Alibaba Cloud ApsaraDB for MySQL

Spatio-temporal Database

Issue: 20190819

MORE THAN JUST CLOUD | Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. **You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.**
2. **No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.**
3. **The content of this document may be changed due to product version upgrades , adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.**
4. **This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults " and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity , applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use**

or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	Warning: Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click OK.
`Courier font`	It is used for commands.	Run the `cd / d C :/ windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*

Style	Description	Example
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich {stand | slave}`

Contents

Legal disclaimer..... I
Generic conventions..... I
1 Introduction..... 1
2 Models..... 2
2.1 Geometry model..... 2
2.2 Raster model..... 7
2.3 Path model..... 8
2.4 Point cloud model..... 11
2.5 Trajectory model..... 14
3 Advanced usage..... 18
3.1 Enable spatio-temporal parallel query..... 18
3.2 Enable GPU-accelerated computing..... 19
4 Raster SQL reference..... 22
4.1 Basic concepts..... 22
4.2 Raster creation..... 22
4.2.1 ST_CreateRast..... 23
4.3 Import and export..... 23
4.3.1 ST_ImportFrom..... 24
4.3.2 ST_ExportTo..... 24
4.4 Pyramid operations..... 25
4.4.1 ST_BuildPyramid..... 25
4.4.2 ST_BestPyramidLevel..... 26
4.5 Coordinate system conversion..... 26
4.5.1 ST_Rast2WorldCoord..... 27
4.5.2 ST_World2RastCoord..... 27
4.5.3 ST_ClipDimension..... 28
4.5.4 ST_Clip..... 28
4.5.5 ST_Update..... 29
4.5.6 ST_MosaicFrom..... 30
4.5.7 ST_MosaicTo..... 31
4.5.8 ST_CheckGPU..... 32
4.6 Overview operations..... 32
4.6.1 ST_BuildOverview..... 32
4.6.2 ST_UpdateOverview..... 33
4.6.3 ST_EraseOverview..... 34
4.7 Attribute query and update..... 34
4.7.1 ST_MetaData..... 35
4.7.2 ST_Width..... 35
4.7.3 ST_Height..... 36

4.7.4 ST_NumBands....36
4.7.5 ST_RasterID....37
4.7.6 ST_CellDepth....37
4.7.7 ST_CellType....38
4.7.8 ST_InterleavingType....38
4.7.9 ST_TopPyramidLevel....39
4.7.10 ST_Extent....39
4.7.11 ST_Srid....40
4.7.12 ST_Georeference....40
4.7.13 ST_IsGeoreferenced....41
4.7.14 ST_NoData....41
4.7.15 ST_ColorTable....42
4.7.16 ST_Statistics....43
4.7.17 ST_SummaryStats....44
4.7.18 ST_ColorInterp....44
4.7.19 ST_Histogram....45
4.7.20 ST_BuildHistogram....47
5 Point cloud SQL reference....48
5.1 Constructors....48
5.1.1 ST_makePoint....48
5.1.2 ST_makePatch....48
5.1.3 ST_Patch....49
5.2 Attribute functions....49
5.2.1 ST_asText....49
5.2.2 ST_pcID....50
5.2.3 ST_get....50
5.2.4 ST_numPoints....51
5.2.5 ST_summary....52
5.3 Object operations....52
5.3.1 ST_compress....52
5.3.2 ST_unCompress....53
5.3.3 ST_union....54
5.3.4 ST_explode....54
5.3.5 ST_patchAvg....55
5.3.6 ST_patchMax....56
5.3.7 ST_patchMin....56
5.3.8 ST_patchAvg....57
5.3.9 ST_patchMax....57
5.3.10 ST_patchMin....58
5.3.11 ST_filterGreaterThan....58
5.3.12 ST_filterLessThan....59
5.3.13 ST_filterEquals....60
5.3.14 ST_filterBetween....60
5.3.15 ST_pointN....61
5.3.16 ST_isSorted....62

5.3.17 ST_sort....62
5.3.18 ST_range....63
5.3.19 ST_setPcid....63
5.3.20 ST_transform....64
5.3.21 ST_envelopeGeometry....65
5.3.22 ST_boundingDiagonalGeometry....66
5.4 OGC WKB operations....66
5.4.1 ST_asBinary....67
5.4.2 ST_envelopeAsBinary....67
5.4.3 ST_boundingDiagonalAsBinary....68
5.5 Spatial relationship judgment....68
5.5.1 ST_intersects....68
5.6 Spatial processing....69
5.6.1 ST_intersection....69
6 Trajectory SQL reference....71
6.1 Basic concepts....71
6.2 Constructors....72
6.2.1 Constructor overview....72
6.2.2 ST_makeTrajectory....72
6.2.3 ST_append....77
6.3 Edit and process functions....78
6.3.1 ST_Compress....79
6.3.2 ST_CompressSED....84
6.3.3 ST_attrDeduplicate....85
6.3.4 ST_sort....86
6.3.5 ST_deviation....87
6.4 Attribute metadata....88
6.4.1 ST_attrDefinition....88
6.4.2 ST_attrSize....89
6.4.3 ST_attrName....89
6.4.4 ST_attrType....90
6.4.5 ST_attrLength....92
6.4.6 ST_attrNullable....93
6.5 Event functions....94
6.5.1 ST_addEvent....94
6.5.2 ST_eventTimes....95
6.5.3 ST_eventTime....96
6.5.4 ST_eventTypes....96
6.5.5 ST_eventType....97
6.6 Attribute functions....98
6.6.1 ST_startTime....98
6.6.2 ST_endTime....98
6.6.3 ST_trajectorySpatial....99
6.6.4 ST_trajectoryTemporal....99
6.6.5 ST_trajAttrs....100

6.6.6 ST_attrIntMax..... 100
6.6.7 ST_attrIntMin..... 101
6.6.8 ST_attrIntAverage..... 102
6.6.9 ST_attrFloatMax..... 103
6.6.10 ST_attrFloatMin..... 104
6.6.11 ST_attrFloatAverage..... 105
6.6.12 ST_leafType..... 106
6.6.13 ST_leafCount..... 106
6.6.14 ST_Duration..... 107
6.6.15 ST_timeAtPoint..... 107
6.6.16 ST_pointAtTime..... 108
6.6.17 ST_velocityAtTime..... 108
6.6.18 ST_accelerationAtTime..... 109
6.6.19 ST_timeToDistance..... 109
6.6.20 ST_timeAtDistance..... 110
6.6.21 ST_cumulativeDistanceAtTime..... 110
6.6.22 ST_timeAtCumulativeDistance..... 111
6.6.23 ST_subTrajectory..... 111
6.6.24 ST_subTrajectorySpatial..... 112
6.6.25 ST_samplingInterval..... 112
6.6.26 ST_trajAttrsAsText..... 113
6.6.27 ST_trajAttrsAsInteger..... 114
6.6.28 ST_trajAttrsAsDouble..... 115
6.6.29 ST_trajAttrsAsBool..... 116
6.6.30 ST_trajAttrsAsTimestamp..... 117
6.6.31 ST_attrIntFilter..... 117
6.6.32 ST_attrFloatFilter..... 120
6.6.33 ST_attrTimestampFilter..... 121
6.6.34 ST_attrNullFilter..... 122
6.6.35 ST_attrNotNullFilter..... 123
6.7 Spatial relationship judgment..... 124
6.7.1 ST_intersects..... 124
6.7.2 ST_equals..... 125
6.7.3 ST_distanceWithin..... 126
6.8 Spatial processing..... 126
6.8.1 ST_intersection..... 127
6.8.2 ST_difference..... 127
6.9 Spatial statistics..... 128
6.9.1 ST_nearestApproachPoint..... 128
6.9.2 ST_nearestApproachDistance..... 129
6.10 Spatio-temporal relationship judgment..... 129
6.10.1 ST_intersects..... 129
6.10.2 ST_equals..... 130
6.10.3 ST_distanceWithin..... 131
6.10.4 ST_durationWithin..... 131

6.11 Spatio-temporal processing....132
6.11.1 ST_intersection....132
6.12 Spatio-temporal statistics....133
6.12.1 ST_nearestApproachPoint....133
6.12.2 ST_nearestApproachDistance....134
6.13 Distance measurement....134
6.13.1 ST_length....135
6.13.2 ST_euclideanDistance....135
6.13.3 ST_mdistance....136
7 Trajectory best practices....137
8 Trajectory FAQ....139

1 Introduction

Overview

Spatio-temporal data is graphic and image data that contains both space and time information. Spatio-temporal data consists of multidimensional information about objects, such as the location, shape, change, and size distribution. As a spatio-temporal engine, ApsaraDB PostgreSQL Ganos (Ganos) provides a series of data types , functions, and stored procedures for ApsaraDB for RDS to efficiently store, index, query, analyze, and compute spatio-temporal data.

This document describes how to use Ganos to manage and analyze spatio-temporal data.

To seek technical support, you can contact a customer service representative at +86 57195187. You can also log on to the RDS console and choose More > Support > Open a new ticket in the top navigation bar. If your business is complex, you can purchase a support plan to obtain support from IM enterprise groups, technical account managers (TAMs), and service managers.

Pricing

Ganos is included in ApsaraDB for PostgreSQL and free of charge.

2 Models

2.1 Geometry model

Overview

Ganos Geometry is a spatial geometry extension of PostgreSQL. Ganos Geometry complies with OpenGIS specifications and enables PostgreSQL to store and manage 2D (X, Y), 3D (X, Y, Z), and 4D (X, Y, Z, M) spatial geometry data. In addition, Ganos Geometry provides a wide range of features such as spatial geometry objects, indexes , functions, and operators. The geometry model is fully compatible with PostGIS functions and allows you to smoothly migrate existing applications.

Quick start

- Create extensions.

```
-- Create a geometry extension .
Create extension ganos_geom etry cascade ;

-- Create a geometry topology extension .
Create extension ganos_geom etry_topol ogy ;

-- Create an SFCGAL plug - in extension .
Create extension ganos_geom etry_sfcga l ;
```

- Create a geometry table.

```
-- Method 1 : Create a table with a geometry field .
CREATE TABLE ROADS ( ID int4 , ROAD_NAME varchar ( 25 ),
geom geometry ( LINESTRING , 4326 ));

-- Method 2 : Create a common table and then add a
  geometry field .
CREATE TABLE ROADS ( ID int4 , ROAD_NAME varchar ( 25 ));
SELECT AddGeometr yColumn (' roads ', ' geom ', 4326 , '
LINESTRING ', 2 );
```

- Add geometry constraints.

```
ALTER TABLE ROADS ADD CONSTRAINT geometry_v alid_check
CHECK ( ST_IsValid ( geom ));
```

- Import geometry data.

```
INSERT INTO roads ( id , geom , road_name )
  VALUES ( 1 , ST_GeomFro mText (' LINESTRING ( 191232 243118 ,
191108 243242 )',- 1 ),' North Fifth - Ring Road ');
INSERT INTO roads ( id , geom , road_name )
```

```
  VALUES  ( 2 , ST_GeomFro  mText (' LINESTRING ( 189141   244158 ,
189265   244817 )',- 1 ),' East   Fifth - Ring   Road ');
INSERT   INTO   roads  ( id ,  geom ,  road_name )
  VALUES  ( 3 , ST_GeomFro  mText (' LINESTRING ( 192783   228138 ,
192612   229814 )',- 1 ),' South   Fifth - Ring   Road ');
INSERT   INTO   roads  ( id ,  geom ,  road_name )
  VALUES  ( 4 , ST_GeomFro  mText (' LINESTRING ( 189412   252431 ,
189631   259122 )',- 1 ),' West   Fifth - Ring   Road ');
INSERT   INTO   roads  ( id ,  geom ,  road_name )
  VALUES  ( 5 , ST_GeomFro  mText (' LINESTRING ( 190131   224148 ,
190871   228134 )',- 1 ),' East   Chang ' an   Avenue ');
INSERT   INTO   roads  ( id ,  geom ,  road_name )
  VALUES  ( 6 , ST_GeomFro  mText (' LINESTRING ( 198231   263418 ,
198213   268322 )',- 1 ),' West   Chang ' an   Avenue ');
```

- **Query geometry object information.**

```
SELECT   id ,  ST_AsText ( geom )  AS   geom ,  road_name   FROM
roads ;

--------------------------------
      id  |  geom                                        |  road_name
--------+----------------------------------------------+-----------
    1  |  LINESTRING ( 191232   243118 , 191108   243242 ) |  North
   Fifth - Ring   Road
    2  |  LINESTRING ( 189141   244158 , 189265   244817 ) |  East
Fifth - Ring   Road
    3  |  LINESTRING ( 192783   228138 , 192612   229814 ) |  South
   Fifth - Ring   Road
    4  |  LINESTRING ( 189412   252431 , 189631   259122 ) |  West
Fifth - Ring   Road
    5  |  LINESTRING ( 190131   224148 , 190871   228134 ) |  East
Chang ' an   Avenue
    6  |  LINESTRING ( 198231   263418 , 198213   268322 ) |  West
Chang ' an   Avenue
( 6   rows )
```

- **Create indexes.**

```
--  Create   a   GiST   index .
CREATE   INDEX  [ indexname ]  ON  [ tablename ]  USING   GIST  ([
geometryfi  eld ]);
CREATE   INDEX  [ indexname ]  ON  [ tablename ]  USING   GIST  ([
geometryfi  eld ]  gist_geome  try_ops_nd );
VACUUM   ANALYZE  [ table_name ] [( column_nam  e )];

--  Example
Create   INDEX   sp_geom_in  dex   ON   ROADS   USING   GIST ( geom
);
VACUUM   ANALYZE   ROADS  ( geom );

--  Create   a   Block   Range   Index  ( BRIN ).
CREATE   INDEX  [ indexname ]  ON  [ tablename ]  USING   BRIN  ([
geometryfi  eld ]);
CREATE   INDEX  [ indexname ]  ON  [ tablename ]  USING   BRIN  ([
geometryfi  eld ]  brin_geome  try_inclus  ion_ops_3d );
CREATE   INDEX  [ indexname ]  ON  [ tablename ]  USING   BRIN  ([
geometryfi  eld ]  brin_geome  try_inclus  ion_ops_4d );
--  Create   a   BRIN   index   with   a   specified   range .
```

```
CREATE INDEX [ indexname ] ON [ tablename ] USING BRIN ([
geometryfi eld ]) WITH ( pages_per_ range = [ number ]);
```

- **Access geometry objects.**

```
-- Determine whether a spatial geometry object consists
   of only simple elements .
  SELECT ST_IsSimpl e ( ST_GeomFro mText (' POLYGON (( 1 2 ,
 3 4 , 5 6 , 1 2 ))'));
  st_issimpl e
-------------
  t
( 1 row )

  SELECT ST_IsSimpl e ( ST_GeomFro mText (' LINESTRING ( 1 1 ,
 2 2 , 2 3 . 5 , 1 3 , 1 2 , 2 1 )'));
  st_issimpl e
-------------
  f
( 1 row )


-- Query the largest city with roundabout s in the
 terrain .
 SELECT gid , name , ST_Area ( the_geom ) AS area
 FROM bc_municip ality
 WHERE ST_NRings ( the_geom ) > 1
 ORDER BY area DESC LIMIT 1 ;

 gid | name | area
-----+-------------+-----------------
 12 | Anning | 257374619 . 430216
( 1 row )
```

- **Measure and analyze spatial data, and judge spatial relationships.**

```
-- Create the bc_roads table .
 Create table bc_roads ( gid serial , name varchar ,
 the_geom geometry );

-- Create the bc_municip ality table .
 Create table bc_municip ality ( gid serial , code integer
 , name varchar , the_geom geometry );

-- Compute the length .
 SELECT sum ( ST_Length ( the_geom ))/ 1000 AS km_roads FROM
   bc_roads ;

 km_roads
------------------
 70842 . 1243039643
( 1 row )

-- Compute the area .
 SELECT ST_Area ( the_geom )/ 10000 AS hectares FROM
 bc_municip ality WHERE name = ' PRINCE GEORGE ';

 hectares
------------------
 32657 . 9103824927
( 1 row )
-- Use the ST_Contain s function .
```

```
SELECT m . name , sum ( ST_Length ( r . the_geom ))/ 1000 as
roads_km
FROM
  bc_roads AS r , bc_municip ality AS m
WHERE
  ST_Contain s ( m . the_geom , r . the_geom )
GROUP BY m . name
ORDER BY roads_km ;

name                          | roads_km
------------------------------+------------------
SURREY                        | 1539 . 4755355124 2
VANCOUVER                     | 1450 . 3309348657 6
LANGLEY DISTRICT              | 833 . 7933925356 62
BURNABY                       | 773 . 7690914043 38
PRINCE GEORGE                 | 694 . 3755436914 7


-- Use the ST_Covers function .
SELECT ST_Covers ( smallc , smallc ) As smallinsma ll ,
 ST_Covers ( smallc , bigc ) As smallcover sbig ,
 ST_Covers ( bigc , ST_Exterio rRing ( bigc )) As bigcoverse
xterior ,
 ST_Contain s ( bigc , ST_Exterio rRing ( bigc )) As
bigcontain sexterior
FROM ( SELECT ST_Buffer ( ST_GeomFro mText (' POINT ( 1 2
)'), 10 ) As smallc ,
 ST_Buffer ( ST_GeomFro mText (' POINT ( 1 2 )'), 20 ) As
bigc ) As foo ;
-- Result
 smallinsma ll | smallcover sbig | bigcoverse xterior |
bigcontain sexterior
-------------+---------------+------------------
+--------------------
 t            | f             | t                 | f
( 1 row )

-- Use the ST_Disjoin t function .
SELECT ST_Disjoin t (' POINT ( 0 0 )':: geometry , '
LINESTRING ( 2 0 , 0 2 )':: geometry );
 st_disjoin t
--------------
 t
( 1 row )
SELECT ST_Disjoin t (' POINT ( 0 0 )':: geometry , '
LINESTRING ( 0 0 , 0 2 )':: geometry );
 st_disjoin t
--------------
 f
( 1 row )

-- Use the ST_Overlap s function .
SELECT ST_Overlap s ( a , b ) As a_overlap_ b ,
 ST_Crosses ( a , b ) As a_crosses_ b ,
  ST_Interse cts ( a , b ) As a_intersec ts_b , ST_Contain
s ( b , a ) As b_contains _a
FROM ( SELECT ST_GeomFro mText (' POINT ( 1 0 . 5 )') As a
, ST_GeomFro mText (' LINESTRING ( 1 0 , 1 1 , 3 5 )')
As b )
 As foo

a_overlap_ b | a_crosses_ b | a_intersec ts_b |
b_contains _a
-----------+------------+---------------+-------------
```

```
 f            |  f            |  t               |  t

--  Use   the   ST_Relate   function .
 SELECT   ST_Relate ( ST_Geometr  yFromText (' POINT ( 1   2 )'),
 ST_Buffer ( ST_Geometr  yFromText (' POINT ( 1   2 )'), 2 ), '
 0FFFFF212 ');
 St_relate
-----------
 t

--  Use   the   ST_Touches   function .
 SELECT   ST_Touches (' LINESTRING ( 0   0 ,  1   1 ,  0   2 )'::
 geometry , ' POINT ( 1   1 )':: geometry );
  st_touches
------------
  f
( 1   row )

 SELECT   ST_Touches (' LINESTRING ( 0   0 ,  1   1 ,  0   2 )'::
 geometry , ' POINT ( 0   2 )':: geometry );
  st_touches
------------
  t
( 1   row )

--  Use   the   ST_Within   function .
 SELECT   ST_Within ( smallc , smallc )  As   smallinsma  ll ,
  ST_Within ( smallc ,  bigc )  As   smallinbig ,
  ST_Within ( bigc , smallc )  As   biginsmall ,
  ST_Within ( ST_Union ( smallc ,  bigc ),  bigc )  as   unioninbig
 ,
  ST_Within ( bigc ,  ST_Union ( smallc ,  bigc ))  as   biginunion
 ,
  ST_Equals ( bigc ,  ST_Union ( smallc ,  bigc ))  as   bigisunion
 FROM
(
 SELECT   ST_Buffer ( ST_GeomFro  mText (' POINT ( 50   50 )'),  20
 )  As   smallc ,
  ST_Buffer ( ST_GeomFro  mText (' POINT ( 50   50 )'),  40 )  As
 bigc )  As   foo ;
--  Result
  smallinsma  ll  |  smallinbig  |  biginsmall  |  unioninbig  |
 biginunion  |  bigisunion
--------------+------------+------------+------------+------------
+------------
  t             |  t          |  f          |  t          |  t
       |  t
( 1   row )
```

- **Delete extensions.**

```
 Drop   extension   ganos_geom  etry   cascade ;
```

SQL reference

For more information, see the official PostGIS reference.

2.2 Raster model

Overview

Raster data consists of a matrix of cells (or pixels) organized into rows and columns (or a grid) where each cell contains a value that represents information, such as temperature.

Raster data can be digital aerial photographs, imagery from satellites, digital pictures, or even scanned maps.

By implementing the raster model in ApsaraDB for PostgreSQL, Ganos Raster can efficiently store and analyze raster data with the help of database technologies and methods.

Quick start

- Create an extension.

```
Create Extension Ganos_Rast er cascade ;
```

- Create a raster table.

```
Create Table raster_tab le ( id integer , raster_obj
raster );
```

- Import raster data from OSS.

```
Insert into raster_tab le Values ( 1 , ST_ImportF rom
(' chunk_tabl e ',' OSS :// ABCDEFG : 1234567890 @ oss - cn .
aliyuncs . com / mybucket / data / 4 . tif '))
```

- Query raster object information.

```
Select ST_Height ( raster_obj ), ST_Width ( raster_obj ) From
raster_tab le Where id = 1 ;
```

- Create a pyramid.

```
Update raster_tab le Set raster_obj = ST_BuildPy ramid (
raster_obj ) Where id = 1 ;
```

- Compute the best pyramid level based on the world space, width, and height of a viewport.

```
Select ST_BestPyr amidLevel ( raster_obj , '(( 128 . 0 , 30 . 0
),( 128 . 5 , 30 . 5 ))', 800 , 600 ) from raster_tab le
where id = 10 ;

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

```
3
```

- Obtain a pixel matrix from the specified space of a raster object.

```
Select ST_Clip ( raster_obj , 0 , '(( 128 . 980 , 30 . 0 ),( 129
. 0 , 30 . 2 ))', ' World ') From raster_tab le Where id
= 1 ;
```

- Compute the raster space of a clipped area.

```
Select ST_ClipDim ension ( raster_obj , 2 , '(( 128 . 0 , 30 .
0 ),( 128 . 5 , 30 . 5 ))') from raster_tab le where id
= 10 ;

-------------------------------------
'(( 600 , 720 ),( 200 , 300 ))'
```

- Use GPU-accelerated computing.

  If you purchase an ApsaraDB for RDS instance and have a GPU in the runtime environment, GPU-accelerated computing is automatically enabled. You do not need to set any parameters. For example, use the ST_BuildPyramid function that supports GPU-accelerated computing to create a pyramid.

```
-- Resampling during pyramid creation supports GPU -
accelerate d computing . The resampling type can be
set to Near , Average , Cubic , or Bilinear .
Update raster_tab le set rast = ST_BuildPy ramid ( rast
) where id = 1 ;
Update raster_tab le set rast = ST_BuildPy ramid ( rast
, 6 , ' Bilinear ') where id = 1 ;
Update raster_tab le set rast = ST_BuildPy ramid ( rast
, 6 , ' Bilinear ', ' chunk_tabl e ') where id = 1 ;
```

  For more information about GPU-accelerated computing, see #unique_7.

- Delete the extension.

```
Drop Extension Ganos_rast er cascade ;
```

SQL reference

For more information, see Raster SQL reference.

2.3 Path model

Overview

Path data is a geometric network diagram that consists of edges and nodes. It is used to build road and traffic networks.

Ganos Networking is an extension of PostgreSQL. Ganos Networking provides a series of functions and stored procedures to find the fastest, shortest, and optimal paths based on cost models. If the cost is time, the fastest path is the optimal path. If the cost is distance, the shortest path is the optimal path. The path model can be used for path planning on a road network, path search and planning in GPS navigation on an electronic map, and routing. The path model is fully compatible with pgRouting functions and allows you to smoothly migrate existing applications.

Quick start

- Create an extension.

```
Create Extension Ganos_Netw orking cascade ;
```

- Create a table.

```
CREATE TABLE edge_table (
    id BIGSERIAL ,
    dir character varying ,
    source BIGINT ,
    target BIGINT ,
    cost FLOAT ,
    reverse_co st FLOAT ,
    capacity BIGINT ,
    reverse_ca pacity BIGINT ,
    category_i d INTEGER ,
    reverse_ca tegory_id INTEGER ,
    x1 FLOAT ,
    y1 FLOAT ,
    x2 FLOAT ,
    y2 FLOAT ,
    the_geom geometry
);
```

- Insert records.

```
INSERT INTO edge_table (
    category_i d , reverse_ca tegory_id ,
    cost , reverse_co st ,
    capacity , reverse_ca pacity ,
    x1 , y1 ,
    x2 , y2 ) VALUES
( 3 , 1 , 1 , 1 , 80 , 130 , 2 , 0 , 2 , 1 ),
( 3 , 2 , - 1 , 1 , - 1 , 100 , 2 , 1 , 3 , 1 ),
( 2 , 1 , - 1 , 1 , - 1 , 130 , 3 , 1 , 4 , 1 ),
( 2 , 4 , 1 , 1 , 100 , 50 , 2 , 1 , 2 , 2 ),
( 1 , 4 , 1 , - 1 , 130 , - 1 , 3 , 1 , 3 , 2 ),
( 4 , 2 , 1 , 1 , 50 , 100 , 0 , 2 , 1 , 2 ),
( 4 , 1 , 1 , 1 , 50 , 130 , 1 , 2 , 2 , 2 ),
( 2 , 1 , 1 , 1 , 100 , 130 , 2 , 2 , 3 , 2 ),
( 1 , 3 , 1 , 1 , 130 , 80 , 3 , 2 , 4 , 2 ),
( 1 , 4 , 1 , 1 , 130 , 50 , 2 , 2 , 2 , 3 ),
( 1 , 2 , 1 , - 1 , 130 , - 1 , 3 , 2 , 3 , 3 ),
( 2 , 3 , 1 , - 1 , 100 , - 1 , 2 , 3 , 3 , 3 ),
( 2 , 4 , 1 , - 1 , 100 , - 1 , 3 , 3 , 4 , 3 ),
( 3 , 1 , 1 , 1 , 80 , 130 , 2 , 3 , 2 , 4 ),
```

```
( 3 , 4 , 1 , 1 , 80 , 50 , 4 , 2 , 4 , 3 ),
( 3 , 3 , 1 , 1 , 80 , 80 , 4 , 1 , 4 , 2 ),
( 1 , 2 , 1 , 1 , 130 , 100 , 0 . 5 , 3 . 5 , 1 .
 9999999999 99 , 3 . 5 ),
( 4 , 1 , 1 , 1 , 50 , 130 , 3 . 5 , 2 . 3 , 3 . 5 ,
 4 );
```

- **Update table attributes.**

```
 UPDATE edge_table SET the_geom = st_makelin e ( st_point (
 x1 , y1 ), st_point ( x2 , y2 )),
 dir = CASE WHEN ( cost > 0 AND reverse_co st > 0 ) THEN
 ' B '
           WHEN ( cost > 0 AND reverse_co st < 0 ) THEN '
 FT '
           WHEN ( cost < 0 AND reverse_co st > 0 ) THEN '
 TF '
           ELSE '' END ;
```

- **Create a topology.**

```
 SELECT pgr_create Topology (' edge_table ', 0 . 001 );
```

- **Query the shortest path.**

```
-- Use Dijkstra ' s algorithm to query the shortest
 path .
 SELECT * FROM pgr_dijkst ra (
    ' SELECT id , source , target , cost , reverse_co st
 FROM edge_table ',
     2 , 3
);

  seq | path_seq | node | edge | cost | agg_cost
-----+----------+------+------+------+----------
    1 |        1 |    2 |    4 |    1 |        0
    2 |        2 |    5 |    8 |    1 |        1
    3 |        3 |    6 |    9 |    1 |        2
    4 |        4 |    9 |   16 |    1 |        3
    5 |        5 |    4 |    3 |    1 |        4
    6 |        6 |    3 |  - 1 |    0 |        5
( 6 rows )

-- Use the A * algorithm to query the shortest path
 .
 SELECT * FROM pgr_astar (
    ' SELECT id , source , target , cost , reverse_co st ,
 x1 , y1 , x2 , y2 FROM edge_table ',
     2 , 12 ,
     directed := false , heuristic := 2 );

  seq | path_seq | node | edge | cost | agg_cost
-----+----------+------+------+------+----------
    1 |        1 |    2 |    2 |    1 |        0
    2 |        2 |    3 |    3 |    1 |        1
    3 |        3 |    4 |   16 |    1 |        2
    4 |        4 |    9 |   15 |    1 |        3
    5 |        5 |   12 |  - 1 |    0 |        4
( 5 rows )

-- Use the Turn Restricted Shortest Path ( TRSP )
 algorithm to query the shortest path .
```

```
SELECT * FROM pgr_trsp (
       ' SELECT id :: INTEGER , source :: INTEGER , target ::
INTEGER , cost FROM edge_table ',
       2 , 7 , false , false ,
       ' SELECT to_cost , target_id :: int4 ,
        from_edge || coalesce ('','' || via_path , '''') AS
via_path
        FROM restrictio ns '
   );

 seq | id1 | id2 | cost
-----+-----+-----+------
   0 |   2 |   4 |    1
   1 |   5 |  10 |    1
   2 |  10 |  12 |    1
   3 |  11 |  11 |    1
   4 |   6 |   8 |    1
   5 |   5 |   7 |    1
   6 |   8 |   6 |    1
   7 |   7 | - 1 |    0
( 8 rows )
```

- Delete the extension.

```
Drop Extension Ganos_Netw orking cascade ;
```

SQL reference

For more information, see the official pgRouting manual.

2.4 Point cloud model

Overview

Point cloud data consists of the points generated by a 3D scanner when it scans an object. Each point is represented by a set of 3D coordinates, and some may contain RGB or intensity information. Point cloud data contains spatial coordinate informatio n, and has a large number of points and complex attribute dimensions.

Ganos PointCloud is an extension of PostgreSQL. It enables PostgreSQL to store and manage point cloud data efficiently and fast. Ganos PointCloud provides features such as point cloud data compression, decompression, and attribute statistics collection. It can also work with Ganos Geometry to support a spatial analysis of point cloud data .

Point cloud data types

Point cloud data types include pcpoint and pcpatch. For the pcpoint type, each point is stored as a row. The dimensions of a point are defined in the point cloud schema. For the pcpatch type, points are stored as a collection. Point cloud data of the pcpatch

type can be compressed to reduce storage space and supports spatial queries. The compression method is specified by the compression parameter in the point cloud schema.

Point cloud schema

The pointcloud_formats table stores point cloud schemas. Each schema lists the attribute dimensions of a point cloud object and provides the information about each dimension, such as the size, type, name, and description.

Quick start

- Create extensions.

```
Create extension ganos_poin tcloud cascade ;
Create extension ganos_poin tcloud_geo metry cascade ;
```

- Insert a point cloud schema.

```
INSERT INTO pointcloud _formats ( pcid , srid , schema )
VALUES ( 1 , 4326 ,
'<? xml version =" 1 . 0 " encoding =" UTF - 8 "? >
< pc : PointCloud Schema xmlns : pc =" http :// pointcloud . org
/ schemas / PC / 1 . 1 "
    xmlns : xsi =" http :// www . w3 . org / 2001 / XMLSchema -
instance ">
 < pc : dimension >
   < pc : position > 1 </ pc : position >
   < pc : size > 4 </ pc : size >
   < pc : descriptio n > X coordinate as a long integer
. You must use the
                  scale and offset informatio n of the
  header to
                  determine the double value .</ pc :
descriptio n >
   < pc : name > X </ pc : name >
   < pc : interpreta tion > int32_t </ pc : interpreta tion >
   < pc : scale > 0 . 01 </ pc : scale >
 </ pc : dimension >
 < pc : dimension >
   < pc : position > 2 </ pc : position >
   < pc : size > 4 </ pc : size >
   < pc : descriptio n > Y coordinate as a long integer
. You must use the
                  scale and offset informatio n of the
  header to
                  determine the double value .</ pc :
descriptio n >
   < pc : name > Y </ pc : name >
   < pc : interpreta tion > int32_t </ pc : interpreta tion >
   < pc : scale > 0 . 01 </ pc : scale >
 </ pc : dimension >
 < pc : dimension >
   < pc : position > 3 </ pc : position >
   < pc : size > 4 </ pc : size >
   < pc : descriptio n > Z coordinate as a long integer
. You must use the
```

```
                    scale   and   offset   informatio  n   of   the
  header   to
                    determine   the   double   value .</ pc :
descriptio  n >
   < pc : name > Z </ pc : name >
   < pc : interpreta  tion > int32_t </ pc : interpreta  tion >
   < pc : scale > 0 . 01 </ pc : scale >
 </ pc : dimension >
 < pc : dimension >
   < pc : position > 4 </ pc : position >
   < pc : size > 2 </ pc : size >
   < pc : descriptio  n > The   intensity   value   is   the
integer   representa  tion
                    of   the   pulse   return   magnitude .  This
value   is   optional
                    and   system - specific .  However ,  it
should   always   be
                    included   if   available .</ pc : descriptio
n >
   < pc : name > Intensity </ pc : name >
   < pc : interpreta  tion > uint16_t </ pc : interpreta  tion >
   < pc : scale > 1 </ pc : scale >
 </ pc : dimension >
 < pc : metadata >
   < Metadata   name =" compressio  n "> dimensiona  l </ Metadata
>
 </ pc : metadata >
</ pc : PointCloud  Schema >');
```

- **Create a point cloud table.**

```
--  Use   the   pcpoint   data   type .
 CREATE   TABLE   points  (
     id   SERIAL   PRIMARY   KEY ,
     pt   PCPOINT ( 1 )
);

--  Use   the   pcpatch   data   type .
 CREATE   TABLE   patches  (
     id   SERIAL   PRIMARY   KEY ,
     pa   PCPATCH ( 1 )
);
```

- **Insert pcpoint data.**

```
 INSERT   INTO   points  ( pt )
 SELECT   ST_MakePoi  nt ( 1 ,  ARRAY [ x , y , z , intensity ])
 FROM  (
   SELECT
  - 127 + a / 100 . 0   AS   x ,
     45 + a / 100 . 0   AS   y ,
          1 . 0 * a   AS   z ,
           a / 10   AS   intensity
   FROM   generate_s  eries ( 1 , 100 )  AS   a
)  AS   values ;

 SELECT   ST_MakePoi  nt ( 1 ,  ARRAY [- 127 ,  45 ,  124 . 0 ,  4 .
 0 ]);
--------------------------
 0101000000  64CEFFFF94  1100007030  00000400

 SELECT   ST_AsText (' 0101000000  64CEFFFF94  1100007030  00000400
 ':: pcpoint );
```

```
-------------------------
{" pcid ": 1 ," pt ":[- 127 , 45 , 124 , 4 ]}
```

- Insert pcpatch data.

```
 INSERT   INTO   patches  ( pa )
 SELECT   ST_Patch ( pt )  FROM   points   GROUP   BY   id / 10 ;

 SELECT   ST_AsText ( ST_MakePat  ch ( 1 ,  ARRAY [- 126 . 99 , 45 .
 01 , 1 , 0 , - 126 . 98 , 45 . 02 , 2 , 0 , - 126 . 97 , 45 . 03 ,
 3 , 0 ]));
-------------------------
{" pcid ": 1 ," pts ":[
 [- 126 . 99 , 45 . 01 , 1 , 0 ],[- 126 . 98 , 45 . 02 , 2 , 0 ],[-
 126 . 97 , 45 . 03 , 3 , 0 ]
]}
```

- Compute the average values of all attribute dimensions in a pcpatch object.

```
 SELECT   ST_AsText ( ST_PatchAv  g ( pa ))  FROM   patches   WHERE
   id  =  7 ;
-------------------------
{" pcid ": 1 ," pt ":[- 126 . 46 , 45 . 54 , 54 . 5 , 5 ]}
```

- Delete extensions.

```
 Drop   extension   ganos_poin  tcloud_geo  metry ;
 Drop   extension   ganos_poin  tcloud   cascade ;
```

SQL reference

For more information, see Point cloud SQL reference.

2.5 Trajectory model

Overview

Trajectory data records the continuous location change information about a moving object, such as a vehicle or a person. Trajectory data is typical spatio-temporal data. You can analyze and understand trajectory data to study many important issues.

Ganos Trajectory is an extension of PostgreSQL. Ganos Trajectory provides a series of data types, functions, and stored procedures to help you efficiently manage, query, and analyze spatio-temporal trajectory data.

Important notes

Ganos Trajectory 1.6 is incompatible with Ganos Trajectory 1.0. To upgrade Ganos Trajectory from version 1.0 to 1.6, you need to contact Alibaba Cloud technical support personnel.

Quick start

- Create an extension.

```
Create Extension Ganos_traj ectory cascade ;
```

- Create the enumeration type for a spatial geometry object.

```
CREATE TYPE leaftype AS ENUM (' STPOINT ', ' STPOLYGON ');
```

- Create a trajectory table.

```
Create Table traj_table ( id integer , traj trajectory );
```

- Insert trajectory data.

```
insert into traj_table values ( 1 , ST_MakeTra jectory ('
STPOINT ':: leaftype , st_geomfro mtext (' LINESTRING ( 114
35 , 115 36 , 116 37 )', 4326 ), '[ 2010 - 01 - 01 14 :
30 , 2010 - 01 - 01 15 : 30 )':: tsrange , '{" leafcount ": 3
," attributes ": {" velocity ": {" type ":" integer "," length ":
4 ," nullable ": false ," value ":[ 120 , 130 , 140 ]}," accuracy
":{" type ":" integer "," length ": 4 ," nullable ": false ," value
":[ 120 , 130 , 140 ]}," bearing ":{" type ":" float "," length
": 4 ," nullable ": false ," value ":[ 120 , 130 , 140 ]},"
accelerati on ":{" type ":" float "," length ": 4 ," nullable ":
false ," value ":[ 120 , 130 , 140 ]}}}')), ( 2 , ST_MakeTra
jectory (' STPOINT ':: leaftype , st_geomfro mtext (' LINESTRING
( 114 35 , 115 36 , 116 37 )', 4326 ), ' 2010 - 01 - 01
14 : 30 ':: timestamp , ' 2010 - 01 - 01 15 : 30 ':: timestamp
, '{" leafcount ": 3 ," attributes ": {" velocity ": {" type ":"
integer "," length ": 4 ," nullable ": false ," value ":[ 120 ,
130 , 140 ]}," accuracy ":{" type ":" integer "," length ": 4
," nullable ": false ," value ":[ 120 , 130 , 140 ]}," bearing
":{" type ":" float "," length ": 4 ," nullable ": false ," value
":[ 120 , 130 , 140 ]}," accelerati on ":{" type ":" float ","
length ": 4 ," nullable ": false ," value ":[ 120 , 130 , 140
]}}}')), ( 3 , ST_MakeTra jectory (' STPOINT ':: leaftype ,
st_geomfro mtext (' LINESTRING ( 114 35 , 115 36 , 116
37 )', 4326 ), ARRAY [' 2010 - 01 - 01 14 : 30 ':: timestamp , '
2010 - 01 - 01 15 : 00 ':: timestamp , ' 2010 - 01 - 01 15 : 30
':: timestamp ], '{" leafcount ": 3 ," attributes ": {" velocity
": {" type ":" integer "," length ": 4 ," nullable ": false ,"
value ":[ 120 , 130 , 140 ]}," accuracy ":{" type ":" integer
"," length ": 4 ," nullable ": false ," value ":[ 120 , 130 ,
140 ]}," bearing ":{" type ":" float "," length ": 4 ," nullable
": false ," value ":[ 120 , 130 , 140 ]}," accelerati on ":{"
type ":" float "," length ": 4 ," nullable ": false ," value ":[
120 , 130 , 140 ]}}}')), ( 4 , ST_MakeTra jectory (' STPOINT
':: leaftype , st_geomfro mtext (' LINESTRING ( 114 35 , 115
36 , 116 37 )', 4326 ), '[ 2010 - 01 - 01 14 : 30 , 2010 -
01 - 01 15 : 30 )':: tsrange , null ));
```

- Create a spatial index for the trajectory table.

```
-- Create a function - based spatial index to
accelerate the filtering of spatial data .
create index tr_spatial _geometry_ index on trajtab
using gist ( st_traject oryspatial ( traj ));
```

```
-- Accelerate the filtering of spatial data for a
 spatial data query .
 select id , traj_id from traj_test where st_interse
 cts ( st_traject oryspatial ( traj ), ST_GeomFro mText ('
 POLYGON (( 116 . 4674785180 5917 39 . 9231796415 5052 , 116 .
 4986540687 358 39 . 9231796415 5052 , 116 . 4986540687 358
 39 . 9445240171 1516 , 116 . 4674785180 5917 39 . 9445240171
 1516 , 116 . 4674785180 5917 39 . 9231796415 5052 ))')) =
 true and st_interse cts ( st_traject oryspatial ( traj
 ), ST_GeomFro mText (' POLYGON (( 116 . 5172424485 498 39
 . 9049847328 32744 , 116 . 5543342491 403 39 . 9049847328
 32744 , 116 . 5543342491 403 39 . 9329491808 2651 , 116 .
 5172424485 498 39 . 9329491808 2651 , 116 . 5172424485 498
 39 . 9049847328 32744 ))')) = true ;
```

- **Create temporal indexes for the trajectory table.**

```
-- Create a function - based temporal index on the
 time range to accelerate the filtering of time .
 create index tr_timespa n_time_ind ex on trajtab using
   gist ( st_timespa n ( traj ));

-- Create function - based indexes on the start time
 and end time of a trajectory object .
 create index tr_startti me_index on trajtab using
 btree ( st_startti me ( traj ));
 create index tr_endtime _index on trajtab using btree
 ( st_endtime ( traj ));

-- Accelerate the filtering of time for a query .
 select id , traj_id from traj_split where st_startti
 me ( traj ) > ' 2008 - 02 - 02 13 : 30 : 44 ':: timestamp and
 st_endtime ( traj ) < ' 2008 - 02 - 03 17 : 30 : 44 ':: timestamp
 ;
```

- **Create a spatio-temporal composite index (btree_gist) for the trajectory table.**

  btree_gist can be used to create a spatio-temporal composite index that is applicable to trajectory data queries that involve the filtering of both time and spatial data. A spatio-temporal composite index simplifies SQL statements for spatial-temporal data queries and improves query efficiency.

```
-- Use btree_gist to create a spatio - temporal
 composite index on the start time , end time , and
 spatial data of a trajectory object .
 create index tr_traj_te st_stm_etm _sp_index on traj_test
   using gist ( st_startti me ( traj ), st_endtime ( traj ),
 st_traject oryspatial ( traj ));

-- Query spatio - temporal data .
 select id , traj_id from traj_test where st_startti
 me ( traj ) > ' 2008 - 02 - 02 13 : 30 : 44 ':: timestamp
 and st_endtime ( traj ) < ' 2008 - 02 - 03 17 : 30 : 44 ':
 timestamp and st_interse cts ( st_traject oryspatial ( traj
 ), ST_GeomFro mText (' POLYGON (( 116 . 4674785180 5917 39
 . 9231796415 5052 , 116 . 4986540687 358 39 . 9231796415
 5052 , 116 . 4986540687 358 39 . 9445240171 1516 , 116 .
 4674785180 5917 39 . 9445240171 1516 , 116 . 4674785180
 5917 39 . 9231796415 5052 ))')) = true and st_interse
 cts ( st_traject oryspatial ( traj ), ST_GeomFro mText ('
```

```
 POLYGON (( 116 . 5172424485  498   39 . 9049847328  32744 , 116 .
 5543342491  403   39 . 9049847328  32744 , 116 . 5543342491  403
   39 . 9329491808  2651 , 116 . 5172424485  498   39 . 9329491808
 2651 , 116 . 5172424485  498   39 . 9049847328  32744 ))')) =  true
 ;
```

- Query the start time and end time of trajectory objects.

```
 select   st_startTi  me ( traj ),  st_endTime ( traj )  from
 traj_table ;
     st_startti  me      |      st_endtime
---------------------+---------------------
  2010 - 01 - 01   14 : 30 : 00  |  2010 - 01 - 01   15 : 30 : 00
  2010 - 01 - 01   14 : 30 : 00  |  2010 - 01 - 01   15 : 30 : 00
  2010 - 01 - 01   14 : 30 : 00  |  2010 - 01 - 01   15 : 30 : 00
  2010 - 01 - 01   14 : 30 : 00  |  2010 - 01 - 01   15 : 30 : 00
  2010 - 01 - 01   14 : 30 : 00  |  2010 - 01 - 01   15 : 30 : 00
  2010 - 01 - 01   11 : 30 : 00  |  2010 - 01 - 01   15 : 00 : 00
  2010 - 01 - 01   11 : 30 : 00  |  2010 - 01 - 01   15 : 00 : 00
  2010 - 01 - 01   11 : 30 : 00  |  2010 - 01 - 01   15 : 00 : 00
( 8   rows )
```

- Query trajectory object information.

```
--  Use   an   interpolat  ion   function   to   query   the
 attributes   of   trajectory   points .
 Select   ST_velocit  yAtTime ( traj , ' 2010 - 01 - 01   12 : 45
 ')  from   traj_table   where   id  >  5 ;
 st_velocit  yattime
-------------------
 5
 5
 4 . 1666666666  6667
( 3   rows )
```

- Analyze the proximity between trajectory objects.

```
 postgres =#  Select   ST_euclide  anDistance (( Select   traj
 From   traj_table   Where   id  =  6 ), ( Select   traj   From
 traj_table   Where   id  =  7 ));
  st_euclide  andistance
----------------------
    0 . 0334968923  954815
( 1   row )
```

- Delete the extension.

```
 Drop   Extension   Ganos_traj  ectory   cascade ;
```

SQL reference

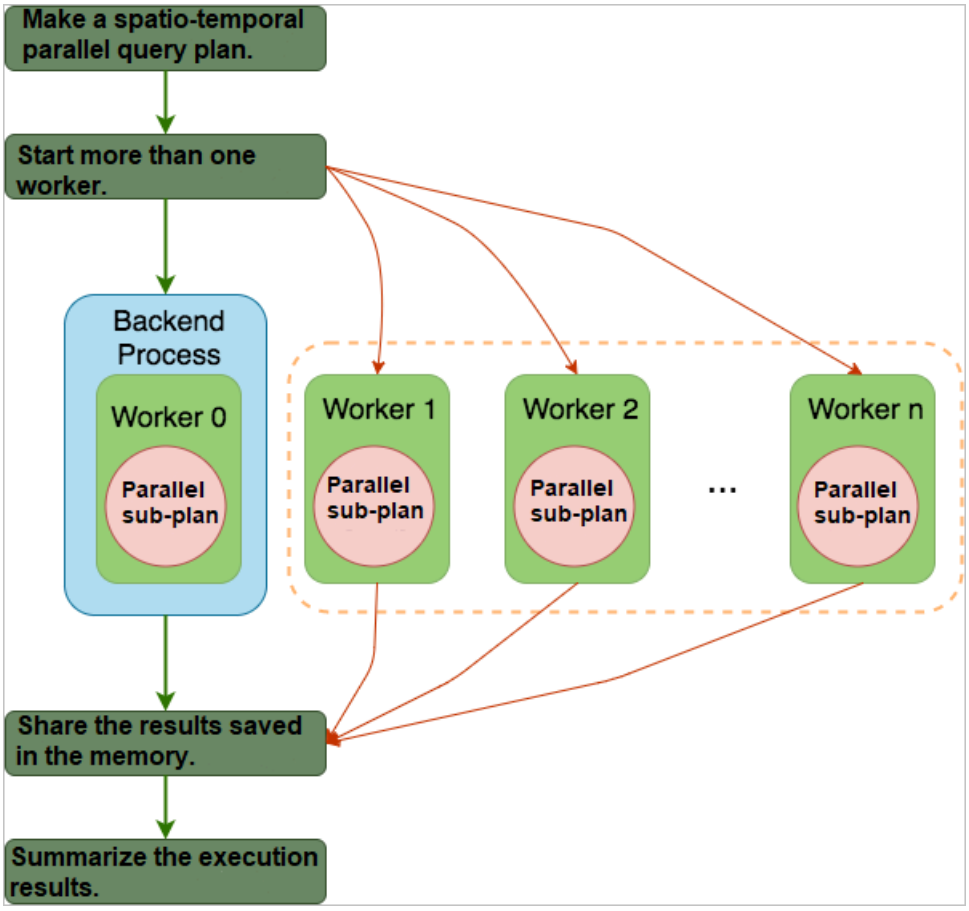
- For more information, see Trajectory SQL reference.

3 Advanced usage

3.1 Enable spatio-temporal parallel query

Parallel query principles

Ganos can use the table-level parallel query capability of PostgreSQL to accelerate complex spatio-temporal data queries that involve a large amount of data. The following figure shows the PostgreSQL parallel query process.



Precautions

- A larger number of workers used for parallel query means a higher CPU load. If the CPU load is already high in some scenarios, we recommend that you set the number of workers to 2, that is, set the value of max_parallel_workers_per_gather to 2.
- When enabling parallel query for highly concurrent access requests on a server with limited memory, you need to properly set the work_mem parameter (

minimum value: 64 KB). You must ensure that the number of concurrent access requests multiplied by the number of parallel workers multiplied by the value of the work_mem parameter does not exceed 60% of the server memory.

Procedure

To enable Ganos parallel query, perform the following operations:

1. Set parallel query parameters in the PostgreSQL configuration file postgresql.conf.
   - Set the max_parallel_workers parameter to specify the total number of parallel workers that can be enabled, in the range of 8 to 32. The value of this parameter must be smaller than that of the max_worker_processes parameter.
   - Set the max_parallel_workers_per_gather parameter to specify the maximum number of parallel workers for a single query gather, in the range of 2 to 4. The value of this parameter must be smaller than that of the max_parallel_workers parameter.
   - To forcibly enable parallel query, set the force_parallel_mode parameter to on.
   - To set the number of parallel workers for a single table, execute the following SQL statement: alter table table_name set (parallel_workers=n).

     Note:
     In the preceding SQL statement, n indicates the number of parallel workers. For more information about how to set an appropriate value for n, see the description of the max_parallel_workers_per_gather parameter.
2. Increase the cost of Ganos functions.

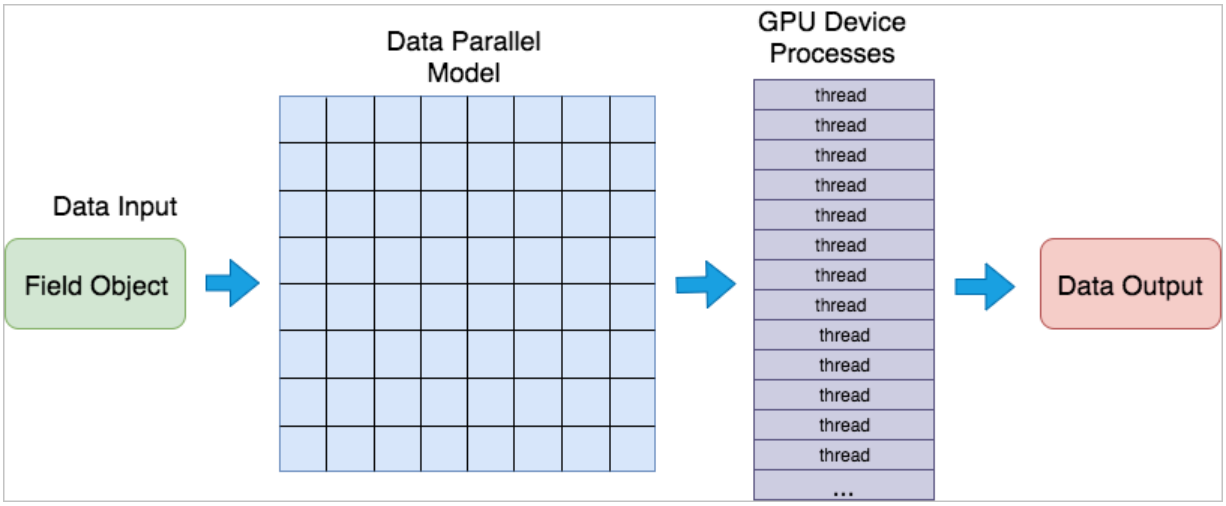
   After a Ganos module extension is created, functions have a default cost. If the data size of the table for the module extension is small, parallel query is disabled by default. If a function is computing-intensive and parallel query is suitable for the function, you must increase the cost of the function before you can enable parallel query.

3.2 Enable GPU-accelerated computing

Acceleration principles

Due to its special hardware architecture, a GPU has great advantages over a CPU in processing computing-intensive and easy-to-parallel code. In a database, GPU parallel

acceleration is performed at the object level. Ganos converts a single field object to a model suitable for parallel computing, and uses the multiple cores of the GPU for parallel computing. The following figure shows the parallel computing process.



Precautions

A single GPU has limited resources. Therefore, we recommend that you disable GPU-accelerated computing in a session for high-concurrency scenarios.

Procedure

When detecting a GPU, Ganos automatically enables GPU-accelerated computing. You do not need to set any parameters. GPU-accelerated computing is transparent and imperceptible to you. In addition, Ganos allows you to enable or disable GPU-accelerated computing at the session level.

1. Check whether a GPU is available in the current environment.
   a. Execute the create extension ganos_raster cascade statement to create a Ganos Raster extension.
   b. Execute the select st_checkgpu() statement.
      - If Ganos detects a GPU in the current environment, it returns the GPU information.

```
rasterdb =#  select   st_checkgp  u ();
                   st_checkgp  u

-------------------------------------------------------------------------
[ GPU ( 0 )  prop ] multiProce  ssorCount = 20 ;  sharedMemP
erBlock = 49152 ;  totalGloba  lMem =- 608239616 ;
maxThreads  PerBlock = 1024 ;  maxThreads  PerMultiPr  ocessor
= 2048 ;  cudaThread  GetLimit = 1024 .
```

```
( 1   row )
```

- If Ganos does not detect a GPU in the current environment, it returns an error message.

```
 rasterdb =#  select   st_checkgp  u ();
                   st_checkgp  u

-------------------------------------------------------------------------
 There   is   not   gpu   device   on   current   environmen  t
 ,  cuda_error  code = 35 ,  errormsg = CUDA   driver   version
   is   insufficie  nt   for   CUDA   runtime   version .
( 1   row )
```

Note:

GPU-accelerated computing can be enabled only in an environment where a GPU is available.

2. Enable or disable GPU-accelerated computing at the session level.

   - In an environment with a GPU, Ganos enables GPU-accelerated computing by default. To disable GPU-accelerated computing to use the original CPU computing mode, perform the following operation in a session:

     Execute the set ganos.raster.use_cuda=off statement.

```
 rasterdb =#  set   ganos . raster . use_cuda = off ;
 SET
 rasterdb =#  show   ganos . raster . use_cuda ;
  ganos . raster . use_cuda
----------------------
  off
( 1   row )
```

   - To enable GPU-accelerated computing again, execute the set ganos.raster.use_cuda=on statement.

```
 rasterdb =#  set   ganos . raster . use_cuda = on ;
 SET
 rasterdb =#  show   ganos . raster . use_cuda ;
  ganos . raster . use_cuda
----------------------
  on
( 1   row )
```

3. Implement GPU-accelerated computing in a Ganos module.

Note:

Currently, you can implement GPU-accelerated computing only in the Ganos Raster module. GPU-accelerated computing will be supported by the Ganos Trajectory and Ganos Geometry modules in the future.

4 Raster SQL reference

4.1 Basic concepts

Concept	Description
raster object	The regular grid that represents a space. Each cell in a raster object is assigned an attribute value. A raster object can be a satellite image, a digital elevation model (DEM), or a picture.
raster cell	The cell in a raster object, which is also called a pixel.
band	The layer of cell value matrix in a raster object. A raster object can have multiple bands.
chunk	The portion of a raster object. The size of a chunk can be customized, such as 256 × 256 × 3.
pyramid	The downsampled version of a raster object. A pyramid can contain multiple downsample layers. Consecutive pyramid layers are downsampled at a scale of 2:1.
pyramid level	The level in a pyramid.
mosaic	The operation to integrate multiple raster objects into an existing raster object.
interleaving	The arrangement of pixels in a raster object. The interleaving types include band sequential (BSQ), band interleaved by pixel (BIP), and band interleaved by line (BIL).
world space	The world coordinates (geographic coordinates) of a raster object.
raster space	The pixel coordinates of a raster object.

4.2 Raster creation

4.2.1 ST_CreateRast

This function creates a raster object based on Alibaba Cloud OSS.

Syntax

```
raster ST_CreateR ast ( cstring url );
```

Parameters

Parameter	Description
url	The path of OSS raster files.

Description

The path of OSS files is in the following format: oss://access_id:secrect_key@endpoint/path_to/file. The parameter *endpoint* can be omitted, and the system will automatically find the corresponding endpoint. If the endpoint is omitted, the path must start with the forward slash (/).

The endpoint is the address used to access OSS from the intranet. To ensure that data is imported at the fastest speed, make sure that the ApsaraDB RDS for PostgreSQL instance is in the same region as the OSS bucket. For more information, see #unique_21.

Examples

```
-- Specify the AccessKey ID , AccessKey Secret , and
endpoint .
Select ST_CreateR ast (' OSS :// ABCDEFG : 1234567890 @ oss - cn .
aliyuncs . com / mybucket / data / 4 . tif ');

-- Specify the AccessKey ID and AccessKey Secret .
Select ST_CreateR ast (' OSS :// ABCDEFG : 1234567890 @/ mybucket
/ data / 4 . tif ');

-- Specify the URL in https format . AccessKey - based
access has better performanc e than URL - based access .
Select ST_CreateR ast (' https :// mybuckets . oss - cn . aliyuncs
. com / data / 4 . tif ');
```

4.3 Import and export

4.3.1 ST_ImportFrom

This function imports an OSS object into a raster object in ApsaraDB for PostgreSQL that uses Ganos.

Syntax

```
raster ST_ImportF rom ( cstring chunkTable Name , cstring
url );
```

Parameters

Parameter	Description
chunkTable Name	The chunk table name, which must comply with the table naming rules of ApsaraDB for PostgreSQL.
u rl	The URL of the external OSS object. For more information, see the description of the url parameter in #unique_24.

Description

This function creates a raster object and imports an external OSS object into the raster object.

Examples

```
Select ST_ImportF rom (' chunk_tabl e ',' OSS :// ABCDEFG :
1234567890 @ oss - cn . aliyuncs . com / mybucket / data / 4 . tif
');
```

4.3.2 ST_ExportTo

This function exports a raster object as an OSS object.

Syntax

```
boolean ST_ExportT o ( raster source , cstring format ,
cstring url , integer level = 0 );
```

Parameters

Parameter	Description
source	The raster object.
format	The format of the exported data, such as GTiff or BMP.
url	The URL of the exported OSS object. For more information, see the description of the url parameter in #unique_24.
level	The pyramid level.

Description

If the raster object is successfully exported, the function returns true. If the raster object fails to be exported, the function returns false.

The format parameter specifies the format of the exported data. The following table lists the common formats.

Format	Full name
BMP	Microsoft Windows Device Independent Bitmap (.bmp)
ECW	ERDAS Compressed Wavelets (.ecw)
EHdr	ESRI .hdr Labelled
GIF	Graphics Interchange Format (.gif)
GPKG	GeoPackage
GTiff	Tagged Image File Format (TIFF), BigTIFF, or GeoTIFF (.tif)
HDF4	Hierarchical Data Format Release 4
PDF	Geospatial PDF
PNG	Portable Network Graphics (.png)

Examples

```
Select   ST_ExportT  o ( raster , ' GTiff ', ' OSS :// ABCDEFG :
1234567890 @ oss - cn . aliyuncs . com / mybucket / data / 4 . tif
')  from   raster_tab  le   where   id = 1 ;
```

4.4 Pyramid operations

4.4.1 ST_BuildPyramid

This function creates a pyramid for a raster object.

Syntax

```
raster   ST_BuildPy  ramid ( raster   source );
raster   ST_BuildPy  ramid ( raster   source ,  cstring   chunkTable
Name );
```

Parameters

Parameter	Description
source	The raster object.

Parameter	Description
chunkTable Name	The name of the chunk table for storing the pyramid.

Examples

```
Update raster_tab le set raster_obj = ST_BuildPy ramid (
raster_obj ) where id = 1 ;

Update raster_tab le set raster_obj = ST_BuildPy ramid (
raster_obj , ' chunk_tabl e ') where id = 2 ;
```

4.4.2 ST_BestPyramidLevel

This function computes the best pyramid level based on the world space, width, and height of a viewport.

Syntax

```
integer ST_BestPyr amidLevel ( raster raster_obj , Box extent
, integer width , integer height );
```

Parameters

Parameter	Description
raster_obj	The raster object.
box	The world space of the viewport.
width	The width of the viewport, in pixels.
height	The height of the viewport, in pixels.

Description

The raster object must have a valid spatial reference system identifier (SRID).

Examples

```
Select ST_BestPyr amidLevel ( raster_obj , '(( 128 . 0 , 30 .
0 ),( 128 . 5 , 30 . 5 ))', 800 , 600 ) from raster_tab le
where id = 10 ;

---------------------
3
```

4.5 Coordinate system conversion

4.5.1 ST_Rast2WorldCoord

This function computes the world coordinates of a cell by using an affine transformation formula based on the pixel coordinates and pyramid level of the cell.

Syntax

```
point   ST_Rast2Wo  rldCoord ( raster   raster_obj ,  integer
pyramidLev  el ,  integer   row ,  integer   column );
```

Parameters

Parameter	Description
raster_obj	The raster object.
pyramidLev el	The pyramid level.
row	The row number.
column	The column number.

Description

The raster object must have a valid spatial reference system identifier (SRID).

Examples

```
Select   ST_Rast2Wo  rldCoord ( raster_obj ,  0 ,  0 ,  0 )  from
raster_tab  le ;
```

4.5.2 ST_World2RastCoord

This function computes the pixel coordinates of a cell by using an inverse affine transformation formula based on the world coordinates and pyramid level of the cell.

Syntax

```
point   ST_World2R  astCoord ( raster   raster_obj ,  integer
pyramidLev  el ,  point   coord );
```

Parameters

Parameter	Description
raster_obj	The raster object.
pyramidLev el	The pyramid level.
coord	The world space.

Description

The raster object must have a valid spatial reference system identifier (SRID).

Examples

```
Select   ST_World2R  astCoord ( raster_obj ,  0 , '( 27 . 9 , 128 . 6
)')  from   raster_tab  le ;
```

4.5.3 ST_ClipDimension

This function computes the raster space of a clipped area.

Syntax

```
box   ST_ClipDim  ension ( raster   raster_obj ,  integer
pyramidLev  el ,  box   extent );
```

Parameters

Parameter	Description
raster_obj	The raster object.
pyramidLev el	The pyramid level.
box	The world space of the clipped area.

Description

The raster object must have a valid spatial reference system identifier (SRID).

Examples

```
Select   ST_ClipDim  ension ( raster_obj ,  2 , '(( 128 . 0 ,  30 . 0
),( 128 . 5 ,  30 . 5 ))')  from   raster_tab  le   where   id  =  10
;

-----------------------------------
'(( 200 ,  300 ),( 600 ,  720 ))'
```

4.5.4 ST_Clip

This function clips a raster object.

Syntax

```
bytea   ST_Clip ( raster   raster_obj , integer   pyramidLev  el ,
box   extent ,  BoxType   boxType );
```

```
bytea   ST_Clip ( raster   raster_obj , integer   pyramidLev  el ,
box   extent ,  BoxType   boxType ,  integer   destSrid );
```

Parameters

Parameter	Description
raster_obj	The raster object.
pyramidLev el	The pyramid level.
extent	The area to be clipped, in the format of '((minX,minY),(maxX, maxY))'.
boxType	The coordinate type of the area to be clipped. The value must be Raster or World. Valid values: · Raster: indicates pixel coordinates. · World: indicates world coordinates.
destSrid	The spatial reference system identifier (SRID) of the output cell subset.

Description

This function returns a cell matrix of the input window size. The size cannot exceed 100 MB.

Examples

```
Select   ST_Clip ( raster_obj ,  0 , '(( 128 . 980 , 30 . 0 ),( 129 .
0 , 30 . 2 ))', ' World ');
Select   ST_Clip ( raster_obj ,  0 , '(( 128 . 980 , 30 . 0 ),( 129 .
0 , 30 . 2 ))', ' World ',  4326 );
```

4.5.5 ST_Update

This function uses a source raster object to update a destination raster object.

Syntax

```
raster   ST_Update ( raster   source ,  raster   dest );
```

Parameters

Parameter	Description
source	The source raster object.
dest	The destination raster object.

Examples

```
Update raster_tab le Set raster_obj = ST_Update ( raster_obj
, ( Select raster_obj from raster_tab le where id = 2 ))
where id = 1 ;
```

4.5.6 ST_MosaicFrom

This function performs a mosaic operation to combine multiple raster objects into a new raster object.

Syntax

```
raster ST_MosaicF rom ( raster source [], cstring chunkTable
Name );
```

Parameters

Parameter	Description
source	**The source raster objects.**
chunkTable Name	**The name of the chunk table for storing the created raster object. The name must comply with the table naming rules of ApsaraDB for PostgreSQL.**

Description

This function creates a raster object.

All the specified raster objects must meet the following requirements:

- **They have the same number of bands.**
- **Either all of them are geographically referenced, or none of them is geographic ally referenced. If all of them are geographically referenced, world coordinates (geographic coordinates) are used for the mosaic operation.**
- **Their pixel types can be different. If world coordinates are used for the mosaic operation, they must have the same spatial reference system identifier (SRID) and affine parameters.**

Examples

```
Insert Into raster_obj Values ( 1 , ST_MosaicF rom ( Array (
select raster_obj from raster_tab le where id < 10 ), '
chunk_tabl e_mosaic '))
```

```
Update raster_tab le Set raster_obj = ST_MosaicF rom (
Array ( select raster_obj from raster_tab le where id <
10 ), ' chunk_tabl e_mosaic ') where id = 11 ;
```

4.5.7 ST_MosaicTo

This function performs a mosaic operation to integrate multiple raster objects into an existing raster object.

Syntax

```
raster ST_MosaicT o ( raster raster_obj , raster source []);
```

Parameters

Parameter	Description
raster_obj	The destination raster object.
source	The source raster objects.

Description

All the specified raster objects must meet the following requirements:

- They have the same number of bands.
- Either all of them are geographically referenced, or none of them is geographic ally referenced. If all of them are geographically referenced, world coordinates (geographic coordinates) are used for the mosaic operation.
- Their pixel types can be different. If world coordinates are used for the mosaic operation, they must have the same spatial reference system identifier (SRID) and affine parameters.

Examples

```
Update raster_tab le Set raster_obj = ST_MosaicT o (
raster_obj , Array ( select raster_obj from raster_tab le
where id < 10 )) where id = 11 ;
```

4.5.8 ST_CheckGPU

This function checks whether a GPU is available in the current environment.

Syntax

```
text ST_CheckGP U ();
```

Parameters

This function checks whether a GPU can be identified in the current runtime environment.

Examples

```
select st_checkgp u ();


------------------------------------------------------------------------------
[ GPU prop ] multiProce ssorCount = 20 ; sharedMemP erBlock =
49152 ; maxThreads PerBlock = 1024

( 1 row )
```

4.6 Overview operations

4.6.1 ST_BuildOverview

This function creates an overview.

Syntax

```
raster ST_BuildOv erview ( cstring srcTableNa me , cstring
  srcColumnN ame , integer srcPyramid Level , cstring
chunkTable Name );
```

Parameters

Parameter	Description
tableName	The table name.
columnName	The name of the raster object column.
pyramidLev el	The pyramid level.

Parameter	Description
chunkTable Name	The name of the chunk table for storing the raster object created as an overview.

This function creates a table-based raster overview.

All the specified raster objects must meet the following requirements:

- They have the same number of bands.
- Either all of them are geographically referenced, or none of them is geographic ally referenced. If all of them are geographically referenced, world coordinates (geographic coordinates) are used for the mosaic operation.
- Their pixel types can be different. If world coordinates are used for the mosaic operation, they must have the same spatial reference system identifier (SRID) and pixel resolution.

Examples

```
Insert into raster_tab le_overvie w values ( 1 , ST_BuildOv
erview (' raster_tab le ',' raster_obj ', 0 ,' chunk_tabl
e_overview '));
```

4.6.2 ST_UpdateOverview

This function uses raster objects to update an overview.

Syntax

```
raster ST_UpdateO verview ( raster raster_obj , raster source
[]);
```

Parameters

Parameter	Description
raster_obj	The destination raster object.
source	The source raster objects.

Description

All the specified raster objects must meet the following requirements:

- They have the same number of bands.

- Either all of them are geographically referenced, or none of them is geographic ally referenced. If all of them are geographically referenced, world coordinates (geographic coordinates) are used for the mosaic operation.
- Their pixel types can be different. If world coordinates are used for the mosaic operation, they must have the same spatial reference system identifier (SRID) and pixel resolution.

Examples

```
Update raster_tab le set raster_obj = ST_UpdateO verview (
raster_obj , Array ( select raster_obj from raster_tab le_new
)) where id = 1 ;
```

4.6.3 ST_EraseOverview

This function clears the specified area of a raster object.

Syntax

```
raster ST_EraseOv erview ( raster raster_obj , Box extent ,
BoxType type , boolean useNodata );
```

Parameters

Parameter	Description
raster_obj	The raster object.
extent	The area to be cleared, in the format of '((minX,minY),(maxX, maxY))'.
type	The coordinate type of the area to be cleared. The value must be Raster or World, where Raster indicates pixel coordinates and World indicates world coordinates.
useNodata	Indicates whether to use the predefined NoData value to fill the area to be cleared. If this parameter is set to false or the NoData value is not defined, 0 is used to fill the area.

Examples

```
Select ST_EraseOv erview ( raster_obj , '(( 0 , 0 ),( 100 , 100
))', ' Raster ', false ) from raster_tab le where id = 100 ;
```

4.7 Attribute query and update

4.7.1 ST_MetaData

This function returns the metadata of a raster object in JSON format.

Syntax

```
text   ST_MetaDat  a ( raster   raster_obj );
```

Parameters

Parameter	Description
raster_obj	The raster object.

Examples

```
select   ST_MetaDat  a ( raster_obj )  from   raster_tab  le ;
```

4.7.2 ST_Width

This function returns the width of a raster object. For more information about how to obtain the width of a chunk, see ST_ChuckWidth.

Syntax

```
integer   ST_Width ( raster   raster_obj );
```

Parameters

Parameter	Description
raster_obj	The raster object.

Examples

```
select   ST_Width ( raster_obj )  from   raster_tab  le ;
_____________________________
```

```
10060
```

4.7.3 ST_Height

This function returns the height of a raster object.

Syntax

```
integer   ST_Height ( raster   raster_obj );
```

Parameters

Parameter	Description
raster_obj	**The raster object.**

Examples

```
select   ST_Height ( raster_obj )  from   raster_tab  le ;
```

4.7.4 ST_NumBands

This function returns the number of bands in a raster object.

Syntax

```
integer   ST_NumBand  s ( raster   raster_obj );
```

Parameters

Parameter	Description
raster_obj	**The raster object.**

Examples

```
select   ST_NumBand  s ( raster_obj )  from   raster_tab  le ;
_____________________________________
```

```
3
```

4.7.5 ST_RasterID

This function returns the universally unique identifier (UUID) of a raster object.

Syntax

```
text   ST_RasterI  D ( raster   raster_obj );
```

Parameters

Parameter	Description
raster_obj	The raster object.

Examples

```
select   ST_RasterI  D ( raster_obj )  from   raster_tab  le ;
_____________________________________
4e692ed0 - 74e2 - 42a3 - a10d - c28d4ae319  82
```

4.7.6 ST_CellDepth

This function returns the pixel depth of a raster object. The pixel depth can be 0, 1, 2, 4, 8, 16, 32, or 64, where 0 indicates that the pixel depth is unknown.

Syntax

```
integer   ST_CellDep  th ( raster   raster_obj );
```

Parameters

Parameter	Description
raster_obj	The raster object.

Examples

```
select   ST_CellDep  th ( raster_obj )  from   raster_tab  le ;

__________  __________  __________  ____
```

```
8
```

4.7.7 ST_CellType

This function returns the pixel type of a raster object. The pixel type can be 8BSI, 8BUI, 16BSI, 16BUI, 32BSI, 32BUI, 32BF, or 64BF.

Syntax

```
text   ST_CellTyp  e ( raster   raster_obj );
```

Parameters

Parameter	Description
raster_obj	**The raster object.**

Examples

```
select   st_celltyp  e ( raster_obj )  from   raster_tab  le ;

_________  _________  _________  ____
8BUI
```

4.7.8 ST_InterleavingType

This function returns the interleaving type of a raster object. The interleaving type can be BSQ, BIL, or BIP.

Syntax

```
text   ST_Interle  avingType ( rasterrast  er_obj );
```

Parameters

Parameter	Description
raster_obj	**The raster object.**

Examples

```
select   ST_Interle  avingType ( raster_obj )  from   raster_tab  le
;

_________  _________  _________  ____
```

```
BSQ
```

4.7.9 ST_TopPyramidLevel

This function returns the highest pyramid level of a raster object.

Syntax

```
integer   TopPyramid  Level ( raster   raster_obj );
```

Parameters

Parameter	Description
raster_obj	The raster object.

Examples

```
select   ST_TopPyra  midLevel ( raster_obj )  from   raster_tab  le ;

__________  __________  __________  ____
6
```

4.7.10 ST_Extent

This function returns the coordinate range of a raster object. The return value is a box object of PostgreSQL in the format of '((minX,minY),(maxX,maxY))'.

Syntax

```
BOX   ST_Extent ( raster   raster_obj , CoorSpatia  lOption
csOption  = ' WorldFirst ')
```

Parameters

Parameter	Description
raster_obj	The raster object.
CoorSpatia lOption	The coordinate type.

Description

The CoorSpatialOption parameter specifies the coordinate type. Valid values:

- Raster: indicates the raster space. If this value is used, the function returns pixel coordinates.
- World: indicates the world space. If this value is used, the function returns world coordinates.

- WorldFirst: indicates that the world space takes precedence. If the raster object is geographically referenced, the function returns world coordinates. If the raster object is not geographically referenced, the function returns pixel coordinates.

Examples

```
select   ST_Extent ( raster_obj , ' Raster ')  from   raster_tab  le
;

__________ __________ __________  ____
(( 0 , 0 ),( 255 , 255 ))
```

4.7.11 ST_Srid

This function returns the spatial reference system identifier (SRID) of a raster object. The SRID and its definition are stored in the spatial_ref_sys table.

Syntax

```
integer   ST_Srid ( raster   raster_obj );
```

Parameters

Parameter	Description
raster_obj	The raster object.

Examples

```
select   ST_Srid ( raster_obj )  from   raster_tab  le   where   id =
1 ;

__________  __________  __________  ____
4326
```

4.7.12 ST_Georeference

This function returns the geographic reference information about a raster object. Affine parameters in the text format of "A,B,C,D,E,F" are returned.

Syntax

```
text   ST_Georefe  rence ( raster   raster_obj );
```

Parameters

Parameter	Description
raster_obj	The raster object.

Examples

```
select   ST_Georefe  rence ( raster_obj )  from   raster_tab  le
where   id = 1 ;

__________  __________  __________  ____
2 . 5000000000  00000 , 0 . 0000000000  00000 , 38604686 .
7500000000  00000 , 0 . 0000000000  00000 ,- 2 . 5000000000  00000 ,
4573895 . 7500000000  00000
```

4.7.13 ST_IsGeoreferenced

This function returns true if a raster object is geographically referenced. The return value is t or f in Boolean format.

Syntax

```
boolean   ST_IsGeore  ferenced ( raster   raster_obj );
```

Parameters

Parameter	Description
raster_obj	**The raster object.**

Description

In the returned result, t indicates true and f indicates false.

Examples

```
select   ST_IsGeore  ferenced ( raster_obj )  from   raster_tab  le
where   id = 1 ;

__________  __________  __________  ____
t
```

4.7.14 ST_NoData

This function returns the predefined NoData value for a band of a raster object. If the NoData value is not defined, the function returns null.

Syntax

```
float8   ST_NoData ( raster   raster_obj ,  integer   band );
```

Parameters

Parameter	Description
raster_obj	**The raster object.**

Parameter	Description
band	The band sequence number, starting from 0.

Examples

```
select   ST_NoData ( raster_obj ,  0 )  from   raster_tab  le   where
  id = 1 ;

__________  __________  __________  ____
0 . 000
```

4.7.15 ST_ColorTable

This function returns the color table of a band of a raster object in JSON format.

Syntax

```
text   ST_ColorTa  ble ( raster   raster_obj ,  integer   band );
```

Parameters

Parameter	Description
raster_obj	The raster object.
band	The band sequence number, starting from 0.

Description

The following code shows color tables in JSON format.

- Four color components:

```
'{" compsCount ": 4 ,
    " entries ":[
        {" value ": 0 ," c1 ": 0 ," c2 ": 0 ," c3 ": 0 ," c4 ": 255
 },
        {" value ": 1 ," c1 ": 0 ," c2 ": 0 ," c3 ": 85 ," c4 ": 255
 },
        {" value ": 2 ," c1 ": 0 ," c2 ": 0 ," c3 ": 170 ," c4 ":
 255 }
    ]
}'
```

- Three color components:

```
'{" compsCount ": 3 ,
    " entries ":[
        {" value ": 0 ," c1 ": 0 ," c2 ": 0 ," c3 ": 0 },
        {" value ": 1 ," c1 ": 0 ," c2 ": 0 ," c3 ": 85 },
        {" value ": 2 ," c1 ": 0 ," c2 ": 0 ," c3 ": 170 }
    ]
```

```
}'
```

If the band does not have a color table, the function returns null.

Examples

```
select   ST_ColorTa  ble ( raster_obj , 0 )  from   raster_tab  le
where   id  =  1 ;

----------  ----------  ----------  ----
'{" compsCount ": 3 ,
    " entries ":
    [
        {" value ": 0 ," c1 ": 0 ," c2 ": 0 ," c3 ": 0 },
        {" value ": 1 ," c1 ": 0 ," c2 ": 0 ," c3 ": 85 },
        {" value ": 2 ," c1 ": 0 ," c2 ": 0 ," c3 ": 170 }
    ]
}'
```

4.7.16 ST_Statistics

This function returns the statistics about a band of a raster object in JSON format. If the band does not have statistics, the function returns null.

Syntax

```
text   ST_Statist  ics ( raster   raster_obj ,  integer   band );
```

Parameters

Parameter	Description
raster_obj	The raster object.
band	The band sequence number, starting from 0.

Examples

```
select   ST_Statist  ics ( raster_obj ,  0 )  from   raster_tab  le
where   id = 1 ;

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

```
'{ " min ":  0 . 00 , " max ":  255 . 00 , " mean ":  125 . 00 , " std
 ":  23 . 123 , " approx ":  false }'
```

4.7.17 ST_SummaryStats

This function computes the statistics about a band set of a raster object.

Syntax

```
raster  ST_Summary  Stats ( raster  raster_obj );
```

Parameters

Parameter	Description
raster_obj	The raster object.

Examples

```
update  raster_obj  set  raster_obj = ST_Summary  Stats (
raster_obj )  where  id  =  1 ;
 ----------  ----------  ----------  ----
( 1  row )
```

4.7.18 ST_ColorInterp

This function returns the color interpretation type of a band of a raster object.

Syntax

```
text  ST_ColorIn  terp ( raster  raster_obj ,  integer  band );
```

Parameters

Parameter	Description
raster_obj	The raster object.
band	The band sequence number, starting from 0.

The following table describes the values of the interp parameter that is returned.

Value	Description
Undefined	The color interpretation type undefined.
GrayIndex	The gray value index.
PaletteInd ex	The color table index.
RedBand	The red band in the RGB color model.
GreenBand	The green band in the RGB color model.

Value	Description
BlueBand	The blue band in the RGB color model.
AlphaBand	The alpha band in the RGBA color model.
HueBand	The hue band in the HSL color model.
Saturation Band	The saturation band in the HSL color model.
LightnessB and	The lightness band in the HSL color model.
CyanBand	The cyan band in the CMYK color model.
MagentaBan d	The magenta band in the CMYK color model.
YellowBand	The yellow band in the CMYK color model.
BlackBand	The black band in the CMYK color model.
YCbCr_YBan d	The Y band in the YCbCr color model.
YCbCr_CbBa nd	The Cb band in the YCbCr color model.
YCbCr_CrBa nd	The Cr band in the YCbCr color model.

Examples

```
select   ST_ColorIn  terp ( raster_obj , 0 )  from   raster_tab  le
where   id  =  1 ;

__________  __________  __________  ____
RedBand
```

4.7.19 ST_Histogram

This function returns the histogram of a band of a raster object in text format. If the band does not have a histogram, the function returns null.

Syntax

```
text   ST_Histogr  am ( raster   raster_obj ,  integer   band );
```

Parameters

Parameter	Description
raster_obj	The raster object.
band	The band sequence number, starting from 0.

Examples

```
 select   ST_Histogr  am ( raster_obj ,  0 )  from   raster_tab  le
 where   id = 1 ;
 ----------  ----------  ----------  ----
{
 " approximat  e ": false ,
 " histsCount  s ":
 [ 2 , 1 , 1 , 0 , 8 , 17 , 47 , 101 , 193 , 345 , 443 , 640 , 877 ,
 1189 , 1560 , 1847 , 2087 , 2560 , 2816 , 3193 , 3567 , 3840 , 4101
 , 4415 , 4498 , 3876 , 3235 , 2458 , 1800 , 1598 , 1087 , 731 , 638 ,
 426 , 264 , 198 , 147 , 126 , 104 , 104 , 80 , 84 , 86 , 71 , 80 , 62
 , 74 , 85 , 72 , 80 , 70 , 88 , 69 , 68 , 62 , 58 , 63 , 51 , 53 , 55
 , 54 , 56 , 55 , 63 , 47 , 39 , 49 , 59 , 66 , 62 , 64 , 73 , 66 , 72
 , 67 , 84 , 86 , 79 , 91 , 92 , 117 , 138 , 136 , 142 , 157 , 225 ,
 287 , 285 , 382 , 449 , 567 , 628 , 750 , 855 , 1021 , 1142 , 1242 ,
 1410 , 1504 , 1590 , 1786 , 1870 , 2044 , 2099 , 2277 , 2373 , 2451 ,
 2585 , 2646 , 2882 , 2878 , 3091 , 3396 , 3620 , 3911 , 4124 , 4304 ,
 4700 , 4893 , 5314 , 5446 , 5657 , 5765 , 5649 , 5749 , 5753 , 5601 ,
 5335 , 5161 , 4943 , 4592 , 4445 , 4207 , 4083 , 4090 , 4270 , 4465 ,
 4514 , 4844 , 5204 , 5331 , 5597 , 5777 , 5838 , 6004 , 6316 , 6095 ,
 5762 , 5567 , 5465 , 4923 , 4677 , 4220 , 3843 , 3401 , 3041 , 2571 ,
 2345 , 1972 , 1725 , 1376 , 1140 , 1008 , 841 , 716 , 548 , 442 , 373
 , 308 , 212 , 133 , 78 , 68 , 31 , 24 , 12 , 2 , 2 , 1 ],
 " binFunctio  n ":
 {
  " type ":" unknown ",
  " binRange ":
  {
   " minValue ": 29 . 0 ,
   " maxValue ": 208 . 0 ,
   " outRange ":" include ",
   " binValues ":
   [ 29 . 0 , 30 . 0 , 31 . 0 , 32 . 0 , 33 . 0 , 34 . 0 , 35 . 0 , 36
 . 0 , 37 . 0 , 38 . 0 , 39 . 0 , 40 . 0 , 41 . 0 , 42 . 0 , 43 . 0 ,
 44 . 0 , 45 . 0 , 46 . 0 , 47 . 0 , 48 . 0 , 49 . 0 , 50 . 0 , 51 . 0
 , 52 . 0 , 53 . 0 , 54 . 0 , 55 . 0 , 56 . 0 , 57 . 0 , 58 . 0 , 59 .
 0 , 60 . 0 , 61 . 0 , 62 . 0 , 63 . 0 , 64 . 0 , 65 . 0 , 66 . 0 , 67
 . 0 , 68 . 0 , 69 . 0 , 70 . 0 , 71 . 0 , 72 . 0 , 73 . 0 , 74 . 0 ,
 75 . 0 , 76 . 0 , 77 . 0 , 78 . 0 , 79 . 0 , 80 . 0 , 81 . 0 , 82 . 0
 , 83 . 0 , 84 . 0 , 85 . 0 , 86 . 0 , 87 . 0 , 88 . 0 , 89 . 0 , 90 .
 0 , 91 . 0 , 92 . 0 , 93 . 0 , 94 . 0 , 95 . 0 , 96 . 0 , 97 . 0 , 98
 . 0 , 99 . 0 , 100 . 0 , 101 . 0 , 102 . 0 , 103 . 0 , 104 . 0 , 105
 . 0 , 106 . 0 , 107 . 0 , 108 . 0 , 109 . 0 , 110 . 0 , 111 . 0 , 112
 . 0 , 113 . 0 , 114 . 0 , 115 . 0 , 116 . 0 , 117 . 0 , 118 . 0 , 119
 . 0 , 120 . 0 , 121 . 0 , 122 . 0 , 123 . 0 , 124 . 0 , 125 . 0 , 126
 . 0 , 127 . 0 , 128 . 0 , 129 . 0 , 130 . 0 , 131 . 0 , 132 . 0 , 133
 . 0 , 134 . 0 , 135 . 0 , 136 . 0 , 137 . 0 , 138 . 0 , 139 . 0 , 140
 . 0 , 141 . 0 , 142 . 0 , 143 . 0 , 144 . 0 , 145 . 0 , 146 . 0 , 147
 . 0 , 148 . 0 , 149 . 0 , 150 . 0 , 151 . 0 , 152 . 0 , 153 . 0 , 154
 . 0 , 155 . 0 , 156 . 0 , 157 . 0 , 158 . 0 , 159 . 0 , 160 . 0 , 161
 . 0 , 162 . 0 , 163 . 0 , 164 . 0 , 165 . 0 , 166 . 0 , 167 . 0 , 168
 . 0 , 169 . 0 , 170 . 0 , 171 . 0 , 172 . 0 , 173 . 0 , 174 . 0 , 175
 . 0 , 176 . 0 , 177 . 0 , 178 . 0 , 179 . 0 , 180 . 0 , 181 . 0 , 182
 . 0 , 183 . 0 , 184 . 0 , 185 . 0 , 186 . 0 , 187 . 0 , 188 . 0 , 189
 . 0 , 190 . 0 , 191 . 0 , 192 . 0 , 193 . 0 , 194 . 0 , 195 . 0 , 196
 . 0 , 197 . 0 , 198 . 0 , 199 . 0 , 200 . 0 , 201 . 0 , 202 . 0 , 203
 . 0 , 204 . 0 , 205 . 0 , 206 . 0 , 207 . 0 ]
  }
 }
```

```
}
```

4.7.20 ST_BuildHistogram

This function computes the histogram of a band set of a raster object.

Syntax

```
 raster   ST_BuildHi  stogram ( raster   raster_obj );
```

Parameters

Parameter	Description
raster_obj	**The raster object.**

Examples

```
 UPDATE   raster_tab  le   SET   raster_obj  =  st_buildhi  stogram (
 raster_obj )  WHERE   id  =  1 ;
--------------------------------
( 1   row )
```

5 Point cloud SQL reference

5.1 Constructors

5.1.1 ST_makePoint

This function constructs a pcpoint object.

Syntax

```
pcpoint  ST_makePoi  nt ( integer  pcid ,  float8 []  vals );
```

Parameters

Parameter	Description
pcid	The ID of the schema, which comes from the pointcloud_formats table.
float8 []	The float8 array. The number of entries in the array is equal to the number of dimensions specified in the schema.

Examples

```
SELECT  ST_makePoi  nt ( 1 ,  ARRAY [- 127 ,  45 ,  124 . 0 ,  4 . 0
]);
-------------------------------------------------
0101000000  64CEFFFF94  1100007030  00000400
```

5.1.2 ST_makePatch

This function constructs a pcpatch object.

Syntax

```
pcpatch  ST_makePat  ch ( integer  pcid ,  float8 []  vals );
```

Parameters

Parameter	Description
pcid	The ID of the schema, which comes from the pointcloud_formats table.

Parameter	Description
float8 []	The float8 array. The number of entries in the array is an integral multiple of the number of dimensions specified in the schema.

Examples

```
 SELECT   ST_asText ( ST_MakePat  ch ( 1 ,  ARRAY [- 126 . 99 , 45 .
 01 , 1 , 0 , - 126 . 98 , 45 . 02 , 2 , 0 , - 126 . 97 , 45 . 03 , 3
 , 0 ]));
-----------------------------------------------
{" pcid ": 1 ," pts ":[
 [- 126 . 99 , 45 . 01 , 1 , 0 ],[- 126 . 98 , 45 . 02 , 2 , 0 ],[-
 126 . 97 , 45 . 03 , 3 , 0 ]
]}
```

5.1.3 ST_Patch

This function constructs a pcpatch object from a pcpoint array.

Syntax

```
 pcpatch   ST_Patch ( pcpoint []  pts );
```

Parameters

Parameter	Description
pts	The pcpoint array.

Examples

```
 INSERT   INTO   patches  ( pa )
 SELECT   ST_Patch ( pt )  FROM   points   GROUP   BY   id / 10 ;
```

5.2 Attribute functions

5.2.1 ST_asText

This function converts a pcpoint or pcpatch object into a JSON-formatted string.

Syntax

```
 text   ST_asText ( pcpatch   pp );
```

```
 text   ST_asText ( pcpoint   pt );
```

Parameters

Parameter	Description
pp	The pcpatch object.
pt	The pcpoint object.

Examples

```
 SELECT   ST_asText (' 0101000000  64CEFFFF94  1100007030  00000400
 ':: pcpoint );
--------------------------------
{" pcid ": 1 ," pt ":[- 127 , 45 , 124 , 4 ]}
```

5.2.2 ST_pcID

This function returns the schema ID of a pcpoint or pcpatch object.

Syntax

```
 integer   ST_pcID ( pcpoint   pt );
 integer   ST_pcID ( pcpatch   pp );
```

Parameters

Parameter	Description
pt	The pcpoint object.
pp	The pcpatch object.

Examples

```
 SELECT   ST_pcID (' 0101000000  64CEFFFF94  1100007030  00000400 ':: 
 pcpoint );
--------------------------------
 1
```

5.2.3 ST_get

This function returns the values of attribute dimensions in a pcpoint object.

Syntax

```
 float8 []  ST_get ( pcpoint   pc );
```

```
 numeric   ST_get ( pcpoint   pc ,  text   dimname );
```

Parameters

Parameter	Description
pc	The pcpoint object.
dimname	The name of the specified attribute dimension.

Examples

```
 SELECT   ST_Get (' 0101000000  64CEFFFF94  1100007030  00000400 ':: 
 pcpoint , ' Intensity ');
-------------------
 4

 SELECT   ST_Get (' 0101000000  64CEFFFF94  1100007030  00000400 ':: 
 pcpoint );
--------------------
{- 127 , 45 , 124 , 4 }
```

5.2.4 ST_numPoints

This function returns the number of pcpoint objects in a pcpatch object.

Syntax

```
 integer   ST_numPoin  ts ( pcpatch   pc );
```

Parameters

Parameter	Description
pc	The pcpatch object.

Examples

```
 SELECT   ST_NumPoin  ts ( pa )  FROM   patches   LIMIT   1 ;
---------------------------
```

```
9
```

5.2.5 ST_summary

This function returns the summary of a pcpatch object.

Syntax

```
text   ST_summary ( pcpatch   pc );
```

Parameters

Parameter	Description
pc	The pcpatch object.

Examples

```
 SELECT   ST_Summary ( pa )  FROM   patches   LIMIT   1 ;
----------------------------------
{" pcid ": 1 , " npts ": 9 , " srid ": 4326 , " compr ":" dimensiona
 l "," dims ":[{" pos ": 0 ," name ":" X "," size ": 4 ," type ":"
 int32_t "," compr ":" sigbits "," stats ":{" min ":- 126 . 99 ," max
 ":- 126 . 91 ," avg ":- 126 . 95 }},{" pos ": 1 ," name ":" Y ","
 size ": 4 ," type ":" int32_t "," compr ":" sigbits "," stats ":{"
 min ": 45 . 01 ," max ": 45 . 09 ," avg ": 45 . 05 }},{" pos ": 2 ,"
 name ":" Z "," size ": 4 ," type ":" int32_t "," compr ":" sigbits
 "," stats ":{" min ": 1 ," max ": 9 ," avg ": 5 }},{" pos ": 3 ,"
 name ":" Intensity "," size ": 2 ," type ":" uint16_t "," compr ":"
 rle "," stats ":{" min ": 0 ," max ": 0 ," avg ": 0 }}]}
```

5.3 Object operations

5.3.1 ST_compress

This function compresses a pcpatch object based on the specified compression method.

Syntax

```
pcpatch   ST_compres  s ( pcpatch   pc ,  text   global_com
pression_s  chema   default  '',  text   compressio  n_config
default  '');
```

Parameters

Parameter	Description
pc	The pcpatch object.
global_com pression_s chema	The compression schema.

Parameter	Description
compressio n_config	The compression configuration item that specifies the compression algorithm for specific dimensions.

Description

Valid values of the global_compression_schema parameter include:

```
auto            --  determined   by   pcid
dimension
laz              --  no   compressio  n   config   supported
ght             --  is   discarded
```

If the global_compression_schema parameter is set to dimension, valid values of the compression_config parameter include:

```
auto  --  determined   automatica  lly ,  from   values   stats
zlib  --  deflate   compressio  n
sigbits  --  significan  t   bits   removal
rle  --  run - length   encoding
```

Examples

```
SELECT   ST_asText ( ST_Compres  s ( ST_MakePat  ch ( 1 ,  ARRAY [-
126 . 99 , 45 . 01 , 1 , 0 , - 126 . 98 , 45 . 02 , 2 , 0 , - 126 .
97 , 45 . 03 , 3 , 0 ])));

-------------------------------------------------
{" pcid ": 1 ," pts ":[
 [- 126 . 99 , 45 . 01 , 1 , 0 ],[- 126 . 98 , 45 . 02 , 2 , 0 ],[-
 126 . 97 , 45 . 03 , 3 , 0 ]
]}
```

5.3.2 ST_unCompress

This function returns an uncompressed version of the pcpatch object.

Syntax

```
pcpatch   ST_unCompr  ess ( pcpatch   pc );
```

Parameters

Parameter	Description
pc	The pcpatch object.

Description

All the pcpatch objects returned by functions are compressed based on the compression method specified in the schema before they are returned.

Examples

```
 SELECT   ST_Uncompr  ess ( pa )  FROM   patches   WHERE   ST_NumPoin
 ts ( pa ) =  1 ;
--------------------------------------
 0101000000  0000000001  000000C8CE  FFFFF81100  0010270000  0A00
```

5.3.3 ST_union

This function merges an array of pcpatch entries into a pcpatch object.

Syntax

```
 pcpatch   ST_union ( pcpatch []  pcs );
```

Parameters

Parameter	Description
pcs	The pcpatch array.

Examples

```
--  Compare   npoints ( sum ( patches ))  with   sum ( npoints (
 patches )).
 SELECT   ST_NumPoin  ts ( ST_Union ( pa ))  FROM   patches ;
 SELECT   Sum ( ST_NumPoin  ts ( pa ))  FROM   patches ;

 100
```

5.3.4 ST_explode

This function separates a pcpatch object into pcpoint objects in multiple rows.

Syntax

```
 setof [ pcpoint ]  ST_explode ( pcpatch   pc );
```

Parameters

Parameter	Description
pc	The pcpatch object.

Examples

```
 SELECT   ST_AsText ( ST_Explode ( pa )),  id   FROM   patches   WHERE
   id  =  7 ;
-----------------------------------------
              st_astext                |  id
---------------------------------------+----
 {" pcid ": 1 ," pt ":[- 126 . 5 , 45 . 5 , 50 , 5 ]}   |   7
 {" pcid ": 1 ," pt ":[- 126 . 49 , 45 . 51 , 51 , 5 ]} |   7
 {" pcid ": 1 ," pt ":[- 126 . 48 , 45 . 52 , 52 , 5 ]} |   7
 {" pcid ": 1 ," pt ":[- 126 . 47 , 45 . 53 , 53 , 5 ]} |   7
 {" pcid ": 1 ," pt ":[- 126 . 46 , 45 . 54 , 54 , 5 ]} |   7
 {" pcid ": 1 ," pt ":[- 126 . 45 , 45 . 55 , 55 , 5 ]} |   7
 {" pcid ": 1 ," pt ":[- 126 . 44 , 45 . 56 , 56 , 5 ]} |   7
 {" pcid ": 1 ," pt ":[- 126 . 43 , 45 . 57 , 57 , 5 ]} |   7
 {" pcid ": 1 ," pt ":[- 126 . 42 , 45 . 58 , 58 , 5 ]} |   7
 {" pcid ": 1 ," pt ":[- 126 . 41 , 45 . 59 , 59 , 5 ]} |   7
```

5.3.5 ST_patchAvg

This function computes the average values of all the attribute dimensions for all the pcpoint objects in a pcpatch object and returns a new pcpoint object whose attribute dimensions are set to these average values.

Syntax

```
 pcpoint   ST_patchAv  g ( pcpatch   pc );
```

Parameters

Parameter	Description
pc	The pcpatch object.

Examples

```
 SELECT   ST_AsText ( ST_PatchAv  g ( pa ))  FROM   patches   WHERE
 id  =  7 ;
-------------------------------------
```

```
{" pcid ": 1 ," pt ":[- 126 . 46 , 45 . 54 , 54 . 5 , 5 ]}
```

5.3.6 ST_patchMax

This function computes the maximum values of all the attribute dimensions for all the pcpoint objects in a pcpatch object and returns a new pcpoint object whose attribute dimensions are set to these maximum values.

Syntax

```
 pcpoint   ST_patchMa  x ( pcpatch   pc );
```

Parameters

Parameter	Description
pc	The pcpatch object.

Examples

```
 SELECT   ST_PatchMa  x ( pa )  FROM   patches   WHERE   id  =  7 ;
---------------------------------
{" pcid ": 1 ," pt ":[- 126 . 41 , 45 . 59 , 59 , 5 ]}
```

5.3.7 ST_patchMin

This function computes the minimum values of all the attribute dimensions for all the pcpoint objects in a pcpatch object and returns a new pcpoint object whose attribute dimensions are set to these minimum values.

Syntax

```
 numeric   ST_patchMi  n ( pcpatch   pc );
```

Parameters

Parameter	Description
pc	The pcpatch object.

Examples

```
 SELECT   ST_PatchMi  n ( pa )  FROM   patches   WHERE   id  =  7 ;
---------------------------------
```

```
{" pcid ": 1 ," pt ":[- 126 . 5 , 45 . 5 , 50 , 5 ]}
```

5.3.8 ST_patchAvg

This function computes the average value of an attribute dimension for all the pcpoint objects in a pcpatch object.

Syntax

```
numeric   ST_patchAv  g ( pcpatch   pc ,  text   dimname );
```

Parameters

Parameter	Description
pc	The pcpatch object.
dimname	The name of the specified attribute dimension.

Examples

```
 SELECT   ST_PatchAv  g ( pa , ' intensity ')  FROM   patches   WHERE
   id  =  7 ;
--------------------------
 5 . 0000000000  000000
```

5.3.9 ST_patchMax

This function computes the maximum value of an attribute dimension for all the pcpoint objects in a pcpatch object.

Syntax

```
numeric   ST_patchMa  x ( pcpatch   pc ,  text   dimname );
```

Parameters

Parameter	Description
pc	The pcpatch object.
dimname	The name of the specified attribute dimension.

Examples

```
 SELECT   ST_PatchMa  x ( pa , ' intensity ')  FROM   patches   WHERE
   id  =  7 ;
--------------------------
```

```
125 . 0000000000  000000
```

5.3.10 ST_patchMin

This function computes the minimum value of an attribute dimension for all the pcpoint objects in a pcpatch object.

Syntax

```
numeric   ST_patchMi  n ( pcpatch   pc ,  text   dimname );
```

Parameters

Parameter	Description
pc	The pcpatch object.
dimname	The name of the specified attribute dimension.

Examples

```
SELECT   ST_PatchMi  n ( pa , ' intensity ')  FROM   patches   WHERE
  id  =  7 ;
--------------------------
0 . 25122
```

5.3.11 ST_filterGreaterThan

This function filters all pcpoint objects to obtain the pcpoint objects whose values of the specified attribute dimension are greater than a fixed value in a pcpatch object and returns a new pcpatch object that is composed of these pcpoint objects.

Syntax

```
pcpatch   ST_filterG  reaterThan ( pcpatch   pc ,  text   dimname ,
float8   value );
```

Parameters

Parameter	Description
pc	The pcpatch object.
dimname	The name of the specified attribute dimension.
value	The fixed value of the attribute dimension.

Examples

```
 SELECT   ST_AsText ( ST_FilterG  reaterThan ( pa , ' y ',  45 . 57
 ))  FROM   patches   WHERE   id  =  7 ;
-------------------------------------------------------------
 {" pcid ": 1 ," pts ":[[- 126 . 42 , 45 . 58 , 58 , 5 ],[- 126 . 41 ,
 45 . 59 , 59 , 5 ]]}
```

5.3.12 ST_filterLessThan

This function filters all pcpoint objects to obtain the pcpoint objects whose values of the specified attribute dimension are smaller than a fixed value in a pcpatch object and returns a new pcpatch object that is composed of these pcpoint objects.

Syntax

```
 pcpatch   ST_filterL  essThan ( pcpatch   pc ,  text   dimname ,
 float8   value );
```

Parameters

Parameter	Description
pc	**The pcpatch object.**
dimname	**The name of the specified attribute dimension.**
value	**The fixed value of the attribute dimension.**

Examples

```
 SELECT   ST_AsText ( ST_FilterL  essThan ( pa , ' y ',  45 . 60 ))
 FROM   patches   WHERE   id  =  7 ;
-------------------------------------------------------------
```

```
{" pcid ": 1 ," pts ":[[- 126 . 42 , 45 . 58 , 58 , 5 ],[- 126 . 41 ,
45 . 59 , 59 , 5 ]]}
```

5.3.13 ST_filterEquals

This function filters all pcpoint objects to obtain the pcpoint objects whose values of the specified attribute dimension are equal to a fixed value in a pcpatch object and returns a new pcpatch object that is composed of these pcpoint objects.

Syntax

```
pcpatch   ST_filterE  quals ( pcpatch   pc ,  text   dimname ,
float8   value );
```

Parameters

Parameter	Description
pc	The pcpatch object.
dimname	The name of the specified attribute dimension.
value	The fixed value of the attribute dimension.

Examples

```
SELECT   ST_AsText ( ST_FilterE  quals ( pa , ' y ',  45 . 57 ))
FROM   patches   WHERE   id  =  7 ;
------------------------------------------------------------
{" pcid ": 1 ," pts ":[[- 126 . 42 , 45 . 57 , 58 , 5 ],[- 126 . 41 ,
45 . 57 , 59 , 5 ]]}
```

5.3.14 ST_filterBetween

This function filters all pcpoint objects to obtain the pcpoint objects whose values of the specified attribute dimension are between two fixed values in a pcpatch object and returns a new pcpatch object that is composed of these pcpoint objects.

Syntax

```
pcpatch   ST_filterB  etween ( pcpatch   pc ,  text   dimname ,
float8   minvalue ,  float8   maxvalue );
```

Parameters

Parameter	Description
pc	The pcpatch object.

Parameter	Description
dimname	The name of the specified attribute dimension.
minvalue	The minimum fixed value of the attribute dimension.
maxvalue	The maximum fixed value of the attribute dimension.

Description

The pcpoint objects whose values of the specified attribute dimension are equal to the minimum or maximum fixed value are not returned.

Examples

```
SELECT   ST_AsText ( ST_FilterB  etween ( pa , ' y ',  45 . 57 ,  45
. 60 ))  FROM   patches   WHERE   id  =  7 ;
-----------------------------------------------------------
{" pcid ": 1 ," pts ":[[- 126 . 42 , 45 . 58 , 58 , 5 ],[- 126 . 41 ,
45 . 59 , 59 , 5 ]]}
```

5.3.15 ST_pointN

This function returns the pcpoint object with the specified sequence number in a pcpatch object.

Syntax

```
pcpoint   ST_pointN ( pcpatch   pc ,  integer   n );
```

Parameters

Parameter	Description
pc	The pcpatch object.
n	The sequence number of the pcpoint object , starting from 1. If n is a negative number , it specifies the pcpoint object with the sequence number of |n| starting from the end of the pcpatch object reversely. For example, a value of -2 indicates the last but one pcpoint object in the pcpatch object.

Examples

```
SELECT   ST_asText ( ST_pointN ( pa ,  4 ))  FROM   patches   WHERE
id  =  7 ;
```

```
---------------------------------
{" pcid ": 1 ," pt ":[- 126 . 41 , 45 . 59 , 59 , 5 ]}
```

5.3.16 ST_isSorted

This function returns true if all the pcpoint objects in a pcpatch object are sorted based on the specified attribute dimensions.

Syntax

```
boolean   ST_isSorte  d ( pcpatch   pc ,  text []  dimnames ,
boolean   strict   default   true );
```

Parameters

Parameter	Description
pc	**The pcpatch object.**
dimnames	**The array of attribute dimension names.**
strict	**Indicates whether each pcpoint object has unique values for the specified attribute dimensions.**

Examples

```
SELECT   ST_isSorte  d ( pa , [' y ',' m ']:: array )  FROM   patches
;
-------------------------    ---------------------------------
  f
( 1   rows )
```

5.3.17 ST_sort

This function sorts all the pcpoint objects in a pcpatch object based on the specified attribute dimensions and returns a new pcpatch object that is composed of the sorted pcpoint objects.

Syntax

```
pcpatch   ST_sort ( pcpatch   pc ,  text []  dimnames );
```

Parameters

Parameter	Description
pc	**The pcpatch object.**
dimnames	**The array of attribute dimension names.**

Examples

```
 update   patches   set   pa = ST_Sort ( pa , [' y ',' m ']:: array );
-------------------------------------
( 1   rows )
```

5.3.18 ST_range

This function obtains n consecutive pcpoint objects that start from a sequence number from a pcpatch object and returns a new pcpatch object that is composed of these pcpoint objects.

Syntax

```
 pcpatch   ST_range ( pcpatch   pc ,  integer   start ,  integer   n
 );
```

Parameters

Parameter	Description
pc	The pcpatch object.
start	The sequence number of the start pcpoint object. The value starts from 1.
n	The number of pcpoint objects after the pcpoint object (included) specified by the start parameter.

Examples

```
 update   patches   set   pa = ST_range ( pa ,  2 ,  16 );
-------------------------------------
( 1   rows )
```

5.3.19 ST_setPcid

This function sets a new schema for a pcpatch object and returns a new pcpatch object.

Syntax

```
 pcpatch   ST_setPcid ( pcpatch   pc ,  integer   pcid ,  float8   def
   default   0 . 0 );
```

Parameters

Parameter	Description
pc	The pcpatch object.

Parameter	Description
pcid	The ID of the new schema, which comes from the pointcloud_formats table.
def	The specified value. Attribute dimensions that exist in the new schema but not in the old schema are set to this value. The default value is 0.0.

Description

Attribute dimensions that exist in the old schema but not in the new schema are discarded.

Examples

```
 update   patches   set   pa = ST_setPcid ( pa ,  2 ,  0 . 0 );
------------------------------------
( 1   rows )
```

5.3.20 ST_transform

This function transforms a pcpatch object based on a new schema and returns a new pcpatch object.

Syntax

```
 pcpatch   ST_transfo  rm ( pcpatch   pc ,  integer   pcid ,  float8
 def   default   0 . 0 );
```

Parameters

Parameter	Description
pc	The pcpatch object.
pcid	The ID of the new schema, which comes from the pointcloud_formats table.
def	The specified value. Attribute dimensions that exist in the new schema but not in the old schema are set to this value. The default value is 0.0.

Description

Different from the ST_setPcid function, the ST_transform function returns a new pcpatch object based on the different dimension interpretations, scales, or offsets in the new schema.

Examples

```
 update   patches   set   pa = ST_transfo  rm ( pa ,  2 ,  0 . 0 );
-------------------------------------
( 1   rows )
```

5.3.21 ST_envelopeGeometry

This function returns the bounding box of a pcpatch object as a geometry object.

Syntax

```
 geometry   ST_envelop  eGeometry ( pcpatch   pc );
```

Parameters

Parameter	Description
pc	The pcpatch object.

Description

A bounding box is a 2D geometry object of Ganos Geometry.

Examples

```
 SELECT   ST_AsText ( ST_Envelop  eGeometry ( pa ))  FROM   patches
 LIMIT   1 ;
------------------------------
 POLYGON ((- 126 . 99   45 . 01 ,- 126 . 99   45 . 09 ,- 126 . 91   45
 . 09 ,- 126 . 91   45 . 01 ,- 126 . 99   45 . 01 ))
```

```
CREATE INDEX ON patches USING GIST ( ST_Envelop eGeometry
( patch ));
```

5.3.22 ST_boundingDiagonalGeometry

This function returns the diagonal of the bounding box of a pcpatch object as a geometry object.

Syntax

```
geometry ST_boundin gDiagonalG eometry ( pcpatch pc );
```

Parameters

Parameter	Description
pc	The pcpatch object.

Description

The diagonal of a bounding box is a 2D linestring object of Ganos Geometry. This function can be used to create an index on a pcpatch object column.

Examples

```
SELECT ST_AsText ( ST_Boundin gDiagonalG eometry ( pa )) FROM
patches ;
                st_astext
-------------------------------------------
LINESTRING Z (- 126 . 99 45 . 01 1 ,- 126 . 91 45 . 09 9 )
LINESTRING Z (- 126 46 100 ,- 126 46 100 )
LINESTRING Z (- 126 . 2 45 . 8 80 ,- 126 . 11 45 . 89 89 )
LINESTRING Z (- 126 . 4 45 . 6 60 ,- 126 . 31 45 . 69 69 )
LINESTRING Z (- 126 . 3 45 . 7 70 ,- 126 . 21 45 . 79 79 )
LINESTRING Z (- 126 . 8 45 . 2 20 ,- 126 . 71 45 . 29 29 )
LINESTRING Z (- 126 . 5 45 . 5 50 ,- 126 . 41 45 . 59 59 )
LINESTRING Z (- 126 . 6 45 . 4 40 ,- 126 . 51 45 . 49 49 )
LINESTRING Z (- 126 . 9 45 . 1 10 ,- 126 . 81 45 . 19 19 )
LINESTRING Z (- 126 . 7 45 . 3 30 ,- 126 . 61 45 . 39 39 )
LINESTRING Z (- 126 . 1 45 . 9 90 ,- 126 . 01 45 . 99 99 )


CREATE INDEX ON patches USING GIST ( ST_Boundin
gDiagonalG eometry ( patch ) gist_geome try_ops_nd );
```

5.4 OGC WKB operations

5.4.1 ST_asBinary

This function returns the well-known binary (WKB) value of a pcpoint object.

Syntax

```
bytea   ST_asBinar  y ( pcpoint   pt );
```

Parameters

Parameter	Description
pt	The pcpoint object.

Examples

```
 SELECT   ST_AsBinar  y (' 0101000000  64CEFFFF94  1100007030
 00000400 ':: pcpoint );
------------------------------------------
\ x010100008  0000000000  0c05fc0000  0000000804  6400000000
 000005f40
```

5.4.2 ST_envelopeAsBinary

This function returns the well-known binary (WKB) value of the bounding box of a pcpatch object.

Syntax

```
bytea   ST_envelop  eAsBinary ( pcpatch   pc );
```

Parameters

Parameter	Description
pc	The pcpatch object.

Description

A bounding box is a 2D geometry object of Ganos Geometry.

Examples

```
 SELECT   ST_Envelop  eAsBinary ( pa )  FROM   patches   LIMIT   1 ;
------------------------------------------
\ x010300000  0010000000  500000090c  2f5285cbf5  fc0e17a
 14ae478146  4090c2f528  5cbf5fc0ec  51b81e858b  46400ad7
 a3703dba5f  c0ec51b81e  858b46400a  d7a3703dba  5fc0e17a
```

```
 14ae478146  4090c2f528  5cbf5fc0e1  7a14ae4781  4640
```

5.4.3 ST_boundingDiagonalAsBinary

This function returns the well-known binary (WKB) value of the bounding box diagonal of a pcpatch object.

Syntax

```
 bytea   ST_boundin  gDiagonalA  sBinary ( pcpatch   pc );
```

Parameters

Parameter	Description
pc	**The pcpatch object.**

Description

A bounding box is a 2D geometry object.

Examples

```
 SELECT   ST_Boundin  gDiagonalA  sBinary (  ST_Patch ( ARRAY [
 ST_MakePoi  nt ( 1 ,  ARRAY [ 0 ., 0 ., 0 ., 10 .]),  ST_MakePoi  nt
 ( 1 ,  ARRAY [ 1 ., 1 ., 1 ., 10 .]),  ST_MakePoi  nt ( 1 ,  ARRAY [
 10 ., 10 ., 10 ., 10 .])]));
-------------------------------------------------------
\ x01020000a  0e61000000  2000000000  0000000000  0000000000
 0000000000  0000000000  0000000000  0000000244  0000000000
 0002440000  0000000002  440
```

5.5 Spatial relationship judgment

5.5.1 ST_intersects

This function returns true if the bounding boxes of two pcpatch objects or the bounding boxes of a pcpatch object and a geometry object intersect.

Syntax

```
 boolean   ST_interse  cts ( pcpatch   pp1 ,  pcpatch   pp2 );
 boolean   ST_interse  cts ( geometry   g ,   pcpatch   pp1 );
```

Parameters

Parameter	Description
pp1	**Pcpatch object 1.**

Parameter	Description
pp2	Pcpatch object 2.
g	The geometry object of Ganos Geometry.

Examples

```
--  Patch   should   intersect   itself
 SELECT   ST_Interse  cts (
        ' 0101000000  0000000001  000000C8CE  FFFFF81100  0010270000
 0A00 ':: pcpatch ,
        ' 0101000000  0000000001  000000C8CE  FFFFF81100  0010270000
 0A00 ':: pcpatch );
------------------------
 t

 SELECT   ST_Interse  cts (' SRID = 4326 ; POINT (- 126 . 451   45 .
 552 )':: geometry ,  pa )  FROM   patches   WHERE   id  =  7 ;
------------------------
 t
```

5.6 Spatial processing

5.6.1 ST_intersection

This function returns a new pcpatch object that indicates the intersection of a pcpatch object and the given geometry object.

Syntax

```
 pcpatch   ST_interse  ction ( pcpatch   pc ,  geometry   geom );
```

Parameters

Parameter	Description
pc	The pcpatch object.
geom	The geometry object of Ganos Geometry.

Examples

```
 SELECT   ST_AsText ( ST_Explode ( ST_Interse  ction (
      pa ,
     ' SRID = 4326 ; POLYGON ((- 126 . 451   45 . 552 , - 126 . 42
   47 . 55 , - 126 . 40   45 . 552 , - 126 . 451   45 . 552 ))'::
 geometry
)))
 FROM   patches   WHERE   id  =  7 ;

             st_astext
-------------------------------------
```

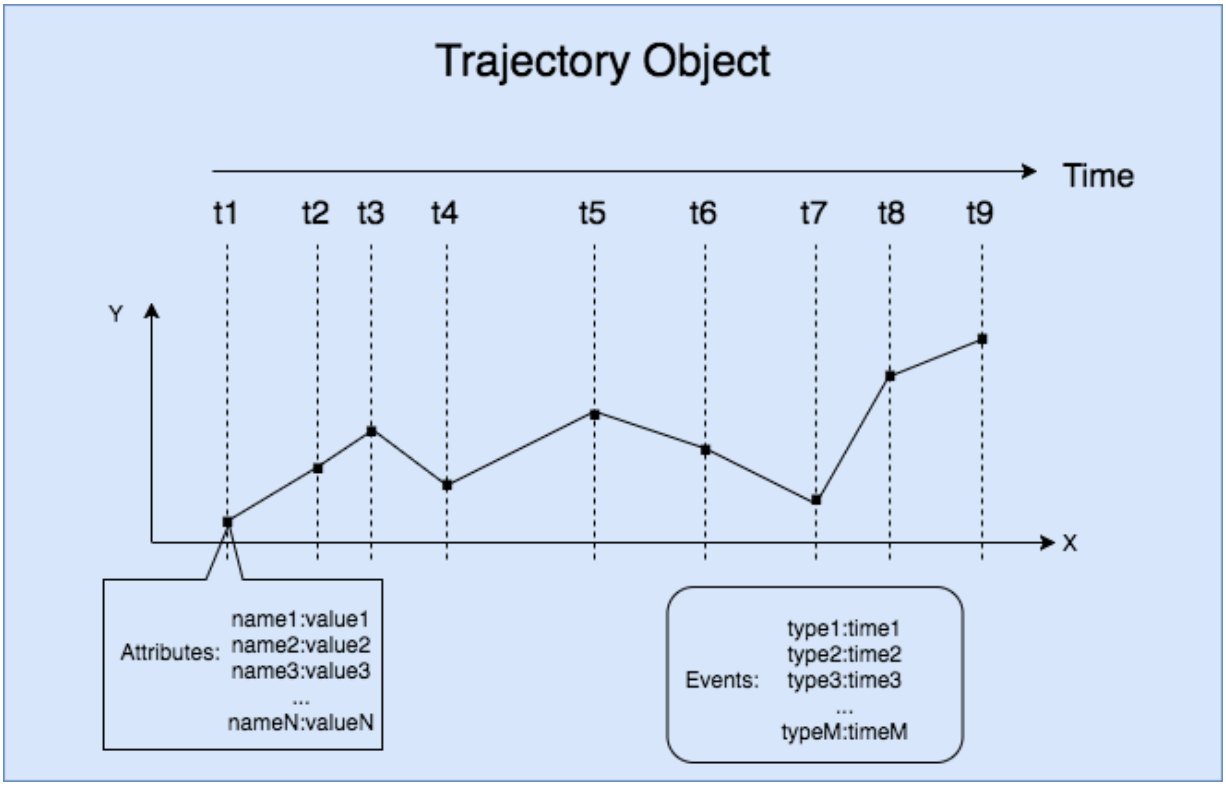
```
{" pcid ": 1 ," pt ":[- 126 . 44 , 45 . 56 , 56 , 5 ]}
{" pcid ": 1 ," pt ":[- 126 . 43 , 45 . 57 , 57 , 5 ]}
{" pcid ": 1 ," pt ":[- 126 . 42 , 45 . 58 , 58 , 5 ]}
{" pcid ": 1 ," pt ":[- 126 . 41 , 45 . 59 , 59 , 5 ]}
```

6 Trajectory SQL reference

6.1 Basic concepts

Concept	Description
`trajectory point`	The spatio-temporal object that consists of the spatial location and attributes of a moving object at a time point. The spatial location can be represented by 2D or 3D coordinates. The attributes support multiple fields of different types.
`trajectory object`	The high-dimensional object that consists of a series of trajectory points and events, including time, space, attributes, and events.
`trajectory timeline`	The time value sequence of a trajectory object.
`trajectory spatial`	The spatial geometry object of a trajectory object, which is of the linestring type.
`trajectory leaf`	The spatial location of a moving object (which is a trajectory point) at a time point.
`trajectory attributes`	The attributes of a moving object at different trajectory points, such as the velocity and direction.
`trajectory attribute field`	The attribute field of a trajectory object, such as the velocity field. The number of values of an attribute field is the same as the number of trajectory points.
`trajectory field value`	The value of a trajectory attribute field at a time point.
`trajectory events`	The events that occur in a trajectory, such as refueling, breakdown, and car lock events in the trajectory of a car. A trajectory event consists of the event type ID and event time.

The following figure shows a trajectory object.



6.2 Constructors

6.2.1 Constructor overview

Constructors include the function for constructing a trajectory object by using JSON-formatted strings or arrays and the function for appending trajectory points or a sub-trajectory to a trajectory object.

6.2.2 ST_makeTrajectory

This function constructs a trajectory object.

Syntax

```
trajectory   ST_makeTra  jectory  ( leaftype   type ,  geometry
spatial ,  tsrange   timespan ,  cstring   attrs_json );
trajectory   ST_makeTra  jectory  ( leaftype   type ,  geometry
  spatial ,  timestamp   start ,  timestamp   end ,  cstring
attrs_json );
trajectory   ST_makeTra  jectory  ( leaftype   type ,  geometry
spatial ,  timestamp []  timeline ,  cstring   attrs_json );
trajectory   ST_makeTra  jectory  ( leaftype   type ,  geometry
[]  points ,  timestamp []  timeline ,  integer   srid ,  text []
```

```
 attr_field  _names ,  int4 []  attr_int4 ,  float8 []  attr_float  8
 ,  text []  attr_cstri  ng ,  anyarray   attr_any );
```

Parameters

Parameter	Description
type	The trajectory type. Only ST_POINT is supported.
spatial	The trajectory path based on a linestring object.
timespan	The time range of the trajectory.
start	The start time of the trajectory.
end	The end time of the trajectory.
timeline	The trajectory timeline. The number of time points must be the same as that of points in the linestring object.
attrs_json	The strings of attribute data, in JSON format.
attr_field _names	The array of the names of all attribute fields for the trajectory.
x	The x-axis (array) for the geometry object.
y	The y-axis (array) for the geometry object.
srid	The spatial reference system identifier (SRID) of the trajectory . This parameter is required.

The format of the attr_json parameter is as follows: `{" leafcount ": 3 ," attributes ":{" velocity ":{" type ":" integer "," length ": 2 ," nullable ": true ," value ":[120 , null , 140]}," accuracy ":{" type ":" float "," length ": 4 ," nullable ": false ," value ":[120 , 130 , 140]}," bearing ":{" type ":" float "," length ": 8 ," nullable ": false ," value ":[120 , 130 , 140]}," vesname ":{" type ":" string "," length ": 20 ," nullable ": true ," value ":[" dsff "," fgsd ", null]}," active ":{" type ":" timestamp "," nullable ": false ," value ":[" Fri Jan 01 14 : 30 : 00 2010 "," Fri Jan 01 15 : 00 : 00 2010 "," Fri Jan 01 15 : 30 : 00 2010 "]}}," events ":[{" 1 ":" Fri Jan 01 14 : 30 : 00 2010 "},{" 2 ":" Fri Jan 01 15 : 00 : 00 2010 "},{" 3 ":" Fri Jan 01 15 : 30 : 00 2010 "}]}`.

leafcount specifies the number of trajectory points in the trajectory object. Its value must be consistent with the number of spatial points in the spatial geometry object. Its value is also the same as the number of values for each attribute field. All attribute fields must have the same number of values.

Attribute field values can only be of the numeric type and are stored as double-type data. The string type is not supported.

attributes specifies trajectory attributes, including the definitions and a sequence of values for all attribute fields. It must co-exist with leafcount. Attribute definitions and requirements are as follows:

- An attribute name can contain a maximum of 60 characters.
- type: the field type. Valid values: integer, float, string, timestamp, and bool.
- length: the field length. The value varies with the field type. integer: 1, 2, 4, or 8. float: 4 or 8. string: customizable (If not specified, the default length is 64 and the maximum length is 253. The length value is the actual number of characters and excludes the end identifier.). timestamp: If not specified, the default length is 8. bool: If not specified, the default length is 1.
- nullable: indicates whether the field can be null. Valid values: true and false. Default value: true.
- value: the sequence of field values, listed in JSON array format. If a value is null, null is displayed.

events specifies trajectory events. Multiple events are listed in JSON array format. Each array element is expressed in key:value format, where key indicates the event type ID and value indicates the event time.

If only the timespan parameter or the start and end parameters are specified as time, such as in syntax 1 and syntax 2, this constructor can interpolate time points based on the number of spatial points to generate the trajectory timeline.

If all the four types of syntax do not meet your requirements, you can customize the parameters following the first six fixed parameters to create a custom ST_makeTrajectory constructor as follows:

```
CREATE   OR   REPLACE   FUNCTION   _ST_MakeTr  ajectory ( type
  leaftype ,  x   float8 [],  y   float8 [],  srid   integer ,
timespan   timestamp [],
attrs_name   cstring [],  attr1   float8 [],  attr2   float4 [],
attr3   timestamp [])
RETURNS   trajectory
```

```
 AS  '$ libdir / libpg - trajectory  xx ',' sqltr_traj  _make_all_
 array '
 LANGUAGE  ' c '  IMMUTABLE   Parallel   SAFE ;
```

Examples

```
-- ( 1 )  ST_MakeTra  jectory   with   a   timestamp   range
 select   ST_MakeTra  jectory (' STPOINT ':: leaftype ,  st_geomfro
 mtext (' LINESTRING  ( 114   35 ,  115   36 ,  116   37 )',  4326 ),
 '[ 2010 - 01 - 01   14 : 30 ,  2010 - 01 - 01   15 : 30 )':: tsrange
 , '{" leafcount ": 3 ," attributes ":{" velocity ": {" type ": "
 integer ", " length ":  2 ," nullable ":  true ," value ": [ 120 ,
 130 ,  140 ]}, " accuracy ": {" type ": " float ", " length ":  4 ,
 " nullable ":  false ," value ": [ 120 ,  130 ,  140 ]}, " bearing
 ": {" type ": " float ", " length ":  8 , " nullable ":  false ,"
 value ": [ 120 ,  130 ,  140 ]}, " vesname ": {" type ": " string ",
 " length ":  20 , " nullable ":  true ," value ": [" adsf ", " sdf
 ", " sdfff "]}, " active ": {" type ": " timestamp ", " nullable ":
 false ," value ": [" Fri   Jan   01   14 : 30 : 00   2010 ", " Fri
   Jan   01   15 : 00 : 00   2010 ", " Fri   Jan   01   15 : 30 : 00
   2010 "]}}, " events ": [{" 1 ":" Fri   Jan   01   14 : 30 : 00
 2010 "}, {" 2 ":" Fri   Jan   01   15 : 00 : 00   2010 "}, {" 3 ":"
 Fri   Jan   01   15 : 30 : 00   2010 "}]}');  st_maketra  jectory
 ----------------------------------------------------------------------------
 {" trajectory ":{" version ": 1 ," type ":" STPOINT ," leafcount
 ": 3 ," start_time ":" 2010 - 01 - 01   14 : 30 : 00 ," end_time
 ":" 2010 - 01 - 01   15 : 30 : 00 ," spatial ":" SRID = 4326 ;
 LINESTRING ( 114   35 , 115   36 , 116   37 )"," timeline ":[" 2010
 - 01 - 01   14 : 30 : 00 ," 2010 - 01 - 01   15 : 00 : 00 ," 2010
 - 01 - 01   15 : 30 : 00 "]," attributes ":{" leafcount ": 3 ,"
 velocity ":{" type ":" integer ," length ": 2 ," nullable ": true
 ," value ":[ 120 , 130 , 140 ]}," accuracy ":{" type ":" float ","
 length ": 4 ," nullable ": false ," value ":[ 120 . 0 , 130 . 0 , 140
 . 0 ]}," bearing ":{" type ":" float ," length ": 8 ," nullable ":
 false ," value ":[ 120 . 0 , 130 . 0 , 140 . 0 ]}," vesname ":{" type
 ":" string ," length ": 20 ," nullable ": true ," value ":[" adsf
 "," sdf "," sdfff "]}," active ":{" type ":" timestamp ," length ":
 8 ," nullable ": false ," value ":[" 2010 - 01 - 01   14 : 30 : 00
 "," 2010 - 01 - 01   15 : 00 : 00 ," 2010 - 01  - 01   15 : 30 : 00
 "]}}," events ":[{" 1 ":" 2010 - 01 - 01   14 : 30 : 00 "},{" 2 ":"
 2010 - 01 - 01   15 : 00 : 00 "},{" 3 ":" 2010 - 01 - 01   15 : 30 :
 00 "}]}} ( 1   row )

-- ( 2 )  ST_MakeTra  jectory   with   a   start   timestamp   and
 end   timestamp
 select   ST_MakeTra  jectory (' STPOINT ':: leaftype ,  st_geomfro
 mtext (' LINESTRING  ( 114   35 ,  115   36 ,  116   37 )',  4326
 ), ' 2010 - 01 - 01   14 : 30 ':: timestamp , ' 2010 - 01 - 01
   15 : 30 ':: timestamp , '{" leafcount ": 3 ," attributes ":{"
 velocity ": {" type ": " integer ", " length ":  2 ," nullable ":
 true ," value ": [ 120 ,  130 ,  140 ]}, " accuracy ": {" type
 ": " float ", " length ":  4 , " nullable ":  false ," value ": [
 120 ,  130 ,  140 ]}, " bearing ": {" type ": " float ", " length
 ":  8 , " nullable ":  false ," value ": [ 120 ,  130 ,  140 ]},
 " vesname ": {" type ": " string ", " length ":  20 , " nullable
 ":  true ," value ": [" adsf ", " sdf ", " sdfff "]}, " active
 ": {" type ": " timestamp ", " nullable ":  false ," value ": ["
 Fri   Jan   01   14 : 30 : 00   2010 ", " Fri   Jan   01   15 :
 00 : 00   2010 ", " Fri   Jan   01   15 : 30 : 00   2010 "]}},
 " events ": [{" 1 ":" Fri   Jan   01   14 : 30 : 00   2010 "},
 {" 2 ":" Fri   Jan   01   15 : 00 : 00   2010 "}, {" 3 ":" Fri
   Jan   01   15 : 30 : 00   2010 "}]}');  st_maketra  jectory
 ----------------------------------------------------------------------------
```

```
 {" trajectory ":{" version ": 1 ," type ":" STPOINT ," leafcount
 ": 3 ," start_time ":" 2010 - 01 - 01   14 : 30 : 00 ," end_time
 ":" 2010 - 01 - 01   15 : 30 : 00 ," spatial ":" SRID = 4326 ;
 LINESTRING ( 114   35 , 115   36 , 116   37 )"," timeline ":[" 2010
 - 01 - 01   14 : 30 : 00 ," 2010 - 01 - 01   15 : 00 : 00 ," 2010
 - 01 - 01   15 : 30 : 00 "]," attributes ":{" leafcount ": 3 ,"
 velocity ":{" type ":" integer ," length ": 2 ," nullable ": true
 ," value ":[ 120 , 130 , 140 ]}," accuracy ":{" type ":" float ","
 length ": 4 ," nullable ": false ," value ":[ 120 . 0 , 130 . 0 , 140
 . 0 ]}," bearing ":{" type ":" float ," length ": 8 ," nullable ":
 false ," value ":[ 120 . 0 , 130 . 0 , 140 . 0 ]}," vesname ":{" type
 ":" string ," length ": 20 ," nullable ": true ," value ":[" adsf
 "," sdf "," sdfff "]}," active ":{" type ":" timestamp ," length ":
 8 ," nullable ": false ," value ":[" 2010 - 01 - 01   14 : 30 : 00
 "," 2010 - 01 - 01   15 : 00 : 00 "," 2010 - 01  - 01   15 : 30 : 00
 "]}}," events ":[{" 1 ":" 2010 - 01 - 01   14 : 30 : 00 "},{" 2 ":"
 2010 - 01 - 01   15 : 00 : 00 "},{" 3 ":" 2010 - 01 - 01   15 : 30 :
 00 "}]}} ( 1   row )

-- ( 3 )  ST_MakeTra  jectory   with   a   timestamp   array
 select   ST_MakeTra  jectory (' STPOINT ':: leaftype ,  st_geomfro
 mtext (' LINESTRING  ( 114   35 ,  115   36 ,  116   37 )',  4326 ),
 ARRAY [' 2010 - 01 - 01   14 : 30 ':: timestamp , ' 2010 - 01 - 01
   15 : 00 ':: timestamp , ' 2010 - 01 - 01   15 : 30 ':: timestamp
 ], '{" leafcount ": 3 ," attributes ":{" velocity ": {" type ": "
 integer ", " length ":  2 ," nullable ":  true ," value ": [ 120 ,
 130 ,  140 ]}, " accuracy ": {" type ": " float ", " length ":  4 ,
 " nullable ":  false ," value ": [ 120 ,  130 ,  140 ]}, " bearing
 ": {" type ": " float ", " length ":  8 , " nullable ":  false ,"
 value ": [ 120 ,  130 ,  140 ]}, " vesname ": {" type ": " string ",
 " length ":  20 , " nullable ":  true ," value ": [" adsf ", " sdf
 ", " sdfff "]}, " active ": {" type ": " timestamp ", " nullable ":
 false ," value ": [" Fri   Jan   01   14 : 30 : 00   2010 ", " Fri
   Jan   01   15 : 00 : 00   2010 ", " Fri   Jan   01   15 : 30 : 00
   2010 "]}}, " events ": [{" 1 ":" Fri   Jan   01   14 : 30 : 00
 2010 "}, {" 2 ":" Fri   Jan   01   15 : 00 : 00   2010 "}, {" 3 ":"
 Fri   Jan   01   15 : 30 : 00   2010 "}]}');  st_maketra  jectory
 ------------------------------------------------------------------------
 {" trajectory ":{" version ": 1 ," type ":" STPOINT ," leafcount
 ": 3 ," start_time ":" 2010 - 01 - 01   14 : 30 : 00 ," end_time
 ":" 2010 - 01 - 01   15 : 30 : 00 ," spatial ":" SRID = 4326 ;
 LINESTRING ( 114   35 , 115   36 , 116   37 )"," timeline ":[" 2010
 - 01 - 01   14 : 30 : 00 ," 2010 - 01 - 01   15 : 00 : 00 ," 2010
 - 01 - 01   15 : 30 : 00 "]," attributes ":{" leafcount ": 3 ,"
 velocity ":{" type ":" integer ," length ": 2 ," nullable ": true
 ," value ":[ 120 , 130 , 140 ]}," accuracy ":{" type ":" float ","
 length ": 4 ," nullable ": false ," value ":[ 120 . 0 , 130 . 0 , 140
 . 0 ]}," bearing ":{" type ":" float ," length ": 8 ," nullable ":
 false ," value ":[ 120 . 0 , 130 . 0 , 140 . 0 ]}," vesname ":{" type
 ":" string ," length ": 20 ," nullable ": true ," value ":[" adsf
 "," sdf "," sdfff "]}," active ":{" type ":" timestamp ," length ":
 8 ," nullable ": false ," value ":[" 2010 - 01 - 01   14 : 30 : 00
 "," 2010 - 01 - 01   15 : 00 : 00 "," 2010 - 01  - 01   15 : 30 : 00
 "]}}," events ":[{" 1 ":" 2010 - 01 - 01   14 : 30 : 00 "},{" 2 ":"
 2010 - 01 - 01   15 : 00 : 00 "},{" 3 ":" 2010 - 01 - 01   15 : 30 :
 00 "}]}} ( 1   row )

-- ( 4 )  ST_MakeTra  jectory   with   null   attrs_json
 select   ST_MakeTra  jectory (' STPOINT ':: leaftype ,
 st_geomfro  mtext (' LINESTRING  ( 114   35 ,  115   36 ,
 116   37 )',  4326 ), '[ 2010 - 01 - 01   14 : 30 ,  2010 - 01
 - 01   15 : 30 )':: tsrange ,  null );  st_maketra  jectory
 ------------------------------------------------------------------------
 {" trajectory ":{" leafsize ": 3 ," starttime ":" Fri   Jan   01
```

```
 14 : 30 : 00   2010 "," endtime ":" Fri   Jan   01   15 : 30 : 00
   2010 "," spatial ":" LINESTRING ( 114   35 , 115   36 , 116   37
 )"," timeline ":[" Fri   Jan   01   14 : 30 : 00   2010 "," Fri   Jan
   01   15 : 00 : 00   2010 "," Fri   Jan   01   15 : 30 : 00   2010
 "]}}( 1   row ) -- ( 5 )  ST_MakeTra  jectory   make   from   points
   select   st_makeTra  jectory (' STPOINT ':: leaftype ,  ARRAY [ 1
 :: float8 ],  ARRAY [ 2 :: float8 ],  4326 ,  ARRAY [' 2010 - 01 -
 01   11 : 30 ':: timestamp ],  ARRAY [' velocity '],  ARRAY [ 1 ::
 int4 ],  NULL ,  NULL ,  NULL :: anyarray );  st_maketra  jectory
 ----------------------------------------------------------------------------
{" trajectory ":{" version ": 1 ," type ":" STPOINT "," leafcount ": 1
 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 "," end_time ":" 2010
 - 01 - 01   11 : 30 : 00 "," spatial ":" SRID = 4326 ; POINT ( 1   2
 )"," timeline ":[" 2010 - 01 - 01   11 : 30 : 00 "]," attributes ":{"
 leafcount ": 1 ," velocity ":{" type ":" integer "," length ": 4 ,"
 nullable ": true ," value ":[ 1 ]}}}}( 1   row )
```

6.2.3 ST_append

This function appends trajectory points or a sub-trajectory to a trajectory object.

Syntax

```
trajectory   ST_append ( trajectory   traj ,  geometry   spatial ,
timestamp []  timespan ,  text   str_theme_  json );
trajectory   ST_append ( trajectory   traj ,  trajectory   tail );
```

Parameters

Parameter	Description
traj	The original trajectory object.
spatial	The spatial geometry object of the appended trajectory object.
timespan	The time array of the appended trajectory object. It can be a timeline.
str_attrs_ json	The attributes of the appended trajectory object. For more information, see ST_makeTrajectory.
tail	The appended trajectory object.

Examples

```
With   traj   AS  ( Select   ST_makeTra  jectory (' STPOINT ', '
LINESTRING ( 1   1 ,  6   6 ,  9   8 )':: geometry , '[ 2010 - 01 -
01   11 : 30 ,  2010 - 01 - 01   15 : 00 )':: tsrange , '{" leafcount
": 3 ," attributes ":{" velocity ": {" type ": " integer ", " length
":  2 ," nullable ":  true ," value ": [ 120 ,  130 ,  140 ]}, "
accuracy ": {" type ": " float ", " length ":  4 , " nullable ":
false ," value ": [ 120 ,  130 ,  140 ]}, " bearing ": {" type ": "
float ", " length ":  8 , " nullable ":  false ," value ": [ 120 ,
130 ,  140 ]}, " accelerati  on ": {" type ": " string ", " length
":  20 , " nullable ":  true ," value ": [" 120 ", " 130 ", " 140
"]}, " active ": {" type ": " timestamp ", " nullable ":  false ,"
value ": [" Fri   Jan   01   14 : 30 : 00   2010 ", " Fri   Jan   01
  15 : 00 : 00   2010 ", " Fri   Jan   01   15 : 30 : 00   2010 "]}},
```

```
 " events ": [{" 1 ":" Fri   Jan   01   14 : 30 : 00   2010 "}, {" 2
 ": " Fri   Jan   01   15 : 00 : 00   2010 "}, {" 3 ": " Fri   Jan
 01   15 : 30 : 00   2010 "}]}')  a ,  ST_makeTra  jectory (' STPOINT
 ', ' LINESTRING ( 7   7 ,  3   4 ,  1   5 )':: geometry , '[ 2010
 - 01 - 02   15 : 30 ,  2010 - 01 - 02   18 : 00 )':: tsrange , '{"
 leafcount ": 3 ," attributes ":{" velocity ": {" type ": " integer ",
 " length ":  2 ," nullable ":  true ," value ": [ 121 ,  131 ,  141
 ]}, " accuracy ": {" type ": " float ", " length ":  4 , " nullable
 ":  false ," value ": [ 121 ,  131 ,  141 ]}, " bearing ": {" type
 ": " float ", " length ":  8 , " nullable ":  false ," value ": [ 121
 ,  131 ,  141 ]}, " accelerati  on ": {" type ": " string ", " length
 ":  20 , " nullable ":  true ," value ": [" 121 ", " 131 ", " 141
 "]}, " active ": {" type ": " timestamp ", " nullable ":  false ,"
 value ": [" Fri   Jan   02   14 : 30 : 00   2010 ", " Fri   Jan   02
   15 : 00 : 00   2010 ", " Fri   Jan   02   15 : 30 : 00   2010 "]}},
 " events ": [{" 1 ":" Fri   Jan   02   14 : 30 : 00   2010 "}, {" 2
 ":" Fri   Jan   02   15 : 00 : 00   2010 "}, {" 3 ":" Fri   Jan   02
   15 : 30 : 00   2010 "}]}')  b ) Select   ST_Append ( a ,  b )  from
   traj ;
                st_append


-------------------------------------------------------------------------------
 {" trajectory ":{" version ": 1 ," type ":" STPOINT ," leafcount ":
 6 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ," end_time ":"
 2010 - 01 - 02   18 : 00 : 00 ," spatial ":" LINESTRING ( 1   1 , 6
   6 , 9   8 , 7   7 , 3   4 , 1   5 )," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 ," 2010 - 01 - 01   13 : 15 : 00 ," 2010 - 01 - 01
 15 : 00 : 00 ," 2010 - 01 - 02   15 : 30 : 00 ," 2010 - 01 - 02
 16 : 45 : 00 ," 2010 - 01 - 02   18 : 00 : 00 "]," attributes ":{"
 leafcount ": 6 ," velocity ":{" type ":" integer ," length ": 2 ,"
 nullable ": true ," value ":[ 120 , 130 , 140 , 121 , 131 , 141 ]},"
 accuracy ":{" type ":" float ," length ": 4 ," nullable ": false
 ," value ":[ 120 . 0 , 130 . 0 , 140 . 0 , 121 . 0 , 131 . 0 , 141
 . 0 ]}," bearing ":{" type ":" float ," length ": 8 ," nullable ":
 false ," value ":[ 120 . 0 , 130 . 0 , 140 . 0 , 121 . 0 , 131 . 0
 , 141 . 0 ]}," accelerati  on ":{" type ":" string ," length ": 20
 ," nullable ": true ," value ":[" 120 ," 130 ," 140 ," 121 ,"
 131 ," 141 "]}," active ":{" type ":" timestamp ," length ": 8 ,"
 nullable ": false ," value ":[" 2010 - 01 - 01   14 : 30 : 00 ,"
 2010 - 01 - 01   15 : 00 : 00 ," 2010 - 01 - 01   15 : 30 : 00 ,"
 2010 - 01 - 02   14 : 30 : 00 ," 2010 - 01 - 02   15 : 00 : 00 ,"
 2010 - 01 - 02   15 : 30 : 00 "]}}," events ":[{" 1 ":" 2010 - 01 -
 01   14 : 30 : 00 "},{" 2 ":" 2010 - 01 - 01   15 : 00 : 00 "},{" 3
 ":" 2010 - 01 - 01   15 : 30 : 00 "},{" 1 ":" 2010 - 01 - 02   14 :
 30 : 00 "},{" 2 ":" 2010 - 01 - 02   15 : 00 : 00 "},{" 3 ":" 2010 -
 01 - 02   15 : 30 : 00 "}]}}
( 1   row )
```

6.3 Edit and process functions

6.3.1 ST_Compress

This function compresses a trajectory object based on the Euclidean distance offset threshold.

Syntax

```
trajectory   ST_Compres  s  ( trajectory   traj ,  float8   dist );
trajectory   ST_Compres  s  ( trajectory   traj ,  float8   dist ,
float8   angle ,  float8   accelerati  on );
trajectory   ST_Compres  s  ( trajectory   traj ,  float8   dist ,
float8   angle ,  float8   accelerati  on ,  cstring   velocity_f
ield );
```

Parameters

Parameter	Description
traj	The original trajectory object.
dist	The Euclidean distance offset threshold. After this value is specified, trajectory points whose Euclidean distance offset is greater than this value are kept, so as to maintain the spatial trend of the original trajectory object.
angle	The angle offset threshold. After this value is specified, trajectory points whose angle changes are greater than this value are kept.
accelerati on	The acceleration threshold. After this value is specified, trajectory points whose velocity changes are greater than this value are kept.
velocity_f ield	The name of the velocity attribute field in the trajectory object. The values of this attribute field are used to compute the acceleration.

Description

This function discards trajectory points based on specified thresholds to compress the trajectory object in lossy mode, and returns the compressed trajectory object.

Syntax 1: Compresses the trajectory object based on the specified spatial distance offset threshold. This function maintains the spatial trend of the original trajectory object, but may delete important trajectory points with large angle or velocity changes.

Syntax 2: Compresses the trajectory object based on the specified spatial distance offset threshold, angle offset threshold, and acceleration threshold. This function

not only maintains the spatial trend of the original trajectory object, but also keeps important trajectory points with large angle or velocity changes. You can specify any one or two of the thresholds for compression.

Syntax 3: This function achieves the same effect as syntax 2. It applies when the trajectory object contains the velocity attribute field. After the name of the velocity attribute field is specified, this method directly computes the acceleration of trajectory points based on the values of the velocity attribute field.

Examples

```
--  Create   data .
 Create   table   If   not   exists   traj_test ( id   integer ,  mmsi
   integer , traj   trajectory );
 INSERT   INTO   traj_test ( mmsi ,  traj )  VALUES ( 477027500 ,
 ST_makeTra  jectory (' STPOINT ':: leaftype , ' LINESTRING (- 179 .
 48077   51 . 72814 ,- 179 . 47416   51 . 73714 ,- 179 . 47187   51 .
 74027 ,- 179 . 46964   51 . 74325 ,- 179 . 46731   51 . 74634 ,- 179
 . 46502   51 . 74934 ,- 179 . 46183   51 . 75378 ,- 179 . 45943   51
 . 75736 ,- 179 . 45560   51 . 76273 ,- 179 . 44845   51 . 77186 ,-
 179 . 43419   51 . 78977 ,- 179 . 42595   51 . 80094 ,- 179 . 42343
   51 . 80411 ,- 179 . 42078   51 . 80719 ,- 179 . 41821   51 . 81025
 ,- 179 . 41562   51 . 81308 ,- 179 . 41259   51 . 81643 ,- 179 .
 41001   51 . 81941 ,- 179 . 40751   51 . 82223 ,- 179 . 40497   51 .
 82505 ,- 179 . 40242   51 . 82796 ,- 179 . 39981   51 . 83095 ,- 179
 . 39734   51 . 83398 ,- 179 . 39499   51 . 83709 ,- 179 . 39264   51
 . 84023 ,- 179 . 39037   51 . 84333 ,- 179 . 38699   51 . 84791 ,-
 179 . 38467   51 . 85114 ,- 179 . 38216   51 . 85439 ,- 179 . 37997
 51 . 85762 ,- 179 . 37772   51 . 86144 ,- 179 . 37474   51 . 86568 ,-
 179 . 37219   51 . 86869 ,- 179 . 36983   51 . 87156 ,- 179 . 36755
 51 . 87467 ,- 179 . 36001   51 . 88423 ,- 179 . 35754   51 . 88712 ,-
 179 . 34216   51 . 90644 ,- 179 . 33935   51 . 90995 ,- 179 . 33704
 51 . 91298 ,- 179 . 18826   52 . 10105 ,- 179 . 18096   52 . 11031 ,-
 179 . 17504   52 . 11786 ,- 179 . 16482   52 . 12996 ,- 179 . 16233
 52 . 13289 ,- 179 . 15967   52 . 13590 ,- 179 . 14599   52 . 15132 ,-
 177 . 76666   52 . 85042 ,- 177 . 48459   52 . 89898 ,- 177 . 47841
 52 . 90001 ,- 177 . 47319   52 . 90084 ,- 177 . 46251   52 . 90268 ,-
 177 . 38188   52 . 91595 ,- 177 . 37102   52 . 91765 ,- 177 . 36378
 52 . 91877 ,- 177 . 34492   52 . 92173 ,- 177 . 33217   52 . 92364 ,-
 177 . 32581   52 . 92468 ,- 177 . 31238   52 . 92697 ,- 177 . 03751
 52 . 97394 ,- 176 . 93063   52 . 99160 ,- 176 . 92406   52 . 99265 ,-
 176 . 91471   52 . 99423 ,- 176 . 90643   52 . 99554 ,- 176 . 89912
 52 . 99674 ,- 176 . 89246   52 . 99791 ,- 176 . 88342   52 . 99942 ,-
 176 . 87697   53 . 00060 ,- 176 . 86594   53 . 00256 ,- 176 . 85946
 53 . 00370 ,- 176 . 85294   53 . 00481 ,- 176 . 84640   53 . 00592 ,-
 176 . 83985   53 . 00705 ,- 176 . 83238   53 . 00830 ,- 176 . 82589
 53 . 00950 ,- 176 . 81848   53 . 01084 ,- 176 . 80553   53 . 01310 ,-
 176 . 79879   53 . 01419 ,- 176 . 79115   53 . 01548 ,- 176 . 78466
 53 . 01668 ,- 176 . 77901   53 . 01765 ,- 176 . 77256   53 . 01879 ,-
 176 . 76301   53 . 02039 ,- 176 . 75649   53 . 02141 ,- 176 . 74700
   53 . 02296 ,- 176 . 73757   53 . 02450 ,- 176 . 71683   53 . 02795
 ,- 176 . 70741   53 . 02950 ,- 176 . 68481   53 . 03327 )':: geometry
 ,  ARRAY [' 2017 - 01 - 15   09 : 06 : 39 ':: timestamp ,' 2017 - 01
 - 15   09 : 10 : 08 ':: timestamp ,' 2017 - 01 - 15   09 : 11 : 20
 ':: timestamp ,' 2017 - 01 - 15   09 : 12 : 29 ':: timestamp ,' 2017
 - 01 - 15   09 : 13 : 39 ':: timestamp ,' 2017 - 01 - 15   09 : 14
 : 48 ':: timestamp ,' 2017 - 01 - 15   09 : 16 : 28 ':: timestamp ,'
 2017 - 01 - 15   09 : 17 : 48 ':: timestamp ,' 2017 - 01 - 15   09 :
```

```
19 : 48 ':: timestamp ,' 2017 - 01 - 15   09 : 23 : 19 ':: timestamp
,' 2017 - 01 - 15   09 : 30 : 28 ':: timestamp ,' 2017 - 01 - 15
  09 : 34 : 40 ':: timestamp ,' 2017 - 01 - 15   09 : 35 : 49 ':
timestamp ,' 2017 - 01 - 15   09 : 36 : 59 ':: timestamp ,' 2017 -
01 - 15   09 : 38 : 09 ':: timestamp ,' 2017 - 01 - 15   09 : 39 : 18
':: timestamp ,' 2017 - 01 - 15   09 : 40 : 40 ':: timestamp ,' 2017
- 01 - 15   09 : 41 : 49 ':: timestamp ,' 2017 - 01 - 15   09 : 42
: 58 ':: timestamp ,' 2017 - 01 - 15   09 : 44 : 08 ':: timestamp ,'
2017 - 01 - 15   09 : 45 : 18 ':: timestamp ,' 2017 - 01 - 15   09 :
46 : 29 ':: timestamp ,' 2017 - 01 - 15   09 : 47 : 38 ':: timestamp
,' 2017 - 01 - 15   09 : 48 : 49 ':: timestamp ,' 2017 - 01 - 15
  09 : 49 : 58 ':: timestamp ,' 2017 - 01 - 15   09 : 51 : 08 ':
timestamp ,' 2017 - 01 - 15   09 : 52 : 49 ':: timestamp ,' 2017 -
01 - 15   09 : 53 : 58 ':: timestamp ,' 2017 - 01 - 15   09 : 55 : 09
':: timestamp ,' 2017 - 01 - 15   09 : 56 : 18 ':: timestamp ,' 2017
- 01 - 15   09 : 57 : 38 ':: timestamp ,' 2017 - 01 - 15   09 : 59
: 09 ':: timestamp ,' 2017 - 01 - 15   10 : 00 : 20 ':: timestamp ,'
2017 - 01 - 15   10 : 01 : 29 ':: timestamp ,' 2017 - 01 - 15   10 :
02 : 39 ':: timestamp ,' 2017 - 01 - 15   10 : 06 : 29 ':: timestamp
,' 2017 - 01 - 15   10 : 07 : 40 ':: timestamp ,' 2017 - 01 - 15
  10 : 15 : 00 ':: timestamp ,' 2017 - 01 - 15   10 : 16 : 20 ':
timestamp ,' 2017 - 01 - 15   10 : 17 : 29 ':: timestamp ,' 2017 -
01 - 15   11 : 30 : 09 ':: timestamp ,' 2017 - 01 - 15   11 : 33 : 58
':: timestamp ,' 2017 - 01 - 15   11 : 36 : 58 ':: timestamp ,' 2017
- 01 - 15   11 : 42 : 00 ':: timestamp ,' 2017 - 01 - 15   11 : 43
: 10 ':: timestamp ,' 2017 - 01 - 15   11 : 44 : 20 ':: timestamp ,'
2017 - 01 - 15   11 : 50 : 28 ':: timestamp ,' 2017 - 01 - 15   18 :
01 : 00 ':: timestamp ,' 2017 - 01 - 15   18 : 54 : 13 ':: timestamp
,' 2017 - 01 - 15   18 : 55 : 21 ':: timestamp ,' 2017 - 01 - 15
  18 : 56 : 22 ':: timestamp ,' 2017 - 01 - 15   18 : 58 : 21 ':
timestamp ,' 2017 - 01 - 15   19 : 13 : 21 ':: timestamp ,' 2017 -
01 - 15   19 : 15 : 21 ':: timestamp ,' 2017 - 01 - 15   19 : 16 : 41
':: timestamp ,' 2017 - 01 - 15   19 : 20 : 11 ':: timestamp ,' 2017
- 01 - 15   19 : 22 : 31 ':: timestamp ,' 2017 - 01 - 15   19 : 23
: 41 ':: timestamp ,' 2017 - 01 - 15   19 : 26 : 10 ':: timestamp ,'
2017 - 01 - 15   20 : 15 : 49 ':: timestamp ,' 2017 - 01 - 15   20 :
34 : 39 ':: timestamp ,' 2017 - 01 - 15   20 : 35 : 49 ':: timestamp
,' 2017 - 01 - 15   20 : 37 : 30 ':: timestamp ,' 2017 - 01 - 15
  20 : 39 : 00 ':: timestamp ,' 2017 - 01 - 15   20 : 40 : 19 ':
timestamp ,' 2017 - 01 - 15   20 : 41 : 30 ':: timestamp ,' 2017 -
01 - 15   20 : 43 : 08 ':: timestamp ,' 2017 - 01 - 15   20 : 44 : 19
':: timestamp ,' 2017 - 01 - 15   20 : 46 : 19 ':: timestamp ,' 2017
- 01 - 15   20 : 47 : 29 ':: timestamp ,' 2017 - 01 - 15   20 : 48
: 40 ':: timestamp ,' 2017 - 01 - 15   20 : 49 : 49 ':: timestamp ,'
2017 - 01 - 15   20 : 50 : 59 ':: timestamp ,' 2017 - 01 - 15   20 :
52 : 21 ':: timestamp ,' 2017 - 01 - 15   20 : 53 : 29 ':: timestamp
,' 2017 - 01 - 15   20 : 54 : 50 ':: timestamp ,' 2017 - 01 - 15
  20 : 57 : 09 ':: timestamp ,' 2017 - 01 - 15   20 : 58 : 20 ':
timestamp ,' 2017 - 01 - 15   20 : 59 : 40 ':: timestamp ,' 2017 -
01 - 15   21 : 00 : 49 ':: timestamp ,' 2017 - 01 - 15   21 : 01 : 50
':: timestamp ,' 2017 - 01 - 15   21 : 02 : 58 ':: timestamp ,' 2017
- 01 - 15   21 : 04 : 40 ':: timestamp ,' 2017 - 01 - 15   21 : 05
: 50 ':: timestamp ,' 2017 - 01 - 15   21 : 07 : 29 ':: timestamp ,'
2017 - 01 - 15   21 : 09 : 11 ':: timestamp ,' 2017 - 01 - 15   21 :
12 : 49 ':: timestamp ,' 2017 - 01 - 15   21 : 14 : 30 ':: timestamp
,' 2017 - 01 - 15   21 : 18 : 30 ':: timestamp ], '{" leafcount ":
89 ," attributes ": {" sog ": {" type ":" float "," length ": 8 ,"
nullable ": false ," value ":[ 10 . 5 , 10 . 4 , 10 . 5 , 10 . 7 , 10
. 8 , 10 . 3 , 10 . 7 , 10 . 4 , 10 . 5 , 10 . 1 , 10 . 2 , 11 . 0 ,
11 . 2 , 10 . 8 , 10 . 3 , 10 . 3 , 10 . 1 , 10 . 7 , 10 . 6 , 10 .
0 , 10 . 3 , 10 . 5 , 10 . 6 , 10 . 3 , 10 . 8 , 10 . 9 , 10 . 8 , 10
. 8 , 10 . 8 , 11 . 0 , 11 . 2 , 11 . 2 , 10 . 3 , 10 . 2 , 10 . 8 ,
10 . 0 , 10 . 4 , 10 . 7 , 10 . 2 , 10 . 6 , 9 . 1 , 10 . 2 , 10 . 1
, 9 . 7 , 10 . 4 , 10 . 6 , 9 . 9 , 12 . 3 , 12 . 1 , 12 . 0 , 12 . 0
```

```
 , 12 . 2 , 12 . 3 , 12 . 2 , 12 . 3 , 12 . 2 , 12 . 3 , 12 . 2 , 12 .
 2 , 12 . 8 , 12 . 8 , 12 . 9 , 12 . 5 , 12 . 6 , 12 . 5 , 12 . 6 , 12
 . 6 , 12 . 3 , 12 . 6 , 12 . 6 , 12 . 5 , 12 . 7 , 12 . 8 , 12 . 5 ,
 12 . 7 , 12 . 5 , 12 . 8 , 13 . 0 , 12 . 9 , 12 . 6 , 12 . 9 , 12 .
 8 , 12 . 7 , 12 . 8 , 13 . 0 , 12 . 7 , 12 . 8 , 12 . 6 , 12 . 7 ]},
 " cog ": {" type ":" float "," length ": 8 ," nullable ": false ,"
 value ":[ 23 . 3 , 25 . 7 , 25 . 9 , 23 . 6 , 25 . 3 , 24 . 1 , 23 .
 0 , 21 . 6 , 20 . 7 , 24 . 8 , 22 . 4 , 28 . 5 , 23 . 1 , 30 . 3 , 26
 . 2 , 28 . 1 , 25 . 1 , 28 . 7 , 31 . 4 , 28 . 2 , 30 . 4 , 29 . 4 ,
 29 . 2 , 23 . 0 , 25 . 1 , 25 . 1 , 23 . 5 , 22 . 7 , 27 . 1 , 23 .
 3 , 19 . 2 , 27 . 1 , 31 . 0 , 28 . 8 , 22 . 0 , 30 . 1 , 24 . 6 , 26
 . 2 , 26 . 7 , 24 . 7 , 26 . 8 , 29 . 5 , 19 . 9 , 30 . 1 , 28 . 8 ,
 28 . 7 , 30 . 0 , 74 . 2 , 69 . 1 , 75 . 2 , 81 . 3 , 81 . 3 , 80 . 1
 , 72 . 6 , 82 . 4 , 74 . 2 , 74 . 9 , 67 . 7 , 73 . 4 , 74 . 2 , 72 .
 2 , 80 . 5 , 78 . 6 , 77 . 3 , 70 . 9 , 80 . 1 , 85 . 4 , 71 . 9 , 67
 . 0 , 77 . 5 , 77 . 5 , 72 . 2 , 70 . 5 , 72 . 6 , 70 . 8 , 77 . 8 ,
 71 . 2 , 71 . 2 , 73 . 8 , 75 . 4 , 67 . 1 , 77 . 5 , 74 . 3 , 76 . 9
 , 80 . 1 , 72 . 8 , 76 . 0 , 75 . 4 , 72 . 9 ]}," heading ": {" type
 ":" float "," length ": 8 ," nullable ": false ," value ":[ 22 . 0 ,
 23 . 0 , 23 . 0 , 23 . 0 , 23 . 0 , 21 . 0 , 21 . 0 , 25 . 0 , 24 . 0
 , 26 . 0 , 25 . 0 , 27 . 0 , 28 . 0 , 29 . 0 , 31 . 0 , 30 . 0 , 28 .
 0 , 29 . 0 , 29 . 0 , 28 . 0 , 28 . 0 , 27 . 0 , 24 . 0 , 24 . 0 , 25
 . 0 , 25 . 0 , 25 . 0 , 26 . 0 , 25 . 0 , 24 . 0 , 25 . 0 , 29 . 0 ,
 31 . 0 , 28 . 0 , 29 . 0 , 31 . 0 , 28 . 0 , 29 . 0 , 29 . 0 , 29 . 0
 , 27 . 0 , 27 . 0 , 26 . 0 , 26 . 0 , 26 . 0 , 27 . 0 , 27 . 0 , 69 .
 0 , 71 . 0 , 72 . 0 , 72 . 0 , 71 . 0 , 73 . 0 , 72 . 0 , 72 . 0 , 72
 . 0 , 72 . 0 , 71 . 0 , 71 . 0 , 72 . 0 , 72 . 0 , 72 . 0 , 73 . 0 ,
 72 . 0 , 71 . 0 , 72 . 0 , 72 . 0 , 72 . 0 , 71 . 0 , 72 . 0 , 72 . 0
 , 73 . 0 , 71 . 0 , 72 . 0 , 71 . 0 , 72 . 0 , 73 . 0 , 72 . 0 , 72 .
 0 , 72 . 0 , 72 . 0 , 73 . 0 , 74 . 0 , 73 . 0 , 73 . 0 , 73 . 0 , 73
 . 0 , 73 . 0 , 73 . 0 ]}}}'));

--  Compress   the   trajectory   object .
 select   st_compres  s ( traj , 0 . 001 )  as   traj   from
 traj_test ;
           traj
-------------------------------------------------------------
 {" trajectory ":{" version ": 1 ," type ":" STPOINT ," leafcount ":
 8 ," start_time ":" 2017 - 01 - 15   09 : 06 : 39 ," end_time ":"
 2017 - 01 - 15   21 : 18 : 30 ," spatial ":" LINESTRING (- 179 .
 48077   51 . 72814 ,- 179 . 42595   51 . 80094 ,- 179 . 39734   51 .
 83398 ,- 179 . 37474   51 . 86568 ,- 179 . 17504   52 . 11786 ,- 179
 . 14599   52 . 15132 ,- 177 . 76666   52 . 85042 ,- 176 . 68481   53
 . 03327 )"," timeline ":[" 2017 - 01 - 15   09 : 06 : 39 ," 2017 -
 01 - 15   09 : 34 : 40 ," 2017 - 01 - 15   09 : 47 : 38 ," 2017 -
 01 - 15   09 : 59 : 09 ," 2017 - 01 - 15   11 : 36 : 58 ," 2017 -
 01 - 15   11 : 50 : 28 ," 2017 - 01 - 15   18 : 01 : 00 ," 2017 -
 01 - 15   21 : 18 : 30 "]," attributes ":{" leafcount ": 8 ," sog ":
{" type ":" float "," length ": 8 ," nullable ": false ," value ":[
 10 . 5 , 11 . 0 , 10 . 6 , 11 . 2 , 10 . 1 , 9 . 9 , 12 . 3 , 12 .
 7 ]}," cog ":{" type ":" float "," length ": 8 ," nullable ": false
 ," value ":[ 23 . 3 , 28 . 5 , 29 . 2 , 27 . 1 , 19 . 9 , 30 . 0 ,
 74 . 2 , 72 . 9 ]}," heading ":{" type ":" float "," length ": 8 ,"
 nullable ": false ," value ":[ 22 . 0 , 27 . 0 , 24 . 0 , 29 . 0 , 26
 . 0 , 27 . 0 , 69 . 0 , 73 . 0 ]}}}}
( 1   row )

 select   st_compres  s ( traj , 0 . 001 , null , null )  as   traj
 from   traj_test   where   traj_id = 5 ;
           traj
-------------------------------------------------------------
{" trajectory ":{" type ":" STPOINT ," leafsize ": 8 ," starttime ":"
 2017 - 01 - 15   09 : 06 : 39 ," endtime ":" 2017 - 01 - 15   21 :
 18 : 30 ," spatial ":" LINESTRING (- 179 . 48077   51 . 72814 ,- 179
```

```
 . 42595   51 . 80094 ,- 179 . 39734   51 . 83398 ,- 179 . 37474   51
 . 86568 ,- 179 . 17504   52 . 11786 ,- 179 . 14599   52 . 15132 ,-
 177 . 76666   52 . 85042 ,- 176 . 68481   53 . 03327 )"," timeline ":
[" 2017 - 01 - 15   09 : 06 : 39 "," 2017 - 01 - 15   09 : 34 : 40 ","
 2017 - 01 - 15   09 : 47 : 38 "," 2017 - 01 - 15   09 : 59 : 09 ","
 2017 - 01 - 15   11 : 36 : 58 "," 2017 - 01 - 15   11 : 50 : 28 ","
 2017 - 01 - 15   18 : 01 : 00 "," 2017 - 01 - 15   21 : 18 : 30 "],"
 themeline ":{" leafs ": 8 ," sog ":[ 10 . 5 , 11 . 0 , 10 . 6 , 11 .
 2 , 10 . 1 , 9 . 9 , 12 . 3 , 12 . 7 ]," cog ":[ 23 . 3 , 28 . 5 , 29
 . 2 , 27 . 1 , 19 . 9 , 30 . 0 , 74 . 2 , 72 . 9 ]," heading ":[ 22 .
 0 , 27 . 0 , 24 . 0 , 29 . 0 , 26 . 0 , 27 . 0 , 69 . 0 , 73 . 0 ]}}}
( 1   row )

 select   st_compres  s ( traj , 0 . 001 , 5 , 0 . 3 )  as   traj
 from   traj_test ;
           traj
------------------------------------------------------------
{" trajectory ":{" version ": 1 ," type ":" STPOINT "," leafcount ":
 13 ," start_time ":" 2017 - 01 - 15   09 : 06 : 39 "," end_time ":"
 2017 - 01 - 15   21 : 18 : 30 "," spatial ":" LINESTRING (- 179 .
 48077   51 . 72814 ,- 179 . 42595   51 . 80094 ,- 179 . 39734   51 .
 83398 ,- 179 . 37474   51 . 86568 ,- 179 . 35754   51 . 88712 ,- 179
 . 18826   52 . 10105 ,- 179 . 17504   52 . 11786 ,- 179 . 14599   52
 . 15132 ,- 177 . 76666   52 . 85042 ,- 177 . 47841   52 . 90001 ,-
 177 . 47319   52 . 90084 ,- 176 . 83238   53 . 0083 ,- 176 . 68481
   53 . 03327 )"," timeline ":[" 2017 - 01 - 15   09 : 06 : 39 ","
 2017 - 01 - 15   09 : 34 : 40 "," 2017 - 01 - 15   09 : 47 : 38 ","
 2017 - 01 - 15   09 : 59 : 09 "," 2017 - 01 - 15   10 : 07 : 40 ","
 2017 - 01 - 15   11 : 30 : 09 "," 2017 - 01 - 15   11 : 36 : 58 ","
 2017 - 01 - 15   11 : 50 : 28 "," 2017 - 01 - 15   18 : 01 : 00 ","
 2017 - 01 - 15   18 : 55 : 21 "," 2017 - 01 - 15   18 : 56 : 22 ","
 2017 - 01 - 15   20 : 52 : 21 "," 2017 - 01 - 15   21 : 18 : 30 "],"
 attributes ":{" leafcount ": 13 ," sog ":{" type ":" float "," length
 ": 8 ," nullable ": false ," value ":[ 10 . 5 , 11 . 0 , 10 . 6 , 11
 . 2 , 10 . 4 , 9 . 1 , 10 . 1 , 9 . 9 , 12 . 3 , 12 . 0 , 12 . 0 , 12
 . 5 , 12 . 7 ]}," cog ":{" type ":" float "," length ": 8 ," nullable
 ": false ," value ":[ 23 . 3 , 28 . 5 , 29 . 2 , 27 . 1 , 24 . 6 ,
 26 . 8 , 19 . 9 , 30 . 0 , 74 . 2 , 75 . 2 , 81 . 3 , 72 . 6 , 72 . 9
 ]}," heading ":{" type ":" float "," length ": 8 ," nullable ": false
 ," value ":[ 22 . 0 , 27 . 0 , 24 . 0 , 29 . 0 , 28 . 0 , 27 . 0 , 26
 . 0 , 27 . 0 , 69 . 0 , 72 . 0 , 72 . 0 , 72 . 0 , 73 . 0 ]}}}}
( 1   row )

 select   st_compres  s ( traj , 0 . 001 , 5 , 1 . 1 ,' sog ')  as
 traj   from   traj_test ;

            traj
------------------------------------------------------------
{" trajectory ":{" version ": 1 ," type ":" STPOINT "," leafcount ":
 10 ," start_time ":" 2017 - 01 - 15   09 : 06 : 39 "," end_time ":"
 2017 - 01 - 15   21 : 18 : 30 "," spatial ":" LINESTRING (- 179 .
 48077   51 . 72814 ,- 179 . 42595   51 . 80094 ,- 179 . 39734   51 .
 83398 ,- 179 . 37474   51 . 86568 ,- 179 . 33704   51 . 91298 ,- 179
 . 18826   52 . 10105 ,- 179 . 17504   52 . 11786 ,- 179 . 14599   52
 . 15132 ,- 177 . 76666   52 . 85042 ,- 176 . 68481   53 . 03327 )","
 timeline ":[" 2017 - 01 - 15   09 : 06 : 39 "," 2017 - 01 - 15   09
 : 34 : 40 "," 2017 - 01 - 15   09 : 47 : 38 "," 2017 - 01 - 15   09
 : 59 : 09 "," 2017 - 01 - 15   10 : 17 : 29 "," 2017 - 01 - 15   11
 : 30 : 09 "," 2017 - 01 - 15   11 : 36 : 58 "," 2017 - 01 - 15   11
 : 50 : 28 "," 2017 - 01 - 15   18 : 01 : 00 "," 2017 - 01 - 15   21
 : 18 : 30 "]," attributes ":{" leafcount ": 10 ," sog ":{" type ":"
 float "," length ": 8 ," nullable ": false ," value ":[ 10 . 5 , 11
 . 0 , 10 . 6 , 11 . 2 , 10 . 6 , 9 . 1 , 10 . 1 , 9 . 9 , 12 . 3 , 12
 . 7 ]}," cog ":{" type ":" float "," length ": 8 ," nullable ": false
```

```
," value ":[ 23 . 3 , 28 . 5 , 29 . 2 , 27 . 1 , 24 . 7 , 26 . 8 , 19
. 9 , 30 . 0 , 74 . 2 , 72 . 9 ]}," heading ":{" type ":" float ","
length ": 8 ," nullable ": false ," value ":[ 22 . 0 , 27 . 0 , 24
. 0 , 29 . 0 , 29 . 0 , 27 . 0 , 26 . 0 , 27 . 0 , 69 . 0 , 73 . 0
]}}}}
```

6.3.2 ST_CompressSED

This function compresses a trajectory object based on the offset threshold of the synchronous Euclidean distance (SED).

Syntax

```
trajectory   ST_Compres  sSed  ( trajectory   traj ,  float8   dist
);
```

Parameters

Parameter	Description
traj	The original trajectory object.
dist	The SED offset threshold. After this value is specified, trajectory points whose SED offset is greater than this value are kept, so as to maintain the spatial trend of the original trajectory object.

Description

This function computes the SED of trajectory points, discards the points whose SED offset is smaller than the SED offset threshold to compress the trajectory object in lossy mode, and returns the compressed trajectory object.

Examples

```
select   st_compres  sSED ( traj ,  0 . 001 )  as   traj   from
traj_test ;
           traj
-----------------------------------------------------------------
{" trajectory ":{" version ": 1 ," type ":" STPOINT ," leafcount ":
12 ," start_time ":" 2017 - 01 - 15   09 : 06 : 39 ," end_time ":"
2017 - 01 - 15   21 : 18 : 30 ," spatial ":" LINESTRING (- 179 .
48077   51 . 72814 ,- 179 . 42595   51 . 80094 ,- 179 . 39734   51 .
83398 ,- 179 . 37474   51 . 86568 ,- 179 . 18826   52 . 10105 ,- 179
. 16482   52 . 12996 ,- 179 . 14599   52 . 15132 ,- 177 . 76666   52
. 85042 ,- 177 . 47319   52 . 90084 ,- 177 . 31238   52 . 92697 ,-
177 . 03751   52 . 97394 ,- 176 . 68481   53 . 03327 )"," timeline ":
[" 2017 - 01 - 15   09 : 06 : 39 ," 2017 - 01 - 15   09 : 34 : 40 ","
2017 - 01 - 15   09 : 47 : 38 ," 2017 - 01 - 15   09 : 59 : 09 ","
2017 - 01 - 15   11 : 30 : 09 ," 2017 - 01 - 15   11 : 42 : 00 ","
2017 - 01 - 15   11 : 50 : 28 ," 2017 - 01 - 15   18 : 01 : 00 ","
2017 - 01 - 15   18 : 56 : 22 ," 2017 - 01 - 15   19 : 26 : 10 ","
2017 - 01 - 15   20 : 15 : 49 ," 2017 - 01 - 15   21 : 18 : 30 "],"
attributes ":{" leafcount ": 12 ," sog ":{" type ":" float ," length
": 8 ," nullable ": false ," value ":[ 10 . 5 , 11 . 0 , 10 . 6 , 11
```

```
 . 2 , 9 . 1 , 9 . 7 , 9 . 9 , 12 . 3 , 12 . 0 , 12 . 2 , 12 . 8 , 12
 . 7 ]},"  cog ":{" type ":" float "," length ": 8 ," nullable ": false
 ," value ":[ 23 . 3 , 28 . 5 , 29 . 2 , 27 . 1 , 26 . 8 , 30 . 1 ,
 30 . 0 , 74 . 2 , 81 . 3 , 73 . 4 , 74 . 2 , 72 . 9 ]},"  heading ":{"
 type ":" float "," length ": 8 ," nullable ": false ," value ":[ 22 .
 0 , 27 . 0 , 24 . 0 , 29 . 0 , 27 . 0 , 26 . 0 , 27 . 0 , 69 . 0 , 72
 . 0 , 71 . 0 , 72 . 0 , 73 . 0 ]}}}}
( 1   row )
```

6.3.3 ST_attrDeduplicate

This function removes trajectory points whose values for the specified attribute field are duplicate from a trajectory object and returns the trajectory object after deduplication. The start and end trajectory points of the trajectory object must be kept.

Syntax

```
trajectory   ST_attrDed  uplicate ( trajectory   traj ,  cstring
attr_field  _name );
```

Parameters

Parameter	Description
traj	The trajectory object.
attr_field _name	The name of the specified attribute field.

Examples

```
select   st_attrDed  uplicate ( ST_makeTra  jectory (' STPOINT '::
leaftype , ' LINESTRING (- 179 . 48077   51 . 72814 ,- 179 . 46731
51 . 74634 ,- 179 . 46502   51 . 74934 ,- 179 . 46183   51 . 75378 ,-
179 . 45943   51 . 75736 ,- 179 . 45560   51 . 76273 ,- 179 . 44845
51 . 77186 ,- 179 . 43419   51 . 78977 ,- 179 . 41259   51 . 81643 ,-
179 . 41001   51 . 81941 ,- 179 . 40751   51 . 82223 ,- 179 . 40497
  51 . 82505 ,- 179 . 40242   51 . 82796 ,- 179 . 39981   51 . 83095
,- 179 . 39734   51 . 83398 ,- 179 . 39499   51 . 83709 )':: geometry
,  ARRAY [' 2017 - 01 - 15   09 : 06 : 39 ':: timestamp ,' 2017 -
01 - 15   09 : 13 : 39 ',' 2017 - 01 - 15   09 : 14 : 48 ',' 2017 -
01 - 15   09 : 16 : 28 ',' 2017 - 01 - 15   09 : 17 : 48 ',' 2017 -
01 - 15   09 : 19 : 48 ',' 2017 - 01 - 15   09 : 23 : 19 ',' 2017 -
01 - 15   09 : 30 : 28 ',' 2017 - 01 - 15   09 : 34 : 40 ',' 2017 -
01 - 15   09 : 36 : 59 ',' 2017 - 01 - 15   09 : 38 : 09 ',' 2017
- 01 - 15   09 : 39 : 18 ',' 2017 - 01 - 15   09 : 40 : 40 ',' 2017
- 01 - 15   09 : 47 : 38 ',' 2017 - 01 - 15   09 : 48 : 49 ',' 2017
- 01 - 15   21 : 18 : 30 '], '{" leafcount ":  16 , " attributes ":
{" heading ": {" type ": " float ", " length ":  4 , " nullable ":
false ," value ": [ 23 . 0 , 23 . 0 , 23 . 0 , 23 . 0 , 21 . 0 , 21 .
0 , 72 . 0 , 72 . 0 , 72 . 0 , 72 . 0 , 73 . 0 , 74 . 0 , 73 . 0 , 73
. 0 , 73 . 0 , 73 . 0 ]}}}'),' heading ')  as   traj ;
```

```
                                                        traj



-------------------------------------------------------------------------------------
 {" trajectory ":{" version ": 1 ," type ":" STPOINT "," leafcount ":
 6 ," start_time ":" 2017 - 01 - 15   09 : 17 : 48 "," end_time ":"
 2017 - 01 - 15   21 : 18 : 30 "," spatial ":" LINESTRING (- 179 .
 45943   51 . 75736 ,- 179 . 44845   51 . 77
 186 ,- 179 . 40751   51 . 82223 ,- 179 . 40497   51 . 82505 ,- 179 .
 39499   51 . 83709 )"," timeline ":[" 2017 - 01 - 15   09 : 17 : 48
 "," 2017 - 01 - 15   09 : 23 : 19 "," 2017 - 01 - 15   09 : 38 : 09
 "," 2017 - 01 - 15   09 : 39 : 18 "," 2017 - 01 - 15   21 : 18 : 3
 0 "]," attributes ":{" leafcount ": 6 ," heading ":{" type ":" float
 "," length ": 4 ," nullable ": false ," value ":[ 23 . 0 , 21 . 0 ,
 72 . 0 , 73 . 0 , 74 . 0 , 73 . 0 ]}}}}
( 1   row )
```

6.3.4 ST_sort

This function sorts the timeline of a trajectory object in ascending order and returns the trajectory object after sorting.

Syntax

```
trajectory   ST_sort ( trajectory   traj );
```

Parameters

Parameter	Description
traj	The trajectory object.

Examples

```
select   st_sort ( ST_makeTra  jectory (' STPOINT ':: leaftype , '
LINESTRING (- 179 . 48077   51 . 72814 ,- 179 . 46731   51 . 74634 ,-
179 . 46502   51 . 74934 ,- 179 . 46183   51 . 75378 ,- 179 . 45943
51 . 75736 ,- 179 . 45560   51 . 76273 ,- 179 . 44845   51 . 77186 ,-
179 . 43419   51 . 78977 ,- 179 . 41259   51 . 81643 ,- 179 . 41001
51 . 81941 ,- 179 . 40751   51 . 82223 ,- 179 . 40497   51 . 82505 ,-
179 . 40242   51 . 82796 ,- 179 . 39981   51 . 83095 ,- 179 . 39734
  51 . 83398 ,- 179 . 39499   51 . 83709 )':: geometry ,  ARRAY ['
2017 - 01 - 15   09 : 06 : 39 ':: timestamp ,' 2017 - 01 - 15   09 :
14 : 48 ':: timestamp ,' 2017 - 01 - 15   09 : 13 : 39 ':: timestamp
,' 2017 - 01 - 15   09 : 16 : 28 ':: timestamp ,' 2017 - 01 - 15
  09 : 19 : 48 ':: timestamp ,' 2017 - 01 - 15   09 : 17 : 48 '::
timestamp ,' 2017 - 01 - 15   09 : 23 : 19 ':: timestamp ,' 2017 -
01 - 15   09 : 34 : 40 ':: timestamp ,' 2017 - 01 - 15   09 : 30 : 28
':: timestamp ,' 2017 - 01 - 15   09 : 36 : 59 ':: timestamp ,' 2017
- 01 - 15   09 : 38 : 09 ':: timestamp ,' 2017 - 01 - 15   09 : 39
: 18 ':: timestamp ,' 2017 - 01 - 15   09 : 40 : 40 ':: timestamp ,'
2017 - 01 - 15   09 : 47 : 38 ':: timestamp ,' 2017 - 01 - 15   21 :
18 : 30 ':: timestamp ,' 2017 - 01 - 15   09 : 48 : 49 ':: timestamp
], '{" leafcount ":  16 , " attributes ": {" heading ": {" type ": "
integer ", " length ":  4 , " nullable ":  false ," value ":[ 0 , 1 ,
2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 ]}}}'));
```

```
                                                                    st_sort


-------------------------------------------------------------------------------
 {" trajectory ":{" version ": 1 ," type ":" STPOINT ," leafcount ":
 16 ," start_time ":" 2017 - 01 - 15   09 : 06 : 39 ," end_time ":"
 2017 - 01 - 15   21 : 18 : 30 ," spatial ":" LINESTRING (- 179 .
 48077   51 . 72814 ,- 179 . 46502   51 . 7
 4934 ,- 179 . 46731   51 . 74634 ,- 179 . 46183   51 . 75378 ,- 179 .
 4556   51 . 76273 ,- 179 . 45943   51 . 75736 ,- 179 . 44845   51 .
 77186 ,- 179 . 41259   51 . 81643 ,- 179 . 43419   51 . 78977 ,- 179
 . 41001   51 . 81941 ,- 179 . 40751   51 . 82223 ,-
 179 . 40497   51 . 82505 ,- 179 . 40242   51 . 82796 ,- 179 . 39981
   51 . 83095 ,- 179 . 39499   51 . 83709 ,- 179 . 39734   51 . 83398
 )"," timeline ":[" 2017 - 01 - 15   09 : 06 : 39 ," 2017 - 01 - 15
 09 : 13 : 39 ," 2017 - 01 - 15   09 : 14 : 48 ," 2017 -
 01 - 15   09 : 16 : 28 ," 2017 - 01 - 15   09 : 17 : 48 ," 2017 -
 01 - 15   09 : 19 : 48 ," 2017 - 01 - 15   09 : 23 : 19 ," 2017 -
 01 - 15   09 : 30 : 28 ," 2017 - 01 - 15   09 : 34 : 40 ," 2017 -
 01 - 15   09 : 36 : 59 ," 2017 - 01 - 15   09 : 38 : 09 ," 2017 -
 01 - 15   09 :
 39 : 18 ," 2017 - 01 - 15   09 : 40 : 40 ," 2017 - 01 - 15   09 :
 47 : 38 ," 2017 - 01 - 15   09 : 48 : 49 ," 2017 - 01 - 15   21 :
 18 : 30 "]," attributes ":{" leafcount ": 16 ," heading ":{" type ":"
 integer "," length ": 4 ," nullable ": false ," val
 ue ":[ 0 , 2 , 1 , 3 , 5 , 4 , 6 , 8 , 7 , 9 , 10 , 11 , 12 , 13 , 15
 , 14 ]}}}}
( 1   row )
```

6.3.5 ST_deviation

This function returns the deviation of the processed trajectory object from the original trajectory object.

Syntax

```
 trajectory   ST_deviati  on ( trajectory   traj ,  trajectory
 after_oper  _traj );
```

Parameters

Parameter	Description
traj	The original trajectory object.
after_oper _traj	The processed trajectory object, such as the compressed trajectory object.

Examples

```
 select   st_deviati  on ( traj ,  st_compres  s ( traj , 0 . 001 ))
 from   traj_test ;
    st_deviati  on
---------------------
 0 . 0091917734  5596219
```

```
( 1   row )
```

6.4 Attribute metadata

6.4.1 ST_attrDefinition

This function returns the definitions of attributes in a trajectory object.

Syntax

```
t   ext   ST_attrDef  inition ( trajectory   traj );
```

Parameters

Parameter	Description
traj	The trajectory object.

Examples

```
With   traj   AS  (
  select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
  11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
Select   st_attrDef  inition ( a )  from   traj ;


st_attrdef  inition


----------------------------------------------------------------------------------
{" size ": 5 ," velocity ":{" type ":" integer "," length ": 4 ,"
nullable ": true }," speed ":{" type ":" float "," length ": 8 ,"
nullable ": true }," angle ":{" type ":" string "," length ": 64 ,"
nullable ": true }," tngel2 ":{" type ":" timestamp "," length ": 8
," nullable ": true }," bearing ":{" type ":" bool "," length ": 1 ,"
nullable ": true }}
```

```
( 1   row )
```

6.4.2 ST_attrSize

This function returns the number of attributes in a trajectory object.

Syntax

```
 integer   ST_attrSiz  e ( trajectory   traj );
```

Parameters

Parameter	Description
traj	The trajectory object.

Examples

```
 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
 Select   st_attrSiz  e ( a )  from   traj ;
  st_attrsiz  e
-------------
            5
( 1   row )
```

6.4.3 ST_attrName

This function returns the name of a trajectory attribute field or the names of all trajectory attribute fields.

Syntax

```
 text []  ST_attrNam  e ( trajectory   traj );
 text   ST_attrNam  e ( trajectory   traj ,  integer   index );
```

Parameters

Parameter	Description
traj	The trajectory object.
i ndex	The attribute index. The value starts from 0.

Examples

```
 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
 Select   st_attrNam  e ( a )  from   traj ;
              st_attrnam  e
---------------------------------------
 { velocity , speed , angle , tngel2 , bearing }
( 1   row )

 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
 Select   st_attrNam  e ( a ,  0 )  from   traj ;
  st_attrnam  e
-------------
  velocity
( 1   row )
```

6.4.4 ST_attrType

This function returns the data type of a trajectory attribute field.

Syntax

```
 text   ST_attrTyp  e ( trajectory   traj ,  integer   index );
 text   ST_attrTyp  e ( trajectory   traj ,  text   name );
```

Parameters

Parameter	Description
traj	**The trajectory object.**
i ndex	**The attribute index.**

Parameter	Description
name	The name of the attribute field.

Description

This function returns one of the following strings to indicate the data type of the trajectory attribute field:

- integer
- float
- string
- timestamp
- bool

Examples

```
 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
 Select   st_attrTyp  e ( a ,  0 )  from   traj ;
  st_attrtyp  e
-------------
  integer
( 1   row )

 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
 Select   st_attrTyp  e ( a , ' velocity ')  from   traj ;
  st_attrtyp  e
-------------
  integer
```

```
( 1   row )
```

6.4.5 ST_attrLength

This function returns the defined length of a trajectory attribute field.

Syntax

```
integer   ST_attrLen  gth ( trajectory   traj ,  integer   index );
integer   ST_attrLen  gth ( trajectory   traj ,  text   name );
```

Parameters

Parameter	Description
traj	The trajectory object.
i ndex	The attribute index.
name	The name of the attribute field.

Examples

```
 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
 Select   st_attrLen  gth ( a ,  0 )  from   traj ;
  st_attrlen  gth
---------------
              4
( 1   row )

 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
 Select   st_attrLen  gth ( a , ' velocity ')  from   traj ;
  st_attrlen  gth
```

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

6.4.6 ST_attrNullable

This function returns true if a trajectory attribute field can be null.

Syntax

```
 bool   ST_attrNul  lable ( trajectory   traj ,  integer   index );
 bool   ST_attrNul  lable ( trajectory   traj ,  text   name );
```

Parameters

Parameter	Description
traj	The trajectory object.
i ndex	The attribute index.
name	The name of the attribute field.

Examples

```
 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
 Select   st_attrNul  lable ( a , ' velocity ')  from   traj ;
  st_attrnul  lable
-----------------
  t
( 1   row )

 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
```

```
 Select   st_attrNul  lable ( a ,  1 )  from   traj ;
  st_attrnul  lable
-----------------
  t
( 1   row )
```

6.5 Event functions

6.5.1 ST_addEvent

This function adds an event to a trajectory object.

Syntax

```
 trajectory   ST_addEven  t ( trajectory   traj ,  integer
 event_type ,  timestamp   event_time );
```

Parameters

Parameter	Description
traj	**The trajectory object.**
event_type	**The event type ID.**
event_time	**The event timestamp.**

Description

Event type IDs are predefined. For example, 1000 indicates unlocking and 2000 indicates locking.

Examples

```
 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 "," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
 Select   st_addeven  t ( a ,  1 , ' 2010 - 01 - 01   11 : 30 : 00 ')
 from   traj ;
```

```
                                                        st_addeven  t
```

```
-------------------------------------------------------------------------------
 {" trajectory ":{" version ": 1 ," type ":" STPOINT ," leafcount
 ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ," end_time
 ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID = 4326 ;
 LINESTRING ( 1   1 , 3   5 )," timeline ":[" 2010 - 01 - 01   11
 : 30 : 00 ," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":{"
 leafcount ": 2 ," velocity ":{" type ":" integer ," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float ," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string ," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}," tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool ," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}," events ":[{" 1 ":"
 2010 - 01 - 01   11 : 30 : 00 "}]}}
( 1   row )
```

6.5.2 ST_eventTimes

This function returns the time of all events in a trajectory object.

Syntax

```
 timestamp []  ST_eventTi  mes ( trajectory   traj );
```

Parameters

Parameter	Description
traj	The trajectory object.

Examples

```
 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 ," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer ," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float ," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string ," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool ," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}," events ":[{" 1 ":"
 Fri   Jan   01   14 : 30 : 00   2010 "},{" 2 ":" Fri   Jan   01   14
 : 30 : 00   2010 "}]}}':: trajectory   a )
 Select   st_eventTi  mes ( a )  from   traj ;
                 st_eventti  mes
-----------------------------------------------
 {" 2010 - 01 - 01   14 : 30 : 00 ," 2010 - 01 - 01   14 : 30 : 00 "}
```

```
( 1   row )
```

6.5.3 ST_eventTime

This function returns the time of an event with the specified index in a trajectory object.

Syntax

```
timestamp   ST_eventTi  me ( trajectory   traj ,  integer   index );
```

Parameters

Parameter	Description
traj	The trajectory object.
index	The event index.

Examples

```
With   traj   AS  (
  select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
  11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
nullable ": true ," value ":[ null , true ]}}," events ":[{" 1 ":"
Fri   Jan   01   14 : 30 : 00   2010 "}]}}':: trajectory   a )
Select   st_eventTi  me ( a ,  0 )  from   traj ;
    st_eventti  me
---------------------
  2010 - 01 - 01   14 : 30 : 00
( 1   row )
```

6.5.4 ST_eventTypes

This function returns the type IDs of all events in a trajectory object.

Syntax

```
integer []  ST_eventTy  pes (  trajectory   traj );
```

Parameters

Parameter	Description
traj	The trajectory object.

Examples

```
 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}," events ":[{" 1 ":"
 Fri   Jan   01   14 : 30 : 00   2010 "},{" 2 ":" Fri   Jan   01   14
 : 30 : 00   2010 "}]}}':: trajectory   a )
 Select   st_eventTy  pes ( a )  from   traj ;
  st_eventty  pes
---------------
 { 1 , 2 }
```

6.5.5 ST_eventType

This function returns the type ID of an event with the specified index in a trajectory object.

Syntax

```
 integer   ST_eventTy  pe ( trajectory   traj ,  integer   index );
```

Parameters

Parameter	Description
traj	**The trajectory object.**
index	**The event index.**

Examples

```
 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}," events ":[{" 1 ":"
```

```
 Fri   Jan   01   14 : 30 : 00   2010 "},{" 2 ":" Fri   Jan   01   14
 : 30 : 00   2010 "}]}}':: trajectory   a )
 Select   st_eventTy  pe ( a ,  0 )  from   traj ;
  st_eventty  pe
--------------
             1
```

6.6 Attribute functions

6.6.1 ST_startTime

This function returns the start time of a trajectory object.

Syntax

```
 timestamp   ST_startTi  me ( trajectory   traj );
```

Parameters

Parameter	Description
traj	The trajectory object.

Examples

```
 Select   ST_startTi  me ( traj )  From   traj_table ;
```

6.6.2 ST_endTime

This function returns the end time of a trajectory object.

Syntax

```
 timestamp   ST_endTime ( trajectory   traj );
```

Parameters

Parameter	Description
traj	The trajectory object.

Examples

```
Select   ST_endTime ( traj )  From   traj_table ;
```

6.6.3 ST_trajectorySpatial

This function returns the spatial geometry object of a trajectory object.

Syntax

```
geometry   ST_traject  orySpatial ( trajectory   traj );
```

Parameters

Parameter	Description
traj	The trajectory object.

Examples

```
Select   AsText ( ST_traject  orySpatial ( traj ))  FROM   traj_table
;
```

6.6.4 ST_trajectoryTemporal

This function returns the timeline of a trajectory object.

Syntax

```
timeline   ST_traject  oryTempora  l ( trajectory   traj );
```

Parameters

Parameter	Description
traj	The trajectory object.

Examples

```
Select   ST_traject  oryTempora  l ( traj )  From   traj_table ;
```

6.6.5 ST_trajAttrs

This function returns the attributes of a trajectory object.

Syntax

```
themeline   ST_trajAtt  rs ( trajectory   traj );
```

Parameters

Parameter	Description
traj	The trajectory object.

Examples

```
Select   ST_trajAtt  rs ( traj )  From   traj_table ;
```

6.6.6 ST_attrIntMax

This function returns the maximum value of an integer-type attribute field.

Syntax

```
int8   ST_attrInt  Max ( trajectory   traj ,  cstring   attr_field
_name );
```

Parameters

Parameter	Description
traj	The trajectory object.
attr_field _name	The name of the attribute field.

Examples

```
select   st_attrInt  Max ( ST_makeTra  jectory (' STPOINT '::
leaftype , ' LINESTRING (- 179 . 48077   51 . 72814 ,- 179 . 46731
51 . 74634 ,- 179 . 46502   51 . 74934 ,- 179 . 46183   51 . 75378 ,-
179 . 45943   51 . 75736 ,- 179 . 45560   51 . 76273 ,- 179 . 44845
51 . 77186 ,- 179 . 43419   51 . 78977 ,- 179 . 41259   51 . 81643 ,-
179 . 41001   51 . 81941 ,- 179 . 40751   51 . 82223 ,- 179 . 40497
  51 . 82505 ,- 179 . 40242   51 . 82796 ,- 179 . 39981   51 . 83095
,- 179 . 39734   51 . 83398 ,- 179 . 39499   51 . 83709 )':: geometry
,  ARRAY [' 2017 - 01 - 15   09 : 06 : 39 ':: timestamp ,' 2017 -
01 - 15   09 : 14 : 48 ',' 2017 - 01 - 15   09 : 13 : 39 ',' 2017 -
01 - 15   09 : 16 : 28 ',' 2017 - 01 - 15   09 : 19 : 48 ',' 2017 -
```

```
 01 - 15   09 : 17 : 48 ',' 2017 - 01 - 15   09 : 23 : 19 ',' 2017 -
 01 - 15   09 : 34 : 40 ',' 2017 - 01 - 15   09 : 30 : 28 ',' 2017 -
 01 - 15   09 : 36 : 59 ',' 2017 - 01 - 15   09 : 38 : 09 ',' 2017 -
 01 - 15   09 : 39 : 18 ',' 2017 - 01 - 15   09 : 40 : 40 ',' 2017 -
 01 - 15   09 : 47 : 38 ',' 2017 - 01 - 15   21 : 18 : 30 ',' 2017 -
 01 - 15   09 : 48 : 49 '], '{" leafcount ":  16 , " attributes ": {"
 heading ": {" type ": " integer ", " length ":  4 , " nullable ":
 false ," value ":[ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 ,
 12 , 13 , 14 , 15 ]}}}'), ' heading ');
  st_attrint  max
---------------
             15
```

6.6.7 ST_attrIntMin

This function returns the minimum value of an integer-type attribute field.

Syntax

```
 int8   ST_attrInt  Min ( trajectory   traj ,  cstring   attr_field
 _name );
```

Parameters

Parameter	Description
traj	The trajectory object.
attr_field _name	The name of the attribute field.

Examples

```
 select   st_attrInt  Min ( ST_makeTra  jectory (' STPOINT '::
 leaftype , ' LINESTRING (- 179 . 48077   51 . 72814 ,- 179 . 46731
 51 . 74634 ,- 179 . 46502   51 . 74934 ,- 179 . 46183   51 . 75378 ,-
 179 . 45943   51 . 75736 ,- 179 . 45560   51 . 76273 ,- 179 . 44845
 51 . 77186 ,- 179 . 43419   51 . 78977 ,- 179 . 41259   51 . 81643 ,-
 179 . 41001   51 . 81941 ,- 179 . 40751   51 . 82223 ,- 179 . 40497
   51 . 82505 ,- 179 . 40242   51 . 82796 ,- 179 . 39981   51 . 83095
 ,- 179 . 39734   51 . 83398 ,- 179 . 39499   51 . 83709 )':: geometry
 ,  ARRAY [' 2017 - 01 - 15   09 : 06 : 39 ':: timestamp ,' 2017 -
 01 - 15   09 : 14 : 48 ',' 2017 - 01 - 15   09 : 13 : 39 ',' 2017 -
 01 - 15   09 : 16 : 28 ',' 2017 - 01 - 15   09 : 19 : 48 ',' 2017 -
 01 - 15   09 : 17 : 48 ',' 2017 - 01 - 15   09 : 23 : 19 ',' 2017 -
 01 - 15   09 : 34 : 40 ',' 2017 - 01 - 15   09 : 30 : 28 ',' 2017 -
 01 - 15   09 : 36 : 59 ',' 2017 - 01 - 15   09 : 38 : 09 ',' 2017 -
 01 - 15   09 : 39 : 18 ',' 2017 - 01 - 15   09 : 40 : 40 ',' 2017 -
 01 - 15   09 : 47 : 38 ',' 2017 - 01 - 15   21 : 18 : 30 ',' 2017 -
 01 - 15   09 : 48 : 49 '], '{" leafcount ":  16 , " attributes ": {"
 heading ": {" type ": " integer ", " length ":  4 , " nullable ":
 false ," value ":[ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 ,
 12 , 13 , 14 , 15 ]}}}'), ' heading ');
  st_attrint  min
---------------
```

```
                0
```

6.6.8 ST_attrIntAverage

This function returns the average value of an integer-type attribute field.

Syntax

```
int8   ST_attrInt  Average ( trajectory   traj ,  cstring
attr_field  _name );
```

Parameters

Parameter	Description
traj	The trajectory object.
attr_field _name	The name of the attribute field.

Examples

```
select   st_attrInt  Average ( ST_makeTra  jectory (' STPOINT '::
leaftype , ' LINESTRING (- 179 . 48077   51 . 72814 ,- 179 . 46731
51 . 74634 ,- 179 . 46502   51 . 74934 ,- 179 . 46183   51 . 75378 ,-
179 . 45943   51 . 75736 ,- 179 . 45560   51 . 76273 ,- 179 . 44845
51 . 77186 ,- 179 . 43419   51 . 78977 ,- 179 . 41259   51 . 81643 ,-
179 . 41001   51 . 81941 ,- 179 . 40751   51 . 82223 ,- 179 . 40497
  51 . 82505 ,- 179 . 40242   51 . 82796 ,- 179 . 39981   51 . 83095
,- 179 . 39734   51 . 83398 ,- 179 . 39499   51 . 83709 )':: geometry
,  ARRAY [' 2017 - 01 - 15   09 : 06 : 39 ':: timestamp ,' 2017 - 01
- 15   09 : 14 : 48 ',' 2017 - 01 - 15   09 : 13 : 39 ',' 2017 - 01
- 15   09 : 16 : 28 ',' 2017 - 01 - 15   09 : 19 : 48 ',' 2017 - 01
- 15   09 : 17 : 48 ',' 2017 - 01 - 15   09 : 23 : 19 ',' 2017 - 01
- 15   09 : 34 : 40 ',' 2017 - 01 - 15   09 : 30 : 28 ',' 2017 - 01
- 15   09 : 36 : 59 ',' 2017 - 01 - 15   09 : 38 : 09 ',' 2017 - 01
- 15   09 : 39 : 18 ',' 2017 - 01 - 15   09 : 40 : 40 ',' 2017 - 01
- 15   09 : 47 : 38 ',' 2017 - 01 - 15   21 : 18 : 30 ',' 2017 - 01
- 15   09 : 48 : 49 '], '{" leafcount ":  16 , " attributes ": {"
heading ": {" type ": " integer ", " length ":  4 , " nullable ":
false ," value ":[ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 ,
12 , 13 , 14 , 15 ]}}}'), ' heading ');
 st_attrint  average
--------------------
```

```
                    7
```

6.6.9 ST_attrFloatMax

This function returns the maximum value of a float-type attribute field.

Syntax

```
float8   ST_attrFlo  atMax ( trajectory   traj ,  cstring
attr_field  _name );
```

Parameters

Parameter	Description
traj	The trajectory object.
attr_field _name	The name of the attribute field.

Examples

```
select   st_attrFlo  atMax ( ST_makeTra  jectory (' STPOINT '::
leaftype , ' LINESTRING (- 179 . 48077   51 . 72814 ,- 179 . 46731
51 . 74634 ,- 179 . 46502   51 . 74934 ,- 179 . 46183   51 . 75378 ,-
179 . 45943   51 . 75736 ,- 179 . 45560   51 . 76273 ,- 179 . 44845
51 . 77186 ,- 179 . 43419   51 . 78977 ,- 179 . 41259   51 . 81643 ,-
179 . 41001   51 . 81941 ,- 179 . 40751   51 . 82223 ,- 179 . 40497
  51 . 82505 ,- 179 . 40242   51 . 82796 ,- 179 . 39981   51 . 83095
,- 179 . 39734   51 . 83398 ,- 179 . 39499   51 . 83709 )':: geometry
,  ARRAY [' 2017 - 01 - 15   09 : 06 : 39 ':: timestamp ,' 2017 -
01 - 15   09 : 13 : 39 ',' 2017 - 01 - 15   09 : 14 : 48 ',' 2017
- 01 - 15   09 : 16 : 28 ',' 2017 - 01 - 15   09 : 17 : 48 ',' 2017
- 01 - 15   09 : 19 : 48 ',' 2017 - 01 - 15   09 : 23 : 19 ',' 2017
- 01 - 15   09 : 30 : 28 ',' 2017 - 01 - 15   09 : 34 : 40 ',' 2017
- 01 - 15   09 : 36 : 59 ',' 2017 - 01 - 15   09 : 38 : 09 ',' 2017
- 01 - 15   09 : 39 : 18 ',' 2017 - 01 - 15   09 : 40 : 40 ',' 2017
- 01 - 15   09 : 47 : 38 ',' 2017 - 01 - 15   09 : 48 : 49 ',' 2017
- 01 - 15   21 : 18 : 30 '], '{" leafcount ":  16 , " attributes ":
{" heading ": {" type ": " float ", " length ":  4 , " nullable ":
false ," value ": [ 23 . 0 , 23 . 0 , 23 . 0 , 23 . 0 , 21 . 0 , 21 .
0 , 72 . 0 , 72 . 0 , 72 . 0 , 72 . 0 , 73 . 0 , 74 . 0 , 73 . 0 , 73
. 0 , 73 . 0 , 73 . 0 ]}}}'),' heading ');
 st_attrflo  atmax
-----------------
```

```
                    74
```

6.6.10 ST_attrFloatMin

This function returns the minimum value of a float-type attribute field.

Syntax

```
float8   ST_attrFlo  atMax ( trajectory   traj ,  cstring
attr_field  _name );
```

Parameters

Parameter	Description
traj	The trajectory object.
attr_field _name	The name of the attribute field.

Examples

```
select   st_attrFlo  atMin ( ST_makeTra  jectory (' STPOINT '::
leaftype , ' LINESTRING (- 179 . 48077   51 . 72814 ,- 179 . 46731
51 . 74634 ,- 179 . 46502   51 . 74934 ,- 179 . 46183   51 . 75378 ,-
179 . 45943   51 . 75736 ,- 179 . 45560   51 . 76273 ,- 179 . 44845
51 . 77186 ,- 179 . 43419   51 . 78977 ,- 179 . 41259   51 . 81643 ,-
179 . 41001   51 . 81941 ,- 179 . 40751   51 . 82223 ,- 179 . 40497
  51 . 82505 ,- 179 . 40242   51 . 82796 ,- 179 . 39981   51 . 83095
,- 179 . 39734   51 . 83398 ,- 179 . 39499   51 . 83709 )':: geometry
,  ARRAY [' 2017 - 01 - 15   09 : 06 : 39 ':: timestamp ,' 2017 -
01 - 15   09 : 13 : 39 ',' 2017 - 01 - 15   09 : 14 : 48 ',' 2017
- 01 - 15   09 : 16 : 28 ',' 2017 - 01 - 15   09 : 17 : 48 ',' 2017
- 01 - 15   09 : 19 : 48 ',' 2017 - 01 - 15   09 : 23 : 19 ',' 2017
- 01 - 15   09 : 30 : 28 ',' 2017 - 01 - 15   09 : 34 : 40 ',' 2017
- 01 - 15   09 : 36 : 59 ',' 2017 - 01 - 15   09 : 38 : 09 ',' 2017
- 01 - 15   09 : 39 : 18 ',' 2017 - 01 - 15   09 : 40 : 40 ',' 2017
- 01 - 15   09 : 47 : 38 ',' 2017 - 01 - 15   09 : 48 : 49 ',' 2017
- 01 - 15   21 : 18 : 30 '], '{" leafcount ":  16 , " attributes ":
{" heading ": {" type ": " float ", " length ":  4 , " nullable ":
false ," value ": [ 23 . 0 , 23 . 0 , 23 . 0 , 23 . 0 , 21 . 0 , 21 .
0 , 72 . 0 , 72 . 0 , 72 . 0 , 72 . 0 , 73 . 0 , 74 . 0 , 73 . 0 , 73
. 0 , 73 . 0 , 73 . 0 ]}}}'),' heading ');
 st_attrflo  atmin
-----------------
```

```
                21
```

6.6.11 ST_attrFloatAverage

This function returns the average value of a float-type attribute field.

Syntax

```
float8   ST_attrFlo  atAverage ( trajectory   traj ,  cstring
attr_field  _name );
```

Parameters

Parameter	Description
traj	The trajectory object.
attr_field _name	The name of the attribute field.

Examples

```
select   st_attrFlo  atAverage ( ST_makeTra  jectory (' STPOINT '::
leaftype , ' LINESTRING (- 179 . 48077   51 . 72814 ,- 179 . 46731
51 . 74634 ,- 179 . 46502   51 . 74934 ,- 179 . 46183   51 . 75378 ,-
179 . 45943   51 . 75736 ,- 179 . 45560   51 . 76273 ,- 179 . 44845
51 . 77186 ,- 179 . 43419   51 . 78977 ,- 179 . 41259   51 . 81643 ,-
179 . 41001   51 . 81941 ,- 179 . 40751   51 . 82223 ,- 179 . 40497
  51 . 82505 ,- 179 . 40242   51 . 82796 ,- 179 . 39981   51 . 83095
,- 179 . 39734   51 . 83398 ,- 179 . 39499   51 . 83709 )':: geometry
,  ARRAY [' 2017 - 01 - 15   09 : 06 : 39 ':: timestamp ,' 2017 -
01 - 15   09 : 13 : 39 ',' 2017 - 01 - 15   09 : 14 : 48 ',' 2017 -
01 - 15   09 : 16 : 28 ',' 2017 - 01 - 15   09 : 17 : 48 ',' 2017 -
01 - 15   09 : 19 : 48 ',' 2017 - 01 - 15   09 : 23 : 19 ',' 2017 -
01 - 15   09 : 30 : 28 ',' 2017 - 01 - 15   09 : 34 : 40 ',' 2017 -
01 - 15   09 : 36 : 59 ',' 2017 - 01 - 15   09 : 38 : 09 ',' 2017 -
01 - 15   09 : 39 : 18 ',' 2017 - 01 - 15   09 : 40 : 40 ',' 2017 -
01 - 15   09 : 47 : 38 ',' 2017 - 01 - 15   09 : 48 : 49 ',' 2017 -
01 - 15   21 : 18 : 30 '], '{" leafcount ":  16 , " attributes ":
{" heading ": {" type ": " float ", " length ":  4 , " nullable ":
false ," value ": [ 23 . 0 , 23 . 0 , 23 . 0 , 23 . 0 , 21 . 0 , 21 .
0 , 72 . 0 , 72 . 0 , 72 . 0 , 72 . 0 , 73 . 0 , 74 . 0 , 73 . 0 , 73
. 0 , 73 . 0 , 73 . 0 ]}}}'),' heading ');
 st_attrflo  ataverage
---------------------
              53 . 8125
```

```
( 1   row )
```

6.6.12 ST_leafType

This function returns the type of leaves in a trajectory object.

Syntax

```
leaftype   ST_leafTyp  e ( trajectory   traj );
```

Parameters

Parameter	Description
traj	The trajectory object.

Only the ST_POINT type is supported.

Examples

```
Select   ST_leafTyp  e ( raj )  from   traj_table ;
```

6.6.13 ST_leafCount

This function returns the number of leaves in a trajectory object.

Syntax

```
integer   ST_leafCou  nt ( trajectory   traj );
```

Parameters

Parameter	Description
traj	The trajectory object.

Examples

```
Select ST_leafCou nt ( raj ) from traj_table ;
```

6.6.14 ST_Duration

This function returns the duration of a trajectory object.

Syntax

```
interval ST_Duratio n ( trajectory traj );
```

Parameters

Parameter	Description
traj	The trajectory object.

Examples

```
Select ST_Duratio n ( traj ) from traj_table ;
```

6.6.15 ST_timeAtPoint

This function returns a set of time points when a trajectory object passes through the specified spatial point.

Syntax

```
timestamp [] ST_timeAtP oint ( trajectory traj , geometry g );
```

Parameters

Parameter	Description
traj	The trajectory object.
g	The spatial point.

6.6.16 ST_pointAtTime

This function returns the spatial geometry object of a trajectory object at the specified time point.

Syntax

```
geometry   ST_pointAt  Time ( trajectory   traj ,  timestamp   t );
```

Parameters

Parameter	Description
traj	The trajectory object.
t	The time point.

This function returns a spatial point.

Examples

```
Select   ST_pointAt  Time ( traj , ' 2010 - 1 - 11   23 : 40 : 00 ')
from   traj_table ;
```

6.6.17 ST_velocityAtTime

This function returns the value of the velocity attribute field at the specified time point.

Syntax

```
float8   ST_velocit  yAtTime  ( trajectory   traj ,  timestamp   t );
```

Parameters

Parameter	Description
traj	The trajectory object.
t	The time point.

Examples

```
Select ST_velocityAtTime ( traj ) From traj_table ;
```

6.6.18 ST_accelerationAtTime

This function returns the value of the acceleration attribute field at the specified time point.

Syntax

```
float8 ST_accelerationAtTime ( trajectory traj , timestamp t );
```

Parameters

Parameter	Description
traj	The trajectory object.
t	The specified time point.

Examples

```
Select ST_accelerationAtTime ( traj ) From traj_table ;
```

6.6.19 ST_timeToDistance

This function returns a line chart in which the time is the x-axis and the Euclidean distance is the y-axis.

Syntax

```
geometry ST_timeToDistance ( trajectory traj );
```

Parameters

Parameter	Description
traj	The trajectory object.

Examples

```
Select ST_timeToD istance ( traj ) from traj_table ;
```

6.6.20 ST_timeAtDistance

This function returns the time point when the movement arrives at a spatial point from the start point for the specified distance.

Syntax

```
timestamp [] ST_timeAtD istance ( trajectory traj , float8 d
);
```

Parameters

Parameter	Description
traj	The trajectory object.
d	The distance of movement.

This function returns an array of timestamps. Multiple spatial points may be at the same distance from the start point.

Examples

```
Select ST_timeAtD istance ( traj , 100 ) from traj_table ;
```

6.6.21 ST_cumulativeDistanceAtTime

This function returns the cumulative distance of movement from the start point to the spatial point arrived at the specified time point.

Syntax

```
float8 ST_cumulat iveDistanc eAtTime ( trajectory traj ,
timestamp t )
;
```

Parameters

Parameter	Description
traj	The trajectory object.
t	The time point.

Examples

```
Select  ST_cumulat  iveDistanc  eAtTime ( traj , ' 2011 - 11 - 1
04 : 30 : 00 ')  from  traj_table ;
```

6.6.22 ST_timeAtCumulativeDistance

This function returns the time point when the movement arrives at a spatial point from the start point for the specified cumulative distance.

Syntax

```
timestamp  ST_timeAtC  umulativeD  istance ( trajectory  traj ,
float  d )
;
```

Parameters

Parameter	Description
traj	**The trajectory object.**
d	**The cumulative distance of movement.**

Examples

```
Select  ST_timeAtC  umulativeD  istance ( traj ,  100 . 0 )  from
traj_table ;
```

6.6.23 ST_subTrajectory

This function returns the sub-trajectory of a trajectory object within the specified time range.

Syntax

```
trajectory  ST_subTraj  ectory ( trajectory  traj ,  timestamp
starttime ,  timestamp  endtime );

trajectory  ST_subTraj  ectory ( trajectory  traj ,  tsrange
range );
```

Parameters

Parameter	Description
traj	**The trajectory object.**
starttime	**The start time.**
endtime	**The end time.**

Parameter	Description
range	The time range.

Examples

```
Select   ST_subTraj  ectory ( traj , ' 2010 - 1 - 11   02 : 45 : 30
', ' 2010 - 1 - 11   03 : 00 : 00 ')  FROM   traj_table ;
```

6.6.24 ST_subTrajectorySpatial

This function returns the spatial geometry object of a trajectory object within the specified time range.

Syntax

```
geometry   ST_subTraj  ectorySpat  ial ( trajectory   traj ,
timestamp   starttime ,  timestamp   endtime );
geometry   ST_subTraj  ectorySpat  ial ( trajectory   traj ,  tsrange
  range );
```

Parameters

Parameter	Description
traj	The trajectory object.
starttime	The start time.
endtime	The end time.
range	The time range.

Examples

```
Select   ST_subTraj  ectorySpat  ial ( traj , ' 2010 - 1 - 11   02 :
45 : 30 ', ' 2010 - 1 - 11   03 : 00 : 00 ')  FROM   traj_table ;
```

6.6.25 ST_samplingInterval

This function returns the sampling interval.

Syntax

```
interval   ST_samplin  gInterval ( trajectory   traj );
```

Parameters

Parameter	Description
traj	The trajectory object.

Examples

```
Select ST_samplinIgInterval ( traj ) from traj_table ;
```

6.6.26 ST_trajAttrsAsText

This function returns an array of values for a text-type attribute field of a trajectory object.

Syntax

```
text [] st_trajAtt rsAsText ( trajectory traj , text attr_name
);
```

Parameters

Parameter	Description
traj	The trajectory object.
a ttr_name	The name of the attribute field.

Examples

```
With traj AS (
  select '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
leafcount ": 2 ," start_time ":" 2010 - 01 - 01 11 : 30 : 00 ","
end_time ":" 2010 - 01 - 01 12 : 30 : 00 ," spatial ":" SRID =
4326 ; LINESTRING ( 1 1 , 3 5 )"," timeline ":[" 2010 - 01 - 01
  11 : 30 : 00 "," 2010 - 01 - 01 12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01 12 :
30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
nullable ": true ," value ":[ null , true ]}}}}':: trajectory a )
Select st_trajAtt rsAsText ( a , ' angle ') from traj ;
 st_trajatt rsastext
--------------------
 { test , NULL }
( 1 row )

With traj AS (
  select '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
leafcount ": 2 ," start_time ":" 2010 - 01 - 01 11 : 30 : 00 ","
end_time ":" 2010 - 01 - 01 12 : 30 : 00 ," spatial ":" SRID =
4326 ; LINESTRING ( 1 1 , 3 5 )"," timeline ":[" 2010 - 01 - 01
  11 : 30 : 00 "," 2010 - 01 - 01 12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01 12 :
```

```
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
 Select   st_trajAtt  rsAsText ( a , ' tngel2 ')  from   traj ;
       st_trajatt  rsastext
------------------------------
 {" 2010 - 01 - 01   12 : 30 : 00 ", NULL }
( 1   row )

 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 "," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
 Select   st_trajAtt  rsAsText ( a , ' bearing ')  from   traj ;
  st_trajatt  rsastext
--------------------
 { NULL , t }
( 1   row )
```

6.6.27 ST_trajAttrsAsInteger

This function returns an array of values for an integer-type attribute field of a trajectory object.

Syntax

```
 integer []  st_trajAtt  rsAsIntege  r ( trajectory   traj ,  text
 attr_name );
```

Parameters

Parameter	Description
traj	The trajectory object.
a ttr_name	The name of the attribute field.

Examples

```
 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 "," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
```

```
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool ," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
 Select   st_trajAtt  rsAsIntege  r ( a , ' speed ')  from   traj ;
  st_trajatt  rsasintege  r
----------------------
 { NULL , 1 }
( 1   row )
```

6.6.28 ST_trajAttrsAsDouble

This function returns an array of values for a double-type attribute field of a trajectory object.

Syntax

```
 float8 []  st_trajAtt  rsAsDouble ( trajectory   traj ,  text
 attr_name );
```

Parameters

Parameter	Description
traj	The trajectory object.
a ttr_name	The name of the attribute field.

Examples

```
 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer ," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float ," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string ," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool ," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
 Select   st_trajAtt  rsAsDouble ( a , ' velocity ')  from   traj ;
  st_trajatt  rsasdouble
----------------------
 { 1 , NULL }
```

```
( 1   row )
```

6.6.29 ST_trajAttrsAsBool

This function returns an array of values for a Boolean-type attribute field of a trajectory object.

Syntax

```
bool []  st_trajAtt  rsAsBool ( trajectory   traj ,  text   attr_name
);
```

Parameters

Parameter	Description
traj	The trajectory object.
a ttr_name	The name of the attribute field.

Examples

```
 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 "," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
 Select   st_trajAtt  rsAsBool ( a , ' velocity ')  from   traj ;
  st_trajatt  rsasbool
--------------------
 { t , NULL }
```

```
( 1   row )
```

6.6.30 ST_trajAttrsAsTimestamp

This function returns an array of values for a timestamp-type attribute field of a trajectory object.

Syntax

```
timestamp []  st_trajAtt  rsAsTimest  amp ( trajectory   traj ,  text
  attr_name );
```

Parameters

Parameter	Description
traj	The trajectory object.
a ttr_name	The name of the attribute field.

Examples

```
With   traj   AS  (
  select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
  11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
," nullable ": true ," value ":[ 1 , null ]}," speed ":{" type ":"
float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
Select   st_trajAtt  rsAsTimest  amp ( a , ' tngel2 ')  from   traj ;
   st_trajatt  rsastimest  amp
------------------------------
{" 2010 - 01 - 01   12 : 30 : 00 ", NULL }
```

6.6.31 ST_attrIntFilter

This function filters all trajectory points to obtain the trajectory points whose values for the specified attribute field meet a filter condition based on a fixed value or a value range and returns an array of the attribute field values of these points.

Syntax

```
int8 []  ST_attrInt  Filter ( trajectory   traj ,  cstring
attr_field  _name , cstring   operator ,  int8   value );
```

```
int8 []  ST_attrInt  Filter ( trajectory   traj ,  cstring
attr_field  _name , cstring   operator ,  int8   value1 ,  int8
value2 );
```

Parameters

Parameter	Description
traj	The trajectory object.
attr_field _name	The name of the specified attribute field.
operator	The filter operator. Valid values: '=', '!=', '>', '<', '>=', '<=', '[]', '()', '[)', and '()'.
value and value1	The fixed value of the attribute field and the minimum fixed value of the attribute field.
value2	The maximum fixed value of the attribute field.

Description

This function supports only integer-type attribute fields.

Examples

```
create   table   traj ( id   integer ,  traj   trajectory );
insert   into   traj   values ( ST_makeTra  jectory (' STPOINT '::
leaftype , ' LINESTRING (- 179 . 48077   51 . 72814 ,- 179 . 46731
51 . 74634 ,- 179 . 46502   51 . 74934 ,- 179 . 46183   51 . 75378 ,-
179 . 45943   51 . 75736 ,- 179 . 45560   51 . 76273 ,- 179 . 44845
51 . 77186 ,- 179 . 43419   51 . 78977 ,- 179 . 41259   51 . 81643 ,-
179 . 41001   51 . 81941 ,- 179 . 40751   51 . 82223 ,- 179 . 40497
  51 . 82505 ,- 179 . 40242   51 . 82796 ,- 179 . 39981   51 . 83095
,- 179 . 39734   51 . 83398 ,- 179 . 39499   51 . 83709 )':: geometry
,  ARRAY [' 2017 - 01 - 15   09 : 06 : 39 ':: timestamp ,' 2017 -
01 - 15   09 : 14 : 48 ',' 2017 - 01 - 15   09 : 13 : 39 ',' 2017 -
01 - 15   09 : 16 : 28 ',' 2017 - 01 - 15   09 : 19 : 48 ',' 2017 -
01 - 15   09 : 17 : 48 ',' 2017 - 01 - 15   09 : 23 : 19 ',' 2017 -
01 - 15   09 : 34 : 40 ',' 2017 - 01 - 15   09 : 30 : 28 ',' 2017 -
01 - 15   09 : 36 : 59 ',' 2017 - 01 - 15   09 : 38 : 09 ',' 2017 -
01 - 15   09 : 39 : 18 ',' 2017 - 01 - 15   09 : 40 : 40 ',' 2017 -
01 - 15   09 : 47 : 38 ',' 2017 - 01 - 15   21 : 18 : 30 ',' 2017 -
01 - 15   09 : 48 : 49 '], '{" leafcount ":  16 , " attributes ": {"
heading ": {" type ": " integer ", " length ":  4 , " nullable ":
false ," value ":[ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 ,
12 , 13 , 14 , 15 ]}}}'));


select   st_attrInt  Filter ( traj , ' heading ', '>',  5 )  from
traj   where   id  =  1 ;
      st_attrint  filter
----------------------------
 { 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 }
( 1   row )
```

```
 select   st_attrInt  Filter ( traj , ' heading ', '>=',  5 )  from
 traj   where   id  =  1 ;
       st_attrint  filter
------------------------------
 { 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 }
( 1   row )

 select   st_attrInt  Filter ( traj , ' heading ', '<',  5 )  from
 traj   where   id  =  1 ;
  st_attrint  filter
------------------
 { 0 , 1 , 2 , 3 , 4 }
( 1   row )

 select   st_attrInt  Filter ( traj , ' heading ', '<=',  5 )  from
 traj   where   id  =  1 ;
  st_attrint  filter
------------------
 { 0 , 1 , 2 , 3 , 4 , 5 }
( 1   row )

 select   st_attrInt  Filter ( traj , ' heading ', '=',  5 )  from
 traj   where   id  =  1 ;
  st_attrint  filter
------------------
 { 5 }
( 1   row )

 select   st_attrInt  Filter ( traj , ' heading ', '! =',  5 )  from
 traj   where   id  =  1 ;
           st_attrint  filter
----------------------------------------
 { 0 , 1 , 2 , 3 , 4 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 }
( 1   row )

 select   st_attrInt  Filter ( traj , ' heading ', '()',  5 , 8 )
 from   traj   where   id  =  1 ;
  st_attrint  filter
------------------
 { 6 , 7 }
( 1   row )

 select   st_attrInt  Filter ( traj , ' heading ', '(]',  5 , 8 )
 from   traj   where   id  =  1 ;
  st_attrint  filter
------------------
 { 6 , 7 , 8 }
( 1   row )

 select   st_attrInt  Filter ( traj , ' heading ', '[)',  5 , 8 )
 from   traj   where   id  =  1 ;
  st_attrint  filter
------------------
 { 5 , 6 , 7 }
( 1   row )

 select   st_attrInt  Filter ( traj , ' heading ', '[]',  5 , 8 )
 from   traj   where   id  =  1 ;
  st_attrint  filter
------------------
 { 5 , 6 , 7 , 8 }
```

```
( 1   row )
```

6.6.32 ST_attrFloatFilter

This function filters all trajectory points to obtain the trajectory points whose values for the specified attribute field meet a filter condition based on a fixed value or a value range and returns an array of the attribute field values of these points.

Syntax

```
float8 []  ST_attrFlo  atFilter ( trajectory   traj ,  cstring
attr_field  _name , cstring   operator ,  float8   value );
float8 []  ST_attrFlo  atFilter ( trajectory   traj ,  cstring
attr_field  _name , cstring   operator ,  float8   value1 ,  float8
value2 );
```

Parameters

Parameter	Description
traj	The trajectory object.
attr_field _name	The name of the specified attribute field.
operator	The filter operator. Valid values: '=', '!=', '>', '<', '>=', '<=', '[]', '(]', '[)', and '()'.
value and value1	The fixed value of the attribute field and the minimum fixed value of the attribute field.
value2	The maximum fixed value of the attribute field.

Description

This function supports only float-type attribute fields.

Examples

```
create   table   traj ( id   integer ,  traj   trajectory );
insert   into   traj   values ( 2 , ST_makeTra  jectory (' STPOINT
':: leaftype , ' LINESTRING (- 179 . 48077   51 . 72814 ,- 179 .
46731   51 . 74634 ,- 179 . 46502   51 . 74934 ,- 179 . 46183   51
. 75378 ,- 179 . 45943   51 . 75736 ,- 179 . 45560   51 . 76273 ,-
179 . 44845   51 . 77186 ,- 179 . 43419   51 . 78977 ,- 179 . 41259
  51 . 81643 ,- 179 . 41001   51 . 81941 ,- 179 . 40751   51 . 82223
,- 179 . 40497   51 . 82505 ,- 179 . 40242   51 . 82796 ,- 179 .
39981   51 . 83095 ,- 179 . 39734   51 . 83398 ,- 179 . 39499   51
. 83709 )':: geometry ,  ARRAY [' 2017 - 01 - 15   09 : 06 : 39 ':
timestamp ,' 2017 - 01 - 15   09 : 14 : 48 ',' 2017 - 01 - 15   09 :
13 : 39 ',' 2017 - 01 - 15   09 : 16 : 28 ',' 2017 - 01 - 15   09 :
19 : 48 ',' 2017 - 01 - 15   09 : 17 : 48 ',' 2017 - 01 - 15   09 :
23 : 19 ',' 2017 - 01 - 15   09 : 34 : 40 ',' 2017 - 01 - 15   09 :
30 : 28 ',' 2017 - 01 - 15   09 : 36 : 59 ',' 2017 - 01 - 15   09 :
38 : 09 ',' 2017 - 01 - 15   09 : 39 : 18 ',' 2017 - 01 - 15   09 :
```

```
 40 : 40 ',' 2017 - 01 - 15   09 : 47 : 38 ',' 2017 - 01 - 15   21 :
 18 : 30 ',' 2017 - 01 - 15   09 : 48 : 49 '], '{" leafcount ":  16 ,
 " attributes ": {" heading ": {" type ": " float ", " length ":  8 ,
 " nullable ":  false ," value ":[ 23 . 0 , 23 . 0 , 23 . 0 , 23 . 0 ,
 21 . 0 , 21 . 0 , 72 . 0 , 72 . 0 , 72 . 0 , 72 . 0 , 73 . 0 , 74 . 0
 , 73 . 0 , 73 . 0 , 73 . 0 , 73 . 0 ]}}}'));


 select   st_attrFlo  atFilter ( traj , ' heading ', '>',  23 . 0 )
 from   traj   where   id  =  2 ;
        st_attrflo  atfilter
--------------------------------
 { 72 , 72 , 72 , 72 , 73 , 74 , 73 , 73 , 73 , 73 }
( 1   row )
```

6.6.33 ST_attrTimestampFilter

This function filters all trajectory points to obtain the trajectory points whose values for the specified attribute field meet a filter condition based on a fixed value or a value range and returns an array of the attribute field values of these points.

Syntax

```
 timestamp []  ST_attrTim  estampFilt  er ( trajectory   traj ,
 cstring   attr_field  _name , cstring   operator ,  timestamp   value
 );
 timestamp []  ST_attrTim  estampFilt  er ( trajectory   traj ,
 cstring   attr_field  _name , cstring   operator ,  timestamp
 value1 ,  timestamp   value2 );
```

Parameters

Parameter	Description
traj	The trajectory object.
attr_field _name	The name of the specified attribute field.
operator	The filter operator. Valid values: '=', '!=', '>', '<', '>=', '<=', '[]', '(]', '[)', and '()'.
value and value1	The fixed value of the attribute field and the minimum fixed value of the attribute field.
value2	The maximum fixed value of the attribute field.

Examples

```
 insert   into   traj   values ( 3 , ST_makeTra  jectory (' STPOINT
 ':: leaftype ,  st_geomfro  mtext (' LINESTRING  ( 114   35 ,  115
   36 ,  116   37 )',  4326 ),  ARRAY [' 2010 - 01 - 01   14 : 30 ':: 
 timestamp , ' 2010 - 01 - 01   15 : 00 ', ' 2010 - 01 - 01   15 : 30
 '], '{" leaftype ":  3 , " attributes ": {" heading ": {" type ": "
 timestamp ", " nullable ":  false ," value ":[" Fri   Jan   01   14
```

```
 : 30 : 00   2010 ", " Fri   Jan   01   15 : 00 : 00   2010 ", " Fri
 Jan   01   15 : 30 : 00   2010 "]}}}'));
 select   st_attrTim  estampFilt  er ( traj , ' heading ', '>', ' Fri
   Jan   01   15 : 00 : 00   2010 ':: timestamp )  from   traj   where
   id  =  3 ;
  st_attrtim  estampfilt  er
-------------------------
{" 2010 - 01 - 01   15 : 30 : 00 "}
( 1   row )
```

6.6.34 ST_attrNullFilter

This function filters all trajectory points to obtain the trajectory points whose values are null for the specified attribute field and returns a new trajectory object that is composed of these points.

Syntax

```
 trajectory   ST_attrNul  lFilter ( trajectory   traj ,  cstring
 attr_field  _name );
 trajectory   ST_attrNul  lFilter ( trajectory   traj ,  cstring
 attr_field  _name );
```

Parameters

Parameter	Description
traj	**The trajectory object.**
attr_field _name	**The name of the specified attribute field.**

Description

This function supports all types of attribute fields.

Examples

```
 select   st_attrNul  lFilter ( ST_makeTra  jectory (' STPOINT ':: 
 leaftype , ' LINESTRING (- 179 . 48077   51 . 72814 ,- 179 . 46731
 51 . 74634 ,- 179 . 46502   51 . 74934 ,- 179 . 46183   51 . 75378 ,-
 179 . 45943   51 . 75736 ,- 179 . 45560   51 . 76273 ,- 179 . 44845
 51 . 77186 ,- 179 . 43419   51 . 78977 ,- 179 . 41259   51 . 81643 ,-
 179 . 41001   51 . 81941 ,- 179 . 40751   51 . 82223 ,- 179 . 40497
   51 . 82505 ,- 179 . 40242   51 . 82796 ,- 179 . 39981   51 . 83095
 ,- 179 . 39734   51 . 83398 ,- 179 . 39499   51 . 83709 )':: geometry
 ,  ARRAY [' 2017 - 01 - 15   09 : 06 : 39 ':: timestamp ,' 2017 -
 01 - 15   09 : 13 : 39 ',' 2017 - 01 - 15   09 : 14 : 48 ',' 2017
 - 01 - 15   09 : 16 : 28 ',' 2017 - 01 - 15   09 : 17 : 48 ',' 2017
 - 01 - 15   09 : 19 : 48 ',' 2017 - 01 - 15   09 : 23 : 19 ',' 2017
 - 01 - 15   09 : 30 : 28 ',' 2017 - 01 - 15   09 : 34 : 40 ',' 2017
 - 01 - 15   09 : 36 : 59 ',' 2017 - 01 - 15   09 : 38 : 09 ',' 2017
 - 01 - 15   09 : 39 : 18 ',' 2017 - 01 - 15   09 : 40 : 40 ',' 2017
 - 01 - 15   09 : 47 : 38 ',' 2017 - 01 - 15   09 : 48 : 49 ',' 2017
 - 01 - 15   21 : 18 : 30 '], '{" leafcount ":  16 , " attributes ":
 {" heading ": {" type ": " float ", " length ":  4 , " nullable ":
```

```
 true ," value ": [ 23 . 0 , 23 . 0 , 23 . 0 , null , 21 . 0 , 21 . 0
 , null , 72 . 0 , 72 . 0 , null , 73 . 0 , 74 . 0 , 73 . 0 , 73 . 0 ,
 null , 73 . 0 ]}}}'),' heading ');


                              st_attrnul  lfilter


-------------------------------------------------------------------------------
 {" trajectory ":{" version ": 1 ," type ":" STPOINT ," leafcount ":
 4 ," start_time ":" 2017 - 01 - 15   09 : 16 : 28 ," end_time ":"
 2017 - 01 - 15   09 : 48 : 49 ," spatial ":" LINESTRING (- 179 .
 46183   51 . 75378 ,- 179 . 44845   51 . 77
 186 ,- 179 . 41001   51 . 81941 ,- 179 . 39734   51 . 83398 )","
 timeline ":[" 2017 - 01 - 15   09 : 16 : 28 ," 2017 - 01 - 15   09
 : 23 : 19 ," 2017 - 01 - 15   09 : 36 : 59 ," 2017 - 01 - 15   09 :
 48 : 49 "]," attributes ":{" leafcount ": 4 ," heading ":
{" type ":" float ," length ": 4 ," nullable ": true ," value ":[
 null , null , null , null ]}}}}
( 1   row )
```

6.6.35 ST_attrNotNullFilter

This function filters all trajectory points to obtain the trajectory points whose values are not null for the specified attribute field and returns a new trajectory object that is composed of these points.

Syntax

```
 trajectory   ST_attrNot  NullFilter ( trajectory   traj ,  cstring
 attr_field  _name );
 trajectory   ST_attrNot  NullFilter ( trajectory   traj ,  cstring
 attr_field  _name );
```

Parameters

Parameter	Description
traj	The trajectory object.
attr_field _name	The name of the specified attribute field.

Description

This function supports all types of attribute fields.

Examples

```
 select   st_attrNot  NullFilter ( ST_makeTra  jectory (' STPOINT '::
 leaftype , ' LINESTRING (- 179 . 48077   51 . 72814 ,- 179 . 46731
 51 . 74634 ,- 179 . 46502   51 . 74934 ,- 179 . 46183   51 . 75378 ,-
 179 . 45943   51 . 75736 ,- 179 . 45560   51 . 76273 ,- 179 . 44845
 51 . 77186 ,- 179 . 43419   51 . 78977 ,- 179 . 41259   51 . 81643 ,-
```

```
 179 . 41001   51 . 81941 ,- 179 . 40751   51 . 82223 ,- 179 . 40497
   51 . 82505 ,- 179 . 40242   51 . 82796 ,- 179 . 39981   51 . 83095
 ,- 179 . 39734   51 . 83398 ,- 179 . 39499   51 . 83709 )':: geometry
 ,  ARRAY [' 2017 - 01 - 15   09 : 06 : 39 ':: timestamp ,' 2017 -
 01 - 15   09 : 13 : 39 ',' 2017 - 01 - 15   09 : 14 : 48 ',' 2017 -
 01 - 15   09 : 16 : 28 ',' 2017 - 01 - 15   09 : 17 : 48 ',' 2017 -
 01 - 15   09 : 19 : 48 ',' 2017 - 01 - 15   09 : 23 : 19 ',' 2017 -
 01 - 15   09 : 30 : 28 ',' 2017 - 01 - 15   09 : 34 : 40 ',' 2017 -
 01 - 15   09 : 36 : 59 ',' 2017 - 01 - 15   09 : 38 : 09 ',' 2017 -
 01 - 15   09 : 39 : 18 ',' 2017 - 01 - 15   09 : 40 : 40 ',' 2017 -
 01 - 15   09 : 47 : 38 ',' 2017 - 01 - 15   09 : 48 : 49 ',' 2017
 - 01 - 15   21 : 18 : 30 '], '{" leafcount ":  16 , " attributes ":
 {" heading ": {" type ": " float ", " length ":  4 , " nullable ":
 true ," value ": [ 23 . 0 , 23 . 0 , 23 . 0 , null , 21 . 0 , 21 . 0
 , null , 72 . 0 , 72 . 0 , null , 73 . 0 , 74 . 0 , 73 . 0 , 73 . 0 ,
 null , 73 . 0 ]}}}'),' heading ');


                                st_attrnot  nullfilter


-------------------------------------------------------------------------------
 {" trajectory ":{" version ": 1 ," type ":" STPOINT ," leafcount ":
 12 ," start_time ":" 2017 - 01 - 15   09 : 06 : 39 ," end_time ":"
 2017 - 01 - 15   21 : 18 : 30 ," spatial ":" LINESTRING (- 179 .
 48077   51 . 72814 ,- 179 . 46731   51 . 7
 4634 ,- 179 . 46502   51 . 74934 ,- 179 . 45943   51 . 75736 ,- 179 .
 4556   51 . 76273 ,- 179 . 43419   51 . 78977 ,- 179 . 41259   51 .
 81643 ,- 179 . 40751   51 . 82223 ,- 179 . 40497   51 . 82505 ,- 179
 . 40242   51 . 82796 ,- 179 . 39981   51 . 83095 ,-
 179 . 39499   51 . 83709 )"," timeline ":[" 2017 - 01 - 15   09 : 06
 : 39 "," 2017 - 01 - 15   09 : 13 : 39 "," 2017 - 01 - 15   09 : 14
 : 48 "," 2017 - 01 - 15   09 : 17 : 48 "," 2017 - 01 - 15   09 : 19
 : 48 "," 2017 - 01 - 15   09 : 30 : 28 "," 2017 - 01 - 15   09 : 34 :
 40
"," 2017 - 01 - 15   09 : 38 : 09 "," 2017 - 01 - 15   09 : 39 : 18
 "," 2017 - 01 - 15   09 : 40 : 40 "," 2017 - 01 - 15   09 : 47 : 38
 "," 2017 - 01 - 15   21 : 18 : 30 "]," attributes ":{" leafcount ":
 12 ," heading ":{" type ":" float "," length ": 4 ," nulla
 ble ": true ," value ":[ 23 . 0 , 23 . 0 , 23 . 0 , 21 . 0 , 21 . 0 ,
 72 . 0 , 72 . 0 , 73 . 0 , 74 . 0 , 73 . 0 , 73 . 0 , 73 . 0 ]}}}}
( 1   row )
```

6.7 Spatial relationship judgment

6.7.1 ST_intersects

This function returns true if a trajectory object and the specified geometry object intersect in space within the specified time range.

Syntax

```
 boolean   ST_interse  cts ( trajectory   traj ,  tsrange   range ,
 geometry   g );
```

```
boolean   ST_interse  cts ( trajectory   traj ,  timestamp   t1 ,
timestamp   t2 ,  geometry   g );
```

Parameters

Parameter	Description
traj	The trajectory object.
t1	The start time.
t2	The end time.
range	The time range.
g	The specified geometry object.

Examples

```
Select   ST_interse  cts ( traj , ' 2010 - 1 - 1   13 : 00 : 00 ', '
2010 - 1 - 1   14 : 00 : 00 ', ' LINESTRING ( 0   0 ,  5   5 ,  9   9
)':: geometry )  from   traj_table ;
```

6.7.2 ST_equals

This function returns true if a trajectory object and the specified geometry object are spatially equal within the specified time range.

Syntax

```
boolean   ST_equals ( trajectory   traj ,  tsrange   range ,
geometry   g );
boolean   ST_equals ( trajectory   traj ,  timestamp   t1 ,
timestamp   t2 ,  geometry   g );
```

Parameters

Parameter	Description
traj	The trajectory object.
t1	The start time.
t2	The end time.
range	The time range.
g	The specified geometry object.

Examples

```
Select   ST_equals ( traj , ' 2010 - 1 - 1   13 : 00 : 00 ', ' 2010 -
1 - 1   14 : 00 : 00 ', ' LINESTRING ( 0   0 ,  5   5 ,  9   9 )':::
geometry )  from   traj_table ;
```

6.7.3 ST_distanceWithin

This function returns true if the distance between a trajectory object and the specified geometry object within the specified time range is within the reference distance.

Syntax

```
boolean   ST_distanc  eWithin ( trajectory   traj ,  tsrange   range
,  geometry   g ,  float8   d );
boolean   ST_distanc  eWithin ( trajectory   traj ,  timestamp   t1
,  timestamp   t2 ,  geometry   g ,  float8   d );
```

Parameters

Parameter	Description
traj	**The trajectory object.**
t1	**The start time.**
t2	**The end time.**
range	**The time range.**
g	**The specified geometry object.**
d	**The reference distance.**

Examples

```
Select   ST_distanc  eWithin ( traj , ' 2010 - 1 - 1   13 : 00 : 00
', ' 2010 - 1 - 1   14 : 00 : 00 ', ' LINESTRING ( 0   0 ,  5   5 ,
9   9 )':: geometry ,  10 )  from   traj_table ;
```

6.8 Spatial processing

6.8.1 ST_intersection

This function returns a new trajectory object that indicates the intersection of a trajectory object and the specified geometry object within the specified time range.

Syntax

```
trajectory []  ST_interse  ction ( trajectory   traj ,  tsrange
range ,  geometry   g );
trajectory []  ST_interse  ction ( trajectory   traj ,  timestamp
t1 ,  timestamp   t2 ,  geometry   g );
```

Parameters

Parameter	Description
traj	The trajectory object.
t1	The start time.
t2	The end time.
range	The time range.
g	The specified geometry object.

If the trajectory object and the geometry object intersect at multiple spatial points, this function returns multiple sub-trajectories.

Examples

```
Select   ST_interse  ction ( traj , ' 2010 - 1 - 1   13 : 00 : 00 ',
' 2010 - 1 - 1   14 : 00 : 00 ', ' LINESTRING ( 0   0 ,  5   5 ,  9
9 )':: geometry )  from   traj_table ;
```

6.8.2 ST_difference

This function returns a new trajectory object that indicates the difference between a trajectory object and the specified geometry object within the specified time range.

Syntax

```
trajectory   ST_differe  nce ( trajectory   traj ,  tsrange   range
,  geometry   g );
trajectory   ST_differe  nce ( trajectory   traj ,  timestamp   t1 ,
timestamp   t2 ,  geometry   g );
```

Parameters

Parameter	Description
traj	The trajectory object.

Parameter	Description
t1	The start time.
t2	The end time.
range	The time range.
g	The specified geometry object.

Examples

```
Select   ST_differe  nce ( traj , ' 2010 - 1 - 1   13 : 00 : 00 ', '
2010 - 1 - 1   14 : 00 : 00 ', ' LINESTRING ( 0   0 ,  5   5 ,  9   9
)':: geometry )  from   traj_table ;
```

6.9 Spatial statistics

6.9.1 ST_nearestApproachPoint

This function returns the spatial point in a trajectory object nearest to the specified geometry object within the specified time range.

Syntax

```
geometry   ST_nearest  ApproachPo  int ( trajectory   traj ,  tsrange
  range ,  geometry   g );
geometry   ST_nearest  ApproachPo  int ( trajectory   traj ,
timestamp   t1 ,  timestamp   t2 ,  geometry   g );
```

Parameters

Parameter	Description
traj	The trajectory object.
t1	The start time.
t2	The end time.
range	The time range.
g	The specified geometry object.

Examples

```
Select   ST   _nearestAp  proachPoin  t ( traj , ' 2010 - 1 - 1   13
: 00 : 00 ', ' 2010 - 1 - 1   14 : 00 : 00 ', ' LINESTRING ( 0   0 ,
5   5 ,  9   9 )':: geometry )  from   traj_table ;
```

6.9.2 ST_nearestApproachDistance

This function returns the nearest distance between a trajectory object and the specified geometry object within the specified time range.

Syntax

```
float8   S   T_nearestA  pproachDis  tance ( trajectory   traj ,
tsrange   range ,  geometry   g );
float8   S   T_nearestA  pproachDis  tance ( trajectory   traj ,
timestamp   t1 ,  timestamp   t2 ,  geometry   g );
```

Parameters

Parameter	Description
traj	The trajectory object.
t1	The start time.
t2	The end time.
range	The time range.
g	The specified geometry object.

Examples

```
Select   ST   _nearestAp  proachDist  ance ( traj , ' 2010 - 1 - 1
13 : 00 : 00 ', ' 2010 - 1 - 1   14 : 00 : 00 ', ' LINESTRING ( 0   0
,  5   5 ,  9   9 )':: geometry )  from   traj_table ;
```

6.10 Spatio-temporal relationship judgment

6.10.1 ST_intersects

This function returns true if trajectory objects 1 and 2 intersect in space within the specified time range.

Syntax

```
boolean   ST_interse  cts ( trajectory   traj1 ,  trajectory   traj2
);
boolean   ST_interse  cts ( trajectory   traj1 ,  trajectory   traj2
,  tsrange   range );
```

```
boolean   ST_interse  cts ( trajectory   traj1 ,  trajectory   traj2
,  timestamp   t1 ,  timestamp   t2 );
```

Parameters

Parameter	Description
traj	The trajectory object.
t1	The start time.
t2	The end time.
range	The time range.

Examples

```
Select   ST_interse  cts (( Select   traj   from   traj_table   where
  id = 1 ), ( Select   traj   from   traj_table   where   id = 2 ), '
2010 - 1 - 1   13 : 00 : 00 ', ' 2010 - 1 - 1   14 : 00 : 00 ');
```

6.10.2 ST_equals

This function returns true if trajectory objects 1 and 2 are spatially equal within the specified time range.

Syntax

```
boolean   ST_equals ( trajectory   traj1 ,  trajectory   traj2 ,
tsrange   range );
boolean   ST_equals ( trajectory   traj1 ,  trajectory   traj2 ,
timestamp   t1 ,  timestamp   t2 );
```

Parameters

Parameter	Description
traj	The trajectory object.
t1	The start time.
t2	The end time.
range	The time range.

Examples

```
Select   ST_equals (( Select   traj   from   traj_table   where   id
= 1 ), ( Select   traj   from   traj_table   where   id = 2 ), ' 2010
- 1 - 1   13 : 00 : 00 ', ' 2010 - 1 - 1   14 : 00 : 00 ');
```

6.10.3 ST_distanceWithin

This function returns true if the distance between trajectory objects 1 and 2 within the specified time range is within the reference distance.

Syntax

```
boolean   ST_distanc  eWithin ( trajectory   traj1 ,  trajectory
traj2 ,  tsrange   range ,  float8   d );
boolean   ST_distanc  eWithin ( trajectory   traj1 ,  trajectory
traj2 ,  timestamp   t1 ,  timestamp   t2 ,  float8   d );
```

Parameters

Parameter	Description
traj	The trajectory object.
t1	The start time.
t2	The end time.
range	The time range.
d	The reference distance.

Examples

```
Select   ST_distanc  eWithin (( Select   traj   from   traj_table
where   id = 1 ), ( Select   traj   from   traj_table   where   id =
2 ), ' 2010 - 1 - 1   13 : 00 : 00 ', ' 2010 - 1 - 1   14 : 00 : 00
',  100 );
```

6.10.4 ST_durationWithin

This function returns true if the difference between the time when trajectory objects 1 and 2 intersect at the same spatial point within the specified time range is within the reference interval.

Syntax

```
boolean   ST_duratio  nWithin ( trajectory   traj1 ,  trajectory
traj2 ,  tsrange   range ,  interval   i );
```

```
boolean   ST_duratio  nWithin ( trajectory   traj1 ,  trajectory
traj2 ,  timestamp   t1 ,  timestamp   t2 ,  interval   i );
```

Parameters

Parameter	Description
traj	The trajectory object.
t1	The start time.
t2	The end time.
range	The time range.
i	The reference interval.

If two trajectory objects intersect at the same spatial point multiple times, this function returns true so long as any time difference is within the reference interval.

Examples

```
Select   ST_duratio  nWithin (( Select   traj   from   traj_table
where   id = 1 ), ( Select   traj   from   traj_table   where   id =
2 ), ' 2010 - 1 - 1   13 : 00 : 00 ', ' 2010 - 1 - 1   14 : 00 : 00
',  INTERVAL  ' 30s ');
```

6.11 Spatio-temporal processing

6.11.1 ST_intersection

This function returns a new trajectory object that indicates the intersection of trajectory objects 1 and 2 within the specified time range.

Syntax

```
geometry   ST_interse  ction ( trajectory   traj1 ,  trajectory
traj2 ,  tsrange   range );
geometry   ST_interse  ction ( trajectory   traj1 ,  trajectory
traj2 ,  timestamp   t1 ,  timestamp   t2 );
```

Parameters

Parameter	Description
traj1	Trajectory object 1.
traj2	Trajectory object 2.
t1	The start time.

Parameter	Description
t2	The end time.
range	The time range.

Examples

```
Select ST_interse ction (( Select traj from traj_table
where id = 1 ), ( Select traj from traj_table where id =
2 ), ' 2010 - 1 - 1 13 : 00 : 00 ', ' 2010 - 1 - 1 14 : 00 : 00
');
```

6.12 Spatio-temporal statistics

6.12.1 ST_nearestApproachPoint

This function returns the spatial point in trajectory object 1 nearest to trajectory object 2 within the specified time range.

Syntax

```
geometry ST_nearest ApproachPo int ( trajectory traj1 ,
trajectory traj2 );
geometry ST_nearest ApproachPo int ( trajectory traj1 ,
trajectory traj2 , tsrange range );
geometry ST_nearest ApproachPo int ( trajectory traj1 ,
trajectory traj2 , timestamp t1 , timestamp t2 );
```

Parameters

Parameter	Description
traj1	Trajectory object 1.
traj2	Trajectory object 2.
t1	The start time.
t2	The end time.
range	The time range.

Examples

```
Select ST_nearest ApproachPo int (( Select traj from
traj_table where id = 1 ), ( Select traj from traj_table
```

```
where   id = 2 ), ' 2010 - 1 - 1   13 : 00 : 00 ', ' 2010 - 1 - 1
14 : 00 : 00 ');
```

6.12.2 ST_nearestApproachDistance

This function returns the nearest distance between trajectory objects 1 and 2 within the specified time range.

Syntax

```
float8   S   T_nearestA  pproachDis  tance ( trajectory   traj ,
trajectory   traj2 );
float8   S   T_nearestA  pproachDis  tance ( trajectory   traj ,
trajectory   traj2 ,  tsrange   range );
float8   S   T_nearestA  pproachDis  tance ( trajectory   traj ,
trajectory   traj2 ,  timestamp   t1 ,  timestamp   t2 );
```

Parameters

Parameter	Description
traj1	**Trajectory object 1.**
traj2	**Trajectory object 2.**
t1	**The start time.**
t2	**The end time.**
range	**The time range.**

Examples

```
Select   ST_nearest  ApproachDi  stance (( Select   traj   from
traj_table   where   id = 1 ), ( Select   traj   from   traj_table
where   id = 2 ), ' 2010 - 1 - 1   13 : 00 : 00 ', ' 2010 - 1 - 1
14 : 00 : 00 ');
```

6.13 Distance measurement

6.13.1 ST_length

This function returns the total length of a trajectory object, in meters.

Syntax

```
float8 ST_length ( trajectory traj , integer srid default
0 );
```

Parameters

Parameter	Description
traj	The trajectory object.
srid	The spatial reference system identifier (SRID) for the coordinates of trajectory points in the trajectory object. Default value: 0.

Examples

```
select st_length ( traj ) from traj where id = 2 ;
    st_length
------------------
  13494 . 6660605311
( 1 row )
```

6.13.2 ST_euclideanDistance

This function returns the nearest Euclidean distance between two trajectory objects at the same time point.

Syntax

```
float ST_euclide anDistance ( trajectory traj1 , trajectory
traj2 );
```

Parameters

Parameter	Description
traj1	Trajectory object 1.
traj2	Trajectory object 2.

The distance has been standardized.

Examples

```
Select   ST_euclide  anDistance (( Select   traj   from   traj_table
  where   id = 1 ), ( Select   traj   from   traj_table   where   id
= 2 ));
```

6.13.3 ST_mdistance

This function returns an array of the Euclidean distance between two trajectory objects at the same time point.

Syntax

```
float []  ST_mdistan  ce ( trajectory   traj1 ,  trajectory   traj2
);
```

Parameters

Parameter	Description
traj1	**Trajectory object 1.**
traj2	**Trajectory object 2.**

The distance has not been standardized.

Examples

```
Select   ST_mDistan  ce (( Select   traj   from   traj_table   where
  id = 1 ), ( Select   traj   from   traj_table   where   id = 2 ));
```

7 Trajectory best practices

Use appropriate spatio-temporal indexes

You can create appropriate indexes based on application requirements to accelerate queries. Trajectory data supports the following index types:

- Spatial index: the index on space, which applies when you query only the spatial data of a trajectory.
- Temporal index: the index on time, which applies when you query only the time range of a trajectory.
- Spatio-temporal composite index: the composite index on both space and time, which applies when you query both the spatial data and time range of a trajectory.

```
-- Create a function - based spatial index to accelerate
   the filtering of spatial data .
 create index tr_spatial _geometry_ index on trajtab using
   gist ( st_traject oryspatial ( traj ));


-- Create a function - based temporal index to accelerate
   the filtering of time .
 create index tr_timespa n_time_ind ex on trajtab using
 gist ( st_timespa n ( traj ));

-- Create function - based indexes on the start time
 and end time of a trajectory object .
 create index tr_startti me_index on trajtab using btree
 ( st_startti me ( traj ));
 create index tr_endtime _index on trajtab using btree (
 st_endtime ( traj ));


-- Create a btree_gist extension .
 create extension btree_gist ;

-- Use btree_gist to create a spatio - temporal composite
   index on the start time , end time , and spatial
 data of a trajectory object .
 create index tr_traj_te st_stm_etm _sp_index on traj_test
   using gist ( st_startti me ( traj ), st_endtime ( traj ),
 st_traject oryspatial ( traj ));
```

Use appropriate partitioned tables

The amount of trajectory data in a database keeps increasing along with the continuous use of the database. As a result, a larger number of database indexes are created and data queries slow down. You can use table partitioning to decrease the data size of a single table.

For more information about table partitioning, see Table partitioning in the PostgreSQL documentation.

Reduce the use of string-type attribute fields

A large number of string-type attribute fields in trajectory data lead to a waste of the storage space and performance deterioration.

- If string-type attribute fields have fixed values, they can be converted into integers . We recommend that you convert the data type in code.
- If string-type attribute fields are required, you can set a default length for them to save space.

  To set a default length for string-type attribute fields, do as follows:

```
-- Set the default length of string - type attribute
 fields to 32 .
 Set ganos . trajectory . attr_strin g_length = 32 ;
```

Use multiple trajectory points to generate a trajectory object

We recommend that you use multiple trajectory points to generate a trajectory object and avoid appending trajectory points one by one.

Use the advanced compression mode

LZ4 is an advanced compression algorithm, with a higher compression ratio and execution speed. To enable the LZ4 compression algorithm, do as follows:

```
-- Enable LZ4 compressio n .
 Set toast_comp ression_us e_lz4 = true ;

-- Disable LZ4 compressio n to use the default
 PostgreSQL compressio n algorithm .
 Set toast_comp ression_us e_lz4 = false ;
```

To enable the LZ4 compression algorithm for the entire database by default, do as follows:

```
-- Enable LZ4 compressio n for the database .
 Alter database dbname Set toast_comp ression_us e_lz4 =
 true ;

-- Disable LZ4 compressio n to use the default
 compressio n algorithm for the database .
 Alter database dbname Set toast_comp ression_us e_lz4 =
 false ;
```

8 Trajectory FAQ

How do I convert the data of coordinate points into a trajectory object?

You can use the ST_makeTrajectory constructor to convert the data of coordinate points into a trajectory object. The procedure is as follows:

```
--  Create   an   extension .
 create   extension   ganos_traj  ectory   cascade ;

--  Create   a   point   table .
 create   table   points ( id   integer ,  x   float8 ,  y   float8 ,
 t   timestamp ,  speed   float8 );
 insert   into   points   values ( 1 ,  128 . 1 ,  28 . 1 , ' 2019 -
 01 - 01   00 : 00 : 00 ',  100 );
 insert   into   points   values ( 2 ,  128 . 2 ,  28 . 2 , ' 2019 -
 01 - 01   00 : 00 : 01 ',  101 );
 insert   into   points   values ( 3 ,  128 . 3 ,  28 . 3 , ' 2019 -
 01 - 01   00 : 00 : 02 ',  102 );
 insert   into   points   values ( 4 ,  128 . 4 ,  28 . 4 , ' 2019 -
 01 - 01   00 : 00 : 04 ',  103 );

--  Create   a   trajectory   table .
 create   table   traj ( id   integer ,  traj   trajectory );

--  Insert   data .
 insert   into   traj ( id ,  traj )
 select   1 ,
     ST_MakeTra  jectory (' STPOINT ':: leaftype ,  x ,  y ,  4326 ,
 t ,  ARRAY [' speed '],  NULL ,  s ,  NULL )
 FROM  ( select   array_agg ( x   order   by   id )  as   x ,
       array_agg ( y   order   by   id )  as   y ,
       array_agg ( t   order   by   id )  as   t ,
       array_agg ( speed   order   by   id )  as   s
   From   points )  a ;
```

What shall I do if the built-in ST_makeTrajectory constructor does not meet my requirements?

If the built-in ST_makeTrajectory constructor does not meet your requirements, you can customize one with required attributes. For example, to create a trajectory object with five attributes, including two of the int8 type, two of the float4 type, and one of the timestamp type, you can customize the following constructor:

```
 CREATE   OR   REPLACE   FUNCTION   ST_MakeTra  jectory ( type
 leaftype ,  x   float8 [],  y   float8 [],
                srid   integer ,  timespan   timestamp [], attrs_name
   cstring [],  attr1   int8 [],
                attr2   int8 [],  attr3   float4 [],  attr4   float4
 [],  attr5   timestamp [])
     RETURNS   trajectory
     AS  '$ libdir / libpg - trajectory  16 ',' sqltr_traj
 _make_all_  array '
```

```
    LANGUAGE  ' c '  IMMUTABLE   Parallel   SAFE ;
```

In this constructor, the first six parameters are fixed, and the last five parameters are customized attributes. You can directly use this constructor with the customized attributes as a user-defined overloaded function.

How do I append trajectory points to a trajectory object?

You can use the ST_append constructor to append trajectory points to an existing trajectory object. The syntax of the ST_append constructor is as follows:

```
trajectory   ST_append ( trajectory   traj ,  geometry   spatial ,
timestamp []  timespan ,  text   str_theme_  json );

trajectory   ST_append ( trajectory   traj ,  trajectory   tail );
```

The following example shows how to append trajectory points to a trajectory object based on a point table.

```
--  Create   an   extension .
 create   extension   ganos_traj  ectory   cascade ;

--  Create   a   point   table .
 create   table   points ( id   integer ,  x   float8 ,  y   float8 ,
 t   timestamp ,  speed   float8 );
 insert   into   points   values ( 1 ,  128 . 1 ,  28 . 1 , ' 2019 -
 01 - 01   00 : 00 : 00 ',  100 );
 insert   into   points   values ( 2 ,  128 . 2 ,  28 . 2 , ' 2019 -
 01 - 01   00 : 00 : 01 ',  101 );
 insert   into   points   values ( 3 ,  128 . 3 ,  28 . 3 , ' 2019 -
 01 - 01   00 : 00 : 02 ',  102 );
 insert   into   points   values ( 4 ,  128 . 4 ,  28 . 4 , ' 2019 -
 01 - 01   00 : 00 : 04 ',  103 );

--  Create   a   trajectory   table .
 create   table   traj ( id   integer ,  traj   trajectory );

--  Insert   data .
 insert   into   traj ( id ,  traj )
 select   1 ,
     ST_MakeTra  jectory (' STPOINT ':: leaftype ,  x ,  y ,  4326 ,
 t ,  ARRAY [' speed '],  NULL ,  s ,  NULL )
 FROM  ( select   array_agg ( x   order   by   id )  as   x ,
       array_agg ( y   order   by   id )  as   y ,
       array_agg ( t   order   by   id )  as   t ,
       array_agg ( speed   order   by   id )  as   s
   From   points )  a ;

--  Insert   trajectory   points .
 insert   into   points   values ( 5 ,  128 . 5 ,  28 . 5 , ' 2019 -
 01 - 01   00 : 00 : 05 ',  105 );
 insert   into   points   values ( 6 ,  128 . 6 ,  28 . 6 , ' 2019 -
 01 - 01   00 : 00 : 06 ',  106 );
 insert   into   points   values ( 7 ,  128 . 7 ,  28 . 7 , ' 2019 -
 01 - 01   00 : 00 : 07 ',  107 );

--  Update   the   trajectory   object   based   on   a   single
 trajectory   point .
```

```
 With   point_traj   as  (
  select   ST_MakeTra  jectory (' STPOINT ':: leaftype ,  x ,  y ,
 4326 ,  t ,  ARRAY [' speed '],  NULL ,  s ,  NULL )  AS   traj
  FROM  ( select   array_agg ( x   order   by   id )  as   x ,
       array_agg ( y   order   by   id )  as   y ,
       array_agg ( t   order   by   id )  as   t ,
       array_agg ( speed   order   by   id )  as   s
       From   points   WHERE   ID  =  5 )  a
)
 Update   traj
 set   traj  =  ST_append ( traj . traj ,  a . traj )
 From   point_traj   a
 WHERE   traj . ID = 1 ;

--  Use   a   generated   SQL   script   to   update   the
 trajectory   object .
 With   point_traj   as  (
  select   ST_MakeTra  jectory (' STPOINT ':: leaftype ,  ARRAY [ 128
 . 5 :: float8 ],
       ARRAY [ 28 . 5 :: float8 ],  4326 ,  ARRAY [' 2019 - 01 - 01
 00 : 00 : 05 ':: timestamp ],
       ARRAY [' speed '],  NULL ,  ARRAY [ 106 :: float8 ],  NULL )
 AS   traj
)
 Update   traj
 set   traj  =  ST_append ( traj . traj ,  a . traj )
 From   point_traj   a
 WHERE   traj . ID  = 1 ;

--  Update   the   trajectory   object   based   on   multiple
 trajectory   points .
 With   point_traj   as  (
  select   ST_MakeTra  jectory (' STPOINT ':: leaftype ,  x ,  y ,
 4326 ,  t ,  ARRAY [' speed '],  NULL ,  s ,  NULL )  AS   traj
  FROM  ( select   array_agg ( x   order   by   id )  as   x ,
       array_agg ( y   order   by   id )  as   y ,
       array_agg ( t   order   by   id )  as   t ,
       array_agg ( speed   order   by   id )  as   s
       From   points   WHERE   ID  >  5 )  a
)
 Update   traj
 set   traj  =  ST_append ( traj . traj ,  a . traj )
 From   point_traj   a
 WHERE   traj . ID  = 1 ;
```

How do I enable advanced compression?

LZ4 is an advanced compression algorithm, with a higher compression ratio and execution speed. To enable the LZ4 compression algorithm, do as follows:

```
--  Enable   LZ4   compressio  n .
 Set   toast_comp  ression_us  e_lz4  =  true ;

--  Disable   LZ4   compressio  n   to   use   the   default
 PostgreSQL   compressio  n   algorithm .
```

```
 Set   toast_comp  ression_us  e_lz4  =  false ;
```

To enable the LZ4 compression algorithm for the entire database by default, do as follows:

```
--  Enable   LZ4   compressio  n   for   the   database .
 Alter   database   dbname   Set   toast_comp  ression_us  e_lz4  =
 true ;

--  Disable   LZ4   compressio  n   to   use   the   default
 compressio  n   algorithm   for   the   database .
 Alter   database   dbname   Set   toast_comp  ression_us  e_lz4  =
 false ;
```

How do I set a default length for string-type attribute fields?

You can use the GUC variable ganos.trajectory.attr_string_length to set a default length for string-type attribute fields. You can do as follows:

```
 Set   ganos . trajectory . attr_strin  g_length  =  32 ;
```

How do I compute the maximum, minimum, or average value of an attribute field?

To compute the maximum, minimum, or average value of an attribute field, do as follows:

```
 With   traj   AS  (
   select  '{" trajectory ":{" version ": 1 ," type ":" STPOINT ","
 leafcount ": 2 ," start_time ":" 2010 - 01 - 01   11 : 30 : 00 ","
 end_time ":" 2010 - 01 - 01   12 : 30 : 00 ," spatial ":" SRID =
 4326 ; LINESTRING ( 1   1 , 3   5 )"," timeline ":[" 2010 - 01 - 01
   11 : 30 : 00 "," 2010 - 01 - 01   12 : 30 : 00 "]," attributes ":
{" leafcount ": 2 ," velocity ":{" type ":" integer "," length ": 4
 ," nullable ": true ," value ":[ 1 , 100 ]}," speed ":{" type ":"
 float "," length ": 8 ," nullable ": true ," value ":[ null , 1 . 0
 ]}," angle ":{" type ":" string "," length ": 64 ," nullable ": true
 ," value ":[" test ", null ]}, " tngel2 ":{" type ":" timestamp ","
 length ": 8 ," nullable ": true ," value ":[" 2010 - 01 - 01   12 :
 30 : 00 ", null ]}," bearing ":{" type ":" bool "," length ": 1 ,"
 nullable ": true ," value ":[ null , true ]}}}}':: trajectory   a )
 select   avg ( v )  from
(
 Select   unnest ( st_trajAtt  rsAsIntege  r ( a , ' velocity '))  as
   v   from   traj
)  t ;
```