial reference system identifier set to *asrid*.

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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_MultiLineString)

CREATE FUNCTION ST_MLineFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    asrid INTEGER)
   RETURNS ST_MultiLineString
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb, asrid) 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 and it must be producible in the BNF for <multilinestring 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 *asrid*.

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 and it must be producible in the BNF for <multilinestring binary representation>.

   b) Return an *ST_MultiLineString* value represented by *awkb* with the spatial reference system identifier set to *asrid*.

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

   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,

   METHOD ST_MultiSurface
      (asurfacearray ST_Surface
         ARRAY[ST_MaxGeometryArrayElements],
       asrid 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_Perimeter()
      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.

Description

1) The *ST_MultiSurface* type provides for public use:

   a) a method *ST_MultiSurface(ST_Surface ARRAY)*,

   b) a method *ST_MultiSurface(ST_Surface ARRAY, INTEGER)*,

   c) a method *ST_Area()*,

   d) a method *ST_Perimeter()*,

   e) a method *ST_Centroid()*,

   f) a method *ST_PointOnSurface()*,

   g) an overriding method *ST_Geometries()*,

   h) an overriding method *ST_Geometries(ST_Geometry ARRAY)*,

   i) a function *ST_MSurfaceFromTxt(CHARACTER LARGE OBJECT)*,

   j) a function *ST_MSurfaceFromTxt(CHARACTER LARGE OBJECT, INTEGER)*,

   k) a function *ST_MSurfaceFromWKB(BINARY LARGE OBJECT)*,

   l) 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.

9.5.2 ST_MultiSurface Methods

Return a specified ST_MultiSurface value.

Definition

```
CREATE METHOD ST_MultiSurface
   (asurfacearray ST_Surface ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiSurface
   FOR ST_MultiSurface
   RETURN SELF.ST_SRID(0).ST_Geometries(asurfacearray)

CREATE METHOD ST_MultiSurface
   (asurfacearray ST_Surface ARRAY[ST_MaxGeometryArrayElements],
    asrid INTEGER)
   RETURNS ST_MultiSurface
   FOR ST_MultiSurface
   RETURN SELF.ST_SRID(asrid).ST_Geometries(asurfacearray)
```

Definitional Rules

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

Description

1) The method *ST_MultiSurface(ST_Surface ARRAY)* takes the following input parameters:

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

2) The null-call type preserving 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*.

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

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

   b) an INTEGER value *asrid*.

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

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

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

9.5.3 ST_Area Method

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;

      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
```

Description

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

2) The null-call method *ST_Area()* returns the sum of the *ST_Area* values of the elements in the *ST_PrivateGeometries* attribute of SELF.

9.5.4 ST_Perimeter Method

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;

      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
```

Description

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

2) The null-call method *ST_Perimeter()* returns the sum of the *ST_Perimeter* value of the elements in the *ST_PrivateGeometries* attribute of SELF.

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
   --
   -- See Description
   --
```

Description

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

2) The null-call method *ST_Centroid()* returns 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.

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
   --
   -- See Description
   --
```

Description

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

2) The null-call method *ST_PointOnSurface()* returns an *ST_Point* value guaranteed to spatially intersect an element in the collection of the *ST_MultiSurface* value.

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 CAST(SELF.ST_PrivateGeometries AS
      ST_Surface ARRAY[ST_MaxGeometryArrayElements])

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 1, 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 parameter';
            ENDIF;
            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) The null-call method *ST_Geometries()* returns 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 parameter*.

      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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt) AS ST_MultiSurface)

CREATE FUNCTION ST_MSurfaceFromTxt
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    asrid INTEGER)
   RETURNS ST_MultiSurface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt, asrid) 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 and it must be producible in the BNF for <multisurface 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 *asrid*.

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 and it must be producible in the BNF for <multisurface text representation>.

   b) Return an *ST_MultiSurface* value represented by *awkt* with the spatial reference system identifier set to *asrid*.

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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb) AS ST_MultiSurface)

CREATE FUNCTION ST_MSurfaceFromWKB
   (awkb BINARY LARGE OBJECT(ST_MaxGeometryAsBinary),
    asrid INTEGER)
   RETURNS ST_MultiSurface
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromWKB(awkb, asrid) 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 and it must be producible in the BNF for <multisurface 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 *asrid*.

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 and it must be producible in the BNF for <multisurface binary representation>.

   b) Return an *ST_MultiSurface* value represented by *awkb* with the spatial reference system identifier set to *asrid*.

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
   NOT INSTANTIABLE
   NOT FINAL

   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,

   METHOD ST_MultiPolygon
      (apolygonarray ST_Polygon
         ARRAY[ST_MaxGeometryArrayElements],
       asrid 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.

Description

1) The *ST_MultiPolygon* type provides for public use:

   a) a method *ST_MultiPolygon(ST_Polygon ARRAY)*,

   b) a method *ST_MultiPolygon(ST_Polygon ARRAY, INTEGER)*,

   c) an overriding method *ST_Geometries()*,

d) an overriding method *ST_Geometries(ST_Geometry ARRAY)*,

e) a function *ST_MPolyFromText(CHARACTER LARGE OBJECT)*,

f) a function *ST_MPolyFromText(CHARACTER LARGE OBJECT, INTEGER)*,

g) a function *ST_MPolyFromWKB(BINARY LARGE OBJECT)*,

h) a function *ST_MPolyFromWKB(BINARY LARGE OBJECT, INTEGER)*,

i) a function *ST_BdMPolyFromText(CHARACTER LARGE OBJECT)*,

j) a function ST_*BdMPolyFromText(CHARACTER LARGE OBJECT, INTEGER)*,

k) a function *ST_BdMPolyFromWKB(BINARY LARGE OBJECT)*,

l) 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 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 all of *ST_Polygon* values in *ST_PrivateGeometries* attribute are well formed.

9.6.2 ST_MultiPolygon Methods

Purpose

Return a specified ST_MultiPolygon value.

Definition

```
CREATE METHOD ST_MultiPolygon
   (apolygonarray ST_Polygon ARRAY[ST_MaxGeometryArrayElements])
   RETURNS ST_MultiPolygon
   FOR ST_MultiPolygon
   RETURN SELF.ST_SRID(0).ST_Geometries(apolygonarray)

CREATE METHOD ST_MultiPolygon
   (apolygonarray ST_Polygon ARRAY[ST_MaxGeometryArrayElements],
    asrid INTEGER)
   RETURNS ST_MultiPolygon
   FOR ST_MultiPolygon
   RETURN SELF.ST_SRID(asrid).ST_Geometries(apolygonarray)
```

Definitional Rules

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

Description

1) The method *ST_MultiPolygon(ST_Polygon ARRAY)* takes the following input parameters:

   a) an *ST_Polygon* value *apolygonarray*.

2) The null-call type preserving 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*.

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

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

   b) an INTEGER value *asrid*.

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

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

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

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 CAST(SELF.ST_PrivateGeometries AS
      ST_Polygon ARRAY[ST_MaxGeometryArrayElements])

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 1, 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 parameter';
            ENDIF;
            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) The null-call method *ST_Geometries()* returns 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 parameter*.

      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
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt) AS ST_MultiPolygon)

CREATE FUNCTION ST_MPolyFromText
   (awkt CHARACTER LARGE OBJECT(ST_MaxGeometryAsText),
    asrid INTEGER)
   RETURNS ST_MultiPolygon
   LANGUAGE SQL
   DETERMINISTIC
   CONTAINS SQL
   RETURNS NULL ON NULL INPUT
   STATIC DISPATCH
   RETURN TREAT(ST_GeomFromText(awkt, asrid) 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 and it must be producible in the BNF for <multipolygon 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 *asrid*.

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 and it must be producible in the BNF for <multipolygon text representation>.

   b) Return an *ST_MultiPolygon* value represented by *awkt* with the spatial reference system identifier set to *asrid*.