IBM® Netezza® Analytics
Release 3.0.0

Netezza Spatial Package
Reference Guide

IBM®
Note: Before using this information and the product that it supports, read the information in "Notices and Trademarks" on page 107.
Contents

Preface

Audience for This Guide.................................................................xiii
Purpose of This Guide........................................................................xiii
Conventions....................................................................................xiii
If You Need Help...............................................................................xiii
Comments on the Documentation...................................................xiv

1 List of functions by category

Spatial.............................................................................................15
Utilities - Actions.............................................................................17

2 Reference Documentation: Spatial

ST_Area - Area of the Geometry ....................................................19
ST_AsBinary - Well-known Binary Representation of the Geometry ...........................................21
ST_AsKML - KML representation of a Geometry .......................................................22
ST_AsText - WKT representation of a Geometry .............................................................23
ST_Boundary - Boundary of the Geometry .......................................................24
ST_Buffer - Buffer around the Geometry .......................................................25
ST_Centroid - Centroid of the Geometry .......................................................27
ST_Collect - Collect multiple point geometries and generate a multipoint geometry from it ......28
ST_Collect - Collect multiple point geometries from a table and generate a table of multipoint
geometries from it ..................................................................................29
ST_Contains - Checks Containment of Two Geometries ..............................................31
ST_ConvexHull - Convex Hull of a Geometry .............................................................32
ST_CoordDim - Coordinate Dimension .............................................................33
ST_Crosses - Checks if Geometries Cross ....................................................................34
ST_Difference - Difference of Two Geometries .......................................................35
ST_Dimension - Dimension of the geometry .............................................................36
ST_Disjoint - Checks if Geometries are Disjoint .......................................................37
ST_Distance - Distance between Geometries .......................................................38
ST_DWithin - Distance Within ...........................................................................39
ST_Ellipse - Ellipse Constructor ............................................................................41
ST_EndPoint - End Point of a Line ...........................................................................43
ST_Envelope - Bounding Box of a Geometry .......................................................44
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST_Equals</td>
<td>Checks if Geometries are Equal</td>
</tr>
<tr>
<td>ST_Expand</td>
<td>Expanded Bounding Rectangle of a Geometry</td>
</tr>
<tr>
<td>ST_ExteriorRing</td>
<td>Exterior Ring of a Polygon</td>
</tr>
<tr>
<td>ST_GeometryN</td>
<td>Nth Geometry from a Geometry Collection</td>
</tr>
<tr>
<td>ST_GeometryType</td>
<td>Type of a Geometry</td>
</tr>
<tr>
<td>ST_GeometryTypeID</td>
<td>Geometry Type of a Geometry</td>
</tr>
<tr>
<td>ST_GeomFromText</td>
<td>Geometry from WKT Representation</td>
</tr>
<tr>
<td>ST_GeomFromWKB</td>
<td>Geometry from WKB Representation</td>
</tr>
<tr>
<td>ST_GrandMBR</td>
<td>Bounding Box from a Set of Geometries</td>
</tr>
<tr>
<td>ST_InteriorRingN</td>
<td>Nth Interior Ring from the Polygon</td>
</tr>
<tr>
<td>ST_Intersection</td>
<td>Intersection of Geometries</td>
</tr>
<tr>
<td>ST_Intersects</td>
<td>Checks if Geometries Intersect</td>
</tr>
<tr>
<td>ST_Is3D</td>
<td>Checks if Geometry has Z Coordinate</td>
</tr>
<tr>
<td>ST_IsClosed</td>
<td>Checks if the Line is Closed</td>
</tr>
<tr>
<td>ST_IsEmpty</td>
<td>Checks if the Geometry is Empty</td>
</tr>
<tr>
<td>ST_IsMeasured</td>
<td>Checks if the Geometry has an M Coordinate</td>
</tr>
<tr>
<td>ST_IsRing</td>
<td>Checks if the Line is a Ring</td>
</tr>
<tr>
<td>ST_IsSimple</td>
<td>Checks if the Geometry is Simple</td>
</tr>
<tr>
<td>ST_Length</td>
<td>Length of the Line</td>
</tr>
<tr>
<td>ST_LineFromMultiPoint</td>
<td>Make a Linestring from a Multipoint geometry</td>
</tr>
<tr>
<td>ST_LocateAlong</td>
<td>Locate Along</td>
</tr>
<tr>
<td>ST_LocateBetween</td>
<td>Locate Between</td>
</tr>
<tr>
<td>ST_M</td>
<td>M Coordinate of a Point</td>
</tr>
<tr>
<td>ST_MaxM</td>
<td>Maximum M Coordinate of a Geometry</td>
</tr>
<tr>
<td>ST_MaxX</td>
<td>Maximum X Coordinate of a Geometry</td>
</tr>
<tr>
<td>ST_MaxY</td>
<td>Maximum Y Coordinate of a Geometry</td>
</tr>
<tr>
<td>ST_MaxZ</td>
<td>Maximum Z Coordinate of a Geometry</td>
</tr>
<tr>
<td>ST_MBR</td>
<td>Bounding Box of a Geometry</td>
</tr>
<tr>
<td>ST_MBRIntersects</td>
<td>Checks if MBRs of the Geometries Intersect</td>
</tr>
<tr>
<td>ST_MinM</td>
<td>Minimum M Coordinate of a Geometry</td>
</tr>
<tr>
<td>ST_MinX</td>
<td>Minimum X Coordinate of a Geometry</td>
</tr>
<tr>
<td>ST_MinY</td>
<td>Minimum Y Coordinate of a Geometry</td>
</tr>
<tr>
<td>ST_MinZ</td>
<td>Minimum Z Coordinate of a Geometry</td>
</tr>
<tr>
<td>ST_NumGeometries</td>
<td>Number of Geometries in a Collection</td>
</tr>
<tr>
<td>ST_NumInteriorRing</td>
<td>Number of Interior Rings</td>
</tr>
<tr>
<td>ST_NumPoints</td>
<td>Number of Vertices of the Geometry</td>
</tr>
<tr>
<td>ST_Overlaps</td>
<td>Checks if Geometries Overlap</td>
</tr>
<tr>
<td>ST_Perimeter</td>
<td>Perimeter of Geometry</td>
</tr>
<tr>
<td>ST_Point</td>
<td>Point Constructor</td>
</tr>
<tr>
<td>ST_PointN</td>
<td>Nth Point in Linestring</td>
</tr>
</tbody>
</table>
3 Reference Documentation: Utilities

ST_CreateGeomColumn - Create the Geometry Column Table ........................................ 111
ST_CreateSpatialRefSys - Create the Spatial Reference System Table ......................... 112

Notices and Trademarks

Notices ......................................................................................................................... 113
Trademarks .................................................................................................................. 114

Regulatory and Compliance ....................................................................................... 115
  Regulatory Notices ................................................................................................. 115
  Homologation Statement ......................................................................................... 115
  FCC - Industry Canada Statement ......................................................................... 115
  WEEE ....................................................................................................................... 115
  CE Statement (Europe) ............................................................................................. 115
  VCCI Statement ...................................................................................................... 116

Index
Preface

This guide describes the IBM Netezza Spatial Package.

Audience for This Guide

This guide is written for developers who intend to use the IBM Netezza Spatial Package with their IBM Netezza systems. This guide does not provide a tutorial on spatial concepts; for more information, see the Netezza Spatial Package User's Guide. Depending on your needs, you should be very familiar with spatial analysis and the OpenGIS standards. You should also be familiar with the basic operation of the IBM Netezza system.

Purpose of This Guide

This guide describes the IBM Netezza Spatial Package. The Package provides spatial analysis functions that can be used on the IBM Netezza database warehouse appliance.

Conventions

Note on Terminology: The terms User-Defined Analytic Process (UDAP) and Analytic Executable (AE) are synonymous.

The following conventions apply:

- *Italics* for emphasis on terms and user-defined values, such as user input.
- **Upper case** for SQL commands, for example, INSERT or DELETE.
- **Bold** for command line input, for example, `nzsystem stop`.
- **Bold** to denote parameter names, argument names, or other named references.
- Angle brackets (`< >`) to indicate a placeholder (variable) that should be replaced with actual text, for example, `nzmat <- nz.matrix("<matrix_name>")`.
- A single backslash (`\`) at the end of a line of code to denote a line continuation. Omit the backslash when using the code at the command line, in a SQL command, or in a file.
- When referencing a sequence of menu and submenu selections, the `"->"` character denotes the different menu options, for example *Menu Name > Submenu Name > Selection*.

If You Need Help

If you are having trouble using the IBM Netezza appliance, IBM Netezza Analytics or any of its components:

1. Retry the action, carefully following the instructions in the documentation.
3. If you have an active service contract maintenance agreement with IBM, you can contact customer support teams via telephone. For individual countries, please visit the Technical
Comments on the Documentation

We welcome any questions, comments, or suggestions that you have for the IBM Netezza documentation. Please send us an e-mail message at netezza-doc@wwpdl.vnet.ibm.com and include the following information:

► The name and version of the manual that you are using
► Any comments that you have about the manual
► Your name, address, and phone number

We appreciate your comments.
CHAPTER 1
List of functions by category

Spatial

ST_Area - Area of the Geometry
ST_AsBinary - Well-known Binary Representation of the Geometry
ST_AsKML - KML representation of a Geometry
ST_AsText - WKT representation of a Geometry
ST_Boundary - Boundary of the Geometry
ST_Buffer - Buffer around the Geometry
ST_Centroid - Centroid of the Geometry
ST_Collect - Collect multiple point geometries and generate a multipoint geometry from it
ST_Collect - Collect multiple point geometries from a table and generate a table of multipoint geometries from it
ST_Contains - Checks Containment of Two Geometries
ST_ConvexHull - Convex Hull of a Geometry
ST_CoordDim - Coordinate Dimension
ST_Crosses - Checks if Geometries Cross
ST_Difference - Difference of Two Geometries
ST_Dimension - Dimension of the geometry
ST_Disjoint - Checks if Geometries are Disjoint
ST_Distance - Distance between Geometries
ST_DWithin - Distance Within
ST_Ellipse - Ellipse Constructor
ST_EndPoint - End Point of a Line
ST_Envelope - Bounding Box of a Geometry
ST_Equals - Checks if Geometries are Equal
ST_Expand - Expanded Bounding Rectangle of a Geometry
ST_ExteriorRing - Exterior Ring of a Polygon
ST_GeometryN - Nth Geometry from a Geometry Collection
ST_GeometryType - Type of a Geometry
ST_GeometryTypeID - Geometry Type of a Geometry
ST_GeomFromText - Geometry from WKT Representation
ST_GeomFromWKB - Geometry from WKB Representation
ST_GrandMBR - Bounding Box from a Set of Geometries
ST_InteriorRingN - Nth Interior Ring from the Polygon
ST_Intersection - Intersection of Geometries
ST_Intersects - Checks if Geometries Intersect
ST_Is3D - Checks if Geometry has Z Coordinate
ST_IsClosed - Checks if the Line is Closed
ST_IsEmpty - Checks if the Geometry is Empty
ST_IsMeasured - Checks if the Geometry has an M Coordinate
ST_IsRing - Checks if the Line is a Ring
ST_IsSimple - Checks if the Geometry is Simple
ST_Length - Length of the Line
ST_LineFromMultiPoint - Make a Linestring from a Multipoint geometry
ST_LocateAlong - Locate Along
ST_LocateBetween - Locate Between
ST_M - M Coordinate of a Point
ST_MaxM - Maximum M Coordinate of a Geometry
ST_MaxX - Maximum X Coordinate of a Geometry
ST_MaxY - Maximum Y Coordinate of a Geometry
ST_MaxZ - Maximum Z Coordinate of a Geometry
ST_MBR - Bounding Box of a Geometry
ST_MBRIntersects - Checks if MBRs of the Geometries Intersect
ST_MinM - Minimum M Coordinate of a Geometry
ST_MinX - Minimum X Coordinate of a Geometry
ST_MinY - Minimum Y Coordinate of a Geometry
List of functions by category

**ST_MinZ** - Minimum Z Coordinate of a Geometry
**ST_NumGeometries** - Number of Geometries in a Collection
**ST_NumInteriorRing** - Number of Interior Rings
**ST_NumPoints** - Number of Vertices of the Geometry
**ST_Overlaps** - Checks if Geometries Overlap
**ST_Perimeter** - Perimeter of Geometry
**ST_Point** - Point Constructor
**ST_PointN** - Nth Point in Linestring
**ST_PointOnSurface** - Point on the Surface
**ST_Relate** - Relation of Geometries
**ST_SRID** - Setter/Getter of the SRID
**ST_StartPoint** - First Point of a Line
**ST_SymDifference** - Symmetric Difference of Geometries
**ST_Touches** - Checks if Geometries Touch
**ST_Union** - Union of Geometries
**ST_Version** - IBM Netezza Spatial Version
**ST_Within** - Checks if the Geometry is Within Another Geometry
**ST_WKBToSQL** - Geometry from WKB Representation
**ST_WKBToWKT** - WKT Representation from WKB Format
**ST_WKTToSQL** - Geometry from WKT Representation
**ST_WKTToWKB** - WKB Representation from WKT Format
**ST_X** - X Coordinate of a Point
**ST_Y** - Y Coordinate of a Point
**ST_Z** - Z Coordinate of a Point

**Utilities - Actions**

**ST_CreateGeomColumn** - Create the Geometry Column Table
**ST_CreateSpatialRefSys** - Create the Spatial Reference System Table
CHAPTER 2
Reference Documentation: Spatial

ST_Area - Area of the Geometry

Determines the area of the specified geometry object having a surface.

Usage

The ST_Area function has the following syntax:

```
ST_Area(VARCHAR(ANY) ST_Geometry, VARCHAR(ANY) unit, VARCHAR(ANY) cSystem);
```

▲ Parameters

► **ST_Geometry**
  The geometry object.
  Type: VARCHAR(ANY)

► **unit**
  The units. Possible values include "meter", "kilometer", "foot", "mile" and "nautical mile".
  Type: VARCHAR(ANY)
  Default: 'meter'

► **cSystem**
  The coordinate system.
  Type: VARCHAR(ANY)
  Default: From geometry's SRID or 'WGS84' if geometry has no SRID.

▲ Returns

DOUBLE The area of the specified geometry object.

Details

This function returns the area of the specified geometry object (in WKB format) having a surface.
Netezza Spatial Package Reference Guide

A polygon, multipolygon, geometry collection. SRID and units are supported. In the cartesian case, this function returns an accurate value. In the spherical case, the area is calculated by dividing the geometry into spherical triangles and calculating their areas using spherical excess / Huiller’s formula. It deals with geometries that cross the 180 meridian or contain the pole. In the spheroidal case, the geometry is projected into a cartesian coordinate system by Behrmann Equal Area projection and its area is calculated in the same way as the cartesian case.

Examples

```
SELECT inza..ST_Area(inza..ST_WKTToSQL('POLYGON((1 1, 1 2, 2 2, 2 1, 1 1))', 1234));

ST_AREA
---------
     1
(1 row)
```

```
SELECT inza..ST_Area(inza..ST_WKTToSQL('POLYGON((1 1, 1 2, 2 2, 2 1, 1 1))', 1111));

ST_AREA
----------------
12367196844.731
(1 row)
```

```
SELECT inza..ST_Area(inza..ST_WKTToSQL('POLYGON((1 1, 1 2, 2 2, 2 1, 1 1))', 4326));

ST_AREA
----------------
12304814950.073
(1 row)
```

```
SELECT inza..ST_Area(inza..ST_WKTToSQL('POLYGON((1 1, 1 2, 2 2, 2 1, 1 1))', 1234), 'foot');

ST_AREA
--------------
10.76391041671
(1 row)
```
SELECT inza..ST_Area(inza..ST_WKTToSQL('POLYGON((1 1, 1 2, 2 2, 2 1, 1 1))', 1234), 'meter', 'sphere');

ST_AREA
-----------------
12367196844.731
(1 row)

SELECT inza..ST_Area(inza..ST_WKTToSQL('POLYGON((1 1, 1 2, 2 2, 2 1, 1 1))'), 'meter', 1111);

ST_AREA
-----------------
12367196844.731
(1 row)

Related Functions
► category Spatial

**ST_AsBinary - Well-known Binary Representation of the Geometry**

Determines the Well-Known Binary (WKB) representation of a geometry object without the SRID.

**Usage**

The ST_AsBinary function has the following syntax:

► **ST_AsBinary(VARCHAR(ANY) ST_Geometry);**

▲ Parameters

► **ST_Geometry**

  The geometry object.
  Type: VARCHAR(ANY)

▲ Returns

  VARCHAR(ANY) The Well-Known Binary (WKB) representation of a geometry object without the SRID.

**Details**

Takes a geometry object and returns its well-known binary representation. ST_AsBinary is the reverse of ST_WKTBToSQL.
Examples

```
SELECT
  inza..ST_AsText(inza..ST_WKBToSQL(inza..ST_AsBinary(inza.
  .ST_Point(1, 2))));
```

```
ST_ASTEXT
-------------
POINT (1 2)
(1 row)
```

Related Functions

- category Spatial
- ST_AsText
- ST_GeomFromText
- ST_GeomFromWKB
- ST_WKBToSQL
- ST_WKTToSQL

ST_AsKML - KML representation of a Geometry

Returns the Keyhole Markup Language (KML) representation of a geometry object.

Usage

The ST_AsKML function has the following syntax:

```
ST_AsKML(VARCHAR(ANY) ST_Geometry); VARCHAR(ANY) = ST_AsKML(VARCHAR(ANY)
ST_Geometry, VARCHAR(ANY) additionalKMLAttributes);
```

- Parameters
  - **ST_Geometry**
    A geometry object.
    Type: VARCHAR(ANY)
  - **additionalKMLAttributes**
    (Optional) Select one of <extrude>, <tessellate> or <altitudeMode>.
    Type: VARCHAR(ANY)
    Default: ''

- Returns
  VARCHAR(ANY) The Keyhole Markup Language (KML) representation of a geometry object.
Details
The "additionalKMLAttributes" are not validated and are simply added to the output.

Examples

```sql
SELECT inza..ST_AsKML(inza..ST_Point(1, 5));
```

```
ST_ASKML
----------
<Point><coordinates>1,5</coordinates></Point>
```

(1 row)

```sql
SELECT inza..ST_AsKML(inza..ST_Point(1, 5),
  '<extrude>1</extrude><altitudeMode>clampToGround</altitudeMode>'
);
```

```
ST_ASKML
----------
<Point><extrude>1</extrude><altitudeMode>clampToGround</altitudeMode><coordinates>1,5</coordinates></Point>
```

(1 row)

Related Functions

► category Spatial

**ST_AsText - WKT representation of a Geometry**

Returns the Well-Known Text (WKT) representation of a geometry object.

Usage

The ST_AsText function has the following syntax:

► **ST_AsText(VARCHAR(ANY) Geometry);**
  ▲ Parameters
  ► **ST_Geometry**
    A geometry object.
    Type: VARCHAR(ANY)
  ▲ Returns
    VARCHAR(ANY) The Well-Known Text (WKT) representation of a geometry object.
Details
Does not return the SRID. ST_AsText is the reverse of ST_GeomFromText.

Examples

```sql
select inza..ST_AsText(inza..ST_Point(1.0, 5.0));
```

```
ST_ASTEXT
-------------
POINT (1 5)
(1 row)
```

```sql
select inza..ST_AsText(inza..ST_WKTToSQL('POLYGON((10 10, 10 20, 20 20, 20 15, 10 10))'));
```

```
ST_ASTEXT
-------------
POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))
(1 row)
```

Related Functions
► category Spatial
► ST_GeomFromText
► ST_GeomFromWKB
► ST_WKTToSQL
► ST_WKBToSQL
► ST_AsBinary

ST_Boundary - Boundary of the Geometry
Determines the boundary of a geometry object.

Usage
The ST_Boundary function has the following syntax:

```sql
ST_Boundary(VARCHAR(ANY) ST_Geometry);
```

▲ Parameters
► ST_Geometry
  A geometry object.
  Type: VARCHAR(ANY)
Returns
VARCHAR(ANY) A geometry object.

Details
ST_Boundary takes a geometry object and returns its combined boundary as a geometry object.

Examples

SELECT
inza..ST_AsText(inza..ST_Boundary(inza..ST_WKTToSQL('POLYGON ((1 1, 1 2, 2 2, 2 1, 1 1))')));

ST_ASTEXT
-------------
LINESTRING (1 1, 1 2, 2 2, 2 1, 1 1)
(1 row)

SELECT
inza..ST_AsText(inza..ST_Boundary(inza..ST_WKTToSQL('LINESTRING (0 0, 1 1, 2 2)')));

ST_ASTEXT
-------------
MULTIPOINT (0 0, 2 2)
(1 row)

Related Functions
► category Spatial

ST_Buffer - Buffer around the Geometry

Determines a buffer region around the specified geometry having the width specified by the distance parameter and a number of segments used to approximate a quarter of a circle.

Usage
The ST_Buffer function has the following syntax:

► ST_Buffer(VARCHAR(ANY) ST_Geometry, DOUBLE distance, INT nSegments, VARCHAR(ANY) unit, VARCHAR(ANY) cSystem);

▲ Parameters
► ST_Geometry
A geometry object.
Type: VARCHAR(ANY)

- **distance**
  The buffer distance.
  Type: DOUBLE

- **nSegments**
  The number of segments used to approximate a quarter of a circle.
  Type: INT
  Default: 8

- **unit**
  The units. Possible values include "meter", "kilometer", "foot", "mile" and "nautical mile".
  Type: VARCHAR(ANY)
  Default: 'meter'

- **cSystem**
  The coordinate system.
  Type: VARCHAR(ANY)
  Default: From geometry's SRID or 'WGS84' if geometry has no SRID.

Returns
VARCHAR(64000) A geometry object that is a buffer region around the specified geometry.

Details
This function supports only points using the spherical and WGS84 coordinate systems.

Examples

```sql
SELECT inza..ST_AsText(inza..ST_Buffer(inza..ST_Point(0, 0, 1234), 1, 2));
```

```
POLYGON ((1 0, 0.707106781186548 -0.707106781186547,
1.61554255216634e-15 -1, -0.707106781186546
-0.707106781186549, -1 -3.23108510433268e-15,
-0.70710678118655 0.707106781186545, -4.62458305157398e-15 1,
0.707106781186544 0.707106781186551, 1 0))
```

(1 row)

```sql
SELECT inza..ST_AsText(inza..ST_Buffer(inza..ST_Point(0, 0, 1234), 1, 2, 'foot'));
```

```
ST_ASTEXT
```

```sql
----------------
```

(1 row)
---

```
POLYGON ((0.3048 0, 0.21552614690566 -0.21552614690566, 4.924173699003e-16 -0.3048, -0.215526146905659 -0.21552614690566, -0.3048 -9.848347398006e-16, -0.215526146905661 0.215526146905659, -1.40957291411975e-15 0.3048, 0.215526146905658 0.215526146905661, 0.3048 0))
```

(1 row)

```
SELECT inza..ST_AsText(inza..ST_Buffer(inza..ST_Point(0, 0, 1234), 1, 2, 'meter', 'wgs84'));
```

---

```
```

(1 row)

Related Functions

- **category Spatial**

**ST_Centroid - Centroid of the Geometry**

Determines the geometric center of a geometry object.

**Usage**

The ST_Centroid function has the following syntax:

```
ST_Centroid(VARCHAR(ANY) ST_Geometry);
```

- **Parameters**
  - **ST_Geometry**
    - A geometry object.
    - Type: VARCHAR(ANY)
  - **Returns**
    - VARCHAR(100) A point geometry object.

**Details**

The geometric center of the geometry is the "average" of the points in the geometry. The result is not guar-
anteed to be on the geometry. If the geometry is empty, then an empty point is returned.

Examples

```
SELECT inza..ST_AsText(inza..ST_Centroid(inza..ST_WKTTToSQL('LINE STRING (0 0, 10 0)')));

ST_ASTEXT
-----------
POINT (5 0)
(1 row)
```

```
SELECT inza..ST_AsText(inza..ST_Centroid(inza..ST_WKTTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))')));

ST_ASTEXT
-----------
POINT (14.4444444444444 16.1111111111111)
(1 row)
```

Related Functions

► category Spatial

**ST_Collect - Collect multiple point geometries and generate a multipoint geometry from it**

Creates a multipoint geometry from a table of points

**Usage**

The ST_Collect aggregate has the following syntax:

```
ST_Collect(VARCHAR(ANY) geometry);
```

▶ Parameters

► geometry
A point geometry object.
Type: VARCHAR(ANY)

▶ Returns
VARCHARG(ANY) The multipoint geometry object.

Examples

    CREATE TABLE points (PointID integer, the_geom varchar(200));
    INSERT INTO points VALUES (1, inza.ST_WKTToSQL('Point (0 0)'));
    INSERT INTO points VALUES (2, inza.ST_WKTToSQL('Point (22 0)'));
    INSERT INTO points VALUES (3, inza.ST_WKTToSQL('Point (33 33)'));
    INSERT INTO points VALUES (4, inza.ST_WKTToSQL('Point (44 44)'));
    SELECT inza.ST_AsText(inza.ST_Collect(the_geom)) from (SELECT the_geom from points order by PointID LIMIT 9999999) points;
    DROP TABLE points;

    ST_ASTEXT
    "MULTIPOINT (0 0, 22 0, 33 33, 44 44)"
    (1 row)

Related Functions

► category Spatial

**ST_Collect** - Collect multiple point geometries from a table and generate a table of multipoint geometries from it

Creates a table of multipoint geometries from a table of points

Usage

The ST_Collect table function has the following syntax:

► **ST_Collect**(VARCHAR(ANY) Varchar);
  ▲ Parameters
      ► **VARCHAR**
        A point geometry object.
        Type: VARCHAR(ANY)
  ▲ Returns
        VARCHAR(ANY) The multipoint geometry object named "multipoint".
Examples

CREATE TABLE trip_points (trip_id integer, geom varchar(200), timestamp integer);

INSERT INTO trip_points VALUES (100, inza..ST_WKTToSQL('Point (100 100)'), 120212);
INSERT INTO trip_points VALUES (100, inza..ST_WKTToSQL('Point (200 200)'), 120312);
INSERT INTO trip_points VALUES (100, inza..ST_WKTToSQL('Point (300 300)'), 120412);
INSERT INTO trip_points VALUES (100, inza..ST_WKTToSQL('Point (400 400)'), 120512);
INSERT INTO trip_points VALUES (200, inza..ST_WKTToSQL('Point (200 200)'), 120212);
INSERT INTO trip_points VALUES (200, inza..ST_WKTToSQL('Point (300 300)'), 120312);
INSERT INTO trip_points VALUES (200, inza..ST_WKTToSQL('Point (100 100)'), 120412);
INSERT INTO trip_points VALUES (200, inza..ST_WKTToSQL('Point (400 400)'), 120512);
INSERT INTO trip_points VALUES (300, inza..ST_WKTToSQL('Point (400 400)'), 120212);
INSERT INTO trip_points VALUES (300, inza..ST_WKTToSQL('Point (300 300)'), 120312);
INSERT INTO trip_points VALUES (300, inza..ST_WKTToSQL('Point (200 200)'), 120412);
INSERT INTO trip_points VALUES (300, inza..ST_WKTToSQL('Point (100 100)'), 120512);

select trip_id, inza..st_astext(tf.multipoint) from (select inza..st_astext(geom), geom, trip_id, lag(0,1,1) over (partition by trip_id order by timestamp) as begin_part, lead(0,1,1) over(partition by trip_id order by timestamp) as end_part from trip_points) as foo, table with final(inza..st_collect(geom, begin_part, end_part)) tf order by trip_id;

DROP TABLE trip_points;

<table>
<thead>
<tr>
<th>TRIP_ID</th>
<th>ST_ASTEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>MULTIPoint (100 100, 200 200, 300 300, 400)</td>
</tr>
</tbody>
</table>
ST_Contains - Checks Containment of Two Geometries

Determines whether the first specified geometry contains the second geometry.

Usage

The ST_Contains function has the following syntax:

```
ST_Contains(VARCHAR(ANY) ST_Geometry1, ARCHAR(ANY) ST_Geometry2);
```

Parameters

- **ST_Geometry1**
  - A geometry object.
  - Type: VARCHAR(ANY)

- **ST_Geometry2**
  - A geometry object.
  - Type: VARCHAR(ANY)

Returns

BOOL TRUE if the first geometry contains the second geometry; otherwise FALSE.

Details

ST_Contains is the reverse of ST_Within. The current implementation performs calculations treating all coordinate systems as cartesian.

Examples

```
SELECT inza..ST_Contains(inza..ST_WKTToSQL('POLYGON ((0 0, 11 0, 11 11, 0 11, 0 0))'), inza..ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))'));
```

```
ST_CONTAINS
-------------
1
(1 row)
```
```
SELECT inza..ST_Contains(inza..ST_WKTToSQL('POLYGON ((0 0, 110 0, 110 110, 0 110, 0 0))'),
inza..ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))'));
```

---

**ST_CONTAINS**

```
t
(1 row)
```

**Related Functions**

- **category Spatial**
- **ST_Relate**
- **ST_Within**

---

**ST_ConvexHull - Convex Hull of a Geometry**

Determines the smallest convex geometry that contains all of the points of the specified geometry object.

**Usage**

The ST_ConvexHull function has the following syntax:

```
ST_ConvexHull(VARCHAR(ANY) ST_Geometry);
```

**Parameters**

- **ST_Geometry**
  A geometry object.

**Returns**

VARCHAR(ANY) A geometry object.

**Examples**

```
SELECT
inza..ST_AsText(inza..ST_ConvexHull(inza..ST_WKTToSQL('POINT (0 0)')));
```

```
SELECT
inza..ST_AsText(inza..ST_ConvexHull(inza..ST_WKTToSQL('LINESTRING (0 0, 10 10, 10 20, 20 20, 20 15, 10 10)')));

ST_ASTEXT
-----------
POLYGON ((0 0, 10 10, 20 10, 0 0))
(1 row)

SELECT
inza..ST_AsText(inza..ST_ConvexHull(inza..ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10)')));

ST_ASTEXT
-----------
POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))
(1 row)

Related Functions
► category Spatial

**ST_CoordDim - Coordinate Dimension**

Determines the coordinate dimension the geometry object.

**Usage**

The ST_CoordDim function has the following syntax:

```
ST_CoordDim(VARCHAR(ANY) ST_Geometry);
```

**Parameters**

- **ST_Geometry**
  A geometry object.
  Type: VARCHAR(ANY)

**Returns**

INT If the geometry only has x and y coordinates, then 2 is returned. If the geometry additionally has z or m, then 3 is returned. If the geometry has x, y, z and m coordinates, then 4 is returned. If the geometry is empty, 2 is returned.

**Examples**

```
SELECT inza..ST_CoordDim(inza..ST_WKTToSQL('POINT (0 0)'));
```
ST_CoordDim(inza..ST_WKTToSQL('POINT (0 0 0)'))

Related Functions
- category Spatial

ST_Crosses - Checks if Geometries Cross

Determines if the first specified geometry crosses the second geometry.

Usage
The ST_Crosses function has the following syntax:

```
ST_Crosses(VARCHAR(ANY) ST_Geometry1, VARCHAR(ANY) ST_Geometry2);
```

Parameters
- **ST_Geometry1**
  A geometry object.
  Type: VARCHAR(ANY)
- **ST_Geometry2**
  A geometry object.
  Type: VARCHAR(ANY)

Returns
BOOL Returns TRUE if the first geometry crosses the second geometry; otherwise FALSE.

Details
Returns TRUE if the DE-9IM intersection matrix for the two geometries is T*T****** or, for two lines, 0********. The current implementation performs calculations treating all coordinate systems as cartesian.
Examples

```sql
SELECT inza..ST_Crosses(inza..ST_WKTToSQL('LINESTRING (0 0, 10 10)'), inza..ST_WKTToSQL('LINESTRING (0 5, 10 5)'));

ST_CROSSES
-----------
\n```

(1 row)

Related Functions

- category Spatial
- ST_Relate

### ST_Difference - Difference of Two Geometries

Determines which points in the first specified geometry are not in the second geometry.

#### Usage

The ST_Difference function has the following syntax:

```sql
ST_Difference(VARCHAR(ANY) ST_Geometry1, VARCHAR(ANY) ST_Geometry2);
```

- Parameters
  - **ST_Geometry1**
    - A geometry object.
    - Type: VARCHAR(ANY)
  - **ST_Geometry2**
    - A geometry object.
    - Type: VARCHAR(ANY)

- Returns
  - VARCHAR(ANY) Returns a geometry object representing the points in the first geometry that are not in the second geometry.

#### Examples

```sql
SELECT inza..ST_AsText(inza..ST_Difference(inza..ST_WKTToSQL('POLYGON ((0 0, 11 0, 11 11, 0 11, 0 0))'), inza..ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))')));

ST_ASTEXT
-----------
```

POLYGON ((11 10.5, 11 0, 0 0, 0 11, 10 11, 10 10, 11 10.5))
Related Functions

- category Spatial

ST_Dimension - Dimension of the geometry

Determines the dimension of a geometry object. Note that the returned dimension is not the coordinate dimension.

Usage

The ST_Dimension function has the following syntax:

```sql
ST_Dimension(VARCHAR(ANY) ST_Geometry);
```

- **Parameters**
  - **ST_Geometry**
    - A geometry object.
    - Type: VARCHAR(ANY)

- **Returns**
  - INT Returns 0 for point, 1 for lines, 2 for polygons, or the highest dimension of the contents of a geometry collection; returns -1 for an empty geometry collection.

Examples

```sql
SELECT inza..ST_Dimension(inza..ST_WKToSQL('POINT (0 0)'));
```

```
ST_DIMENSION
--------------
0
```

(1 row)

```sql
SELECT inza..ST_Dimension(inza..ST_WKToSQL('LINESTRING (0 0, 1 1)'));
```

```
ST_DIMENSION
--------------
1
```

(1 row)
SELECT inza..ST_Dimension(inza..ST_WKTToSQL('POLYGON ((1 1, 1 2, 2 2, 2 1, 1 1))'));

ST_DIMENSION
----------------------
2
(1 row)

Related Functions
► category Spatial
► ST_CoordDim

ST_Disjoint - Checks if Geometries are Disjoint

Determines if the two specified geometries intersect.

Usage

The ST_Disjoint function has the following syntax:

► ST_Disjoint(VARCHAR(ANY) ST_Geometry1, VARCHAR(ANY) ST_Geometry2);

▲ Parameters

► ST_Geometry1
A geometry object.
Type: VARCHAR(ANY)

► ST_Geometry2
A geometry object.
Type: VARCHAR(ANY)

▲ Returns
BOOL TRUE if the two geometries do not intersect; otherwise FALSE.

Details

Returns TRUE if the DE-9IM intersection matrix for the two geometries is FF*FF****. The current implementation performs calculations treating all coordinate systems as cartesian.

Examples

SELECT inza..ST_Disjoint(inza..ST_WKTToSQL('LINESTRING (0 0, 10 10)'), inza..ST_WKTToSQL('LINESTRING (0 5, 10 5)'));

ST_DISJOINT
-------------
Related Functions

- category Spatial
- ST_Intersects
- ST_Relate

**ST_Distance - Distance between Geometries**

Determines the minimum distance between two points or segments.

**Usage**

The ST_Distance function has the following syntax:

```sql
ST_Distance(VARCHAR(ANY) ST_Geometry1, VARCHAR(ANY) ST_Geometry2, VARCHAR(ANY) unit, VARCHAR(ANY) cSystem, BOOL intersectTest);
```

**Parameters**

- **ST_Geometry1**
  A geometry object.
  Type: VARCHAR(ANY)

- **ST_Geometry2**
  A geometry object.
  Type: VARCHAR(ANY)

- **unit**
  The units. Possible values include "meter", "kilometer", "foot", "mile" and "nautical mile".
  Type: VARCHAR(ANY)
  Default: 'meter'

- **cSystem**
  The coordinate system.
  Type: VARCHAR(ANY)
  Default: From geometry's SRID or 'WGS84' if geometry has no SRID.

- **intersectTest**
  (Optional) If enabled, tests for intersection between the geometries and returns 0 if they do. Otherwise, calculate distance between the geometries.
  Type: BOOL
  Default: TRUE
Returns

DOUBLE Returns distance as the minimum distance between two points or segments describing the two ST_Geometry values.

Details

This function returns the minimum distance from any points or segments describing one geometry, ST_Geometry1, to any points or segments describing a second geometry, ST_Geometry2. SRID support.

Examples

```sql
SELECT inza..ST_Distance(inza..ST_Point(0, 0, 1234),
inza..ST_Point(1, 0, 1234));

ST_Distance
-------------
1
(1 row)
```

```sql
SELECT inza..ST_Distance(inza..ST_Point(0, 0, 1234),
inza..ST_Point(1, 0, 1234), 'foot');

ST_Distance
-------------
3.2808398950131
(1 row)
```

```sql
SELECT inza..ST_Distance(inza..ST_Point(0, 0, 1234),
inza..ST_Point(1, 0, 1234), 'meter', 'wgs84');

ST_Distance
-------------
111319.49079323
(1 row)
```

Related Functions

▶ category Spatial

**ST_DWithin - Distance Within**

Determine whether two geometries are within the specified distance of one another.
Usage

The ST_DWithin function has the following syntax:

```
ST_DWithin(VARCHAR(ANY) ST_Geometry1, VARCHAR(ANY) ST_Geometry2, DOUBLE distance, VARCHAR(ANY) unit, VARCHAR(ANY) cSystem);
```

▲ Parameters

- **ST_Geometry1**
  - A geometry object.
  - Type: VARCHAR(ANY)

- **ST_Geometry2**
  - A geometry object.
  - Type: VARCHAR(ANY)

- **distance**
  - The distance within which the geometries must be.
  - Type: DOUBLE

- **unit**
  - The units. Possible values include "meter", "kilometer", "foot", "mile" and "nautical mile".
  - Type: VARCHAR(ANY)
  - Default: 'meter'

- **cSystem**
  - The coordinate system.
  - Type: VARCHAR(ANY)
  - Default: From geometry's SRID or 'WGS84' if geometry has no SRID.

▲ Returns

BOOL Returns TRUE if the geometries are within the specified distance of one another; otherwise FALSE.

Examples

```
SELECT inza..ST_DWithin(inza..ST_Point(0,0),
inza..ST_Point(1,1), 2, 'meter','cartesian');
```

```
<table>
<thead>
<tr>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 row</td>
</tr>
</tbody>
</table>
```

```
SELECT inza..ST_DWithin(inza..ST_Point(0,0),
inza..ST_Point(1,1), 2, 'meter','cartesian');
```

```
<table>
<thead>
<tr>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 row</td>
</tr>
</tbody>
</table>
```

```
SELECT inza..ST_DWithin(inza..ST_Point(0,0),
inza..ST_Point(1,1), 2, 'meter','cartesian');
```

```
<table>
<thead>
<tr>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 row</td>
</tr>
</tbody>
</table>
```
inza..ST_Point(1,1), 2, 'foot', 'cartesian');

ST_DWITHIN
------------
f
(1 row)

SELECT inza..ST_DWithin(inza..ST_Point(0,0),
inza..ST_Point(1,1), 2, 'meter', 'wgs84');

ST_DWITHIN
------------
f
(1 row)

Related Functions
► category Spatial

**ST_Ellipse - Ellipse Constructor**

Specifies an ellipse with the specified center, axes, and tilt.

**Usage**

The ST_Ellipse function has the following syntax:

- **ST_Ellipse**(DOUBLE x0, DOUBLE y0, DOUBLE a, DOUBLE b, DOUBLE tilt, INT nSegment, VARCHAR(ANY) unit, VARCHAR(ANY) cSystems); VARCHAR(64000) = ST_Ellipse(VARCHAR(ANY) ST_Geometry, DOUBLE a, DOUBLE b, DOUBLE tilt, INT nSegments, VARCHAR(ANY) unit, VARCHAR(ANY) cSystem);

▲ Parameters

► **x0**
  The longitude or the x-coordinate of the center.
  Type: DOUBLE

► **y0**
  The latitude or the y-coordinate of the center.
  Type: DOUBLE

► **ST_Geometry**
  A geometry object.
  Type: VARCHAR(ANY)

► **a**
The semi-major axis.
Type: DOUBLE

► \( b \)
The semi-minor axis.
Type: DOUBLE

► \( \text{tilt} \)
The major axis tilt.
Type: DOUBLE

► \( \text{nSegments} \)
The number of segments used to approximate a quarter of a circle.
Type: INT
Default: 8

► \( \text{unit} \)
The units. Possible values include "meter", "kilometer", "foot", "mile" and "nautical mile".
Type: VARCHAR(ANY)
Default: 'meter'

► \( \text{cSystem} \)
The coordinate system.
Type: VARCHAR(ANY)
Default: 'WGS84'

Returns
VARCHAR(64000) Returns an ellipse (polygon) with the specified center, axes, and tilt.

Examples

```sql
SELECT inza..ST_AsText(inza..ST_Ellipse(1.0, 2.0, 100.0, 50.0, 30.0, 2, 'meter', 'cartesian'));
```

```
ST_ASTEXT
-----------
POLYGON ((51 88.6025403784439, 66.9739608441171 45.5595740399158, 44.301271892219 -23, -3.73671727453765 -76.9149130992431, -49 -84.6025403784439, -64.9739608441171 -41.5595740399158, -42.301271892219 27, 5.73671727453765 80.9149130992431, 51 88.6025403784439))
(1 row)
```
SELECT inza..ST_AsText(inza..ST_Ellipse(inza..st_wkttosql('point (1.0 2.0)'), 100.0, 50.0, 30.0, 2, 'meter', 'cartesian'));

ST_ASTEXT
-----------

POLYGON ((51 88.6025403784439, 66.9739608441171 45.5595740399158, 44.3012701892219 -23, -3.73671727453765 -76.9149130992431, -49 -84.6025403784439, -64.9739608441171 -41.5595740399158, -42.3012701892219 27, 5.73671727453765 80.9149130992431, 51 88.6025403784439))

(1 row)

Related Functions

► category Spatial

**ST_EndPoint - End Point of a Line**

Determines the last point of a line.

**Usage**

The ST_EndPoint function has the following syntax:

► **ST_EndPoint(VARCHAR(ANY) ST_Geometry);**

▲ **Parameters**

► **ST_Geometry**

A geometry object, which must be a line.

Type: VARCHAR(ANY)

▲ **Returns**

VARCHAR(100) The point geometry object; NULL if no end point.

**Examples**

SELECT inza..ST_AsText(inza..ST_EndPoint(inza..ST_WKTTosql('LINESTRING (0 0, 1 1, 1 2)'))));

ST_ASTEXT
-----------

POINT (1 2)

(1 row)
Related Functions

► category Spatial

ST_Envelope - Bounding Box of a Geometry
Determines the bounding box of a geometry object.

Usage
The ST_Envelope function has the following syntax:

► ST_Envelope(VARCHAR(ANY) ST_Geometry);

▲ Parameters
► ST_Geometry
A geometry object.
Type: VARCHAR(ANY)

▲ Returns
VARCHAR(200) The bounding box, or envelope, as a geometry object.

Details
This function always returns a rectangle as a polygon. ST_Envelope is exactly the same as ST_MBR.

Examples

SELECT
    inza..ST_AsText(inza..ST_Envelope(inza..ST_WKToSQL('LINE STRING (0 0, 1 -1, 2 2)')));

ST_ASTEXT
-----------
POLYGON ((0 -1, 0 2, 2 2, 2 -1, 0 -1))
(1 row)

Related Functions

► category Spatial
► ST_MBR

ST_Equals - Checks if Geometries are Equal
Determines is two geometries are equal.
Usage
The ST_Equals function has the following syntax:

```
ST_Equals(VARCHAR(ANY) ST_Geometry1, VARCHAR(ANY) ST_Geometry2);
```

▲ Parameters

► **ST_Geometry1**
A geometry object.
Type: VARCHAR(ANY)

► **ST_Geometry2**
A geometry object.
Type: VARCHAR(ANY)

▲ Returns
BOOL TRUE if the two geometries are equal; otherwise FALSE.

Details
Returns TRUE if the DE-9IM intersection matrix for the two geometries is T**F**FFF**F**.

Examples
```
SELECT inza..ST_Equals(inza..ST_WKTToSQL('LINESTRING (0 0, 10 10)'), inza..ST_WKTToSQL('LINESTRING (0 0, 10 10)'));
```

```
ST_EQUALS
 -----------
t
(1 row)
```

Related Functions
► category Spatial
► ST_Relate

ST_Expand - Expanded Bounding Rectangle of a Geometry

Determines the minimum bounding rectangle of a geometry object expanded by the distance parameter.

Usage
The ST_Expand function has the following syntax:

```
ST_Expand(VARCHAR(ANY) ST_Geometry, DOUBLE distance, VARCHAR(ANY) unit, VARCHAR(ANY) cSystem);
```

▲ Parameters

► **ST_Geometry**
A geometry object.
Type: VARCHAR(ANY)

► distance
The distance to expand the bounding box around the input geometry's MBR.
Type: DOUBLE

► unit
The units. Possible values include "meter", "kilometer", "foot", "mile" and "nautical mile".
Type: VARCHAR(ANY)
Default: 'meter'

► cSystem
The coordinate system.
Type: VARCHAR(ANY)
Default: From geometry's SRID or 'WGS84' if the geometry has no SRID.

▲ Returns
VARCHAR(200) A polygon that is the new MBR of the input geometry's MBR, expanded by the specified distance value.

Examples

```
SELECT
inza..ST_AsText(inza..ST_Expand(inza..ST_WKTTtoSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))', 1234), 100));
```

```
ST_ASTEXT
------------------
POLYGON ((-90 -90, -90 120, 120 120, 120 -90, -90 -90))
```

(1 row)

```
SELECT
inza..ST_AsText(inza..ST_Expand(inza..ST_WKTTtoSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))', 1234), 100,
'foot'));
```

```
ST_ASTEXT
------------------
POLYGON ((-20.48 -20.48, -20.48 50.48, 50.48 50.48, 50.48 -20.48, -20.48 -20.48))
```

(1 row)
SELECT
inza..ST_AsText(inza..ST_Expand(inza..ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))', 1234), 100, 'meter', 'wgs84'));

ST_ASTEXT

---------------


(1 row)

Related Functions

► category Spatial
► ST_MBR

ST_ExteriorRing - Exterior Ring of a Polygon

Determines the exterior ring from the specified polygon object.

Usage

The ST_ExteriorRing function has the following syntax:

► ST_ExteriorRing(VARCHAR(ANY) ST_Geometry);

▲ Parameters

► ST_Geometry
A geometry object.
Type: VARCHAR(ANY)

▲ Returns
VARCHAR(64000) A line geometry object; NULL if the polygon is empty.

Examples

SELECT
inza..ST_AsText(inza..ST_ExteriorRing(inza..ST_WKTToSQL('POLYGON ((0 0, 100 0, 100 100, 0 100, 0 0), (10 10, 10 20, 20 20, 20 15, 10 10))')));

ST_ASTEXT

------------

LINESTRING (0 0, 100 0, 100 100, 0 100, 0 0)

(1 row)
Related Functions

- category Spatial
- ST_InteriorRingN

**ST_GeometryN - Nth Geometry from a Geometry Collection**

Determines the Nth geometry from a geometry collection.

**Usage**

The ST_GeometryN function has the following syntax:

```
ST_GeometryN(VARCHAR(ANY) ST_Geometry, INT n);
```

- **Parameters**
  - **ST_Geometry**
    A geometry object.
    Type: VARCHAR(ANY)
  - **n**
    A 1-based index.
    Type: INT

- **Returns**
  VARCHAR(ANY) A geometry object; NULL if the specified geometry is not a collection of geometries.

**Examples**

```
SELECT inza..ST_AsText(inza..ST_GeometryN(inza..ST_WKToSQL('GEO
MERTYCOLLECTION (POLYGON ((10 10, 10 20, 20 20, 20 15, 10
10)), POINT (5 6))'), 2));
```

```
ST_ASTEXT

------------

POINT (5 6)

(1 row)
```

**Related Functions**

- category Spatial
**ST_GeometryType - Type of a Geometry**

Determines the geometry type of the geometry object.

**Usage**

The `ST_GeometryType` function has the following syntax:

```sql
ST_GeometryType(VARCHAR(ANY) ST_Geometry);
```

**Parameters**

- **ST_Geometry**
  A geometry object.
  Type: VARCHAR(ANY)

**Returns**

VARCHAR(50) The geometry type of the geometry object as a string.

**Examples**

```sql
SELECT inza..ST_GeometryType(inza..ST_WKTToSQL('POINT (0 0)'));

ST_GeOMETRYTYPE
-----------------
ST_POINT
(1 row)

SELECT inza..ST_GeometryType(inza..ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))'));

ST_GEOMETRYTYPE
-----------------
ST_POLYGON
(1 row)
```

**Related Functions**

- **category Spatial**
- **ST_GeometryTypeID**

**ST_GeometryTypeID - Geometry Type of a Geometry**

Determines the geometry type of the geometry object according to the OGC standard.

**Usage**

The `ST_GeometryTypeID` function has the following syntax:
**ST_GeometryTypeID(VARCHAR(ANY) ST_Geometry)**

▲ Parameters

▶ **ST_Geometry**

A geometry object.

Type: VARCHAR(ANY)

▲ Returns

INT The geometry type of the geometry object as a number.

**Examples**

```
SELECT inza..ST_GeometryTypeID(inza..ST_WKTTToSQL('POINT (0 0)'));

<table>
<thead>
<tr>
<th>ST_GEOMETRYTYPEID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

(1 row)
```

```
SELECT inza..ST_GeometryTypeID(inza..ST_WKTTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))'));

<table>
<thead>
<tr>
<th>ST_GEOMETRYTYPEID</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

(1 row)
```

**Related Functions**

▶ category Spatial

▶ ST_GeometryType

---

**ST_GeomFromText - Geometry from WKT Representation**

Determines a geometry object from the Well-Known Text (WKT) representation.

**Usage**

The ST_GeomFromText function has the following syntax:

▶ **ST_GeomFromText(VARCHAR(ANY) WKTString); VARCHAR(ANY) =**
**Parameters**

- **ST_Geometry**
  A geometry object.
  Type: VARCHAR(ANY)

- **SRID**
  The Spatial Reference System Identifier.
  Type: INT4
  Default: 4326

- **computeMBRFlag**
  (Optional) Indicates whether the MBR should be computed.
  Type: BOOL
  Default: TRUE

**Returns**

VARCHAR(ANY) A geometry object.

**Details**

`ST_GeomFromWKT` is exactly the same as `ST_WKTToSQL`. `ST_WKTToSQL` is the reverse of `ST_AsText`.

**Examples**

```sql
SELECT inza..ST_GeomFromText('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))');

ST_GEOMFROMTEXT
-------------
g
(1 row)

SELECT inza..ST_GeomFromText('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))', 4326);

ST_GEOMFROMTEXT
-------------
g
(1 row)

SELECT inza..ST_ASTEXT(inza..ST_GeomFromText('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))', 4326, true));

ST_ASTEXT
----------
POLYGON (((10 10, 10 20, 20 20, 20 15, 10 10)))
(1 row)
```
Related Functions

- category Spatial
- ST_WKTToSQL
- ST_GeomFromWKB
- ST_WKBToSQL
- ST_AsText
- ST_AsBinary

**ST_GeomFromWKB - Geometry from WKB Representation**

Determines a geometry object from the Well-Known Binary (WKB) representation.

**Usage**

The ST_GeomFromWKB function has the following syntax:

```sql
ST_GeomFromWKB(VARCHAR(ANY) WKB); VARCHAR(ANY) = ST_GeomFromWKB(VARCHAR(ANY) WKB, INT4 Srid); VARCHAR(ANY) = ST_GeomFromWKB(VARCHAR(ANY) WKB, INT4 Srid, BOOL ComputeMBR);
```

**Parameters**

- **ST_Geometry**
  A geometry object.
  Type: VARCHAR(ANY)

- **SRID**
  The Spatial Reference System Identifier.
  Type: INT4
  Default: 4326

- **computeMBRFlag**
  Indicates whether the MBR should be computed.
  Type: BOOL
  Default: TRUE

**Returns**

VARCHAR(ANY) A geometry object.
Details

ST_GeomFromWKB is exactly the same as ST_WKBToSQL. ST_WKBToSQL is the reverse of ST_AsBinary.

Examples

```sql
select
  inza..ST_AsText(inza..ST_GeomFromWKB(inza..ST_AsBinary(inza..ST_WKTToSQL('POLYGON ((0 0, 11 0, 11 11, 0 11, 0 0))'))));
```

```
ST_ASTEXT
---------------------------------------------------------------
---------------------------------------------------------------
---------------------------------------------------------------
----------------------
POLYGON ((0 0, 11 0, 11 11, 0 11, 0 0))
(1 row)
```

```sql
select
  inza..ST_AsText(inza..ST_GeomFromWKB(inza..ST_AsBinary(inza..ST_WKTToSQL('LINESTRING(0 0, 3 4, -1 1)'))));
```

```
ST_ASTEXT
---------------------------------------------------------------
---------------------------------------------------------------
-----
LINESTRING (0 0, 3 4, -1 1)
(1 row)
```

```sql
select
  inza..ST_AsText(inza..ST_GeomFromWKB(inza..ST_AsBinary(inza..ST_Point(1, 5))));
```

```
ST_ASTEXT
----------
POINT (1 5)
(1 row)
```

Related Functions

- ST_GeomFromText
- ST_WKBToSQL
ST_GrandMBR - Bounding Box from a Set of Geometries

Determines the bounding box from a set of geometry objects.

Usage

The ST_GrandMBR function has the following syntax:

```sql
ST_GrandMBR(VARCHAR(ANY) ST_Geometry);
```

Parameters

- **ST_Geometry**
  - A geometry object.
  - Type: VARCHAR(ANY)

Returns

VARCHAR(200) The bounding box as a geometry object.

Examples

```sql
CREATE TABLE polygons (PointID integer, the_geom varchar(200));

INSERT INTO polygons VALUES (1, inza..ST_WKTToSQL('Polygon ((0 0, 11 0, 11 11, 0 11, 0 0))'));

INSERT INTO polygons VALUES (2, inza..ST_WKTToSQL('Polygon ((0 0, 22 0, 22 22, 0 22, 0 0))'));

INSERT INTO polygons VALUES (3, inza..ST_WKTToSQL('Polygon ((0 0, 33 0, 33 33, 0 33, 0 0))'));

INSERT INTO polygons VALUES (4, inza..ST_WKTToSQL('Polygon ((0 0, 44 0, 44 44, 0 44, 0 0))'));

SELECT inza..ST_AsText(inza..ST_GrandMBR(the_geom)) from polygons;
DROP TABLE polygons;
```

ST_ASTEXT

-----------------------------------------------
Reference Documentation: Spatial

POLYGON ((0 0, 0 44, 44 44, 44 0, 0 0))
(1 row)

Related Functions
► category Spatial
► ST_MBR

**ST_InteriorRingN - Nth Interior Ring from the Polygon**

Determines the Nth interior ring from the specified polygon object.

**Usage**
The ST_InteriorRingN function has the following syntax:

- ST_InteriorRingN(VARCHAR(ANY) Geometry, INT4 N);

▲ Parameters
- **ST_Geometry**
  A geometry object.
  Type: VARCHAR(ANY)
- **N**
  A 1-based index.
  Type: INT4

▲ Returns
VARCHAR(ANY) A line geometry object; NULL if the Nth interior ring is not found.

**Examples**

```
SELECT inza..ST_AsText(inza..ST_InteriorRingN(inza..ST_WKTToSQL('POLYGON ((0 0, 100 0, 100 100, 0 100, 0 0), (10 10, 10 20, 20 20, 20 15, 10 10))'), 1));
```

```
ST_ASTEXT
---------
LINESTRING (10 10, 10 20, 20 20, 20 15, 10 10)
(1 row)
```

```
SELECT inza..ST_InteriorRingN(inza..ST_WKTToSQL('Polygon ((10 10, 10 20, 20 20, 20 15, 10 10))'), 5);
```
**ST_INTERIORRINGN**

------------------

(1 row)

Related Functions

► category Spatial
► ST_ExteriorRing

**ST_Intersection - Intersection of Geometries**

Determines a geometry object representing the points shared by the specified geometries.

**Usage**

The ST_Intersection function has the following syntax:

► **ST_Intersection(VARCHAR(ANY) Geometry1, VARCHAR(ANY) Geometry2);**

▲ Parameters

► **ST_Geometry1**
  
  A geometry object.
  
  Type: VARCHAR(ANY)

► **ST_Geometry2**
  
  A geometry object.
  
  Type: VARCHAR(ANY)

▲ Returns

  VARCHAR(ANY) The shared geometry object. The geometry object may be empty if no points are shared.

**Details**

The current implementation performs calculations treating all coordinate systems as cartesian.

**Examples**

```sql
SELECT
  inza..ST_AsText(inza..ST_Intersection(inza..ST_WKTToSQL('POLYGON ((0 0, 11 0, 11 11, 0 11, 0 0))'),
  inza..ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))')));
```

**ST_ASTEXT**

----------
SELECT
inza..ST_AsText(inza..ST_Intersection(inza..ST_WKTToSQL('POLYGON
((0 0, 0 5, 5 5, 5 0, 0 0))'), inza..ST_WKTToSQL('POLYGON
((10 10, 10 20, 20 20, 20 15, 10 10))')));

ST_ASTEXT
--------------
GEOMETRYCOLLECTION EMPTY

SELECT
inza..ST_AsText(inza..ST_Intersection(inza..ST_WKTToSQL('POLYGON
((0 0, 0 15, 15 15, 15 0, 0 0))'), inza..ST_WKTToSQL('POLYGON
((10 10, 10 20, 20 20, 20 15, 10 10))')));

ST_ASTEXT
--------------
POLYGON ((10 15, 15 15, 15 12.5, 10 10, 10 15))

(1 row)

Related Functions

► category Spatial
► ST_Intersects
► ST_SymDifference

ST_Intersects - Checks if Geometries Intersect

Determines whether two geometries intersect.

Usage

The ST_Intersects function has the following syntax:

► ST_Intersects(VARCHAR(ANY) Geometry1, VARCHAR(ANY) Geometry2);

▲ Parameters

► ST_Geometry1
A geometry object.
Type: VARCHAR(ANY)
ST_Geometry2
A geometry object.
Type: VARCHAR(ANY)

Returns
BOOL Returns TRUE if the two geometries intersect; otherwise FALSE.

Details
Returns TRUE if the DE-9IM intersection matrix for the two geometries is not FF*FF****. The current implementation performs calculations treating all coordinate systems as cartesian.

Examples

```sql
SELECT inza..ST_Intersects(inza..ST_WKTToSQL('POLYGON ((0 0, 11 0, 11 11, 0 11, 0 0))'), inza..ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))'));
```

```
ST_INTERSECTS
---------
t
(1 row)
```

```sql
SELECT inza..ST_Intersects(inza..ST_WKTToSQL('POLYGON ((0 0, 0 5, 5 5, 5 0, 0 0))'), inza..ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))'));
```

```
ST_INTERSECTS
---------
f
(1 row)
```

```sql
SELECT inza..ST_Intersects(inza..ST_WKTToSQL('POLYGON ((0 0, 0 4, 4 0, 0 0))'), inza..ST_WKTToSQL('POLYGON ((0 5, 5 5, 5 0, 0 5))'));
```

```
ST_INTERSECTS
---------
f
(1 row)
```
Related Functions

- category Spatial
- ST_MBRIntersects
- ST_Intersection

**ST_Is3D - Checks if Geometry has Z Coordinate**

Determines if the geometry has x, y, and z values.

**Usage**

The ST_Is3D function has the following syntax:

```
ST_Is3D(VARCHAR(ANY) Geometry);
```

**Parameters**

- **ST_Geometry**
  - A geometry object.
  - Type: VARCHAR(ANY)

**Returns**

- **BOOL** Returns TRUE if the geometry has x, y, and z values; otherwise FALSE.

**Examples**

```
SELECT inza..ST_Is3D(inza..ST_WKTToSQL('LINESTRING (0 0, 11 0, 11 11, 0 0)'));
```

```
ST_IS3D
--------
T
(1 row)
```

```
SELECT inza..ST_Is3D(inza..ST_WKTToSQL('LINESTRING (0 0 0, 11 0 5, 11 11 9, 0 0 0)'));
```

```
ST_IS3D
--------
T
(1 row)
```

```
SELECT inza..ST_Is3D(inza..ST_POINT(0.0, 1.0, 2.0, 3.0));
```

```
ST_IS3D
--------
F
(1 row)
```
Related Functions

- category Spatial
- ST_IsMeasured

**ST_IsClosed - Checks if the Line is Closed**

Determines if a line is closed.

**Usage**

The ST_IsClosed function has the following syntax:

```sql
ST_IsClosed(VARCHAR(ANY) Geometry);
```

**Parameters**

- **ST_Geometry**
  A geometry object, which must be a line.
  Type: VARCHAR(ANY)

**Returns**

BOOL Returns TRUE if the line is closed; otherwise FALSE.

**Details**

A line is closed if the start and end points are equal.

**Examples**

```sql
SELECT inza..ST_IsClosed(inza..ST_WKTToSQL('LINESTRING (0 0, 11 0, 11 11, 0 0)'));
```

```
ST_ISCLOSED
------------
1
```

```sql
SELECT inza..ST_IsClosed(inza..ST_WKTToSQL('LINESTRING (1 2 3 4, 8 8 8 8, 9 9 9 9, 1 2 3 4)'));
```

```
ST_ISCLOSED
------------
0
```
SELECT inza..ST_IsClosed(inza..ST_WKToSQL('LINESTRING (0 0, 11 0, 11 11)'));

\textbf{ST_ISCLOSED}

\begin{verbatim}
 f
\end{verbatim}

(1 row)

Related Functions

\begin{itemize}
\item \textbf{category} \text{Spatial}
\end{itemize}

\section*{ST_IsEmpty - Checks if the Geometry is Empty}

Determines whether a geometry is empty, that is, has no points.

\subsection*{Usage}

The \text{ST_IsEmpty} function has the following syntax:

\begin{verbatim}
ST_IsEmpty(VARCHAR(ANY) Geometry); 
\end{verbatim}

\begin{itemize}
\item ▲ Parameters
  \begin{itemize}
  \item \textbf{ST_Geometry} \\
      A geometry object.
  \end{itemize}
  \item Type: VARCHAR(ANY)
  \item Returns
      BOOL Returns TRUE if the geometry is empty; otherwise FALSE.
\end{itemize}

\subsection*{Examples}

\begin{verbatim}
SELECT inza..ST_IsEmpty(inza..ST_WKToSQL('POLYGON ((0 0, 11 0, 11 11, 0 11, 0 0))'));
\end{verbatim}

\textbf{ST_ISEMPTY}

\begin{verbatim}
 f
\end{verbatim}

(1 row)

\begin{verbatim}
SELECT inza..ST_IsEmpty(inza..ST_WKToSQL('POLYGON EMPTY')); 
\end{verbatim}
**ST_ISEMPTY**

-------------
t
(1 row)

```
SELECT inza..ST_IsEmpty(inza..ST_WKTToSQL('LINESTRING EMPTY'));
```

**ST_ISEMPTY**

-------------
t
(1 row)

**Related Functions**

► category Spatial

**ST_IsMeasured - Checks if the Geometry has an M Coordinate**

Determine whether the geometry has an m value.

**Usage**

The ST_IsMeasured function has the following syntax:

```
ST_IsMeasured(VARCHAR(ANY) Geometry);
```

▲ Parameters

► **ST_Geometry**

A geometry object.

Type: VARCHAR(ANY)

▲ Returns

BOOL Returns TRUE if the geometry has an m value; otherwise FALSE.

**Examples**

```
SELECT inza..ST_IsMeasured(inza..ST_WKTToSQL('LINESTRING (0 0, 11 0, 11 11, 0 0)'));
```

```
ST_ISMEASURED
-------------
f
```
SELECT inza..ST_IsMeasured(inza..ST_WKTToSQL('LINESTRING (0 0 0
0, 11 0 5 5, 11 11 9 8, 0 0 0 0))'));

    ST_ISMEASURED
---------------
    t
    (1 row)

SELECT inza..ST_IsMeasured(inza..ST_Point(1.0, 5.0, 8.0,
False));

    ST_ISMEASURED
---------------
    t
    (1 row)

Related Functions
► category Spatial
► ST_Is3D

ST_IsRing - Checks if the Line is a Ring
Determines whether a line is a ring.

Usage
The ST_IsRing function has the following syntax:

► ST_IsRing(VARCHAR(ANY) Geometry);
  ▲ Parameters
    ► ST_Geometry
      A geometry object, which must be a line.
      Type: VARCHAR(ANY)
  ▲ Returns
      BOOL Returns TRUE if the line is a ring; otherwise FALSE.

Details
A line is a ring if it is closed and simple.
Examples

```
SELECT inza..ST_IsRing(inza..ST_WKTToSQL('LINESTRING (0 0, 11 0, 11 11, 0 0)'));

ST_ISRING
------------

  t
  (1 row)
```

```
SELECT inza..ST_IsRing(inza..ST_WKTToSQL('LINESTRING (1 2, 3 4, 5 6)'));

ST_ISRING
------------

  f
  (1 row)
```

Related Functions

- category Spatial

**ST_IsSimple - Checks if the Geometry is Simple**

Determines if a geometry is simple.

**Usage**

The ST_IsSimple function has the following syntax:

```
ST_IsSimple(VARCHAR(ANY) Geometry);
```

**Parameters**

- **ST_Geometry**
  A geometry object.
  Type: VARCHAR(ANY)

**Returns**

- BOOL Returns TRUE if the geometry is simple; otherwise FALSE.

**Details**

One example of not being simple is a geometry that intersects with itself.
Examples

```
SELECT inza..ST_IsSimple(inza..ST_WKTTToSQL('POINT (1 1)'));

ST_ISSIMPLE
-------------
t
(1 row)
```

Related Functions

- category Spatial

**ST_Length - Length of the Line**

Determines the length of the linestring or multilinestring geometry.

**Usage**

The ST_Length function has the following syntax:

```
ST_Length(VARCHAR(ANY) Geometry); DOUBLE = ST_Length(VARCHAR(ANY) Geometry, VARCHAR(ANY) unit); DOUBLE = ST_Length(VARCHAR(ANY) Geometry, VARCHAR(ANY) unit, VARCHAR(ANY) cSystem);
```

- **Parameters**
  - **ST_GEOMETRY**
    - A geometry object, which must be a LineString or MultiLineString.
    - Type: VARCHAR(ANY)
  - **unit**
    - The units. Possible values include "meter", "kilometer", "foot", "mile" and "nautical mile".
    - Type: VARCHAR(ANY)
    - Default: 'meter'
  - **cSystem**
    - The coordinate system.
    - Type: VARCHAR(ANY)
    - Default: From geometry's SRID or 'WGS84' if geometry has no SRID.

- **Returns**
  - DOUBLE The length of the line.

**Details**

Function takes WKB (Well Known Binary) as an input. Geometric objects must be specified in terms of latitude/longitude on a spherical earth model.
Examples

```sql
SELECT inza..ST_Length(inza..ST_WKTToSQL('LINESTRING(0 0, 3 4, -1 1)'),'meter', 'cartesian');

ST_LENGTH
-----------
10
(1 row)
```

```sql
SELECT inza..ST_Length(inza..ST_WKTToSQL('MULTILINESTRING((0 0, 3 4, -1 1), (100 100, 400 500, 800 800))'),'meter', 'cartesian');

ST_LENGTH
-----------
1010
(1 row)
```

```sql
SELECT inza..ST_Length(inza..ST_WKTToSQL('MULTILINESTRING((0 0, 1 0), (0 1, 1 0))', 4326));

ST_LENGTH
-----------
268219.05906783
(1 row)
```

Related Functions

► category Spatial

**ST_LineFromMultiPoint - Make a Linestring from a Multipoint geometry**

Makes a Linestring from a Multipoint geometry

**Usage**

The ST_LineFromMultiPoint function has the following syntax:
ST_LineFromMultiPoint(VARCHAR(ANY) ST_Geometry);

Parameters
- **ST_Geometry**
  A Multipoint geometry object.
  Type: VARCHAR(ANY)

Returns
VARCHAR(ANY) A Linestring geometry object.

Details
ST_LineFromMultiPoint takes a Multipoint geometry object and returns a Linestring geometry object that connects all of the points.

Examples

```
SELECT inza..ST_AsText(inza..ST_LineFromMultiPoint(inza..ST_WKTToSQL('MULTIPOINT (1 1, 1 2, 2 2, 2 1, 1 1)', 4326, false, true)));
```

```
ST_ASTEXT
---------------------------------------------
LINESTRING (1 1, 1 2, 2 2, 2 1, 1 1)
```

(1 row)

```
SELECT inza..ST_AsText(inza..ST_LineFromMultiPoint(inza..ST_WKTToSQL('MULTIPOINT (0 0, 1 1, 2 2)')));
```

```
ST_ASTEXT
-----------
LINESTRING (0 0, 1 1, 2 2)
```

(1 row)

Related Functions
- category Spatial

**ST_LocateAlong - Locate Along**

Specifies a derived geometry collection value that matches the specified value of the m coordinate.

Usage
The ST_LocateAlong function has the following syntax:
ST_LocateAlong(VARCHAR(ANY) Geometry, DOUBLE m);

- **Parameters**
  - **ST_Geometry**
    - A geometry object.
    - Type: VARCHAR(ANY)
  - **m**
    - The start range of the measure to find.
    - Type: VARCHAR(ANY)

- **Returns**
  - VARCHAR(ANY) Returns a derived geometry collection value that matches the specified value of m coordinate; returns an empty GeometryCollection for geometries without m values.

**Details**
Points, MultiPoints, LineStrings and MultiLineStrings are supported. This function operates only for cartesian coordinates.

**Examples**

```sql
SELECT
  inza..ST_AsText(inza..ST_LocateAlong(inza..ST_WKTTToSQL('MULTIPOINT (0 0 0 4, 100 0 0 5, 100 100 5 6, 0 100 7 8, 1 0 9 5)'), 5));

ST_ASTEXT
---------------
MULTIPOINT ZM (100 0 0 5, 1 0 9 5)
(1 row)
```

**Related Functions**
- category Spatial
- ST_LocateBetween

**ST_LocateBetween - Locate Between**

Determines a derived geometry collection value that matches the specified range of m coordinate values inclusively. Points, MultiPoints, LineStrings and MultiLineStrings are supported. This function operates only for cartesian coordinates.

**Usage**

The ST_LocateBetween function has the following syntax:
**ST_LocateBetween**(VARCHAR(ANY) Geometry, DOUBLE m1, DOUBLE m2);

▲ Parameters

► **ST_Geometry**
   A geometry object.
   Type: VARCHAR(ANY)

► **m1**
   The start range of the measure to find.
   Type: VARCHAR(ANY)

► **m2**
   The end range of the measure to find.
   Type: VARCHAR(ANY)

▲ Returns

VARCHAR(ANY) Returns a derived geometry collection value that matches the specified range of m coordinate values inclusively; returns an empty GeometryCollection for geometries without m values.

**Examples**

```
SELECT
    inza..ST_AsText(inza..ST_LocateBetween(inza..ST_WKTToSQL('MULTIPOINT (0 0 0 4, 100 0 0 5, 100 100 5 6, 0 100 7 8, 1 0 9 5)'), 5, 8));
```

```
ST_ASTEXT
------------
MULTIPOINT ZM (100 0 0 5, 100 100 5 6, 0 100 7 8, 1 0 9 5)
(1 row)
```

**Related Functions**

► category Spatial
► **ST_LocateAlong**

---

**ST_M - M Coordinate of a Point**

Determines the m coordinate of a point object or sets the M coordinate of a point object to the specified M value.

**Usage**

The ST_M function has the following syntax:

```
ST_M(VARCHAR(ANY) Geometry); VARCHAR(100) = ST_M(VARCHAR(ANY) Geometry, DOUBLE M);
```
### Parameters

- **ST_Geometry**
  - A geometry object, which must be a point object.
  - Type: VARCHAR(ANY)

- **m**
  - (Optional) The value to set for m. If NULL, the m value is removed from the point.
  - Type: DOUBLE

### Returns

DOUBLE Or VARCHAR(100) The m coordinate or a geometry object with the m coordinate set to the value specified by m.

### Examples

```sql
SELECT inza..ST_M(inza..ST_Point(0.0, 1.0, 2.0, 3.0));
```

```
ST_M
-----
   3
(1 row)
```

```sql
SELECT inza..ST_M(inza..ST_Point(0.0, 1.0, 2.0, false));
```

```
ST_M
-----
   2
(1 row)
```

```sql
SELECT inza..ST_AsText(inza..ST_M(inza..ST_WKTTOSQL('POINT (0.0 1.0)'), 5.0));
```

```
ST_ASTEXT
----------
POINT M (0 1 5)
(1 row)
```

### Related Functions

- **category Spatial**
- **ST_X**
ST_MaxM - Maximum M Coordinate of a Geometry

Determines the maximum m coordinate of a geometry object.

Usage

The ST_MaxM function has the following syntax:

```
ST_MaxM(VARCHAR(ANY) Geometry);
```

▲ Parameters

- ST_Geometry
  A geometry object.
  Type: VARCHAR(ANY)

▲ Returns

- DOUBLE The maximum m coordinate; NULL if geometry is empty.

Examples

```
SELECT inza..ST_MaxM(inza..ST_WKTToSQL('POLYGON ((10 10 0 50, 10 20 1 60, 20 20 2 70, 20 15 3 80, 10 10 0 50))'));

ST_MAXM
---------
80
(1 row)
```

```
SELECT inza..ST_MaxM(inza..ST_WKTToSQL('LINESTRING EMPTY'));

ST_MAXM
---------
(1 row)
```

Related Functions

- category Spatial
- ST_MaxX
- ST_MaxY
- ST_MaxZ
- ST_MinX
- ST_MinY
ST_MaxX - Maximum X Coordinate of a Geometry

Determines the maximum x coordinate of a geometry object.

Usage

The ST_MaxX function has the following syntax:

**ST_MaxX(VARCHAR(ANY) Geometry);**

- **Parameters**
  - **ST_Geometry**
    - A geometry object.
    - Type: VARCHAR(ANY)

- **Returns**
  - DOUBLE Maximum x coordinate; NULL if the geometry is empty.

Examples

```sql
SELECT inza..ST_MaxX(inza..ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))'));

  ST_MAXX
  ------
   20
(1 row)
```

```sql
SELECT inza..ST_MaxX(inza..ST_WKTToSQL('GeometryCollection(POINT (10 10), POINT (30 30), LINESTRING (15 15, 20 20))'));

  ST_MAXX
  ------
   30
(1 row)
```

```sql
SELECT inza..ST_MaxX(inza..ST_WKTToSQL('LINESTRING EMPTY'));
```
**ST_MAXX**

---------

(1 row)

Related Functions
► category Spatial
► ST_MaxY
► ST_MaxZ
► ST_MaxM
► ST_MinX
► ST_MinY
► ST_MinZ
► ST_MinM
► ST_Envelope

**ST_MaxY - Maximum Y Coordinate of a Geometry**

Determines the maximum y coordinate of a geometry object.

Usage
The ST_MaxY function has the following syntax:

► **ST_MaxY(VARCHAR(ANY));**
  ▲ Parameters
  ► **ST_Geometry**
    A geometry object.
    Type: VARCHAR(ANY)
  ▲ Returns
    DOUBLE The maximum y coordinate; NULL if the geometry is empty.

Examples

```
SELECT inza..ST_MaxY(inza..ST_WKTToSQL('POLYGON ((10 10, 10 30, 20 20, 20 15, 10 10))'));
ST_MAXY
---------
30
(1 row)
```

```
SELECT inza..ST_MaxY(inza..ST_WKTToSQL('GeometryCollection(POINT (10 10), POINT (30 15), LINESTRING (15 15, 20 20))'));
```
ST_MAXY

---------

20

(1 row)

SELECT inza..ST_MaxY(inza..ST_WKToSQL('LINESTRING EMPTY'));

ST_MAXY

---------

(1 row)

**Related Functions**

- category Spatial
- ST_MaxX
- ST_MaxZ
- ST_MaxM
- ST_MinX
- ST_MinY
- ST_MinZ
- ST_MinM
- ST_Envelope

**ST_MaxZ - Maximum Z Coordinate of a Geometry**

Determines the maximum z coordinate of a geometry object.

**Usage**

The ST_MaxZ function has the following syntax:

```
ST_MaxZ(VARCHAR(ANY) Geometry);
```

▲ Parameters

- **ST_Geometry**
  - A geometry object.
  - Type: VARCHAR(ANY)

▲ Returns

  DOUBLE The maximum z coordinate; NULL if the geometry is empty
Examples

```sql
SELECT inza..ST_MaxZ(inza..ST_WKTToSQL('POLYGON ((10 10 0, 10 20 1, 20 20 2, 20 15 3, 10 10 0))'));

ST_MAXZ
---------
   3
(1 row)
```

```sql
SELECT inza..ST_MaxZ(inza..ST_WKTToSQL('POLYGON ((10 10 10, 10 20 20, 20 15 15, 10 10 10))'));

ST_MAXZ
---------
  20
(1 row)
```

```sql
SELECT inza..ST_MaxZ(inza..ST_WKTToSQL('LINESTRING EMPTY'));

ST_MAXZ
---------
   (1 row)
```

Related Functions

- category Spatial
- ST_MaxX
- ST_MaxY
- ST_MaxM
- ST_MinX
- ST_MinY
- ST_MinZ
- ST_MinM
- ST_Envelope

ST_MBR - Bounding Box of a Geometry

Determine the bounding box of a geometry object.

Usage

The ST_MBR function has the following syntax:
ST_MBR(VARCHAR(ANY) Geometry);

Parameters

ST_Geometry
A geometry object.

Type: VARCHAR(ANY)

Returns

VARCHAR(200) The bounding box as a geometry object.

Details

This function always returns a rectangle as a polygon. ST_MBR is exactly the same as ST_Envelope.

Examples

```
SELECT inza..ST_AsText(inza..ST_MBR(inza..ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))')));

ST_ASTEXT
--------

POLYGON ((10 10, 10 20, 20 20, 20 10, 10 10))

(1 row)
```

```
SELECT inza..ST_AsText(inza..ST_MBR(inza..ST_WKTToSQL('POINT (0 1)')));

ST_ASTEXT
--------

POLYGON ((0 1, 0 1, 0 1, 0 1, 0 1))

(1 row)
```

Related Functions

- category Spatial
- ST_Point
- ST_Envelope

ST_MBRIntersects - Checks if MBRs of the Geometries Intersect

Determines whether the minimum bounding rectangles of the two geometries intersect.
Usage
The ST_MBRIntersects function has the following syntax:

```sql
ST_MBRIntersects(VARCHAR(ANY) Geometry1, VARCHAR(ANY) Geometry2);
```

▲ Parameters
► **ST_Geometry1**
  A geometry object.
  Type: VARCHAR(ANY)
► **ST_Geometry2**
  A geometry object.
  Type: VARCHAR(ANY)

▲ Returns
 BOOL Returns TRUE if the minimum bounding rectangles of the two geometries intersect; otherwise FALSE.

Details
Returns TRUE if the DE-9IM intersection matrix for the minimum bounding rectangles of the two geometries is not FF*FF****.

Examples
```sql
SELECT inza..ST_MBRIntersects(inza..ST_WKTToSQL('POLYGON ((0 0, 11 0, 11 11, 0 11, 0 0))'), inza..ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))'));
```

```
ST_MBRINTERSECTS
------------------
t
(1 row)
```

Related Functions
► category Spatial
► ST_Intersects

**ST_MinM - Minimum M Coordinate of a Geometry**

Determines the minimum m coordinate of a geometry object.

Usage
The ST_MinM function has the following syntax:

```sql
ST_MinM(VARCHAR(ANY) Geometry);
```
▲ Parameters

► **ST_Geometry**
A geometry object.
Type: VARCHAR(ANY)

▲ Returns

DOUBLE The minimum m coordinate; NULL if the geometry is empty

Examples

```
SELECT inza..ST_MinM(inza..ST_WKToSQL('POLYGON ((10 10 0
50, 10 20 1 60, 20 20 2 70, 20 15 3 80, 10 10 0 50))'));

ST_MINM
--------
  50
(1 row)
```

```
SELECT inza..ST_MinM(inza..ST_WKToSQL('GeometryCollection(POINT
(10 10 10 10), POINT (30 30 30 30), LINESTRING (15 15 15
, 20 20 20 20))'));

ST_MINM
--------
  10
(1 row)
```

```
SELECT inza..ST_MinM(inza..ST_WKToSQL('LINESTRING EMPTY'));

ST_MINM
--------

(1 row)
```

Related Functions

► category Spatial
► ST_MaxX
► ST_MaxY
► ST_MaxZ
► ST_MaxM
ST_MinX - Minimum X Coordinate of a Geometry

Determines the minimum x coordinate of a geometry object.

Usage
The ST_MinX function has the following syntax:

```
ST_MinX(VARCHAR(ANY) Geometry);
```

- Parameters
  - **ST_Geometry**
    A geometry object.
    Type: VARCHAR(ANY)

- Returns
  DOUBLE Returns the minimum x coordinate of a geometry object.

Examples

```
SELECT inza..ST_MinX(inza..ST_WKTTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))'));

ST_MINX
-------
10
(1 row)
```

```
SELECT inza..ST_MinX(inza..ST_WKTTToSQL('GeometryCollection(POINT (10 5), POINT (30 15), LINESTRING (15 15, 20 20))'));

ST_MINX
-------
10
(1 row)
```

```
SELECT inza..ST_MinX(inza..ST_WKTTToSQL('LINESTRING EMPTY'));

ST_MINX
-------
```

Related Functions

- category Spatial
- ST_MaxX
- ST_MaxY
- ST_MaxZ
- ST_MaxM
- ST_MinY
- ST_MinZ
- ST_MinM
- ST_Envelope

ST_MinY - Minimum Y Coordinate of a Geometry

Determines the minimum y coordinate of a geometry object.

Usage

The ST_MinY function has the following syntax:

```
ST_MinY(VARCHAR(ANY) Geometry);
```

Parameters

- **ST_Geometry**
  A geometry object.
  Type: VARCHAR(ANY)

Returns

DOUBLE The minimum y coordinate of the geometry object; NULL if the geometry is empty.

Examples

```sql
SELECT inza..ST_MinY(inza..ST_WKToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))'));
```

```
ST_MINY
---------
 10
(1 row)
```

```sql
SELECT inza..ST_MinY(inza..ST_WKToSQL('GeometryCollection(POINT
```
ST_MINY
---------
5
(1 row)

SELECT inza..ST_MinY(inza..ST_WKTTToSQL('LINESTRING EMPTY'));

ST_MINY
---------

(1 row)

Related Functions
► category Spatial
► ST_MaxX
► ST_MaxY
► ST_MinX
► ST_MinY
► ST_MinZ
► ST_MaxZ
► ST_MaxM
► ST_MinM
► ST_Envelope

ST_MinZ - Minimum Z Coordinate of a Geometry

Determines the minimum z coordinate of a geometry object.

Usage
The ST_MinZ function has the following syntax:

► ST_MinZ(VARCHAR(ANY) Geometry);

Parameters
► ST_Geometry
A geometry object.
Type: VARCHAR(ANY)

Returns
DOUBLE The minimum z coordinate; NULL if the geometry is empty.

Examples
SELECT inza..ST_MinZ(inza..ST_WKTTToSQL('POLYGON ((10 10 0, 10 20')));

ST_MINY
---------
5
(1 row)
ST_MinZ
---------
0
(1 row)

SELECT inza..ST_MinZ(inza..ST_WKToSQL('POLYGON ((10 10 10 10, 10 20 20 20, 20 20 20 20, 20 15 15 40, 10 10 10 10))')));
ST_MinZ
---------
10
(1 row)

SELECT inza..ST_MinZ(inza..ST_WKToSQL('LINESTRING EMPTY')));
ST_MinZ
---------
(1 row)

Related Functions
► category Spatial
► ST_MaxX
► ST_MaxY
► ST_MaxZ
► ST_MaxM
► ST_MinY
► ST_MinZ
► ST_MinM
► ST_Envelope

ST_NumGeometries - Number of Geometries in a Collection

Determines the number of geometries in the geometry object.

Usage

The ST_NumGeometries function has the following syntax:
ST_NumGeometries(VARCHAR(ANY) Geometry);

Parameters

- ST_Geometry
  A geometry object.

  Type: VARCHAR(ANY)

Returns

INT4 The number of geometries in the geometry object; returns 1 for geometries that are not collections.

Examples

```sql
SELECT inza..ST_NumGeometries(inza..ST_WKTToSQL('GEOMETRYCOLLECTION (POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10)), POINT (5 6))'));
```

```
ST_NUMGEOMETRIES
------------------
2
(1 row)
```

```sql
SELECT inza..ST_NumGeometries(inza..ST_POINT(1, 5));
```

```
ST_NUMGEOMETRIES
------------------
1
(1 row)
```

Related Functions

- category Spatial

ST_NumInteriorRing - Number of Interior Rings

Determines the number of interior rings of the polygon.

Usage

The ST_NumInteriorRing function has the following syntax:

ST_NumInteriorRing(VARCHAR(ANY) Geometry);

Parameters

- ST_Geometry
  A geometry object, which must be a polygon.
Type: VARCHAR(ANY)

▲ Returns
INT4 Number of interior rings of the polygon.

Examples

SELECT
inza..ST_NumInteriorRing(inza..ST_WKTToSQL('POLYGON ((0 0, 100 0, 100 100, 0 100, 0 0), (10 10, 10 20, 20 20, 20 15, 10 10))'));

ST_NUMINTERIORRING
-------------------

1

(1 row)

SELECT
inza..ST_NumInteriorRing(inza..ST_WKTToSQL('POLYGON ((0 0, 100 0, 100 100, 0 100, 0 0))'));

ST_NUMINTERIORRING
-------------------

0

(1 row)

SELECT
inza..ST_NumInteriorRing(inza..ST_WKTToSQL('POLYGON EMPTY'));

ST_NUMINTERIORRING
-------------------

0

(1 row)

Related Functions

► category Spatial

**ST_NumPoints - Number of Vertices of the Geometry**

Determines the number of vertices of the geometry object.
Usage

The ST_NumPoints function has the following syntax:

► **ST_NumPoints(VARCHAR(ANY) Geometry);**

▲ **Parameters**

► **ST_Geometry**
  A geometry object.
  
  Type: VARCHAR(ANY)

▲ **Returns**

INT4 The number of vertices of the geometry object.

Examples

```
SELECT inza..ST_NumPoints(inza..ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))'));

ST_NUMPOINTS
--------------
   5
(1 row)
```

```
SELECT inza..ST_NumPoints(inza..ST_WKTToSQL('GeometryCollection(POINT (10 10), POINT (30 30), LINESTRING (15 15, 20 20))'));

ST_NUMPOINTS
--------------
   4
(1 row)
```

```
SELECT inza..ST_NumPoints(inza..ST_WKTToSQL('POLYGON ((0 0, 110 0, 110 110, 0 110, 0 0), (10 10, 10 20, 20 20, 20 15, 10 10))'));

ST_NUMPOINTS
--------------
   10
(1 row)
```

Related Functions

► category Spatial
**ST_Overlaps - Checks if Geometries Overlap**

Determines if two geometries overlap.

**Usage**

The `ST_Overlaps` function has the following syntax:

```sql
ST_Overlaps(VARCHAR(ANY) Geometry1, VARCHAR(ANY) Geometry2);
```

**Parameters**

- **ST_Geometry1**
  - A geometry object.
  - Type: VARCHAR(ANY)

- **ST_Geometry2**
  - A geometry object.
  - Type: VARCHAR(ANY)

**Returns**

BOOL Returns TRUE if the two geometries overlap; otherwise FALSE.

**Details**

Returns TRUE if the DE-9IM intersection matrix for the two geometries is T*T***T**, or, for two lines, 1*T***T**. Current implementation performs calculations treating all coordinate systems as cartesian.

**Examples**

```sql
SELECT inza..ST_Overlaps(inza..ST_WKTToSQL('LINESTRING (0 0, 10 10)'), inza..ST_WKTToSQL('LINESTRING (5 5, 11 11)'));
```

```
ST_OVERLAPS
-------------
t
(1 row)
```

**Related Functions**

- **category Spatial**
- **ST_Relate**
**ST_Perimeter - Perimeter of Geometry**

Computes the perimeter of the specified geometry.

**Usage**

The ST_Perimeter function has the following syntax:

- **ST_Perimeter(VARCHAR(ANY) Geometry); DOUBLE = ST_Perimeter(VARCHAR(ANY) Geometry, VARCHAR(ANY) unit); DOUBLE = ST_Perimeter(VARCHAR(ANY) Geometry, VARCHAR(ANY) unit, VARCHAR(ANY) cSystem);**

  ▲ **Parameters**

  ▶ **ST_Geometry**
  
  A geometry object.
  
  Type: VARCHAR(ANY)

  ▶ **unit**
  
  The units. Possible values include "meter", "kilometer", "foot", "mile" and "nautical mile".
  
  Type: VARCHAR(ANY)
  
  Default: 'meter'

  ▶ **cSystem**
  
  The coordinate system.
  
  Type: VARCHAR(ANY)
  
  Default: The geometry's SRID or 'WGS84' if geometry has no SRID.

  ▲ **Returns**

  DOUBLE The perimeter of the geometry object.

**Examples**

```
SELECT inza..ST_Perimeter(inza..ST_WKTToSQL('POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))', 1234));

ST_PERIMETER
-------------

 4

(1 row)
```

```
SELECT inza..ST_Perimeter(inza..ST_WKTToSQL('MULTIPOLYGON(((0 0, 0 1, 1 1, 1 0, 0 0)), ((0 0, 0 4, 5 4, 5 0, 0 0)))', 1234));

ST_PERIMETER
-------------

 22

(1 row)
```
Related Functions

- category Spatial

**ST_Point - Point Constructor**

Specifies a point geometry object with the specified coordinates.

**Usage**

The ST_Point function has the following syntax:

- `ST_Point(DOUBLE X, DOUBLE Y); VARCHAR(50) = ST_Point(DOUBLE X, DOUBLE Y, DOUBLE Z); VARCHAR(50) = ST_Point(DOUBLE X, DOUBLE Y, DOUBLE Z, DOUBLE M); VARCHAR(50) = ST_Point(DOUBLE X, DOUBLE Y, DOUBLE ZorM, BOOL isZ); VARCHAR(50) = ST_Point(DOUBLE X, DOUBLE Y, DOUBLE Z, INT4 SRID); VARCHAR(50) = ST_Point(DOUBLE X, DOUBLE Y, DOUBLE Z, DOUBLE M, INT4 SRID); VARCHAR(50) = ST_Point(DOUBLE X, DOUBLE Y, DOUBLE ZorM, BOOL isZ, INT4 SRID);

- **Parameters**
  - x
    - The x coordinate.
    - Type: DOUBLE
  - y
    - The y coordinate.
    - Type: DOUBLE
  - z
    - (Optional) The z coordinate.
    - Type: DOUBLE
  - m
    - (Optional) The m coordinate.
    - Type: DOUBLE
  - isZ
    - (Optional) TRUE indicates that the third parameter is z; false indicates m.
    - Type: BOOL
  - SRID
    - (Optional) The Spatial reference system identifier.
    - Type: INT4

- **Returns**
VARCHAR(50) A point geometry object.

Examples

```
SELECT inza..ST_AsText(inza..ST_Point(1.0, 5.0));
ST_ASTEXT
----------
POINT (1 5)
(1 row)
```

```
SELECT inza..ST_AsText(inza..ST_Point(1.0, 5.0, 6.0));
ST_ASTEXT
----------
POINT Z (1 5 6)
(1 row)
```

```
SELECT inza..ST_AsText(inza..ST_Point(1.0, 5.0, 8.0, False, 4326));
ST_ASTEXT
----------
POINT M (1 5 8)
(1 row)
```

Related Functions

► category Spatial

**ST_PointN - Nth Point in Linestring**

Determines the Nth point from the linestring.

**Usage**

The ST_PointN function has the following syntax:

```
ST_PointN(VARCHAR(ANY), INT4);
```

▲ Parameters

► **ST_Geometry**
A geometry object.

Type: VARCHAR(ANY)
Netezza Spatial Package Reference Guide

- \( n \)
  A 1-based index.
  Type: INT4

- Returns
  VARCHAR(100) A point object.

Examples

```sql
SELECT inza..ST_AsText(inza..ST_POINTN(inza..ST_WKToSQL('LINESTRING (0 0, 1 1, 2 2, 10 10)'), 2));
```

```
ST_ASTEXT
-----------------
POINT (1 1)
(1 row)
```

Related Functions

- category Spatial

ST_PointOnSurface - Point on the Surface

Finds a point that is guaranteed to be in the interior of the surface or multisurface.

Usage

The ST_PointOnSurface function has the following syntax:

```sql
ST_PointOnSurface(VARCHAR(ANY) Geometry)
```

- Parameters
  - `ST_Geometry`
    A geometry object.
    Type: VARCHAR(ANY)

- Returns
  VARCHAR(100) A point in the interior of the specified surface or multisurface.

Examples

```sql
SELECT inza..ST_ASTEXT(inza..ST_POINTONSURFACE(inza..ST_WKToSQL('POLYGON ((30 110, 50 110, 50 130, 30 130, 30 110))'));
```

```
ST_ASTEXT
```

ST_Relate - Relation of Geometries

Determines whether the intersection matrix of two geometries equals the specified intersection matrix.

Usage

The ST_Relate function has the following syntax:

```
ST_Relate(VARCHAR(ANY) Geometry1, VARCHAR(ANY) Geometry2, CHAR(9) intersectionMatrix);
```

**Parameters**

- **ST_Geometry1**
  A geometry object.
  Type: VARCHAR(ANY)

- **ST_Geometry2**
  A geometry object.
  Type: VARCHAR(ANY)

- **intersectionMatrix**
  Dimensionally Extended 9 Intersection Model (DE-9IM).
  Type: CHAR(9)

**Returns**

BOOL Returns TRUE if the intersection matrix of two geometries equals the specified intersection matrix; otherwise FALSE.

**Examples**

```sql
SELECT inza..ST_RELATE(inza..ST_WKTTOSQL('LINESTRING (0 0, 10 10)'), inza..ST_WKTTOSQL('LINESTRING (5 5, 11 11)'), '1*T***T**');
```

```
t
(1 row)
```

```sql
SELECT inza..ST_RELATE(inza..ST_WKTTOSQL('POLYGON ((40 120, 90 00J2350-03 Rev. 2')), inza..ST_WKTTOSQL('POLYGON ((40 120, 90 00J2350-03 Rev. 2)), '1*T***T**');
```

```
t
(1 row)
```
120, 90 150, 40 150, 40 120))'),
inza..ST_WKToSQL('POLYGON ((30 110, 50 110, 50 130, 30 130, 30 110))'), 'T*T***FF*');

ST_RELATE
-----------

f
(1 row)

Related Functions
► category Spatial
► ST_Contains
► ST_Crosses
► ST_Disjoint
► ST_Equals
► ST_Relate
► ST_Touches
► ST_Within

ST_SRID - Setter/Getter of the SRID

Sets or gets the SRID from a geometry object.

Usage
The ST_SRID function has the following syntax:

► ST_SRID(VARCHAR(ANY) Geometry); VARCHAR(30) = ST_SRID(VARCHAR(ANY) Geometry, INT4 SRID);

▲ Parameters
► ST_Geometry
  A geometry object.
  Type: VARCHAR(ANY)
► SRID
  (Optional) If specified, sets the Spatial Reference System Identifier.
  Type: INT4
▲ Returns
INT4_Or_VARCHAR(200) The SRID for the get. A geometry with the SRID for the set.

Examples
SELECT inza..ST_SRID(inza..ST_Point(1.0, 5.0, 4326));
ST_SRID
---------
4326
(1 row)

SELECT inza..ST_SRID(inza..ST_WKTToSQL('POLYGON((10 10, 10 20, 20 20, 20 15, 10 10))')));

ST_SRID
---------
4326
(1 row)

SELECT inza..ST_SRID(inza..ST_SRID(inza..ST_WKTToSQL('POLYGON((10 10, 10 20, 20 20, 20 15, 10 10))'), 1234));

ST_SRID
---------
1234
(1 row)

Related Functions
▶ category Spatial

ST_StartPoint - First Point of a Line

Determines the first point of a line.

Usage
The ST_StartPoint function has the following syntax:

▶ $ST_{StartPoint}($VARCHAR(ANY) Geometry$);

▲ Parameters
▶ $ST_{Geometry}$
A geometry object, which must be a line.
Type: VARCHAR(ANY)

▲ Returns
VARCHAR(50) Returns a point geometry object representing the first point of the line; NULL if there
is no start point.

Examples

```sql
SELECT inza..ST_AsText(inza..ST_StartPoint(inza..ST_WKTToSQL('LINESTRING (0 0, 11 0, 11 11)')));
```

```
ST_ASTEXT
------------
POINT (0 0)
(1 row)
```

```sql
SELECT inza..ST_AsText(inza..ST_StartPoint(inza..ST_WKTToSQL('LINESTRING (0 0 1, 11 0 5, 11 11 7)')));
```

```
ST_ASTEXT
------------
POINT Z (0 0 1)
(1 row)
```

```sql
SELECT inza..ST_StartPoint(inza..ST_WKTToSQL('LINESTRING EMPTY'));
```

```
ST_STARTPOINT
--------------
(1 row)
```

Related Functions

- **category Spatial**

**ST_SymDifference - Symmetric Difference of Geometries**

Determines a geometry object representing the non-intersecting parts of the specified geometries.

Usage

The `ST_SymDifference` function has the following syntax:
ST_SymDifference(VARCHAR(ANY) Geometry1, VARCHAR(ANY) Geometry2);

▲ Parameters
► ST_Geometry1
A geometry object.
Type: VARCHAR(ANY)
► ST_Geometry2
A geometry object.
Type: VARCHAR(ANY)

▲ Returns
VARCHAR(ANY) A geometry object representing the non-intersecting parts of the specified geometries.

Examples

```sql
SELECT
    inza..ST_ASTEXT(inza..ST_SYMDIFFERENCE(inza..ST_WKTToSQL('POLYGON ((0 0, 11 0, 11 11, 0 11, 0 0))'), inza..ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))')));

ST_ASTEXT
---------------
MULTIPOLYGON (((11 10.5, 11 0, 0 0, 0 11, 10 11, 10 10, 11 10.5)), ((11 10.5, 11 11, 10 11, 10 20, 20 20, 20 15, 11 10.5)))
(1 row)
```

Related Functions
► category Spatial
► ST_Intersection

**ST_Touches - Checks if Geometries Touch**

Determines whether the two geometries touch, but their interiors do not intersect.

**Usage**

The ST_Touches function has the following syntax:

► ST_Touches(VARCHAR(ANY) Geometry1, VARCHAR(ANY) Geometry2);

▲ Parameters
► ST_Geometry1
A geometry object.
Type: VARCHAR(ANY)
Geometry2
A geometry object.
Type: VARCHAR(ANY)

Returns
BOOL TRUE if the two geometries touch, but their interiors do not intersect; otherwise FALSE.

Details
Returns TRUE if the DE-9IM intersection matrix for the two geometries is FT******, F**T***** or F***T****. The current implementation performs calculations treating all coordinate systems as cartesian.

Examples

```sql
SELECT inza..ST_Touches(inza..ST_WKTToSQL('LINESTRING (0 0, 10 10)'), inza..ST_WKTToSQL('LINESTRING (5 5, 11 11)'));

ST_TOUCHES
-----------
| f |
(1 row)
```

```sql
SELECT inza..ST_Touches(inza..ST_WKTToSQL('LINESTRING (0 0, 10 10)'), inza..ST_WKTToSQL('LINESTRING (10 10, 11 11)'));

ST_TOUCHES
-----------
| t |
(1 row)
```

Related Functions
- category Spatial
- ST_Relate

ST_Union - Union of Geometries

Finds a geometry object representing the union of the specified geometries.
Usage
The ST_Union function has the following syntax:

► **ST_Union**(VARCHAR(ANY) Geometry1, VARCHAR(ANY) Geometry2);

▲ Parameters
► **ST_Geometry1**
  A geometry object.
  Type: VARCHAR(ANY)

► **ST_Geometry2**
  A geometry object.
  Type: VARCHAR(ANY)

▲ Returns
VARCHARG(ANY) A geometry object representing the union of the specified geometries.

Examples

```
SELECT inza..ST_AsText(inza..ST_Union(inza..ST_WKTToSQL( 'POLYGON ((0 0, 5 0, 5 5, 0 5, 0 0))'), inza..ST_WKTToSQL( 'POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))')));
```

```
ST_ASTEXT
-----------
MULTIPOLYGON (((0 0, 5 0, 5 5, 0 5, 0 0)), ((10 10, 10 20, 20 20, 20 15, 10 10)))
(1 row)
```

Related Functions
► category Spatial

**ST_Version - IBM Netezza Spatial Version**

Returns the version of the IBM Netezza Spatial Package.

Usage
The ST_Version function has the following syntax:

► **ST_Version();**

▲ Returns
VARCHARG(20) The version of IBM Netezza Spatial Package.

Related Functions
► category Spatial
ST_Within - Checks if the Geometry is Within Another Geometry

Determines if the first specified geometry is completely within the second geometry.

Usage

The ST_Within function has the following syntax:

```
ST_Within(VARCHAR(ANY) Geometry1, VARCHAR(ANY) Geometry2, VARCHAR(ANY) cSystem);
```

▲ Parameters

- **ST_Geometry1**
  - A geometry object.
  - Type: VARCHAR(ANY)

- **ST_Geometry2**
  - A geometry object.
  - Type: VARCHAR(ANY)

- **cSystem**
  - The coordinate system.
  - Type: VARCHAR(ANY)
  - Default: From geometry's SRID or 'WGS84' if geometry has no SRID.

▲ Returns

- BOOL Returns TRUE if the first geometry is completely within the second geometry; otherwise FALSE.

Details

Returns TRUE if the DE-9IM intersection matrix for the two geometries is T*F***F***. ST_Within is the reverse of ST_Contains. The current implementation performs calculations treating all coordinate systems as cartesian.

Examples

```
SELECT inza..ST_Within(inza..ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))', 1234),
inza..ST_WKTToSQL('POLYGON ((0 0, 110 0, 110 110, 0 110, 0 0))', 1234));

ST_WITHIN
----------
  t
(1 row)
```
Related Functions

- category Spatial
- ST_Contains
- ST_Relate

ST_WKBToSQL - Geometry from WKB Representation

Determines a geometry object from the Well-Known Binary (WKB) representation.

Usage

The ST_WKBToSQL function has the following syntax:

```
ST_WKBToSQL(VARCHAR(ANY) WKB); VARCHAR(ANY) = ST_WKBToSQL(VARCHAR(ANY) WKB, INT4 Srid); VARCHAR(ANY) = ST_WKBToSQL(VARCHAR(ANY) WKB, INT4 Srid, BOOL ComputeMBR); VARCHAR(ANY) = ST_WKBToSQL(VARCHAR(ANY) WKB, INT4 Srid, BOOL ComputeMBR, BOOL SkipSimpleTest);
```

▲ Parameters

- **ST_Geometry**
  A geometry object.
  Type: VARCHAR(ANY)

- **SRID**
  The Spatial Reference System Identifier.
  Type: INT4
  Default: 4326

- **computeMBRFlag**
  A Boolean value indicating whether the MBR should be computed.
  Type: BOOL
  Default: TRUE

- **skipSimpleTest**
  (Optional) A Boolean value indicating whether the simple geometry test should be skipped.
  Type: BOOL
  Default: FALSE

▲ Returns

VARCHARG(ANY) A geometry object.

Details

ST_WKBToSQL is exactly the same as ST_GeomFromWKB. ST_WKBToSQL is the reverse of ST_AsBinary.
Examples

```sql
select
  inza..ST_AsText(inza..ST_WKToSQL(inza..ST_AsBinary(inza.
  .ST_WKToSQL('POLYGON ((0 0, 11 0, 11 11, 0 11, 0
  0))'))));

ST_ASTEXT
---------------------------------------------------------
POLYGON ((0 0, 11 0, 11 11, 0 11, 0 0))
(1 row)
```

```sql
select
  inza..ST_AsText(inza..ST_WKToSQL(inza..ST_AsBinary(inza.
  .ST_WKToSQL('LINESTRING(0 0, 3 4, -1 1)'))));

ST_ASTEXT
---------------------------------------------------------
LINESTRING (0 0, 3 4, -1 1)
(1 row)
```

```sql
select
  inza..ST_AsText(inza..ST_WKToSQL(inza..ST_AsBinary(inza.
  .ST_Point(1, 5))));

ST_ASTEXT
-----------
POINT (1 5)
(1 row)
```

Related Functions

- category Spatial
- ST_GeomFromText
ST_WKBTOWKT - WKT Representation from WKB Format

Returns the Well-Known Text (WKT) representation of a geometry object.

Usage

The ST_WKBTOWKT function has the following syntax:

```
ST_WKBTOWKT(VARCHAR(ANY) WKB);
```

Parameters

- **ST_Geometry**: A geometry object.
  
  Type: VARCHAR(ANY)

Returns

VARCHAR(ANY) The Well-Known Text (WKT) representation of a geometry object.

Details

Does not return the SRID. ST_WKBTOWKT is the reverse of ST_GeomFromText.

Examples

```sql
SELECT inza..ST_WKBTOWKT(inza..ST_WKBTOWKT('POLYGON((0 0, 1 2, 3 -1, 0 0))'));
```

```
ST_WKBTOWKT

---------------------

POLYGON ((0 0, 1 2, 3 -1, 0 0))
```

(1 row)

```sql
SELECT inza..ST_WKBTOWKT(inza..ST_WKBTOWKT('POLYGON((0 0, 0 2, 1 4, 2 2, 2 0, 0 0))'));
```

```
ST_WKBTOWKT

---------------------

POLYGON ((0 0, 0 2, 1 4, 2 2, 2 0, 0 0))
```

(1 row)
ST_WKTToSQL - Geometry from WKT Representation

Determines a geometry object from the Well-Known Text (WKT) representation.

**Usage**

The ST_WKTToSQL function has the following syntax:

```sql
ST_WKTToSQL(VARCHAR(ANY) WKTString); VARCHAR(ANY) =
ST_WKTToSQL(VARCHAR(ANY) WKTString, INT4 Srid); VARCHAR(ANY) =
ST_WKTToSQL(VARCHAR(ANY) WKTString, INT4 Srid, BOOL ComputeMBR); VARCHAR(ANY) =
ST_WKTToSQL(VARCHAR(ANY) WKTString, INT4 Srid, BOOL ComputeMBR, BOOL SkipSimpleTest);
```

**Parameters**

- **ST_Geometry**
  A geometry object.
  Type: VARCHAR(ANY)

- **SRID**
  The Spatial Reference System Identifier.
  Type: INT4
  Default: 4326

- **computeMBRFlag**
  (Optional) A Boolean value indicating whether the MBR should be computed.
  Type: BOOL
  Default: TRUE

- **skipSimpleTest**
  (Optional) A Boolean value indicating whether the simple geometry test should be skipped.
  Type: BOOL
  Default: FALSE

**Returns**

VARCHAR(ANY) A geometry object.
Details

ST_WKTTToSQL is exactly the same as ST_GeomFromWKLT. ST_WKTTToSQL is the reverse of ST_AsText.

Examples

```
SELECT inza..ST_AsText(inza..ST_WKTTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))'));

ST_ASTEXT

POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))

(1 row)

SELECT inza..ST_AsText(inza..ST_WKTTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))', 4326));

ST_ASTEXT

POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))

(1 row)

SELECT inza..ST_AsText(inza..ST_WKTTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))', 4326, true));

ST_ASTEXT

POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))

(1 row)
```

Related Functions

- category Spatial
- ST_GeomFromText
- ST_GeomFromWKLT
- ST_WKBToSQL
- ST_AsText
ST_WKTTowkb - WKB Representation from WKT Format

Finds a Well-Known Binary (WKB) geometry object from the Well-Known Text (WKT) representation.

Usage

The ST_WKTTowkb function has the following syntax:

```
ST_WKTTowkb(VARCHAR(ANY) WKTString);
```

▲ Parameters

- **ST_Geometry**
  A geometry object.
  Type: VARCHAR(ANY)

▲ Returns

VARCHAR(ANY) A WKB geometry object.

Details

ST_WKTTowkb is exactly the same as ST_GeomFromWKT. ST_WKTTowkb is the reverse of ST_AsText.

Examples

```
SELECT inza..ST_WKBTowkb(inza..ST_WKTTowkb('POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))'));
```

```
+-----------------------------------------------+
| ST_WKBTowkb                                   |
| POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10)) |
+-----------------------------------------------+
```

(1 row)

Related Functions

- category Spatial
- ST_GeomFromText
- ST_GeomFromWKB
- ST_WKBtoSQL
- ST_AsText
- ST_AsBinary
**ST_X - X Coordinate of a Point**

Determines the x coordinate of a point object, or sets the x coordinate of a point object, to the specified value.

**Usage**

The ST_X function has the following syntax:

```
ST_X(VARCHAR(ANY) Geometry); VARCHAR(100) = ST_X(VARCHAR(ANY) Geometry, DOUBLE X);
```

- **Parameters**
  - **ST_Geometry**
    A geometry object, which must be a point object.
    Type: VARCHAR(ANY)
  - **x**
    (Optional) Sets the value for the x coordinate.
    Type: DOUBLE

- **Returns**
  DOUBLE_Or_VARCHAR(100) The x coordinate, or a geometry object with the x coordinate set to the specified x value.

**Examples**

```
SELECT inza..ST_X(inza..ST_Point(0.0, 1.0));
```

```
ST_X
------
0
(1 row)
```

```
SELECT inza..ST_AsText(inza..ST_X(inza..ST_Point(0.0, 1.0), 5));
```

```
ST_ASTEXT
----------
POINT (5 1)
(1 row)
```

```
SELECT inza..ST_AsText(inza..ST_X(inza..ST_Point(0.0, 1.0, 2.0), 5));
```

```
ST_ASTEXT
----------
POINT Z (5 1 2)
```
SELECT inza..ST_AsText(inza..ST_X(inza..ST_Point(0.0, 1.0, 2.0, 3.0), 5));

ST_ASTEXT
-----------

POINT ZM (5 1 2 3)
(1 row)

Related Functions
► category Spatial
► ST_Y
► ST_Z
► ST_M
► ST_Point

ST_Y - Y Coordinate of a Point

Determine the y coordinate of a point object, or sets the y coordinate of a point object, to the specified value.

Usage

The ST_Y function has the following syntax:

► ST_Y(VARCHAR(ANY) Geometry); VARCHAR(100) = ST_Y(VARCHAR(ANY) Geometry, DOUBLE Y);

▲ Parameters

► ST_Geometry
   A geometry object, which must be a point object.
   Type: VARCHAR(ANY)

► y
   (Optional) Sets the value for the y coordinate.
   Type: DOUBLE

▲ Returns
   DOUBLE Or VARCHAR(100) The y coordinate, or a geometry object with the y coordinate set to the value specified by y.
Examples

```sql
SELECT inza..ST_Y(inza..ST_Point(0.0, 1.0));

ST_Y
-----
  1
(1 row)

SELECT inza..ST_AsText(inza..ST_Y(inza..ST_Point(0.0, 1.0), 5));

ST_ASTEXT
-----------
  POINT (0 5)
(1 row)

SELECT inza..ST_AsText(inza..ST_Y(inza..ST_Point(0.0, 1.0, 2.0, 3.0), 5));

ST_ASTEXT
-----------
  POINT ZM (0 5 2 3)
(1 row)
```

Related Functions

- category Spatial
- ST_Y
- ST_Z
- ST_M
- ST_Point

**ST_Z - Z Coordinate of a Point**

Determines the z coordinate of a point object or sets the z coordinate of a point object to the specified value.

**Usage**

The ST_Z function has the following syntax:

```sql
ST_Z(VARCHAR(ANY) Geometry); VARCHAR(100) = ST_Z(VARCHAR(ANY) Geometry, DOUBLE Z);
```

Parameters
► **ST_Geometry**

A geometry object, which must be a point object.

Type: VARCHAR(ANY)

► **z**

(Optional) Sets the value for the z coordinate. If NULL, the z value is removed from the point.

Type: DOUBLE

▲ Returns

DOUBLE Or VARCHAR(100) The z coordinate or the geometry object with the z coordinate set to the specified z value.

**Examples**

```
SELECT inza..ST_Z(inza..ST_Point(0.0, 1.0, 2.0));
ST_Z
-----
  2
(1 row)

SELECT inza..ST_AsText(inza..ST_Z(inza..ST_Point(0.0, 1.0, 2.0), 5));
ST_ASTEXT
-----------
 POINT Z (0 1 5)
(1 row)

SELECT inza..ST_AsText(inza..ST_Z(inza..ST_Point(0.0, 1.0), 5));
ST_ASTEXT
-----------
 POINT Z (0 1 5)
(1 row)

SELECT inza..ST_AsText(inza..ST_Z(inza..ST_Point (3, 8, 23, 7, 4326), 40));
ST_ASTEXT
```
---------------

POINT ZM (3 8 40 7)

(1 row)

Related Functions

- category Spatial
- ST_X
- ST_Y
- ST_M
- ST_Point
ST_CreateGeomColumn - Create the Geometry Column Table

This stored procedure creates the Geometry Column Table if it does not exists. It will preserve the table if it already exists.

Usage

The ST_CreateGeomColumn stored procedure has the following syntax:

\[ \text{ST_CreateGeomColumn()} \]

- Returns
  - BOOLEAN true on success and false on failure.

Examples

\c inza
set PATH=inza.inza;
call inza..ST_CreateGeomColumn();
\n
\text{ST_CREATEGEOMCOLUMN}

\[ \text{---------------------} \]

\[ f \]

(1 row)

Related Functions

- category Utilities - Actions
**ST_CreateSpatialRefSys - Create the Spatial Reference System Table**

This stored procedure creates the Spatial Reference System Table if it does not exist. It will preserve the table if it already exists.

**Usage**

The ST_CreateSpatialRefSys stored procedure has the following syntax:

- **ST_CreateSpatialRefSys()**
  - Returns BOOLEAN true on success and false on failure.

**Examples**

```
\c inza
set PATH=inza.inza;
call inza..ST_CreateSpatialRefSys();
ST_CREATESPATIALREFSYS
-------------
f
(1 row)
```

**Related Functions**

- category Utilities - Actions
Notices and Trademarks

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
26 Forest Street
Marlborough, MA 01752 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement.
or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only. This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. (enter the year or years). All rights reserved.

Trademarks

IBM, the IBM logo, ibm.com and Netezza are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

NEC is a registered trademark of NEC Corporation.

UNIX is a registered trademark of The Open Group in the United States and other countries.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Red Hat is a trademark or registered trademark of Red Hat, Inc. in the United States and/or other countries.

D-CC, D-C++, Diab+, FastJ, pSOS+, SingleStep, Tornado, VxWorks, Wind River, and the Wind River logo are trademarks, registered trademarks, or service marks of Wind River Systems, Inc. Tornado patent pending.

APC and the APC logo are trademarks or registered trademarks of American Power Conversion Corporation.

Other company, product or service names may be trademarks or service marks of others.

Regulatory and Compliance

Regulatory Notices

Install the NPS system in a restricted-access location. Ensure that only those trained to operate or service the equipment have physical access to it. Install each AC power outlet near the NPS rack that plugs into it, and keep it freely accessible. Provide approved 30A circuit breakers on all power sources.

Product may be powered by redundant power sources. Disconnect ALL power sources before servicing. High leakage current. Earth connection essential before connecting supply. Courant de fuite élevé. Raccordement à la terre indispensable avant le raccordement au réseau.

Homologation Statement

Attention: This product is not intended to be connected directly or indirectly by any means whatsoever to interfaces of public telecommunications networks, neither to be used in a Public Services Network.

FCC - Industry Canada Statement

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio-frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case users will be required to correct the interference at their own expense.

This Class A digital apparatus meets all requirements of the Canadian Interference-Causing Equipment Regulations.

Cet appareil numérique de la classe A respecte toutes les exigences du Règlement sur le matériel brouilleur du Canada.

WEEE

Netezza Corporation is committed to meeting the requirements of the European Union (EU) Waste Electrical and Electronic Equipment (WEEE) Directive. This Directive requires producers of electrical and electronic equipment to finance the takeback, for reuse or recycling, of their products placed on the EU market after August 13, 2005.

CE Statement (Europe)


Warning: This is a class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.
VCCI Statement

この装置は、情報処理装置等電波障害自主規制協議会（VCCI）の基準に基づくクラスA情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起こすことがあります。この場合には使用者が適切な対策を講ずるよう要求されることがあります。
Index

S

ST_Area, 13
ST_AsBinary, 15
ST_AsKML, 16
ST_AsText, 17
ST_Boundary, 18
ST_Buffer, 19
ST_Centroid, 21
ST_Collect, 22
ST_Contains, 25
ST_CoordDim, 27
ST_CreateGeomColumn, 105
ST_CreateSpatialRefSys, 106
ST_Crosses, 28
ST_Difference, 29
ST_Dimension, 30
ST_Disjoint, 31
ST_Distance, 32
ST_DWithin, 34
ST_Ellipse, 35
ST_EndPoint, 37
ST_Envelope, 38
ST_Equals, 39
ST_Expand, 39
ST_ExteriorRing, 41
ST_GeometryN, 42
ST_GeometryType, 43
ST_GeometryTypeID, 44
ST_GeomFromText, 45
ST_GeomFromWKB, 46
ST_GrandMBR, 48
ST_InteriorRingN, 49
ST_Intersection, 50
ST_Intersects, 51
ST_Is3D, 53
ST_IsClosed, 54
ST_IsEmpty, 55
ST_IsMeasured, 56
ST_IsRing, 57
ST_IsSimple, 58

ST_Length, 59
ST_LineFromMultiPoint, 61
ST_LocateAlong, 62
ST_LocateBetween, 63
ST_M, 63
ST_MaxM, 65
ST_MaxX, 66
ST_MaxY, 67
ST_MaxZ, 68
ST_MBR, 70
ST_MBRIntersects, 71
ST_MinM, 71
ST_MinX, 73
ST_MinY, 74
ST_MinZ, 75
ST_NumGeometries, 77
ST_NumInteriorRing, 77
ST_NumPoints, 79
ST_Overlaps, 80
ST_Perimeter, 81
ST_Point, 82
ST_PointN, 83
ST_PointOnSurface, 84
ST_Relate, 85
ST_SRID, 86
ST_StartPoint, 87
ST_SymDifference, 89
ST_Touches, 89
ST_Union, 91
ST_Version, 91
ST_Within, 92
ST_WKBToSQL, 93
ST_WKBTowkt, 95
ST_WKTToSQL, 96
ST_WKTToWKB, 98
ST_X, 99
ST_Y, 100
ST_Z, 101