**PostGIS:**

Dimension():Integer获取几何对象的几何维数

CoordinateDimension():Integer获取几何对象坐标维数

GeometryType():String获取几何对象的数据类型，如点、线、面等

SRID():Integer用于获取几何类型的空间参考系

Envelope():Geometry用于获取最小边界矩形

AsText():String返回WKT (Well-Known Text)的表达形式，不包含SRID元数据

AsBinary():String返WKB(Well-Known Binary)的表达形式，不包含SRID元数据

IsEmpty():Boolean判断几何类型是否为空

IsSimple():Boolean 判断几何类型是否是简单的

Is3D():Boolean判断是否有z坐标

IsMeasured():Boolean判断是否有M值

Boundary():Geometry 获取边界

Distance(another:Geometry):Distance求距离

Buffer(distance:Distance):Geometry求缓冲区

ConvexHull():Geometry 求本Geom的凸包

Intersection(another:Geometry):Geometry 求本Geom与另一个Geom的交

Union(another:Geometry):Geometry 求并

Difference(another:Geometry):Geometry求差

SymDifference(another:Geometry):Geometry求对称差

Equals(another:Geometry):Boolean判断是否相等

Disjoint(another:Geometry):Boolean是否相离

Intersects(another:Geometry):Boolean是否相交

Touches(another:Geometry):Boolean是否相接

Crosses(another:Geometry):Boolean是否穿越another

Within(another:Geometry):Boolean是否包含于another

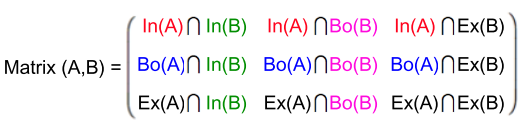
Contains(another:Geometry):Boolean是否包含another

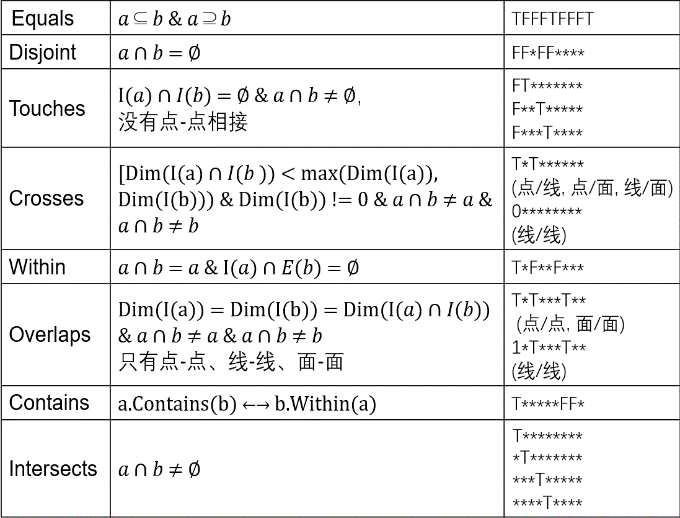
Overlaps(another:Geometry):Boolean是否交叠

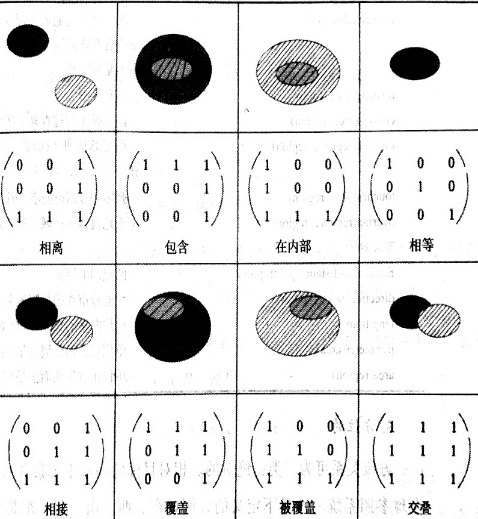
Relates(another:Geometry,matrix:String):Boolean 是否符合给定的九交矩阵

LocateAlong(mValue:Double):Geometry选取M值为mValue的点，形成一个新的Geometry

LocateBetween(mStart:Double,mEnd:Double):Geometry 选取M值∈[mStart~mEnd]的点，形成新Geome

**空间关系** **- 九交矩阵**





类型转换

**CAST**(x AS typename) or x**::**typename

SELECT seq, node, edge, cost

FROM **pgr\_dijkstra**(

'SELECT LINK\_ID as id, START\_NODE\_ID as source, END\_NODE\_ID as target, COST FROM AIRPORT\_LINK',

(SELECT AIRPORT\_ID FROM AIRPORT\_LIST WHERE AIRPORT\_NAME='Dillingham, AK'),

(SELECT AIRPORT\_ID FROM AIRPORT\_LIST WHERE AIRPORT\_NAME='Gainesville, FL') );

更新road\_network中的source和target属性

varchar **pgr\_createTopology**(text edge\_table,  
 double precision tolerance, text the\_geom,  
 text id,text source, text target,  
 text rows\_where:=’true’, boolean clean:=false)

SELECT pgr\_createTopology('road\_network', 0.00001, 'geom', 'id', 'source','target', 'true');

分析网络边和顶点

SELECT **pgr\_analyzeGraph**('road\_network',0.00001, 'geom','id','source','target', 'true');

创建相交点修正网络

SELECT **pgr\_nodeNetwork**('road\_network', 0.00011, the\_geom:='geom',id:='id',table\_ending:='1');

**pgr\_aStar**(edges\_sql, starts\_vid, end\_vids, directed, heuristic, factor, epsilon)

RETURNS SET OF (seq, path\_seq [, start\_vid] [, end\_vid], node, edge, cost, agg\_cost) OR EMPTY SET

SELECT \* FROM pgr\_astar('SELECT id, source, target, cost, reverse\_cost, x1, y1, x2, y2 FROM edge\_table', 2, 12)

最少转机次数

**WITH RECURSIVE** X(end\_node,times) AS(

(SELECT END\_NODE\_ID, 0

FROM AIRPORT\_LINK,AIRPORT\_LIST

WHERE AIRPORT\_LINK.START\_NODE\_ID=AIRPORT\_LIST.AIRPORT\_ID AND AIRPORT\_LIST.AIRPORT\_NAME='Dillingham, AK') UNION

(SELECT AIRPORT\_LINK.END\_NODE\_ID, X.TIMES+1

FROM AIRPORT\_LINK, X

WHERE AIRPORT\_LINK.START\_NODE\_ID=X.end\_node AND X.TIMES<10))

SELECT min(times)

FROM X,AIRPORT\_LIST

WHERE X.end\_node=AIRPORT\_LIST.AIRPORT\_ID and AIRPORT\_LIST.AIRPORT\_NAME='Gainesville, FL'

所有可到达的地铁站：

WITH recursive X(node, depth, path, circle) as (

select fromstation, 0, array[fromstation], false from link,station where station.id=link.fromstation and station.**name like '%世界之窗%'**

UNION

select tostation, depth + 1, path || tostation, tostation = any(path)

from link, X

where fromstation = node and not circle and depth<30)

SELECT min(station.id) as id,station.name,min(depth) as depth,min(path) as path

FROM (SELECT node,min(depth) FROM X GROUP BY node)M, X, station

WHERE X.node=M.node and X.depth=M.min and X.node=station.id

GROUP BY name ORDER BY depth

空间关联查询（事故最多的公路）

SELECT ushighways.gid, ushighways.full\_name as name, ushighways.geom as geom, B.max\_count as accidents\_count

FROM(SELECT max(accidents\_count) as max\_count

FROM(SELECT ushighways.gid as gid, COUNT(\*) as accidents\_count

FROM usaccidents, ushighways

WHERE ST\_DWithin(usaccidents.geom::geography, ushighways.geom::geography, 500) AND (usaccidents.month=7 OR usaccidents.month=8)

GROUP BY ushighways.gid

)A)B,

(SELECT ushighways.gid as gid, COUNT(\*) as accidents\_count

FROM usaccidents, ushighways

WHERE ST\_DWithin(usaccidents.geom::geography, ushighways.geom::geography, 500) AND (usaccidents.month=7 OR usaccidents.month=8)

GROUP BY ushighways.gid)C,

ushighways

WHERE C.accidents\_count=B.max\_count AND C.gid=ushighways.gid

最邻近查询、格式化文本

SELECT DISTINCT (name||'in'||uscities.state)as name

FROM uscities, usaccidents,

(SELECT MIN(ST\_DISTANCE(uscities.geom,(SELECT geom from usaccidents where ST\_CASE=10001),false)) as min\_dist from uscities)a

WHERE ST\_DISTANCE(uscities.geom,(SELECT geom from usaccidents where ST\_CASE=101),false)=min\_dist

增加几何属性列

select **AddGeometryColumn**('uscities', 'geom', 4326, 'POINT', 2);

UPDATE uscities SET geom = ST\_**SetSRID**(ST\_**Makepoint**(longitude,latitude),4326);

数据插入（最大id+1）

**INSERT INTO** trip (id, start\_time, start\_station\_id, end\_station\_id)

VALUES ((SELECT max(id) from trip)+1, '2015-8-31 23:26', 50, 70);

空间关联查询（事故最多的公路）

SELECT ushighways.gid, ushighways.full\_name as name, ushighways.geom as geom, B.max\_count as accidents\_count

FROM(SELECT max(accidents\_count) as max\_count

FROM(SELECT ushighways.gid as gid, COUNT(\*) as accidents\_count

FROM usaccidents, ushighways

WHERE ST\_DWithin(usaccidents.geom::geography, ushighways.geom::geography, 500) AND (usaccidents.month=7 OR usaccidents.month=8)

GROUP BY ushighways.gid

)A)B,

(SELECT ushighways.gid as gid, COUNT(\*) as accidents\_count

FROM usaccidents, ushighways

WHERE ST\_DWithin(usaccidents.geom::geography, ushighways.geom::geography, 500) AND (usaccidents.month=7 OR usaccidents.month=8)

GROUP BY ushighways.gid)C,

ushighways

WHERE C.accidents\_count=B.max\_count AND C.gid=ushighways.gid

DROP TABLE if exists trip;

CREATE TABLE trip(

id int **not null**,

start\_station\_id smallint,

bike\_id smallint,

**PRIMARY KEY**(id),

**FOREIGN KEY**(start\_station\_id) **REFERENCES** station(station\_id) );

**timestamp** (strat\_time)的转化(to date type)与截取：

date\_part('year', start\_time)

GROUP BY start\_time::date

最新的记录, **DESC**降序排序

SELECT B1.endstationid as stationid, COUNT(\*)

FROM (SELECT bikeid,MAX(endTime) as latestTime

FROM trip

GROUP BY bikeId )A,

(SELECT bikeId, endTime, endStationId

FROM trip )B

WHERE A.bikeId=B.bikeId AND A.lastTime=B.endTime

GROUP BY stationId

ORDER BY count DESC, stationId

从磁盘导入数据

**copy** uscity (gid,name,state,latitude,longitude) from 'E:\\usdata\\uscity.txt' (DELIMITER '#')

触发器

CREATE OR REPLACE FUNCTION usagestamp()

RETURNS **TRIGGER** AS $$

BEGIN

IF TG\_OP = 'INSERT' THEN

NEW.created\_by = SESSION\_USER;

NEW.created\_at = CURRENT\_TIMESTAMP;

ELSE

NEW.created\_by = OLD.created\_by;

NEW.created\_at = OLD.created\_at;

END IF;

NEW.last\_changed\_by = SESSION\_USER;

NEW.last\_changed\_at = CURRENT\_TIMESTAMP;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER usagestamp

BEFORE INSERT OR UPDATE ON modify\_test

FOR EACH ROW

EXECUTE PROCEDURE usagestamp();

视图

CREATE **VIEW** current\_employees AS

SELECT NAME, ID, SALARY

FROM EMPLOYEES;

//DROP VIEW current\_employees;

**GRANT** SELECT(,INSERT,UPDATE,DELETE,RULE,ALL)

ON Table\_A TO UserX with grant option;

**REVOKE** SELECT on Table\_A from UserX (restrict, cascade) //如果添加**restrict**,下属的引用行不被删除;如果是**cascade**,下属的引用行一起被删除