IBM® Netezza® Analytics
Release 3.3.5.0

Netezza Spatial ESRI Package Reference Guide

Part Number 00J2577-01 Rev. 1

Note: Before using this information and the product that it supports, read the information in "Notices and Trademarks" on page 122.

Contents

3 Reference Documentation: Utilities

Notices and Trademarks

Index

Preface

This guide describes the IBM Netezza Spatial ESRI Package.

Audience for This Guide

This guide is written for users who wish to use the IBM Netezza Spatial ESRI Package with their IBM Netezza systems. This guide does not provide a tutorial on spatial concepts. Depending on their needs, users should be very familiar with some or all of these topics. Users should also be somewhat familiar with the basic operation and concepts of the IBM Netezza system.

Purpose of This Guide

This guide describes the IBM Netezza Spatial ESRI Package. The IBM Netezza Spatial ESRI Package provides spatial analysis functions that can be used on the IBM Netezza database warehouse appliance. The content provided consists of reference documentation for each Spatial ESRI UDF, UDA and stored procedure.

Conventions

The following conventions apply:

- In the technical literature, both the guides and reference guides, the term "Analytic Executable" or "AE" is used. In marketing materials, the term "User-Defined Analytic Process" or "UDAP" is used. The terms User-Defined Analytic Process and UDAP are synonymous with the terms Analytic Executable and AE.
- Upper case for SQL commands; for example INSERT, DELETE
- In code samples, a single backslash ("\") at the end of a line denotes a line continuation and should be omitted when using the code at the command line, a SQL command or in a file.
- When referencing a sequence of menu and submenu selections, the ">" character denotes the different menu options in the form: "Menu Name > Submenu Name > Selection". Note that not all commands use submenus, while some selections may utilize a number of nested submenus.

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.
2. Go to the IBM Support Portal at http://www.ibm.com/support. Log in using your IBM ID and password. You can search the Support Portal for solutions. To submit a support request, click the 'Service Requests & PMRs' tab.
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 Support section of the IBM Directory of worldwide contacts:

http://www14.software.ibm.com/webapp/set2/sas/f/handbook/contacts.html#phone.

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_AsSdeBinary - ESRI SDE Binary Representation of the 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_Convert - Converts a Geometry Between Two Supported Formats

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 - Check 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 - Check 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 Multi Geometry

ST_GeometryType - Type of a Geometry

ST_GeometryTypeID - Geometry Type ID of a Geometry

ST_GeomFromSDE - Geometry from SDE Representation

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 - Create a geometry that is the intersection of the geometries in a given table

ST_Intersection - Intersection of Geometries

ST_Intersects - Checks if Geometries Intersect

ST_Intersects - Create a geometry that is the union of a table of geometries

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 Linstring 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

ST_MinZ - Minimum Z-coordinate of a Geometry

ST_NumGeometries - Number of Geometries in a Multi Geometry

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_SDEToSQL - Geometry from SDE representation

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_Transform - Transforms the spatial reference system of a geometry

ST_Union - Create a geometry that is the union of a table of geometries

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

ST_CreateSRS - Create a Custom Spatial Reference System

ST_DropSRS - Drop a Custom Spatial Reference System

ST_MapPolygonsToGrid - Maps polygons to a grid

ST_SpatialGridIndex - Creates a spatial grid index

Utilities - Checking

ISDATE_TINY - Check if a string has the compact date format YYYYMMDD

Utilities - Preprocessing

drand64 - 64 bits pseudo-random number generator

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(ST_GEOMETRY(ANY) ST_Geometry, VARCHAR(ANY) unit);**
  - Parameters
    - **ST_Geometry**
      The geometry object.

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

      Type: VARCHAR(ANY)

      Default: 'meter'
  - Returns
    DOUBLE The area of the specified geometry object.

Details

This function returns the area of the specified geometry object having a surface: polygon, multipolygon, geometry collection. Units are supported. The coordinate system of the geometry is used.

Examples

```
SELECT inza..ST_Area(inza..ST_WKTToSQL('POLYGON((1 1, 1 2,
2 2, 2 1, 1 1))', 27700));
```

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

```
SELECT inza..ST_Area(inza..ST_WKTToSQL('POLYGON((1 1, 1
2, 2 2, 2 1, 1 1))', 4269));
     ST_AREA
-----------------
 12304814949.668
(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))', 27700), 'foot');
    ST_AREA
----------------
 10.76391041671
(1 row)
```

```
SELECT
inza..ST_Area(inza..ST_Transform(inza..ST_WKTToSQL('POLYG
ON((1 1, 1 2, 2 2, 2 1, 1 1))', 27700), 4326), 'meter');
     ST_AREA
-----------------
 0.9373240361496
(1 row)
```

```
SELECT
inza..ST_Area(inza..ST_Transform(inza..ST_WKTToSQL('POLYGON((1
1, 1 2, 2 2, 2 1, 1 1))'), 4269), 'meter');

     ST_AREA

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

 12305124757.028

(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.

Usage

The ST_AsBinary function has the following syntax:

- **ST_AsBinary(ST_GEOMETRY(ANY) ST_Geometry);**
  - Parameters
    - **ST_Geometry**
      The geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    VARBINARY(ANY) The Well-Known Binary (WKB) representation of a geometry object. The WKB output does not contain the SRID.

Details

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

Examples

```
SELECT
inza..ST_AsText(inza..ST_WKBToSQL(inza..ST_AsBinary(inza..ST_Poi
nt(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(ST_GEOMETRY(ANY) ST_Geometry); VARCHAR(ANY) = ST_AsKML(ST_GEOMETRY(ANY) ST_Geometry, INT4 precision); VARCHAR(ANY) = ST_AsKML(ST_GEOMETRY(ANY) ST_Geometry, VARCHAR(ANY) additionalKMLAttributes); VARCHAR(ANY) = ST_AsKML(ST_GEOMETRY(ANY) ST_Geometry, VARCHAR(ANY) additionalKMLAttributes, INT4 precision);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
    - **additionalKMLAttributes**
      (Optional) One of <extrude>, <tessellate> or <altitudeMode>.

      Type: VARCHAR(ANY)

      Default: ''
    - **precision**
      (Optional) The number of decimal places for all coordinates.

      Type: INT4

      Default: 16
  - 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

```
SELECT inza..ST_AsKML(inza..ST_Point(1, 5), 2);
 ST_ASKML
 ----------
 <Point><coordinates>1.00,5.00</coordinates></Point>
(1 row)


SELECT inza..ST_AsKML(inza..ST_Point(1, 5),
'<extrude>1</extrude><altitudeMode>clampToGround</altitudeMode>'
, 2);
ST_ASKML
----------
 <Point><extrude>1</extrude><altitudeMode>clampToGround</altitu
deMode><coordinates>1.00,5.00</coordinates></Point>
(1 row)
```

Related Functions

- category Spatial

ST_AsSdeBinary - ESRI SDE Binary Representation of the Geometry

Determines the ESRI SDE Binary representation of a geometry object.

Usage

The ST_AsSdeBinary function has the following syntax:

- **ST_AsSdeBinary(ST_GEOMETRY(ANY) ST_Geometry);**
  - Parameters
    - **ST_Geometry**
      The geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    VARCHAR(ANY) The ESRI SDE Binary representation of a geometry object. The ESRI SDE Binary will not contain the SRID.

Details

Takes a geometry object and returns its ESRI SDE Binary representation. ST_AsSdeBinary is the reverse of ST_SDEToSQL.

Examples

```
SELECT
inza..ST_AsText(inza..ST_SDEToSQL(inza..ST_AsSdeBinary(in
za..ST_Point(1, 2)), 4326));
  ST_ASTEXT
-------------
 POINT (1 2)
(1 row)
```

Related Functions

- category Spatial
- ST_AsText
- ST_GeomFromText
- ST_GeomFromSDE
- ST_SDEToSQL
- ST_WKTToSQL

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(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(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

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

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

```
select inza..ST_AsText(inza..ST_WKTToSQL('POLYGON((10 10, 10 20,
20 20, 20 15, 10 10))'));
                    ST_ASTEXT
-----------------------------------------------
 POLYGON ((10 10, 20 15, 20 20, 10 20, 10 10))
(1 row)
```

Related Functions

- category Spatial
- ST_GeomFromText
- ST_WKTToSQL

ST_Boundary - Boundary of the Geometry

Determines the boundary of a geometry object.

Usage

The ST_Boundary function has the following syntax:

- **ST_Boundary(ST_GEOMETRY(ANY) ST_Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    ST_GEOMETRY(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
```

```
-------------------------------------------

 MULTILINESTRING ((1 1, 2 1, 2 2, 1 2, 1 1))

(1 row)
```

```
SELECT
inza..ST_AsText(inza..ST_Boundary(inza..ST_WKTToSQL('LINE
STRING (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.

Usage

The ST_Buffer function has the following syntax:

- **ST_Buffer(ST_GEOMETRY(ANY) ST_Geometry, DOUBLE distance, INT nSegments, VARCHAR(ANY) unit);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(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 unit of the distance parameter. Possible values include "meter", "kilometer", "foot", "mile" and "nautical mile".

  Type: VARCHAR(ANY)

  Default: 'meter'

- Returns
  ST_GEOMETRY(ANY) A geometry object.

Examples

```
SELECT inza..ST_AsText(inza..ST_Buffer(inza..ST_Point(0, 0,
27700), 1, 2));


ST_ASTEXT

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

 POLYGON ((1 0, 0.70710000023246 0.70710000023246, 0 1,
-0.70710000023246 0.70710000023246, -1 0, -0.70710000023246
-0.70710000023246, 0 -1, 0.70710000023246 -0.70710000023246, 1
0))

(1 row)

ST_ASTEXT


SELECT inza..ST_AsText(inza..ST_Buffer(inza..ST_Point(0, 0,
27700), 1, 2, 'foot'));


ST_ASTEXT

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

 POLYGON ((0.30480000004172 0, 0.21549999993294
0.21550000086427, 0 0.30480000004172, -0.21549999993294
0.21550000086427, -0.30480000004172 0, -0.21549999993294
-0.21550000086427, 0 -0.30480000004172, 0.21549999993294
-0.21550000086427, 0.30480000004172 0))

(1 row)


SELECT inza..ST_AsText(inza..ST_Buffer(inza..ST_Point(0, 0,
27700), 1, 2, 'meter'));
```

```
ST_ASTEXT

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

 POLYGON ((1 0, 0.70710000023246 0.70710000023246, 0 1,
-0.70710000023246 0.70710000023246, -1 0,
-0.70710000023246 -0.70710000023246, 0 -1,
0.70710000023246 -0.70710000023246, 1 0))

(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(ST_GEOMETRY(ANY) ST_Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    ST_GEOMETRY(126) A point geometry object.

Details

The geometric center of the geometry is the "average" of the points in the geometry. The result is not guaranteed to be on the geometry.

Examples

```
SELECT
inza..ST_AsText(inza..ST_Centroid(inza..ST_WKTToSQL('POLY
GON ((0 0, 10 0, 10 10, 0 10, 0 0))')));

ST_ASTEXT
```

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


SELECT
inza..ST_AsText(inza..ST_Centroid(inza..ST_WKTToSQL('POLYGON
((10 10, 10 20, 20 20, 20 15, 10 10))')));
ST_ASTEXT
-----------
 POINT (15 15)
(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(ST_GEOMETRY(ANY) ST_Geometry);**
  - Parameters
    - **ST_Geometry**
      A point geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    ST_GEOMETRY(ANY) The multipoint geometry object.

Examples

```
CREATE TABLE points (PointID integer, the_geom
st_geometry(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 function has the following syntax:

- **ST_Collect(ST_GEOMETRY(ANY) ST_Geometry, INT4 Start, INT4 End);**
  - Parameters
    - **ST_Geometry**
      A point geometry object.

      Type: ST_GEOMETRY(ANY)
    - **Start**
      Start of the group.

      Type: INT4
    - **End**
      End of the group.

      Type: INT4
  - Returns
    ST_GEOMETRY(ANY) The multipoint geometry object named "multipoint".

Examples

```
CREATE TABLE trip_points (trip_id integer, geom
st_geometry(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;
```

```
 TRIP_ID |                    ST_ASTEXT
---------+-------------------------------------------------
     100 | MULTIPOINT (100 100, 200 200, 300 300, 400 400)
     200 | MULTIPOINT (200 200, 300 300, 100 100, 400 400)
     300 | MULTIPOINT (400 400, 300 300, 200 200, 100 100)
```

```
(3 rows)
```

Related Functions

- category Spatial

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(ST_GEOMETRY(ANY) ST_Geometry1, ST_GEOMETRY(ANY) ST_Geometry2);**
  - Parameters
    - **ST_Geometry1**
      A geometry object.

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

      Type: ST_GEOMETRY(ANY)
  - Returns
    BOOL TRUE if the first geometry contains the second geometry; otherwise FALSE.

Details

ST_Contains is the reverse of ST_Within.

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
-------------
 f
(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_Convert - Converts a Geometry Between Two Supported Formats

Converts geomerties between VARCHAR/WKB geometry and ST_GEOMETRY/ESRIB geometry.

Usage

The ST_Convert function has the following syntax:

- **ST_Convert(ST_GEOMETRY(ANY) ST_Geometry1); ST_Convert(VARCHAR(ANY) ST_Geometry2);**
  - Parameters
    - **ST_Geometry1**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    VARCHAR(ANY) or ST_GEOMETRY(ANY) A point geometry object.

Examples

```
SELECT
inza..ST_AsText(inza..ST_Convert(inza..ST_Convert(inza..ST_WKTTo
SQL('LINESTRING (0 0, 10 0)'))));
```

```
ST_ASTEXT
-----------
 LINESTRING (0 0, 10 0)
(1 row)
```

Related Functions

- category Spatial

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(ST_GEOMETRY(ANY) ST_Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    ST_GEOMETRY(ANY) A geometry object.

Examples

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

ST_ASTEXT

-----------

 POINT (0 0)

(1 row)


SELECT
inza..ST_AsText(inza..ST_ConvexHull(inza..ST_WKTToSQL('LI
NESTRING (0 0, 10 10, 20 10, 30 10)')));

ST_ASTEXT

-----------

 POLYGON ((0 0, 30 10, 20 10, 10 10, 0 0))

(1 row)


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

ST_ASTEXT

-----------
```

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

Related Functions

- category Spatial

ST_CoordDim - Coordinate Dimension

Determines the coordinate dimension of the geometry object.

Usage

The ST_CoordDim function has the following syntax:

- **ST_CoordDim(ST_GEOMETRY(ANY) ST_Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(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
-------------
           2
(1 row)

SELECT inza..ST_CoordDim(inza..ST_WKTToSQL('POINT Z(0 0 0)'));
ST_COORDDIM
-------------
           3
(1 row)
```

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(ST_GEOMETRY(ANY) ST_Geometry1, ST_GEOMETRY(ANY) ST_Geometry2);**
  - Parameters
    - **ST_Geometry1**
      A geometry object.

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

      Type: ST_GEOMETRY(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********.

Examples

```
SELECT inza..ST_Crosses(inza..ST_WKTToSQL('LINESTRING (0
0, 10 10)'), inza..ST_WKTToSQL('LINESTRING (0 5, 10
5)'));
 ST_CROSSES
------------
 t
(1 row)
```

Related Functions

- category Spatial
- ST_Relate

ST_Difference - Difference of Two Geometries

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

Usage

The ST_Difference function has the following syntax:

- **ST_Difference(ST_GEOMETRY(ANY) ST_Geometry1, ST_GEOMETRY(ANY) ST_Geometry2);**
  - Parameters
    - **ST_Geometry1**
      A geometry object.

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

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

Examples

```
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 ((0 0, 11 0, 11 10.5, 10 10, 10 11, 0 11, 0 0))

(1 row)
```

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:

- **ST_Dimension(ST_GEOMETRY(ANY) ST_Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    INT Returns 0 for point, 1 for lines or 2 for polygons.

Examples

```
SELECT inza..ST_Dimension(inza..ST_WKTToSQL('POINT (0
0)'));
 ST_DIMENSION
--------------
            0
(1 row)


SELECT inza..ST_Dimension(inza..ST_WKTToSQL('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 - Check if Geometries are Disjoint

Determines if the two specified geometries do not intersect.

Usage

The ST_Disjoint function has the following syntax:

- **ST_Disjoint(ST_GEOMETRY(ANY) ST_Geometry1, ST_GEOMETRY(ANY) ST_Geometry2);**
  - Parameters
    - **ST_Geometry1**
      A geometry object.

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

      Type: ST_GEOMETRY(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****.

Examples

```
SELECT inza..ST_Disjoint(inza..ST_WKTToSQL('LINESTRING (0 0, 10
10)'), inza..ST_WKTToSQL('LINESTRING (0 5, 10 5)'));
 ST_DISJOINT
-------------
 f
(1 row)
```

Related Functions

- category Spatial
- ST_Intersects
- ST_Relate

ST_Distance - Distance between Geometries

Determines the minimum distance between two geometries.

Usage

The ST_Distance function has the following syntax:

- **ST_Distance(ST_GEOMETRY(ANY) ST_Geometry1, ST_GEOMETRY(ANY) ST_Geometry2, VARCHAR(ANY) unit, BOOL intersectTest);**
  - Parameters
    - **ST_Geometry1**
      A geometry object.

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

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

      Type: VARCHAR(ANY)

      Default: 'meter'
    - **intersectTest**
      (Optional) If enabled, tests for intersection between the geometries and returns 0 if they do. Otherise, calculate distance between the geometries.

      Type: BOOL

      Default: TRUE
  - Returns
    DOUBLE Returns the minimum distance between two geometries.

Details

This function returns the minimum distance from one geometry, ST_Geometry1, to a second geometry, ST_Geometry2.

Examples

```
SELECT inza..ST_Distance(inza..ST_Point(0, 0, 27700),
inza..ST_Point(1, 0, 27700));
 ST_DISTANCE
-------------
           1
(1 row)


SELECT inza..ST_Distance(inza..ST_Point(0, 0, 27700),
inza..ST_Point(1, 0, 27700), 'foot');
   ST_DISTANCE
```

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

```
SELECT inza..ST_Distance(inza..ST_Transform(inza..ST_Point(0, 0,
27700), 4326), inza..ST_Transform(inza..ST_Point(1, 0, 27700),
4326), 'meter');
   ST_DISTANCE
------------------
 0.99836324443601
(1 row)
```

Related Functions

- category Spatial

ST_DWithin - Distance Within

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

Usage

The ST_DWithin function has the following syntax:

- **ST_DWithin(ST_GEOMETRY(ANY) ST_Geometry1, ST_GEOMETRY(ANY) ST_Geometry2, DOUBLE distance, VARCHAR(ANY) unit);**
  - Parameters
    - **ST_Geometry1**
      A geometry object.

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

      Type: ST_GEOMETRY(ANY)
    - **distance**
      The distance within which the geometries must be.

      Type: DOUBLE
    - **unit**
      The units used for the distance parameter. Possible values include "meter", "kilometer", "foot", "mile" and "nautical mile".

      Type: VARCHAR(ANY)

Default: 'meter'

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

Examples

```
SELECT
inza..ST_DWithin(inza..ST_Transform(inza..ST_Point(0,0),
27700), inza..ST_Transform(inza..ST_Point(1,1), 27700),
156987, 'meter');
 ST_DWITHIN
------------
 t
(1 row)


SELECT
inza..ST_DWithin(inza..ST_Transform(inza..ST_Point(0,0),
27700), inza..ST_Transform(inza..ST_Point(1,1), 27700),
2, 'foot');
 ST_DWITHIN
------------
 f
(1 row)


SELECT
inza..ST_DWithin(inza..ST_Transform(inza..ST_Point(0,0),
4326), inza..ST_Transform(inza..ST_Point(1,1), 4326), 2,
'meter');
 ST_DWITHIN
------------
 f
(1 row)
```

Related Functions

- category Spatial

ST_Ellipse - Ellipse Constructor

Specifies an ellipse with a 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 nSegments, VARCHAR(ANY) unit, INTEGER SRID); (ANY) = ST_Ellipse(ST_GEOMETRY(ANY) ST_Geometry, DOUBLE a, DOUBLE b, DOUBLE tilt, INT nSegments, VARCHAR(ANY) unit);**
  - Parameters
    - **x0**
      The longtitude or the x-coordinate of the center.

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

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

      Type: DOUBLE
    - **b**
      The semi-minor axis.

      Type: DOUBLE
    - **tilt**
      The major axis tilt.

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

      Type: INT

      Default: 8
    - **unit**
      The units used for the a and b parameters. Possible values include "meter", "kilometer", "foot", "mile" and "nautical mile".

      Type: VARCHAR(ANY)

      Default: 'meter'
    - **SRID**
      (Optional) If specified, sets the Spatial Reference System Identifier.

      Type: INT4

      Default: '4326'
  - Returns

ST_GEOMETRY(ANY) Returns an ellipse (polygon) with a specified center, axes and tilt.

Examples

```
SELECT inza..ST_AsText(inza..ST_Ellipse(1.0, 2.0, 100.0,
50.0, 30.0, 2, 'meter', 27700));


ST_ASTEXT

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

 POLYGON ((51 88.602499999106, 5.7367000002414
80.914899999276, -42.301300000399 27, -64.974000000395
-41.559599999338, -49 -84.602600000799, -3.7368000000715
-76.914999999106, 44.301199999638 -23, 66.973899999633
45.559499999508, 51 88.602499999106))

(1 row)

ST_ASTEXT


SELECT
inza..ST_AsText(inza..ST_Transform(inza..ST_Ellipse(inza.
.ST_WKTToSql('point (1.0 2.0)'), 100.0, 50.0, 30.0, 2,
'meter'), 27700));


ST_ASTEXT

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

 POLYGON ((733904.5182 -5306149.6444, 733838.1317
-5306175.3022, 733811.2396 -5306211.4765, 733793.5087
-5306252.918, 733804.7382 -5306323.1987, 733871.1246
-5306297.541, 733898.0167 -5306261.3666, 733915.7476
-5306219.9251, 733904.5182 -5306149.6444))

(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(ST_GEOMETRY(ANY) ST_Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object, which must be a line.

      Type: ST_GEOMETRY(ANY)
  - Returns
    ST_GEOMETRY(126) 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(ST_GEOMETRY(ANY) ST_Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)

- Returns
  ST_GEOMETRY(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_WKTToSQL('LINE
STRING (0 0, 1 -1, 2 2)')));

ST_ASTEXT

-----------

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

(1 row)
```

Related Functions

- category Spatial
- ST_MBR

ST_Equals - Check if Geometries are Equal

Determines if two geometries are equal.

Usage

The ST_Equals function has the following syntax:

- **ST_Equals(ST_GEOMETRY(ANY) ST_Geometry1, ST_GEOMETRY(ANY) ST_Geometry2);**
  - Parameters
    - **ST_Geometry1**
      A geometry object.

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

      Type: ST_GEOMETRY(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*.

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(ST_GEOMETRY(ANY) ST_Geometry, DOUBLE distance, VARCHAR(ANY) unit);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
    - **distance**
      The distance to expand the bounding box around the input geometry's MBR.

      Type: DOUBLE
    - **unit**
      The units used for the distance value. Possible values include "meter", "kilometer", "foot", "mile" and "nautical mile".

      Type: VARCHAR(ANY)

      Default: 'meter'
  - Returns
    ST_GEOMETRY(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_WKTToSQL('POLYGO
N ((10 10, 10 20, 20 20, 20 15, 10 10))', 27700), 100));

ST_ASTEXT

-----------

 POLYGON ((-90 -90, 120 -90, 120 120, -90 120, -90 -90))

(1 row)
```

```
SELECT
inza..ST_AsText(inza..ST_Expand(inza..ST_WKTToSQL('POLYGO
N ((10 10, 10 20, 20 20, 20 15, 10 10))', 27700), 100,
'foot'));

ST_ASTEXT

-----------

 POLYGON ((-20.480000000447 -20.480000000447,
50.480000000447 -20.480000000447, 50.480000000447
50.480000000447, -20.480000000447 50.480000000447,
-20.480000000447 -20.480000000447))

(1 row)
```

```
SELECT
inza..ST_AsText(inza..ST_Expand(inza..ST_Transform(inza..
ST_WKTToSQL('POLYGON ((10 10, 10 20, 20 20, 20 15, 10
10))', 27700), 4326), 100, 'meter'));


ST_ASTEXT

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

 POLYGON ((-7.558430215 49.766004321, -7.55551071
49.766004321, -7.55551071 49.767898653, -7.558430215
49.767898653, -7.558430215 49.766004321))

(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(ST_GEOMETRY(ANY) ST_Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    ST_GEOMETRY(ANY) A line geometry object.

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 Multi Geometry

Determines the Nth geometry from a multi geometry.

Usage

The ST_GeometryN function has the following syntax:

- **ST_GeometryN(ST_GEOMETRY(ANY) ST_Geometry, INT n);**
  - Parameters

- **ST_Geometry**
  A geometry object.

  Type: ST_GEOMETRY(ANY)
- **n**
  A 1-based index.

  Type: INT

- Returns
  ST_GEOMETRY (ANY) A geometry object; NULL if the specified geometry is not a multi geometries.

Examples

```
SELECT
inza..ST_AsText(inza..ST_GeometryN(inza..ST_WKTToSQL('MUL
TIPOINT (10 10, 5 6)'), 2));

ST_ASTEXT

-----------

 POINT (5 6)

(1 row)


SELECT
inza..ST_AsText(inza..ST_GeometryN(inza..ST_WKTToSQL('MUL
TIPOINT (10 10, 5 6)'), 1));

ST_ASTEXT

-----------

 POINT (10 10)

(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:

- **ST_GeometryType(ST_GEOMETRY(ANY) ST_Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    VARCHAR(50) The geometry type of the geometry object as a string.

Examples

```
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 ID of a Geometry

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

Usage

The ST_GeometryTypeID function has the following syntax:

- **ST_GeometryTypeID(ST_GEOMETRY(ANY) ST_Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)

- Returns
  INT The geometry type ID of the geometry object as a number.

Examples

```
SELECT inza..ST_GeometryTypeID(inza..ST_WKTToSQL('POINT
(0 0)'));
 ST_GEOMETRYTYPEID
-------------------
                 1
(1 row)

SELECT inza..ST_GeometryTypeID(inza..ST_WKTToSQL('POLYGON
((10 10, 10 20, 20 20, 20 15, 10 10))'));
 ST_GEOMETRYTYPEID
-------------------
                 3
(1 row)
```

Related Functions

- category Spatial
- ST_GeometryType

ST_GeomFromSDE - Geometry from SDE Representation

Determines a geometry object from the ESRI Compressed Binary (SDE) representation.

Usage

The ST_GeomFromSDE function has the following syntax:

- **ST_GeomFromSDE(VARBINARY(ANY) SDE, INT4 Srid);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: VARBINARY(ANY)
    - **SRID**
      The Spatial Reference System Identifier.

Type: INT4

- Returns
  ST_GEOMETRY(ANY) A geometry object.

Details

ST_GeomFromSDE is exactly the same as ST_SDEToSQL. ST_SDEToSQL is the reverse of ST_AsSdeBinary.

Examples

```
select
inza..ST_AsText(inza..ST_GeomFromSDE(inza..ST_AsSdeBinary(inza..
ST_WKTToSQL('POLYGON ((0 0, 11 0, 11 11, 0 11, 0 0))')), 4326));

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


select
inza..ST_AsText(inza..ST_GeomFromSDE(inza..ST_AsSdeBinary(inza..
ST_WKTToSQL('LINESTRING(0 0, 3 4, -1 1)')), 4326));

                                                                ST
_ASTEXT
-------------------------------------------------------------------
-------------------------------------------------------------------
-----
 LINESTRING (0 0, 3 4, -1 1)
(1 row)


select
inza..ST_AsText(inza..ST_GeomFromSDE(inza..ST_AsSdeBinary(inza..
ST_Point(1, 5)), 4326));
ST_ASTEXT
-----------
 POINT (1 5)
(1 row)
```

Related Functions

- category Spatial
- ST_GeomFromText
- ST_SDEToSQL
- ST_WKTToSQL
- ST_AsText
- ST_AsSdeBinary

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); ST_GEOMETRY(ANY) = ST_GeomFromText(VARCHAR(ANY) WKTSTring, INT4 Srid);**
  - Parameters
    - **WKTString**
      A Well Known Text string.

      Type: VARCHAR(ANY)
    - **SRID**
      The Spatial Reference System Identifier.

      Type: INT4

      Default: 4326
  - Returns
    ST_GEOMETRY(ANY) A geometry object.

Details

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

Examples

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

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

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

                    ST_ASTEXT

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

 POLYGON ((10 10, 20 15, 20 20, 10 20, 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:

- **ST_GeomFromWKB(VARCHAR(ANY) WKB); ST_GEOMETRY(ANY) = ST_GeomFromWKB(VARCHAR(ANY) WKB, INT4 Srid);**
  - Parameters
    - **WKB**
      A Well Known Binary.

      Type: VARCHAR(ANY)
    - **SRID**
      The Spatial Reference System Identifier.

      Type: INT4

      Default: 4326
  - Returns
    ST_GEOMETRY(ANY) A geometry object.

Details

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

Examples

```
select
inza..ST_AsText(inza..ST_GeomFromWKB(inza..ST_AsBinary(in
za..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)


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

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

 LINESTRING (0 0, 3 4, -1 1)

(1 row)


select
inza..ST_AsText(inza..ST_GeomFromWKB(inza..ST_AsBinary(in
za..ST_Point(1, 5))));

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

 POINT (1 5)

(1 row)
```

Related Functions

- category Spatial
- ST_GeomFromText
- ST_WKBToSQL

- ST_WKTToSQL
- ST_AsText
- ST_AsBinary

ST_GrandMBR - Bounding Box from a Set of Geometries

Determines the bounding box from a set of geometry objects.

Usage

The ST_GrandMBR aggregate has the following syntax:

- **ST_GrandMBR(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    ST_GEOMETRY(200) The bounding box as a geometry object.

Examples

```
CREATE TABLE polygons (PointID integer, the_geom
st_geometry(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
----------------------------------------

 POLYGON ((0 0, 44 0, 44 44, 0 44, 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(ST_GEOMETRY(ANY) Geometry, INT4 N);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
    - **N**
      A 1-based index.

      Type: INT4
  - Returns
    ST_GEOMETRY(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 - Create a geometry that is the intersection of the geometries in a given table

Creates a geometry by doing an intersection on a table of geometries.

Usage

The ST_Intersection aggregate has the following syntax:

- **ST_Intersection(ST_GEOMETRY(ANY) geometry);**
  - Parameters
    - **geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    ST_GEOMETRY(ANY) The unified geometry object.

Examples

```
CREATE TABLE points (PointID integer, the_geom
st_geometry(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_Intersection(the_geom)) from
(SELECT the_geom from points order by PointID LIMIT 9999999)
points;

DROP TABLE points;

  ST_ASTEXT

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

 POINT EMPTY

(1 row)
```

Related Functions

- category Spatial

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(ST_GEOMETRY(ANY) Geometry1, ST_GEOMETRY(ANY) Geometry2);**
  - Parameters
    - **ST_Geometry1**
      A geometry object.

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

      Type: ST_GEOMETRY(ANY)
  - Returns
    ST_GEOMETRY(ANY) The shared geometry object. The geometry object may be empty if no points are shared.

Examples

```
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
-----------
 POLYGON ((10 10, 11 10.5, 11 11, 10 11, 10 10))
(1 row)
```

```
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
-------------
 POINT EMPTY
(1 row)
```

```
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 10, 15 12.5, 15 15, 10 15, 10 10))
(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(ST_GEOMETRY(ANY) Geometry1, ST_GEOMETRY(ANY) Geometry2);**
  - Parameters
    - **ST_Geometry1**
      A geometry object.

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

      Type: ST_GEOMETRY(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****.

Examples

```
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)


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)


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_Intersects - Create a geometry that is the union of a table of geometries

Determines whether all geometries in a table intersect.

Usage

The ST_Intersects aggregate has the following syntax:

- **ST_Intersects(ST_GEOMETRY(ANY) geometry);**
  - Parameters
    - **geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    BOOL Returns TRUE if the table of geometries intersect; otherwise FALSE.

Examples

```
CREATE TABLE points (PointID integer, the_geom
st_geometry(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_Intersects(the_geom) from (SELECT the_geom from
points order by PointID LIMIT 9999999) points;

DROP TABLE points;

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

Related Functions

- category Spatial

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(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(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
---------
 f
(1 row)


SELECT inza..ST_Is3D(inza..ST_WKTToSQL('LINESTRING Z(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
---------
 t
(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:

- **ST_IsClosed(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object, which must be a line.

      Type: ST_GEOMETRY(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

```
SELECT inza..ST_IsClosed(inza..ST_WKTToSQL('LINESTRING (0 0, 11
0, 11 11, 0 0)'));
 ST_ISCLOSED
-------------
 t
(1 row)


SELECT inza..ST_IsClosed(inza..ST_WKTToSQL('LINESTRING ZM(1 2 3
4, 8 8 8 8, 9 9 9 9, 1 2 3 4)'));
 ST_ISCLOSED
-------------
 t
(1 row)


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

```
(1 row)
```

Related Functions

- category Spatial

ST_IsEmpty - Checks if the Geometry is Empty

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

Usage

The ST_IsEmpty function has the following syntax:

- **ST_IsEmpty(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    BOOL Returns TRUE if the geometry is empty; otherwise FALSE.

Examples

```
SELECT inza..ST_IsEmpty(inza..ST_WKTToSQL('POLYGON ((0 0,
11 0, 11 11, 0 11, 0 0))'));
 ST_ISEMPTY
------------
 f
(1 row)


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

Determines whether the geometry has an m value.

Usage

The ST_IsMeasured function has the following syntax:

- **ST_IsMeasured(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(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
(1 row)
```

```
SELECT inza..ST_IsMeasured(inza..ST_WKTToSQL('LINESTRING ZM(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(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object, which must be a line.

      Type: ST_GEOMETRY(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(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(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_WKTToSQL('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(ST_GEOMETRY(ANY) Geometry); DOUBLE = ST_Length(ST_GEOMETRY(ANY) Geometry, VARCHAR(ANY) unit);**
  - Parameters
    - **ST_GEOMETRY**
      A geometry object, which must be a LineString or MultiLineString.

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

      Type: VARCHAR(ANY)

      Default: 'meter'
  - Returns
    DOUBLE The length of the line.

Details

Function takes an ST_GEOMETRY as an input.

Examples

```
SELECT inza..ST_Length(inza..ST_WKTToSQL('LINESTRING(0 0,
3 4, -1 1)', 27700), 'meter');

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


SELECT
inza..ST_Length(inza..ST_WKTToSQL('MULTILINESTRING((0 0,
3 4, -1 1), (100 100, 400 500, 800 800))', 27700),
'meter');
```

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

```
SELECT inza..ST_Length(inza..ST_WKTToSQL('MULTILINESTRING((0 0,
1 0), (0 1, 1 0))', 4326));
ST_LENGTH
-----------
 268219.05908461
(1 row)
```

Related Functions

- category Spatial

ST_LineFromMultiPoint - Make a Linstring from a Multipoint geometry

Makes a Linstring from a Multipoint geometry

Usage

The ST_LineFromMultiPoint function has the following syntax:

- **ST_LineFromMultiPoint(ST_GEOMETRY(ANY) ST_Geometry);**
  - Parameters
    - **ST_Geometry**
      A Multipoint geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    ST_GEOMETRY(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('M
ULTIPOINT (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_LineFromMultiPoint(inza..ST_WKTT
oSQL('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 that matches the specified value of the m coordinate.

Usage

The ST_LocateAlong function has the following syntax:

- **ST_LocateAlong(ST_GEOMETRY(ANY) Geometry, DOUBLE m);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
    - **m**
      The start range of the measure to find.

      Type: DOUBLE
  - Returns
    ST_GEOMETRY(ANY) Returns a derived geometry that matches the specified value of m coordinate.

Details

Points, MultiPoints, LineStrings and MultiLineStrings are supported.

Examples

```
SELECT
inza..ST_AsText(inza..ST_LocateAlong(inza..ST_WKTToSQL('MULTIPOI
NT ZM(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 that matches the specified range of m coordinate values inclusively. Points, MultiPoints, LineStrings and MultiLineStrings are supported.

Usage

The ST_LocateBetween function has the following syntax:

- **ST_LocateBetween(ST_GEOMETRY(ANY) Geometry, DOUBLE m1, DOUBLE m2);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

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

      Type: DOUBLE
    - **m2**
      The end range of the measure to find.

      Type: DOUBLE
  - Returns
    ST_GEOMETRY(ANY) Returns a derived geometry that matches the specified range of m coordinate values inclusively;

Examples

```
SELECT
inza..ST_AsText(inza..ST_LocateBetween(inza..ST_WKTToSQL('MULTIP
```

```
OINT ZM(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); ST_GEOMETRY(126) = ST_M(ST_GEOMETRY(ANY) Geometry, DOUBLE M);**
  - Parameters
    - **Geometry**
      A geometry object, which must be a point object.

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

      Type: DOUBLE
  - Returns
    DOUBLE_Or_ST_GEOMETRY(126) The m coordinate or a geometry object with the m coordinate set to the value specified by m.

Examples

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

 ST_M

------

    3
```

```
(1 row)

SELECT inza..ST_M(inza..ST_Point(0.0, 1.0, 2.0, false));
ST_M
------
    2
(1 row)

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_Y
- ST_Z
- ST_Point

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(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    DOUBLE The maximum m-coordinate; NULL if geometry is empty.

Examples

```
SELECT inza..ST_MaxM(inza..ST_WKTToSQL('POLYGON ZM((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_MinZ
- ST_MinM
- ST_Envelope

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(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

Type: ST_GEOMETRY(ANY)

- Returns
  DOUBLE The maximum x-coordinate; NULL if the geometry is empty.

Examples

```
SELECT inza..ST_MaxX(inza..ST_WKTToSQL('POLYGON ((10 10, 10 20,
20 20, 20 15, 10 10))'));
 ST_MAXX
---------
      20
(1 row)


SELECT inza..ST_MaxX(inza..ST_WKTToSQL('MULTIPOINT(10 10, 30 30,
15 15)'));
 ST_MAXX
---------
      30
(1 row)


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(ST_GEOMETRY(ANY));**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(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('MULTIPOINT(10 10,
30 15, 15 15, 20 20)'));
ST_MAXY
---------
      20
(1 row)


SELECT inza..ST_MaxY(inza..ST_WKTToSQL('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(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    DOUBLE The maximum z-coordinate; NULL if the geometry is empty

Examples

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

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


SELECT inza..ST_MaxZ(inza..ST_WKTToSQL('POLYGON ZM((10 10 10 10,
10 20 20 20, 20 20 20 20, 20 15 15 40, 10 10 10 10))'));

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


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

Determines the bounding box of a geometry object.

Usage

The ST_MBR function has the following syntax:

- **ST_MBR(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    ST_GEOMETRY(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, 20 10, 20 20, 10 20, 10 10))
(1 row)
```

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

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

 POLYGON ((-1.9999788491987e-09 0.99999999800002,
1.9999788491987e-09 0.99999999800002, 1.9999788491987e-09
1.000000002, -1.9999788491987e-09 1.000000002,
-1.9999788491987e-09 0.99999999800002))
(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:

- **ST_MBRIntersects(ST_GEOMETRY(ANY) Geometry1, ST_GEOMETRY(ANY) Geometry2);**
  - Parameters
    - **ST_Geometry1**
      A geometry object.

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

      Type: ST_GEOMETRY(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

```
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:

- **ST_MinM(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    DOUBLE The minimum m-coordinate; NULL if the geometry is empty

Examples

```
SELECT inza..ST_MinM(inza..ST_WKTToSQL('POLYGON ZM((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_WKTToSQL('MULTIPOINT ZM(10 10 10
10, 30 30 30 30, 15 15 15 15, 20 20 20 20)'));
ST_MINM
---------
      10
(1 row)
```

```
SELECT inza..ST_MinM(inza..ST_WKTToSQL('LINESTRING EMPTY'));
 ST_MINM
---------
(1 row)
```

Related Functions

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

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(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

Type: ST_GEOMETRY(ANY)

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

Examples

```
SELECT inza..ST_MinX(inza..ST_WKTToSQL('POLYGON ((10 10,
10 20, 20 20, 20 15, 10 10))'));
 ST_MINX
---------
      10
(1 row)


SELECT inza..ST_MinX(inza..ST_WKTToSQL('MULTIPOINT (10
5 ,30 15, 15 15, 20 20)'));
 ST_MINX
---------
      10
(1 row)


SELECT inza..ST_MinX(inza..ST_WKTToSQL('LINESTRING
EMPTY'));
 ST_MINX
---------
(1 row)
```

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(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    DOUBLE The minimum y-coordinate if the geometry object; NULL if the geometry is empty

Examples

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

ST_MINY

---------

      10

(1 row)
```

```
SELECT inza..ST_MinY(inza..ST_WKTToSQL('MULTIPOINT (10 5, 30 15,
15 15, 20 20)'));

ST_MINY

---------

       5

(1 row)
```

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

ST_MINY

---------

(1 row)
```

Related Functions

- category Spatial

- ST_MaxX
- ST_MaxY
- ST_MinX
- 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(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object

      Type: ST_GEOMETRY(ANY)
  - Returns
    DOUBLE The minimum z-coordinate; NULL if the geometry is empty.

Examples

```
SELECT inza..ST_MinZ(inza..ST_WKTToSQL('POLYGON Z((10 10
0, 10 20 1, 20 20 2, 20 15 3, 10 10 0))'));
 ST_MINZ
---------
       0
(1 row)


SELECT inza..ST_MinZ(inza..ST_WKTToSQL('POLYGON ZM((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_WKTToSQL('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 Multi Geometry

Determines the number of geometries in a geometry object.

Usage

The ST_NumGeometries function has the following syntax:

- **ST_NumGeometries(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    INT4 The number of geometries in the geometry object; returns 1 for geometries that are not multi geometries.

Examples

```
SELECT inza..ST_NumGeometries(inza..ST_WKTToSQL('MULTIPOLYGON
(((10 10, 10 20, 20 20, 20 15, 10 10)), ((0 0, 1 0, 1 1, 0 1, 0
0)))'));
ST_NUMGEOMETRIES
------------------
```

```
                    2
(1 row)
```

```
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 a polygon.

Usage

The ST_NumInteriorRing function has the following syntax:

- **ST_NumInteriorRing(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object, which must be a polygon.

      Type: ST_GEOMETRY(ANY)
  - Returns
    INT4 The 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(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(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('MULTIPOINT
(10 10, 30 30, 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:

- **ST_Overlaps(ST_GEOMETRY(ANY) Geometry1, ST_GEOMETRY(ANY) Geometry2);**
  - Parameters
    - **ST_Geometry1**
      A geometry object.

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

      Type: ST_GEOMETRY(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**.

Examples

```
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(ST_GEOMETRY(ANY) Geometry); DOUBLE = ST_Perimeter(ST_GEOMETRY(ANY) Geometry, VARCHAR(ANY) unit);**
  - Parameters
    - **Geometry**
      A geometry object.

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

      Type: VARCHAR(ANY)

      Default: 'meter'
  - 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))', 27700));

ST_PERIMETER

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

             4

(1 row)


SELECT inza..ST_Perimeter(inza..ST_WKTToSQL('MULTIPOLYGON
(((0 0, 0 1, 1 1, 1 0, 0 0)), ((10 10, 10 14, 15 14, 15
10, 10 10)))', 27700));

 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); ST_GEOMETRY(126) = ST_Point(DOUBLE X, DOUBLE Y, DOUBLE Z); ST_GEOMETRY(126) = ST_Point(DOUBLE X, DOUBLE Y, DOUBLE Z, DOUBLE M); ST_GEOMETRY(126) = ST_Point(DOUBLE X, DOUBLE Y, DOUBLE ZorM, BOOL isZ); ST_GEOMETRY(126) = ST_Point(DOUBLE X, DOUBLE Y, INT4 SRID); ST_GEOMETRY(126) = ST_Point(DOUBLE X, DOUBLE Y, DOUBLE Z, INT4 SRID); ST_GEOMETRY(126) = ST_Point(DOUBLE X, DOUBLE Y, DOUBLE Z, DOUBLE M, INT4 SRID); ST_GEOMETRY(126) = 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

- **SRID**
  (Optional) The Spatial reference system identifier.

  Type: INT4

▲ Returns
ST_GEOMETRY(126) 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(ST_GEOMETRY(ANY), INT4);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
    - **n**
      A 1-based index.

      Type: INT4
  - Returns
    ST_GEOMETRY(126) A point object.

Examples

```
SELECT
inza..ST_AsText(inza..ST_POINTN(inza..ST_WKTToSQL('LINEST
RING (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:

- **ST_PointOnSurface(ST_GEOMETRY(ANY) Geometry)**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    ST_GEOMETRY(126) A point in the interior of the specified surface or multisurface.

Examples

```
SELECT
inza..ST_ASTEXT(inza..ST_POINTONSURFACE(inza..ST_WKTToSQL('POLYG
ON ((30 110, 50 110, 50 130, 30 130, 30 110))')));
```

```
ST_ASTEXT
-----------
 POINT (40 120)
(1 row)
```

Related Functions

- category Spatial

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(ST_GEOMETRY(ANY) Geometry1, ST_GEOMETRY(ANY) Geometry2, VARCHAR(ANY) intersectionMatrix);**
  - Parameters
    - **ST_Geometry1**
      A geometry object.

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

      Type: ST_GEOMETRY(ANY)

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

  Type: VARCHAR(ANY)

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

Examples

```
SELECT inza..ST_RELATE(inza..ST_WKTToSQL('LINESTRING (0
0, 10 10)'), inza..ST_WKTToSQL('LINESTRING (5 5, 11
11)'), '1*T***T**');
 ST_RELATE
-----------
 t
(1 row)


SELECT inza..ST_RELATE(inza..ST_WKTToSQL('POLYGON ((40
120, 90 120, 90 150, 40 150, 40 120))'),
inza..ST_WKTToSQL('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_SDEToSQL - Geometry from SDE representation

Determines a geometry object from the ESRI Compressed Binary (SDE) representation.

Usage

The ST_SDEToSQL function has the following syntax:

- **ST_SDEToSQL(VARBINARY(ANY) SDE, INT4 Srid);**
  - Parameters
    - **SDE**
      An ESRI Compressed Binary.

      Type: VARBINARY(ANY)
    - **SRID**
      The Spatial Reference System Identifier

      Type: INT4
  - Returns
    ST_GEOMETRY(ANY) A geometry object.

Details

ST_SDEToSQL is exactly the same as ST_GeomFromSDE. ST_SDEToSQL is the reverse of ST_AsSdeBinary.

Examples

```
select
inza..ST_AsText(inza..ST_SDEToSQL(inza..ST_AsSdeBinary(inza..ST_
WKTToSQL('POLYGON ((0 0, 11 0, 11 11, 0 11, 0 0))')), 4326));


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

 POLYGON ((0 0, 11 0, 11 11, 0 11, 0 0))

(1 row)


select
inza..ST_AsText(inza..ST_SDEToSQL(inza..ST_AsSdeBinary(inza..ST_
WKTToSQL('LINESTRING(0 0, 3 4, -1 1)')), 4326));

                                                                ST
_ASTEXT
-------------------------------------------------------------------
-------------------------------------------------------------------
-----

 LINESTRING (0 0, 3 4, -1 1)

(1 row)
```

```
select
inza..ST_AsText(inza..ST_SDEToSQL(inza..ST_AsSdeBinary(in
za..ST_Point(1, 5)), 4326));

ST_ASTEXT

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

 POINT (1 5)

(1 row)
```

Related Functions

- category Spatial
- ST_GeomFromText
- ST_GeomFromSDE
- ST_WKTToSQL
- ST_AsText
- ST_AsSdeBinary

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(ST_GEOMETRY(ANY) Geometry); ST_GEOMETRY(ANY) = ST_SRID(ST_GEOMETRY(ANY) Geometry, INT4 SRID);**
  - Parameters
    - **Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
    - **SRID**
      (Optional) If specified, sets the Spatial Reference System Identifier.

      Type: INT4
  - Returns
    INT4_Or_ST_GEOMETRY(ANY) 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))', 4269));
ST_SRID
---------
    4269
(1 row)


SELECT
inza..ST_SRID(inza..ST_SRID(inza..ST_WKTToSQL('POLYGON((10 10,
10 20, 20 20, 20 15, 10 10))'), 27700));
ST_SRID
---------
  27700
(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(ST_GEOMETRY(ANY) Geometry);**
  - Parameters
    - **ST_Geometry**
      A geometry object, which must be a line.

Type: ST_GEOMETRY(ANY)

- Returns
  ST_GEOMETRY(126) Returns a point geometry object representing the first point of the line; NULL if there is no start point.

Examples

```
SELECT
inza..ST_AsText(inza..ST_StartPoint(inza..ST_WKTToSQL('LI
NESTRING (0 0, 11 0, 11 11)')));

ST_ASTEXT

-----------

 POINT (0 0)

(1 row)


SELECT
inza..ST_AsText(inza..ST_StartPoint(inza..ST_WKTToSQL('LI
NESTRING Z(0 0 1, 11 0 5, 11 11 7)')));

ST_ASTEXT

-----------

 POINT Z (0 0 1)

(1 row)


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(ST_GEOMETRY(ANY) Geometry1, ST_GEOMETRY(ANY) Geometry2);**
  - Parameters
    - **Geometry1**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
    - **Geometry2**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    ST_GEOMETRY(ANY) A geometry object representing the non-intersecting parts of the specified geometries.

Examples

```
SELECT
inza..ST_ASTEXT(inza..ST_SYMDIFFERENCE(inza..ST_WKTToSQL('POLYGO
N ((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 (((0 0, 11 0, 11 10.5, 10 10, 10 11, 0 11, 0 0)),
((10 11, 11 11, 11 10.5, 20 15, 20 20, 10 20, 10 11)))

(1 row)
```

Related Functions

- category Spatial
- ST_Intersection

ST_Touches - Checks if Geometries Touch

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

Usage

The ST_Touches function has the following syntax:

- **ST_Touches(ST_GEOMETRY(ANY) Geometry1, ST_GEOMETRY(ANY) Geometry2);**
  - Parameters
    - **ST_Geometry1**

A geometry object.

Type: ST_GEOMETRY(ANY)

- **Geometry2**
  A geometry object.

  Type: ST_GEOMETRY(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****.

Examples

```
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)


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_Transform - Transforms the spatial reference system of a geometry

Transforms a Geometry object from its current spatial reference system to a geometry object using

the specified spatial reference system.

Usage

The ST_Transform function has the following syntax:

- **ST_Transform(ST_GEOMETRY(ANY) ST_Geometry, INT4 srid);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
    - **SRID**
      Spatial Reference System Identifier.

      Type: INT4
  - Returns
    ST_GEOMETRY(ANY) A new geometry that has been transformed from the old geometry but using a coordinate reference system defined by the SRID.

Examples

```
SELECT
inza..ST_AsText(inza..ST_Transform(inza..ST_WKTToSQL('POLYGON
((10 10, 10 20, 20 20, 20 15, 10 10))'), 27700));


ST_ASTEXT

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

 POLYGON ((1724412.4316 -4397861.1924, 2817375.6518
-3744702.2482, 2745912.5732 -3156572.3234, 1662480.6089
-3270453.2226, 1724412.4316 -4397861.1924))

(1 row)
```

Related Functions

- category Spatial

ST_Union - Create a geometry that is the union of a table of geometries

Creates a geometry by doing a union on a table of geometries.

Usage

The ST_Union aggregate has the following syntax:

- **ST_Union(ST_GEOMETRY(ANY) geometry);**
  - Parameters
    - **geometry**
      A geometry object.

      Type: ST_GEOMETRY(ANY)
  - Returns
    ST_GEOMETRY(ANY) The unified geometry object.

Examples

```
CREATE TABLE points (PointID integer, the_geom
st_geometry(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_Union(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_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(ST_GEOMETRY(ANY) Geometry1, ST_GEOMETRY(ANY) Geometry2);**
  - Parameters
    - **ST_Geometry1**
      A geometry object.

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

      Type: ST_GEOMETRY(ANY)
  - Returns
    ST_GEOMETRY(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, 20 15, 20
20, 10 20, 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
    VARCHAR(100) The version of the IBM Netezza Spatial Package.

    Related Functions

- category Spatial

ST_Within - Checks if the Geometry is Within Another Geometry

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

Usage

The ST_Within function has the following syntax:

- **ST_Within(ST_GEOMETRY(ANY) Geometry1, ST_GEOMETRY(ANY) Geometry2);**
  - Parameters
    - **ST_Geometry1**
      A geometry object.

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

      Type: ST_GEOMETRY(ANY)
  - 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.

Examples

```
SELECT inza..ST_Within(inza..ST_WKTToSQL('POLYGON ((10
10, 10 20, 20 20, 20 15, 10 10))', 27700),
inza..ST_WKTToSQL('POLYGON ((0 0, 110 0, 110 110, 0 110,
0 0))', 27700));

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(VARBINARY(ANY) WKB); ST_GEOMETRY(ANY) = ST_WKBToSQL(VARBINARY(ANY) WKB, INT4 Srid);**
  - Parameters
    - **ST_Geometry**
      A geometry object.

      Type: VARBINARY(ANY)
    - **SRID**
      The Spatial Reference System Identifier.

      Type: INT4

      Default: 4326
  - Returns
    VARCHAR(ANY) A geometry object.

Details

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

Examples

```
select
inza..ST_AsText(inza..ST_WKBToSQL(inza..ST_AsBinary(inza..ST_WKT
ToSQL('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)


select
inza..ST_AsText(inza..ST_WKBToSQL(inza..ST_AsBinary(inza..ST_WKT
ToSQL('LINESTRING(0 0, 3 4, -1 1)'))));

                                                                ST
_ASTEXT
-------------------------------------------------------------------
-------------------------------------------------------------------
-----
 LINESTRING (0 0, 3 4, -1 1)
```

```
(1 row)
```

```
select
inza..ST_AsText(inza..ST_WKBToSQL(inza..ST_AsBinary(inza.
.ST_Point(1, 5))));
ST_ASTEXT
-----------
 POINT (1 5)
(1 row)
```

Related Functions

- category Spatial
- ST_GeomFromText
- ST_GeomFromWKB
- ST_WKTToSQL
- ST_AsText
- ST_AsBinary

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
    - **WKB**
      A well known binary object.

      Type: VARBINARY(ANY)
    - **SRID**
      The Spatial Reference System Identifier.

      Type: INT4

      Default: 4326
  - 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

```
SELECT inza..ST_WKBToWKT(inza..ST_WKTToWKB('POLYGON((0 0, 1 2, 3
-1, 0 0))'));

ST_WKBTOWKT

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

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

(1 row)


SELECT inza..ST_WKBToWKT(inza..ST_WKTToWKB('POLYGON((0 0, 0 2, 1
4, 2 2, 2 0, 0 0))'));

ST_WKBTOWKT

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

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

(1 row)
```

Related Functions

- category Spatial
- ST_GeomFromText
- ST_GeomFromWKB
- ST_WKTToSQL
- ST_WKBToSQL
- ST_AsBinary

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:

- **ST_WKTToSQL(VARCHAR(ANY) WKTString); ST_GEOMETRY(ANY) = ST_WKTToSQL(VARCHAR(ANY) WKTSTring, INT4 Srid);**
  - Parameters
    - **WKTString**
      A well know text string.

      Type: VARCHAR(ANY)

- **SRID**
  The Spatial Reference System Identifier.

  Type: INT4

  Default: 4326

- Returns
  ST_GEOMETRY(ANY) A geometry object.

Details

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

Examples

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

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

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

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

SELECT inza..ST_AsText(inza..ST_WKTToSQL('POLYGON ((10
10, 10 20, 20 20, 20 15, 10 10))', 4326));
ST_ASTEXT
-----------
```

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

Related Functions

- category Spatial
- ST_GeomFromText
- ST_GeomFromWKB
- ST_WKBToSQL
- ST_AsText
- ST_AsBinary

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) WKT);**
  - Parameters
    - **WKT**
      A geometry object.

      Type: VARCHAR(ANY)
    - **SRID**
      The Spatial Reference System Identifier.

      Type: INT4

      Default: 4326
  - Returns
    VARBINARY(ANY) A WKB object.

Details

ST_WKTToWKB is the reverse of ST_WKBToWKT.

Examples

```
SELECT inza..ST_WKBToWKT(inza..ST_WKTToWKB('POLYGON ((10 10, 10
20, 20 20, 20 15, 10 10))'));
                  ST_WKBTOWKT
--------------------------------------------
 POLYGON ((10 10, 20 15, 20 20, 10 20, 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(ST_GEOMETRY(ANY) Geometry); ST_GEOMETRY(126) = ST_X(ST_GEOMETRY(ANY) Geometry, DOUBLE X);**
  - Parameters
    - **Geometry**
      A geometry object, which must be a point object.

      Type: ST_GEOMETRY(ANY)
    - **X**
      (Optional) Sets the value for the x coordinate.

      Type: DOUBLE
  - Returns
    DOUBLE_Or_ST_GEOMETRY(126) 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)
(1 row)

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

Determines 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(ST_GEOMETRY(ANY) Geometry); ST_GEOMETRY(126) = ST_Y(ST_GEOMETRY(ANY) Geometry, DOUBLE Y);**

- Parameters
  - **Geometry**
    A geometry object, which must be a point object.

    Type: ST_GEOMETRY(ANY)
  - **Y**
    (Optional) Sets the value for the y coordinate.

    Type: DOUBLE
- Returns
  DOUBLE_Or_ST_GEOMETRY(126) The y coordinate or a geometry object with the y coordinate set to the value specified by y.

Examples

```
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:

- **ST_Z(ST_GEOMETRY(ANY) Geometry); ST_GEOMETRY(126) = ST_Z(ST_GEOMETRY(ANY) Geometry, DOUBLE Z);**
  - Parameters
    - **Geometry**
      A geometry object, which must be a point object.

      Type: ST_GEOMETRY(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_ST_GEOMETRY(126) 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

CHAPTER 3
Reference Documentation: Utilities

drand64 - 64 bits pseudo-random number generator

This function returns a random number using a data-slice independent 64 bits pseudo-random number generator.

Usage

The drand64 function has the following syntax:

- **drand64(INT8 randomSeed)**
  - Parameters
    - **randomSeed**
      random seed

      Type: INT8
  - Returns
    DOUBLE random number between 0.0 and 1.0

Details

This function delivers a random double value between 0.0 and 1.0. The 64 bits pseudo-random number generator is based on different prime numbers per data slice, ensuring the independence of the sequences of numbers producted on any pair of data slices.

Examples

```
SELECT drand64(123);

    DRAND64

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

 0.35831899940968
```

```
(1 row)
```

Related Functions

- category Utilities - Preprocessing

ISDATE_TINY - Check if a string has the compact date format YYYYMMDD

This function checks that the input string represents a valid date with format YYYYMMDD

Usage

The ISDATE_TINY function has the following syntax:

- **ISDATE_TINY(VARCHAR(100) date)**
  - Parameters
    - **date**
      the date string to check

      Type: VARCHAR(100)
  - Returns
    INT4 1 if the string is a valid date with format YYYYMMDD, NULL if the string is NULL, or 0 otherwise

Details

This function checks that the input string represents a valid date with format YYYYMMDD. A valid date consist of exactly 8 characters, each character must be between '0' and '9'. The month part must be between 01 and 12, the days between 01 and the number of days in the month. Leap years are taken into account.

Note that it is possible to pass date as INT. However, when INT is automatically converted to VARCHAR, its leading zeros are truncated: e.g. 00010522 gets '10522' and is finally not recognized as a valid date.

Examples

```
CREATE TABLE DateAsString(date VARCHAR(50));
INSERT INTO DateAsString VALUES ('2009010');
INSERT INTO DateAsString VALUES ('200901222');
INSERT INTO DateAsString VALUES ('200');
INSERT INTO DateAsString VALUES ('-20091101');
INSERT INTO DateAsString VALUES ('20090001');
INSERT INTO DateAsString VALUES ('20091301');
```

```
INSERT INTO DateAsString VALUES ('20090400');
INSERT INTO DateAsString VALUES ('20090431');
INSERT INTO DateAsString VALUES ('20080229');
INSERT INTO DateAsString VALUES ('20090229');
INSERT INTO DateAsString VALUES ('21000229');
INSERT INTO DateAsString VALUES ('19991919');
INSERT INTO DateAsString VALUES ('20040429');
INSERT INTO DateAsString VALUES ('yyyymmdd');
INSERT INTO DateAsString VALUES ('abcd0101');
INSERT INTO DateAsString VALUES ('Date');
INSERT INTO DateAsString VALUES (NULL);
SELECT ISDATE_TINY(date), date FROM DateAsString ORDER BY
isdate_tiny, date;
CALL DROP_TABLE('DateAsString');
 ISDATE_TINY |   DATE
-------------+-----------
             |
           0 | -20091101
           0 | 19991919
           0 | 200
           0 | 20090001
           0 | 2009010
           0 | 200901222
           0 | 20090229
           0 | 20090400
           0 | 20090431
           0 | 20091301
           0 | 21000229
           0 | Date
           0 | abcd0101
           0 | yyyymmdd
           1 | 20040429
           1 | 20080229
```

```
(17 rows)


 DROP_TABLE
------------
t
(1 row)
```

Related Functions

- category Utilities - Checking

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:

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

Examples

```
\c inza
set PATH=inza.inza;
call inza..ST_CreateGeomColumn();
 ST_CREATEGEOMCOLUMN
------------
 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 exists. 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

ST_CreateSRS - Create a Custom Spatial Reference System

This stored procedure creates a Custom Spatial Reference System

Usage

The ST_CreateSRS stored procedure has the following syntax:

- **ST_CreateSRS(CHARACTER VARYING(512) description, CHARACTER VARYING(256) auth_name, INTEGER auth_srid, DOUBLE PRECISION falsex, DOUBLE PRECISION falsey, DOUBLE PRECISION xyunits, DOUBLE PRECISION falsez, DOUBLE PRECISION zunits, DOUBLE PRECISION falsem, DOUBLE PRECISION munits, CHARACTER VARYING(2048) srtext)**
  - Parameters
    - **description**

Description of the Custom Spatial Reference System.

Type: CHARACTER VARYING(512)

- **auth_name**
  Authority Spatial Reference System Name.

  Type: CHARACTER VARYING(256)
- **auth_srid**
  Authority Spatial Reference System ID.

  Type: INTEGER
- **falsex**
  X offset.

  Type: DOUBLE PRECISION
- **falsey**
  Y offset.

  Type: DOUBLE PRECISION
- **xyunits**
  XY scale.

  Type: DOUBLE PRECISION
- **falsez**
  Z offset.

  Type: DOUBLE PRECISION
- **zunits**
  Z scale.

  Type: DOUBLE PRECISION
- **falsem**
  M offset.

  Type: DOUBLE PRECISION
- **zunitm**
  M scale.

  Type: DOUBLE PRECISION
- **srtext**
  Spatial Reference System Text.

  Type: CHARACTER VARYING(2048)

▲ Returns
INTEGER srid of newly create custom spatial reference system. -1 if and error occurred.

Examples

```
set PATH=inza.inza;
```

```
call inza..st_dropsrs(inza..st_createsrs('', 'EPSG', 4326, -400,
-400, 1000000000, 0, 1, 0, 1,
'GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SP298.257223563]],PRIM
EM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]]'));
 ST_DROPSRS
------------
 t
(1 row)
```

Related Functions

- category Utilities - Actions

ST_DropSRS - Drop a Custom Spatial Reference System

This stored procedure drops a Custom Spatial Reference System

Usage

The ST_DropSRS stored procedure has the following syntax:

- **ST_DropSRS(INTEGER srid)**
  - Parameters
    - **srid**
      The SRID of the Custom Spatial Reference System to drop

      Type: INTEGER
  - Returns
    BOOLEAN true on success and false on failure.

Examples

```
set PATH=inza.inza;
call inza..st_dropsrs(inza..st_createsrs('', 'EPSG', 4326, -400,
-400, 1000000000, 0, 1, 0, 1,
'GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SP298.257223563]],PRIM
EM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]]'));
 ST_DROPSRS
------------
 t
(1 row)
```

Related Functions

- category Utilities - Actions

ST_MapPolygonsToGrid - Maps polygons to a grid

This stored procedure maps a table of geometries to grid cells.

Usage

The ST_MapPolygonsToGrid stored procedure has the following syntax:

- **ST_MapPolygonsToGrid(DOUBLE PRECISION GRIDSIZE, CHARACTER VARYING(ANY) SHAPETABLE, CHARACTER VARYING(ANY) SHAPECOLUMN, BOOLEAN DBG, INTEGER SHAPESRID, CHARACTER VARYING(ANY) SHAPEIDCOLUMN, BOOLEAN INCLUDE_ALL_COLUMNS)**
  - Parameters
    - **GRIDSIZE**
      The Grid size in degrees. Allowable values are 1, .1 and .01.

      Type: DOUBLE PRECISION
    - **SHAPETABLE**
      The table containing the polygons to map.

      Type: CHARACTER VARYING(ANY)
    - **SHAPECOLUMN**
      Geometry column name for input table.

      Type: CHARACTER VARYING(ANY)
    - **DBG**
      Enable/Disable debug messages. This will leave all intermediate tables if you want to debug.

      Type: BOOLEAN
    - **SHAPESRID**
      The SRID used when calling ST_SpatialGridIndex.

      Type: INTEGER
    - **SHAPEIDCOLUMN**
      Identifier column for input data..

      Type: CHARACTER VARYING(ANY)
    - **INCLUDE_ALL_COLUMNS**
      This parameter optionally allows users to include all columns in the input table. If FALSE, it will create an output table with the following columns only ID_COLUMN, GEOM_COLUMN, GRID_ID, GRID_GEOMETRY.

      Type: BOOLEAN

- Returns
  INTEGER 0 if success and 1 if error.

Details

This stored procedure maps the passed in table's geometries to the specified grid cell using the grid table specified (1, 0.1, 0.01). If the grid table doesn't already exist, it will be created. The output is a new table with one row for each input geometry/grid cell intersection.

Examples

```
set PATH=inza.inza;

create table POINTS (ID INTEGER, MY_GEOM ST_GEOMETRY(64000));

insert into POINTS VALUES (1, inza..ST_WKTToSQL('Point (1.1
1.1)'));

insert into POINTS VALUES (2, inza..ST_WKTToSQL('Point (2.2
2.2)'));

insert into POINTS VALUES (3, inza..ST_WKTToSQL('Point (3.3
3.3)'));

alter table points owner to inzauser;

CALL
inza..ST_MAPPOLYGONSTOGRID(1,'POINTS','MY_GEOM',false,4326,'ID',
true);

select ID, inza..ST_ASTEXT(MY_GEOM), ONEGID,
inza..ST_ASTEXT(ONEGRID_GEOM) from POINTS_ONE_GRIDMAP;

drop table POINTS_ONE_GRIDMAP;

drop table POINTS;

drop table GRID_ONE_GEOMETRIES;

delete from GEOMETRY_COLUMNS where
F_TABLE_NAME='POINTS_ONE_GRIDMAP';
 ST_MAPPOLYGONSTOGRID
----------------------
                     0
(1 row)
 ID |    ST_ASTEXT    | ONEGID |               ST_ASTEXT
----+-----------------+--------
+-------------------------------------
  2 | POINT (2.2 2.2) |  92182 | POLYGON ((2 2, 3 2, 3 3, 2 3,
2 2))
  3 | POINT (3.3 3.3) |  93183 | POLYGON ((3 3, 4 3, 4 4, 3 4,
3 3))
```

```
  1 | POINT (1.1 1.1) |  91181 | POLYGON ((1 1, 2 1, 2 2,
1 2, 1 1))

(3 rows)
```

Related Functions

- category Utilities - Actions

ST_SpatialGridIndex - Creates a spatial grid index

This stored procedure creates a grid table.

Usage

The ST_SpatialGridIndex stored procedure has the following syntax:

- **ST_SpatialGridIndex(DOUBLE PRECISION GRIDSIZE, BOOLEAN DBG, INTEGER SRID)**
  - Parameters
    - **GRIDSIZE**
      The Grid size in degrees. Allowable values are 1, .1 and .01.

      Type: DOUBLE PRECISION
    - **DBG**
      Enable/Disable debug messages. This will leave all intermediate tables if you want to debug.

      Type: BOOLEAN
    - **SRID**
      The SRID used to initialize the grid.

      Type: INTEGER
  - Returns
    INTEGER 0 if success and 1 if error.

Details

This stored procedure creates a grid table of either 1, .1 or .01 degrees.

Examples

```
set PATH=inza.inza;

call inza..ST_SPATIALGRIDINDEX(1,'FALSE',4326);

select count(*) from grid_one_geometries;

drop table grid_one_geometries;
```

```
call inza..ST_SPATIALGRIDINDEX(0.1,'FALSE',4326);
select count(*) from grid_point1_geometries;
drop table grid_point1_geometries;
 ST_SPATIALGRIDINDEX
---------------------
                    0


(1 row)
 COUNT
-------
 64800
(1 row)
 ST_SPATIALGRIDINDEX
---------------------
                    0


(1 row)
  COUNT
---------
 6480000
(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

This product may not be certified in your country for connection by any means whatsoever to interfaces of public telecommunications networks. Further certification may be required by law prior to making any such connection. Contact an IBM representative or reseller for any questions.

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.

CE Statement (Europe)

This product complies with the European Low Voltage Directive 73/23/EEC and EMC Directive 89/336/EEC as amended by European Directive 93/68/EEC.

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越すことがあります。この場合には使用者が適切な対策を講ずるう要求されることがあります。

Index

D

I

S