空间数据库原理及开发技术

第5章 空间数据处理与分析

中南大学地信系 李光强
QQ：41733233

教学目标

- 掌握空间度量函数、空间关系函数、空间运算函数的使用
- 熟悉空间聚类函数、网络分析函数的使用
- 了解高级拓扑分析方法和栅格数据处理方法

教学重点

- 空间度量函数
- 空间关系函数
- 空间运算函数

教学内容

- 5.1 概述
- 5.2 空间度量函数
- 5.3 空间关系函数
- 5.4 空间运算函数
- 5.5 空间聚类
- 5.6 网络分析
- 5.7* 高级拓扑分析
- 5.8* 栅格数据处理与分析

5.1 概述

- **空间数据处理与分析**是依据空间关系理论，通过建立目标模型，对空间数据进行处理、分析和模拟，利用地理计算和空间表达，挖掘隐藏其中的空间信息和知识，解决空间相关问题，包括空间目标的**分布模式**、空间目标间的**关系**、空间目标的**发展趋势**等。
- ISO-SQL/MM标准明确定义了空间数据的操作规范，OGC-SFA规范定义了具体的**空间数据处理与分析的相关函数。**
- 空间数据库中引入空间数据处理与分析相关函数有助于快捷方便地实现专题属性与几何图形数据的一体化计算与分析。
- 空间数据处理与分析功能大致分为**度量函数**、**空间关系函数**、**空间运算函数**、**空间聚类函数**、**拓扑处理**、**网络分析**、**栅格数据处理函数**等

5.2 度量函数

- PostGIS提供了
  - 多边形度量函数
  - 线串度量函数
  - 几何图形之间的度量函数

5.2.1 多边形度量函数

（一）ST_Area函数

- ST_Area函数为重载函数，计算几何图形或地理图形的**面积**，基本语法如下：float ST_Area(geometry g1);
- 例5-1. 计算多边形几何图形的面积，语句如下：

```
SELECT ST_Area('POLYGON((0 0,10 0,10 10, 0 10,0 0))'::geometry);
```

- 例5-2. 将SRID=4326的多边形转换为SRID=3857的多边形，并计算该多边形面积，语句如下：

```
SELECT ST_Area(ST_Transform(
        'SRID=4326;POLYGON((112.932150 28.161150,112.933610 28.160610,
                            112.934490 28.159250,112.931610 28.159660,
                            112.932150 28.161150))'::geometry
        ,3857));
```

5.2.1 多边形度量函数

（二）ST_Perimeter函数

- ST_Perimeter函数为重载函数，计算多边形几何图形或地理图形的**周长**，基本语法如下：

```
float ST_Perimeter(geometry g1);
```

- 例5-3. 计算多边形的周长，语句如下：

```
SELECT ST_Perimeter('POLYGON((0 0, 10 0, 10 10, 0 10, 0 0))'::geometry);
```

5.2.2 线度量函数

（一）ST_Angle函数

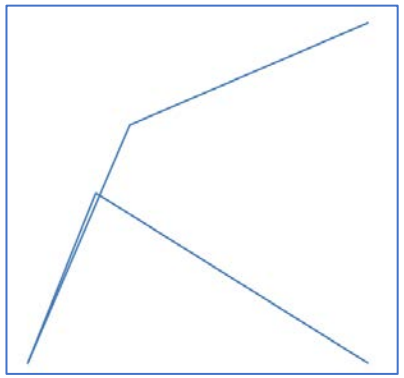
- ST_Angle函数为重载函数，计算两线或两**向量夹角**，向量可以由三个点或四个点定义。向量夹角为顺时针转过的弧度，两线夹角是两线起止点组成的向量夹角。语法如下：
  ① float ST_Angle(geometry point1, geometry point2, geometry point3);
  ② float ST_Angle(geometry point1, geometry point2, geometry point3, geometry point4);
  ③ float ST_Angle(geometry line1, geometry line2);

- 例5-3. 计算三点形成的向量夹角度数，语句如下：

```
SELECT degrees( ST_Angle('POINT(0 0)', 'POINT(10 10)', 'POINT(20 0)') );
```

- 例5-5. 计算两条线串之间的夹角，两线串如右图所示。语句如下：

```
SELECT degrees( ST_Angle( 'LINESTRING(0 0, 0.3 0.7, 1 1)',
                          'LINESTRING(0 0, 0.2 0.5, 1 0)'
));
```



5.2.2 线度量函数

（二）ST_Length函数

- ST_Length函数为重载函数，计算二维线、复合线、曲线、复合曲线等的**长度**，基本语法如下：

  float ST_Length(geometry a_2dlinestring);

- 例5-6. 计算线串的长度，语句如下：

  SELECT ST_Length('LINESTRING(0 0,10 10,50 60,100 80)'::geometry);

5.2.3 图形之间度量函数

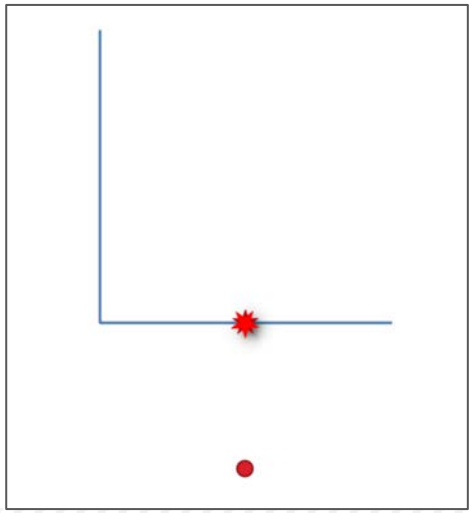
（一）ST_ClosestPoint函数

- ST_ClosestPoint函数计算二维几何图形g1距离g2**最近的点**，语法如下：

  geometry ST_ClosestPoint(geometry g1, geometry g2);

- 例5-7. 计算如图5-2(a)所示线L上距离点P最近的点，语句如下：

```
SELECT
    ST_AsText(ST_ClosestPoint('LINESTRING(10 20,10 10,20 10)'::geometry,
                              'POINT(15 5)'::geometry)
);
```



5.2.3 图形之间度量函数

- （二）ST_Distance函数
  - ST_Distance函数为重载函数，计算地理图形或几何图形的**距离**计算，基本语法如下：

    float ST_Distance(geometry g1, geometry g2);
  - 例5-8. 计算点到线串的距离，语句如下：

    SELECT ST_Distance('LINESTRING(10 20,10 10,20 10)'::geometry, 'POINT(15 5)'::geometry);
  - 例5-9. 计算WGS84坐标系的广州市某地（113.247710, 23.136330）到乌鲁木齐市某地（87.576270, 43.844990）的平面距离，语句如下：

```
SELECT ST_Distance(
                ST_Transform('SRID=4326;POINT(113.247710 23.136330)'::GEOMETRY,3857),
                ST_Transform('SRID=4326;POINT(87.576270 43.844990)'::GEOMETRY,3857)
        );
```

  - 提示：例5-9中的**GEOMETRY**也可以换成**GEOGRAPHY**，二者计算距离有较大差距，在计算地球上距离较远两点距离时，建议使用**GEOGRAPHY**。

5.2.3 图形之间度量函数

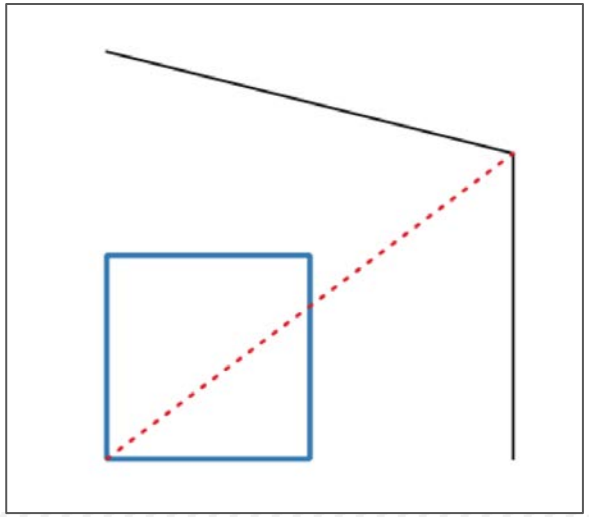
（三）ST_MaxDistance函数

- ST_MaxDistance函数计算两个二维几何图形的**最远距离**，语法如下：

```
float ST_MaxDistance(geometry g1, geometry g2);
```

- 例5-11. 计算图5-3所示两个几何图形之间的最远距离，语句如下：

```
SELECT ST_MaxDistance(
        'POLYGON((0 0, 10 0, 10 10, 0 10, 0 0))'::geometry,
        'LINESTRING(0 20,20 15,20 0)'::geometry
        );
```



5.2.3 图形之间度量函数

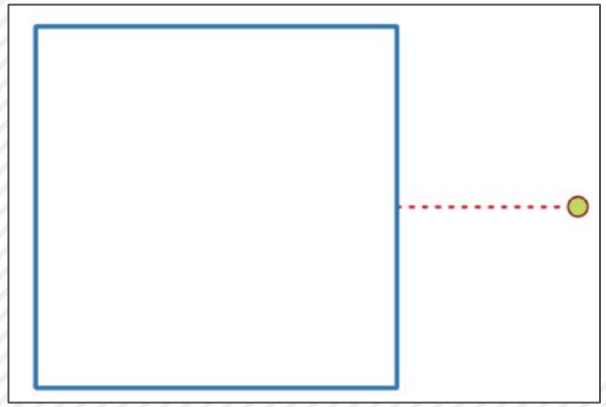
（三）ST_ShortestLine函数

- ST_ShortestLine函数计算两个二维几何图形之间的**最短直线**，语法如下：

```
geometry ST_ShortestLine(geometry g1, geometry g2);
```

- 例5-12. 计算如图5-4所示的点到矩形的最短直线，语句如下：

```
SELECT ST_AsText(
                ST_ShortestLine('POLYGON((0 0, 10 0, 10 10, 0 10, 0 0))'::geometry,
                                'POINT(15 5)'::GEOMETRY
));
```



5.3 空间关系函数

- ISO-SQL/MM标准和OGC-SFA规范都定义了**拓扑关系和距离关系**，并且对相应函数名称等都做了定义和说明。
- PostGIS空间数据库实现了OGC规范的**全部空间关系计算函数**，其中常用的拓扑函数有邻接与远离、包含覆盖、相交与穿过、相等,等.
  - 5.3.1 邻接与远离关系函数
  - 5.3.2 包含与在其中关系函数
  - 5.3.3 相交与穿过关系函数
  - 5.3.4 相等关系函数
  - 5.3.5 距离关系函数

5.3.1 邻接与远离关系函数

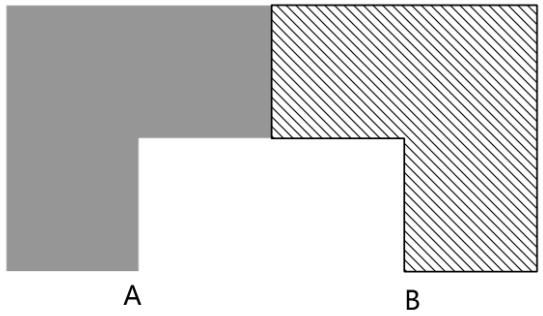
（一）ST_Touches函数

- ST_Touches函数判断几何图形A与几何图形B是否邻接。当且仅当A与B的共同点存在于A和B的边界上时，函数返回真；否则返回假。语法如下：

  boolean ST_Touches(geometry A, geometry B);

- **例5-13.** 判断下图所示两多边形是否邻接，语句如下：

```
SELECT
   ST_Touches('POLYGON((0 0,0 10,10 10, 10 5,5 5,5 0,0 0))'::GEOMETRY,
                  'POLYGON((10 5,10 10,20 10,20 0,15 0,15 5,10 5))'::GEOMETRY);
```



5.3.1 邻接与远离关系函数

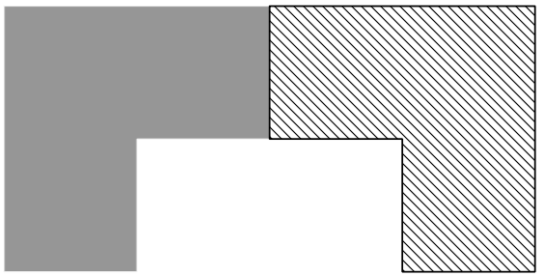
（二）ST_Disjoint函数

- ST_Disjoint函数判断几何图形A与B是否为远离关系。当A与B没有共同点时，函数返回真；否则返回假。语法如下：

  boolean ST_Disjoint(geometry A , geometry B);

- **例5-14.** 判断下图所示两多边形是否远离，语句如下：

```
SELECT
 ST_Disjoint('POLYGON((0 0,0 10,10 10, 10 5,5 5,5 0,0 0))'::GEOMETRY,
                    'POLYGON((10 5,10 10,20 10,20 0,15 0,15 5,10 5))'::GEOMETRY);
```



5.3.2 包含与在其中关系函数

（一）ST_Contains和ST_Within函数

- （1）ST_Contains函数计算几何图形geomA是否包含几何图形geomB。当且仅当geomB没有点在geomA外，函数返回真；否则返回假。语法如下：

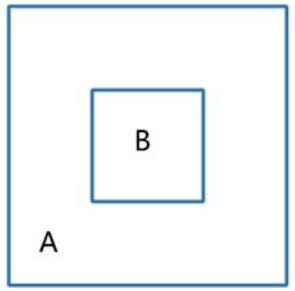
  boolean ST_Contains(geometry geomA, geometry geomB);

- （2）ST_Within函数计算几何图形geomA是否在几何图形geomB内部。当且仅当geomA没有点在geomB外部时，函数返回真；否则返回假。ST_Within是ST_Contains逆函数，语法如下：

  boolean ST_Within(geometry geomA, geometry geomB);

- 例5-15. 判断右图矩形A是否包括B，语句如下：

```
SELECT
    ST_Contains(
        'POLYGON((0 0, 10 0, 10 10, 0 10, 0 0))'::geometry,
        'POLYGON((3 3, 7 3, 7 7, 3 7, 3 3))'::geometry);
```



- 上例中，如何使用ST_Within判断是否包含？

5.3.2 包含与在其中关系函数

（二）ST_Covers和ST_CoveredBy函数

- （1）ST_Covers函数为重载函数，判断几何或地理图形A是否覆盖几何或地理图形B。当且仅当B没有点在A外，函数返回真；否则返回假。基本语法如下：

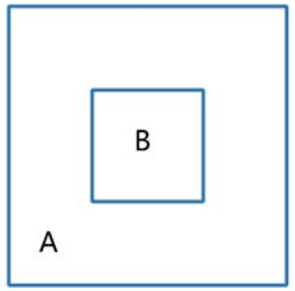
  boolean ST_Covers(geometry A, geometry B);

- （2）ST_CoveredBy函数为重载函数，判断几何或地理图形A是否被几何或地理图形B覆盖。当且仅当A没有点在B外，函数返回真；否则返回假。ST_CoveredBy是ST_Covers的逆函数。基本语法如下：

  boolean ST_CoveredBy(geometry A, geometry B);

- 例5-17. 判断右图矩形A是否覆盖B，语句如下：

```
SELECT
    ST_Covers (
        'POLYGON((0 0, 10 0, 10 10, 0 10, 0 0))'::geometry,
        'POLYGON((3 3, 7 3, 7 7, 3 7, 3 3))'::geometry);
```



- 上例中，如何使用ST_CoveredBy判断是否覆盖？

5.3.2 包含与在其中关系函数

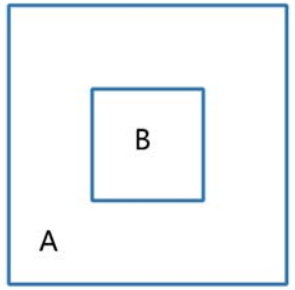
（三）ST_OverLaps函数

- ST_OverLaps函数判断几何图形A和B之间是存在部分重叠关系。当且仅当A与B有共同点集，共同点集不等于A且不等于B时，函数返回真；否则返回假。即当A与B相交部分不等于A也不等于B时，函数返回真，否则返回假。语法如下：

  boolean ST_Overlaps(geometry A, geometry B);

- 例5-19. 判断右图矩形A与B是否重叠，语句如下：

```
SELECT
    ST_OverLaps (
        'POLYGON((0 0, 10 0, 10 10, 0 10, 0 0))'::geometry,
        'POLYGON((3 3, 7 3, 7 7, 3 7, 3 3))'::geometry);
```



5.3.3 相交与穿过关系函数

（一）ST_Intersects函数

- ST_Intersects函数为重载函数，判断几何或地理图形A是否与几何或地理图形B是否相交。当存在A上的点在B上时，函数返回真；否则返回假。基本语法如下：

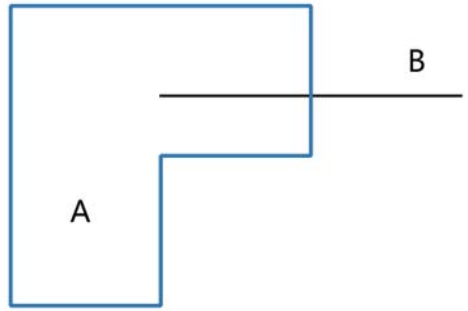
  boolean ST_Intersects(geometry A , geometry B);

- 例5-22. 判断右图所示两线串是否相交，语句如下：

```
SELECT
    ST_Intersects('LINESTRING(0 0,10 10,20 10)'::geometry,
                  'LINESTRING(20 10,30 3,35 3)'::geometry);
```

- 例5-23. 判断右图所示线串与多边形是否相交，语句如下：

```
SELECT
    ST_Intersects('POLYGON((0 0,0 10,10 10, 10 5,5 5,5 0,0 0))'::geometry,
                  'LINESTRING(5 7,15 7)'::geometry);
```

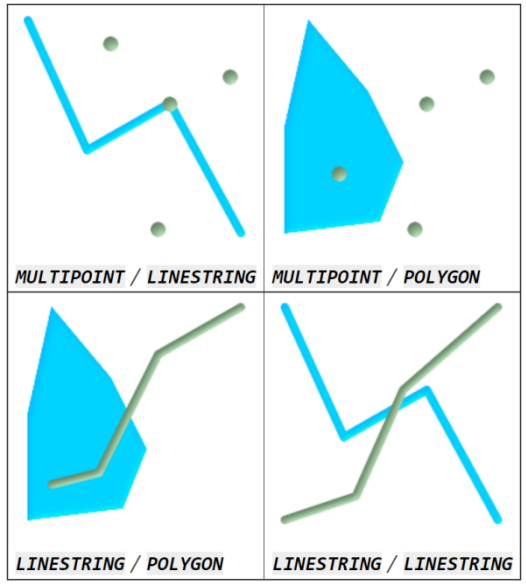


5.3.3 相交与穿过关系函数

（二）ST_Crosses函数

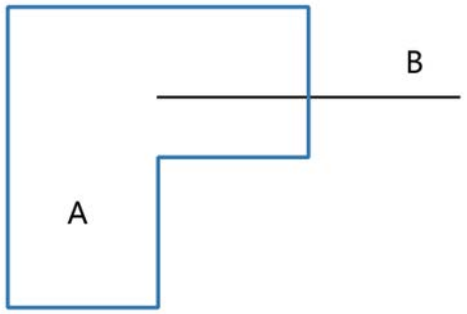
- ST_Crosses函数判断两个几何图形g1和g2是否为穿过或交叉关系。当g1和g2的交集不为空且交集不存在于任一图形全部、交集的维度一定小于两几何图形的最高维度时，函数返回真；否则返回假。下图所示的几何图形均为穿过关系，即ST_Crosses函数均返回真例5-22. 判断右图所示两线串是否相交，语句如下：

boolean ST_Crosses(geometry g1, geometry g2);



例5-24. 判断例5-23之间是否为穿过关系，语句如下：

```
SELECT
  ST_Crosses('POLYGON((0 0,0 10,10 10, 10 5,5 5,5 0,0 0))'::geometry,
                'LINESTRING(5 7,15 7)'::geometry);
```



5.3.4 相等关系函数

（一）ST_Equals函数

- ST_Equals函数判断两几何图形A和B是否相等。当且仅当两图形的点集相等时，函数返回真；否则返回假。或者，当ST_Within(A,B)=True且ST_Within(B,A)=True时，函数返回真；否则返回假。语法如下：

  boolean ST_Equals(geometry A, geometry B);

- 例5-25. 判断线串A=LINESTRING（0 0，10 10）与B=LINESTRING（0 0，5 5，10 10）是否相等，语句如下：

```
SELECT
    ST_Equals('LINESTRING(0 0,10 10)'::GEOMETRY,
                'LINESTRING(0 0,5 5,10 10)'::GEOMETRY);
```

5.3.4 相等关系函数

（二）ST_OrderingEquals函数

- ST_OrderingEquals函数不是ISO-SQL/MM和OGC SFA规范定义的函数，是PostGIS和ArcSDE自定义函数。当且仅当两图形的点集相等，且顶点顺序均相等时，函数返回真；否则返回假。语法如下：

  boolean ST_OrderingEquals(geometry A, geometry B);

- 例5-26. 判断线串A=LINESTRING（0 0，10 10）与B=LINESTRING（0 0，5 5，10 10）是否按序相等，语句如下：

```
SELECT
    ST_OrderingEquals('LINESTRING(0 0,10 10)'::GEOMETRY,
                      'LINESTRING(0 0,5 5,10 10)'::GEOMETRY);
```

5.3.5 距离关系函数

（一）ST_DWithin函数

- ST_DWithin函数为重载函数，判断几何图形或地理图形A和B**是否在给定的距离范围内**。如果输入的两图形距离小于给定的距离阈值，函数返回真；否则返回假。基本语法如下：

```
boolean ST_DWithin(geometry g1, geometry g2, double precision distance_of_srid);
```

其中，参数distance_of_srid是距离度量阈值。

- 例5-26. 判断几何图形A=LINESTRING(10.3 5,20 5)和几何图形B=POLYGON((0 0,10 0,10 10,0 10,0 0))之间距离是否小于0.5，语句如下：

```
SELECT  ST_DWithin('LINESTRING(10.3 5,20 5)'::geometry,
                   'POLYGON((0 0,10 0,10 10,0 10,0 0))'::geometry,
                   0.5);
```

5.3.5 距离关系函数

（二）ST_DFullyWithin函数

- ST_DFullWithin函数判断几何图形A和B**是否完全在给定的距离内**。当且仅当A和B的最大距离小于给定距离阈值时，函数返回真；否则返回假。语法如下：

```
boolean ST_DFullyWithin(geometry g1, geometry g2, double precision_distance);
```

其中，参数precision_distance是距离度量阈值。

- 例5-27. 判断例5-26中两图形是否完全在0.5距离内，语句如下：

```
SELECT ST_DFullyWithin(
                    'LINESTRING(10.3 5,20 5)'::geometry,
                    'POLYGON((0 0,10 0,10 10,0 10,0 0))'::geometry,
                    0.5);
```

5.3.1 邻接与远离关系函数

（二）ST_DFullyWithin函数

- ST_DFullWithin函数判断几何图形A和B**是否完全在给定的距离内**。当且仅当A和B的最大距离小于给定距离阈值时，函数返回真；否则返回假。语法如下：

```
boolean ST_DFullyWithin(geometry g1, geometry g2, double precision_distance);
```

其中，参数precision_distance是距离度量阈值。

- 例5-27. 判断例5-26中两图形是否完全在0.5距离内，语句如下：

```
SELECT ST_DFullyWithin(
                    'LINESTRING(10.3 5,20 5)'::geometry,
                    'POLYGON((0 0,10 0,10 10,0 10,0 0))'::geometry,
                    0.5);
```

5.4 空间运算函数

- ISO-SQL/MM和OGC SFA规范都定义了大量的**空间运算函数**，这些函数又分为**单图形运算函数**和**多图形运算函数**

  - **5.4.1 单图形运算函数**
  - **5.4.2 多图形运算函数**

5.4.1 单图形运算函数

- **单图形运算函数**是对输入的**一个图形参数**进行处理和分析并产生新图形的函数，主要包括ST_Buffer、ST_Centroid、ST_ConvexHull、ST_DelaunayTriangles、ST_OffsetCurve、ST_Simplify、ST_VoronoiLines、ST_VoronoiPolygons等函数。

- （一）ST_Buffer函数
  - ST_Buffer函数为重载函数，计算给定半径的几何或地理图形的缓冲区多边形。基本语法如下：

    geometry ST_Buffer(geometry g1, float radius_of_buffer, text buffer_style_parameters = '');
  - 参数说明：

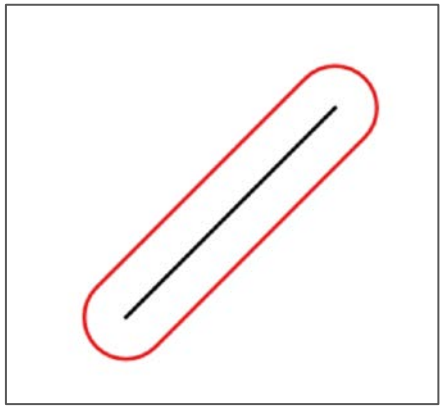
    （1）g1是输入的几何或地理图形。

    （2）radius_of_buffer是缓冲区半径，要设置与g1空间参考系对应的长度值。

    （3）第三个参数控制样式和准确性。

  - 例5-29. 计算线LINESTRING（0 0,5 5）的半径为1的缓冲区,语句如下：

    SELECT ST_AsText(ST_Buffer('LINESTRING(0 0,5 5)'::geometry,1));



5.4.1 单图形运算函数

（二）ST_Centroid函数

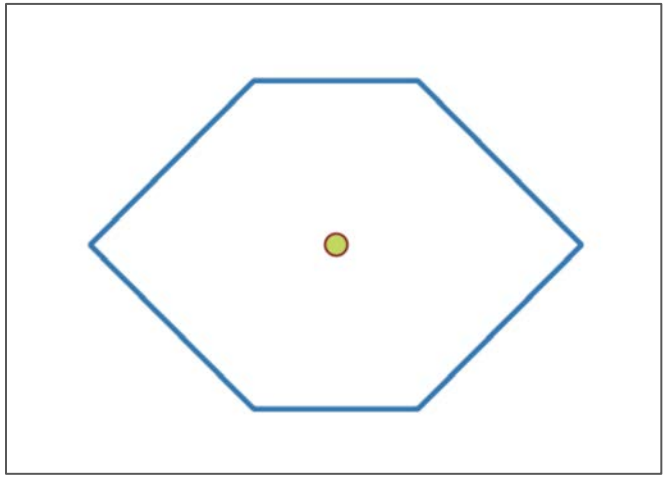
- ST_Centroid函数为重载函数，计算几何图形或地理图形的**几何中心**，基本语法如下：

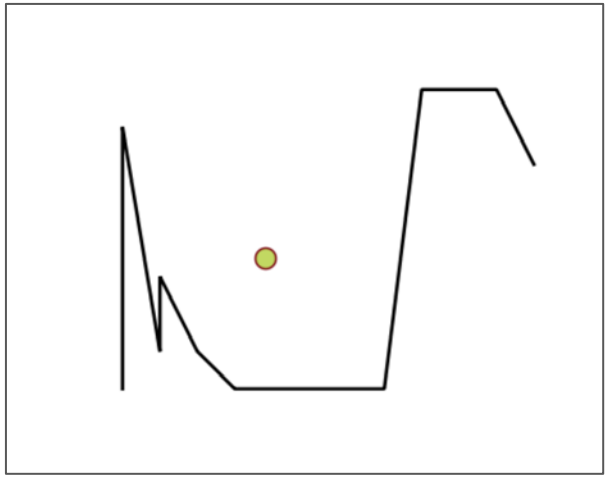
  geometry ST_Centroid(geometry g1);

- 此函数不仅可以计算多边形中心，也可以计算线、点、复合点等所有几何图形的中心。

- 例5-30. 计算图5-11(a)的多边形几何中心，语句如下：

  SELECT ST_AsText(ST_Centroid('POLYGON((0 2,-1 1,0 0,0.5 0,1 0,2 1,1 2,0.5 2,0 2))'::geometry));





5.4.1 单图形运算函数

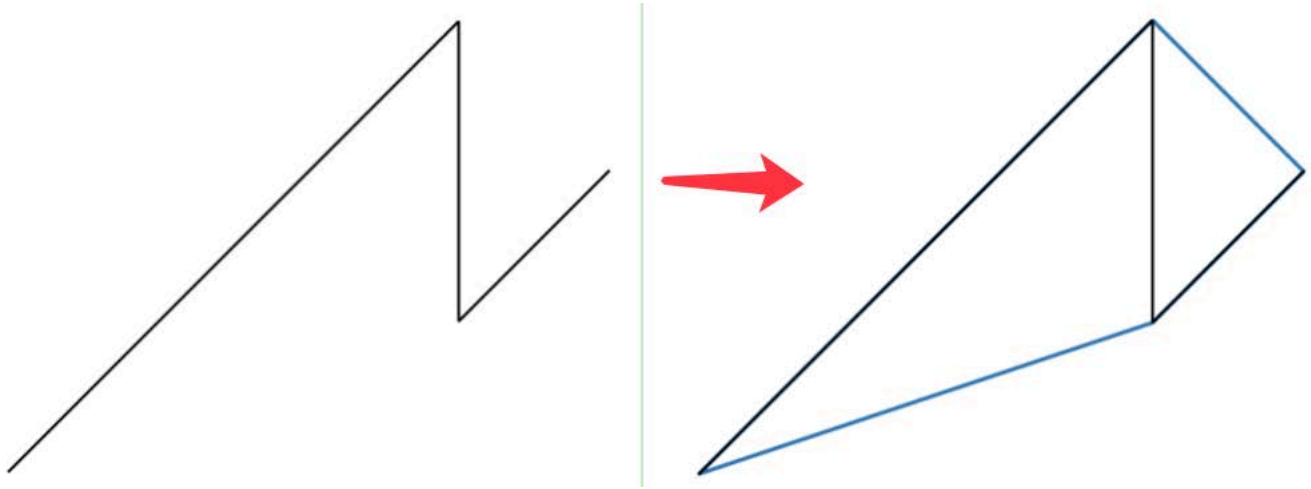
（三）ST_ConvexHull函数

- ST_ConvexHull函数计算几何图形的**最小凸壳（包）多边形**，语法如下：

  geometry ST_ConvexHull(geometry geomA);

- 例5-32. 计算线串“LINESTRING(0 0,15 15,15 5,20 10)”（图5-13(a)）的最小凸包多边形，语句如下：

  SELECT ST_AsText(ST_ConvexHull('LINESTRING(0 0,15 15,15 5,20 10)'::geometry));



5.4.1 单图形运算函数

（四）ST_OffsetCurve函数

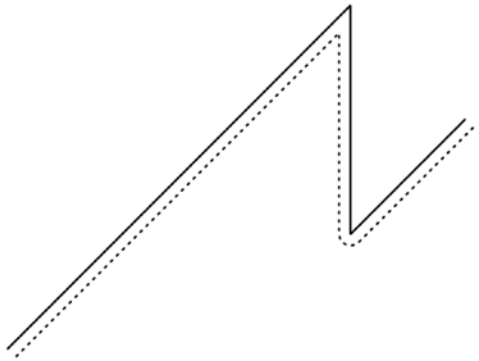
- ST_OffsetCurve函数构建线串几何图形的**平移/平行线串**，语法如下：

  geometry ST_OffsetCurve(geometry line, float signed_distance, text style_parameters='');

- 参数signed_distance表示左右平移距离，负数表示向右平移，正数表示向左平移；参数style_parameters表示构建平移线的样式，同ST_Buffer函数。

- 例5-33. 计算例5-32线串的向右平移0.5的平行线，语句如下：

```
SELECT ST_AsText(
        ST_OffsetCurve(
            'LINESTRING(0 0,15 15,15 5,20 10)'::geometry,
             -0.5,
             'quad_segs=4 join=round'
        ));。
```



5.4.1 单图形运算函数

（五）ST_Polygonize函数

- ST_Polygonize函数为重载函数，该函数根据输入的几何图形集合或数组**构建多边形**。语法如下：

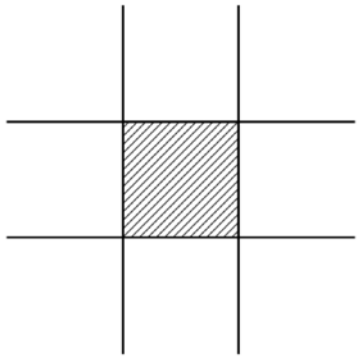
  ① geometry ST_Polygonize(geometry set geomfield);

  ② geometry ST_Polygonize(geometry[] geom_array);

  函数①根据几何图形集合构建多边形，函数②根据几何图形数组构建多边形。

- 例5-34. 利用图5-15(a)所示的复合线串生成多边形，语句如下：

```
SELECT
  ST_Polygonize('MULTILINESTRING((0 5,5 5),(5 5,10 5),(10 5,15 5),(0 10,5 10),(5 10,10 10),(10 10,15 10),
                (5 0,5 5),(5 5,5 10),(5 10,5 15),(10 0,10 5),(10 5,10 10),(10 10,10 15))'::geometry);
```



5.4.1 单图形运算函数

（六）ST_Simplify函数

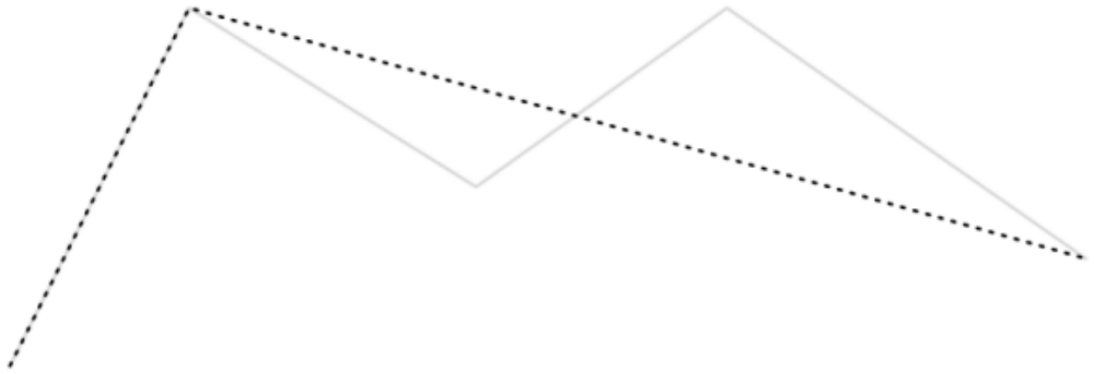
- ST_Simplify函数使用**Douglas-Peucker算法简化线串或多边形**几何图形，语法如下：

  geometry ST_Simplify(geometry geomA, float tolerance, boolean preserveCollapsed);

参数tolerance是Douglas-Peucker简化约束（Douglas etc，1973）；preserveCollapsed是避免因为几何图形长度小于tolerance而被忽略的选项，当此参数设置为真时，将保留小于tolerance的几何图形。

- 例5-35. 使用ST_Simplify函数简化图5-16(a)所示的线串，语句如下：

  SELECT ST_Simplify('LINESTRING(0 0,5 10,13 5,20 10,30 3)'::geometry,6);



5.4.1 单图形运算函数

（七）Delaunay三角网和Voronoi多边形函数

（1）ST_DelaunayTriangles函数

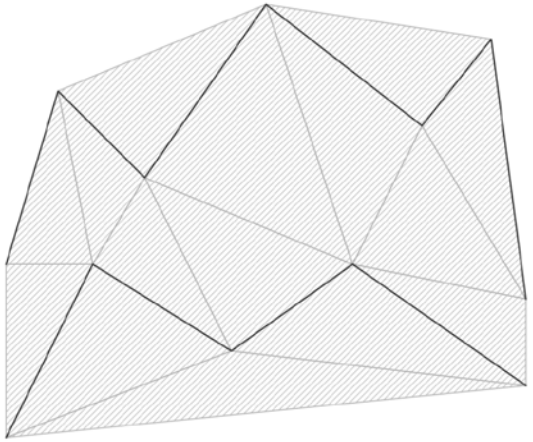
ST_DelaunayTriangles函数根据输入的几何图顶点集构建**Delaunay不规则三角网**，语法如下：

geometry ST_DelaunayTriangles(geometry g1, float tolerance, int4 flags);

参数tolerance控制点集密度，即小于此距离的点将合并为一个点；flags控制生成的三角网类型：0表示生成多边形集合（Polygon Collection），此值为默认值；1表示生成复合线（MultiLineString）类型；2表示生成TIN类型。

- 例5-37. 为图5-18(a)所示的两条线生成Delauney三角网，语句如下：

```
SELECT ST_DelaunayTriangles(ST_Union(
    'LINESTRING(0 0,5 10, 13 5,20 10,30 3)'::geometry,
    'LINESTRING(0 10,3 20, 8 15,15 25,24 18,28 23,30 8)'::geometry));
```



（七）Delauney三角网和Voronoi多边形函数

（2）ST_VoronoiLines函数

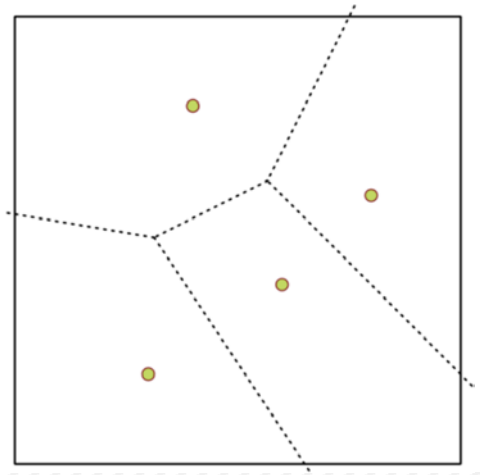
- ST_VoronoiLines函数根据输入的MultiPoint几何图形**生成Voronoi线**，语法如下：

```
geometry ST_VoronoiLines( geometry g1 , float8 tolerance , geometry extend_to );
```

参数tolerance同ST_DelaunayTriangles；extend_to控制Voronoi边线的范围。

- 例5-38. 为图5-19(a)所示的点集生成Voronoi图，语句如下：

```
SELECT ST_VoronoiLines('MULTIPOINT(3 2,4 8,8 6,6 4)'::geometry,
                       0.5,'POLYGON((0 0,10 0,10 10,0 10,0 0))'::geometry));
```



5.4.1 单图形运算函数

（七）Delauney三角网和Voronoi多边形函数

（3）ST_VoronoiPolygons函数

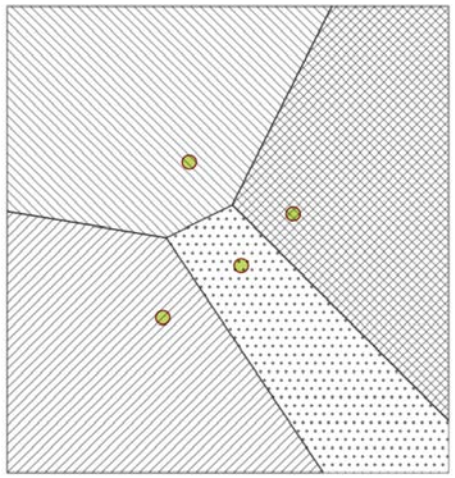
•ST_VoronoiPolygons函数和ST_VoronoiLines相似，都根据输入的MultiPoint点集生成对应的**Voronoi图**，此函数返回GeometryCollection类型。语法如下：

geometry ST_VoronoiPolygons (geometry g1 , float8 tolerance , geometry extend_to);

参数tolerance同ST_DelaunayTriangles；extend_to控制Voronoi边线的范围。

•例5-39. 为图5-19(a)所示的点集生成Voronoi图，语句如下：

```
SELECT ST_VoronoiPolygons ('MULTIPOINT(3 2,4 8,8 6,6 4)'::geometry,
                           0.5,'POLYGON((0 0,10 0,10 10,0 10,0 0))'::geometry));
```



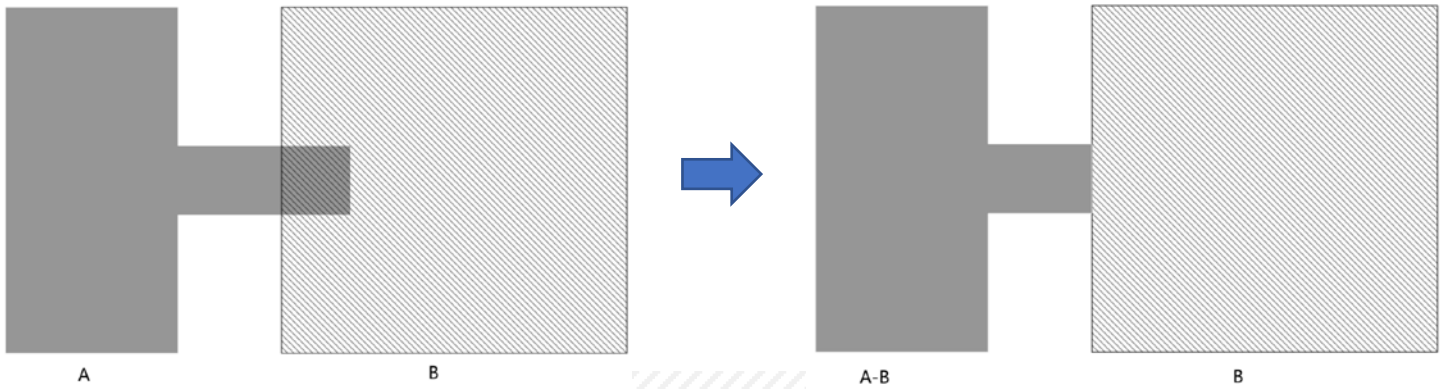
5.4.2 多图形运算函数

- **多图形运算函数**是接收**两个或以上图形**参数，并对其进行处理和分析且产生新图形的函数，主要包括ST_Difference、ST_Intersection、ST_Split、ST_SymDifference、ST_Union、ST_LineMerge等。
- （一）ST_Difference函数
  - ST_Difference函数计算几何图形A去除与几何图形B相交点集以后的结果，即**A的点集去除A与B共同的点集**，表示为A-（A∩B）。语法如下：

    geometry ST_Difference(geometry A, geometry B, float8 gridSize = -1);

    参数gridSize控制A和B是否向网格靠齐，默认为-1，表示不靠齐。
  - 例3-40. 计算图5-21(a)所示的多边形A去除B的结果，语句如下：

```
SELECT ST_Difference('POLYGON((0 0,5 0,5 4,10 4,10 6,5 6,5 10,0 10,0 0))'::geometry,
                     'POLYGON((8 0,18 0,18 10,8 10,8 0))'::geometry));
```



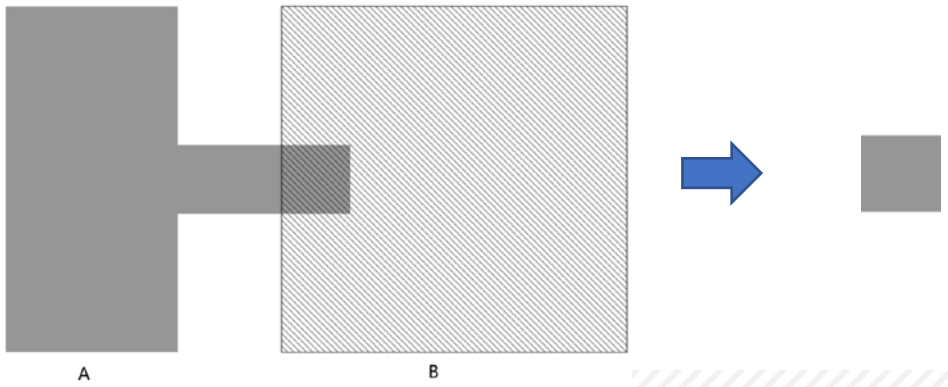
5.4.2 多图形运算函数

- （二）ST_Intersection函数
  - ST_Intersection函数为重载函数，计算几何图形或地理图形**A与B的公共部分**，即返回A与B的公共点集，表示为A∩B。基本语法如下：

    geometry ST_Intersection (geometry A, geometry B, float8 gridSize = -1);

  - 例3-41. 计算图5-21(a)所示的多边形A去除B的结果，语句如下：

```
SELECT ST_Intersection ('POLYGON((0 0,5 0,5 4,10 4,10 6,5 6,5 10,0 10,0 0))'::geometry,
                        'POLYGON((8 0,18 0,18 10,8 10,8 0))'::geometry));
```



5.4.2 多图形运算函数

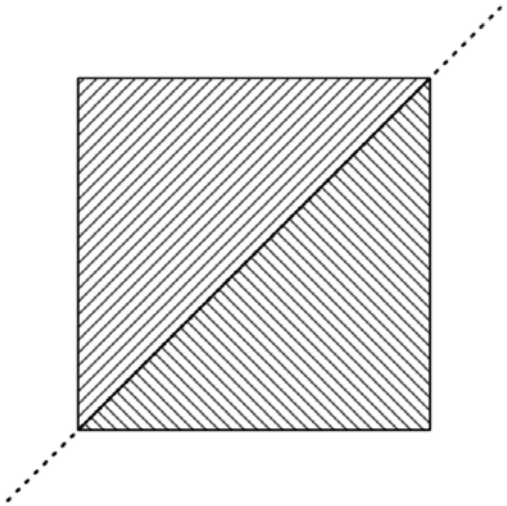
- （三）ST_Split函数
  - ST_Split函数计算几何图形input被几何图形blade的**分割结果**。语法如下：

    geometry ST_Split(geometry input, geometry blade);

    参数blade为线串，input可以是点、复合点、线串、复合线、多边形、复合多边形

  - 例3-42.计算图5-22(a)中多边形被线分割结果，语句如下：

    SELECT ST_Split('POLYGON((0 0,10 0,10 10,0 10,0 0))'::geometry,'LINESTRING(-2 -2,12 12)'::geometry));



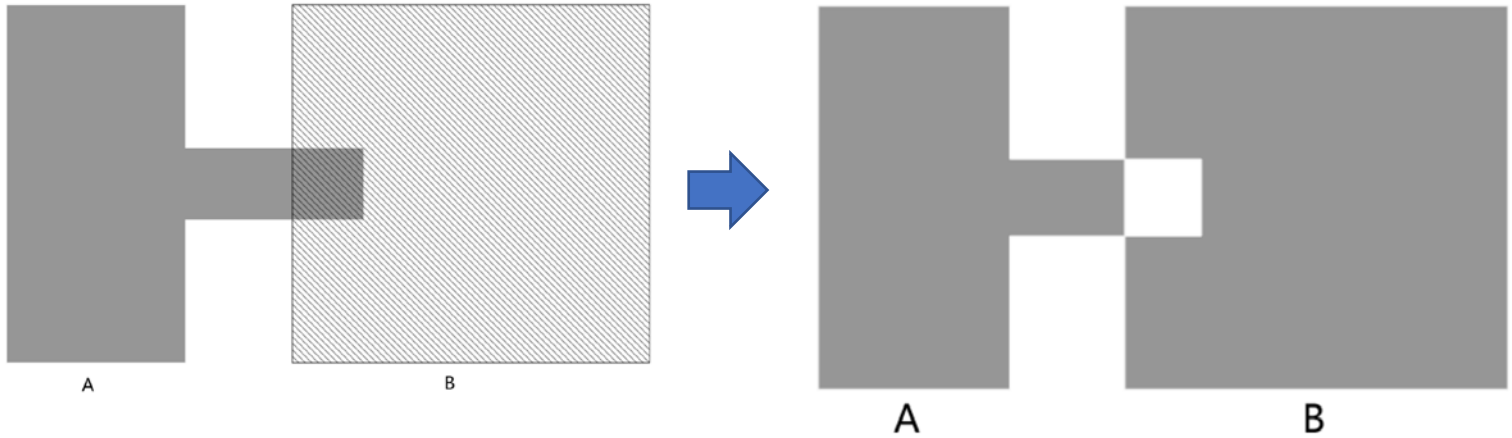
5.4.2 多图形运算函数

- （四）ST_SymDifference函数
  - ST_SymDifference函数称为对称差函数，计算几何图形A与B的**并集减去交集**的结果，表示为（A∪B）-（A∩B）。语法如下：

    geometry ST_SymDifference(geometry A, geometry B, float8 gridSize = -1);

  - 例5-43. 图5-21(a)两多边形的对称差，语句如下：

    SELECT ST_SymDifference('POLYGON((0 0,5 0,5 4,10 4,10 6,5 6,5 10,0 10,0 0))'::geometry,
    'POLYGON((8 0,18 0,18 10,8 10,8 0))'::geometry);

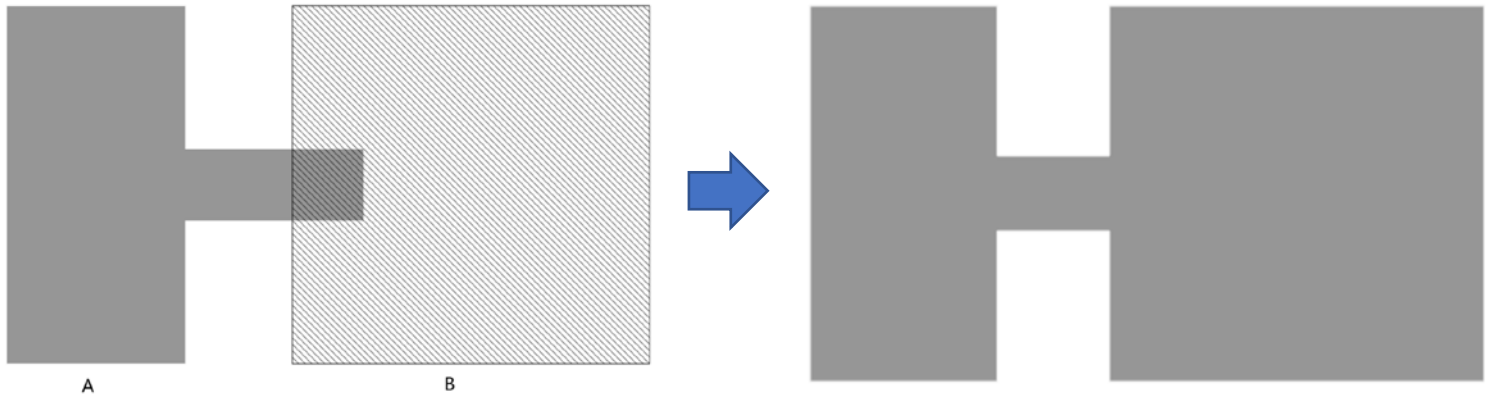


5.4.2 多图形运算函数

- （五）ST_Union函数
  - ST_Union函数为重载函数，计算多个几何图形的**合并结果**。语法如下：
    ① geometry ST_Union(geometry g1, geometry g2);
    ② geometry ST_Union(geometry g1, geometry g2, float8 gridSize);
    ③ geometry ST_Union(geometry[] g1_array);
    ④ geometry ST_Union(geometry set g1field);
    ⑤ geometry ST_Union(geometry set g1field, float8 gridSize);

    此函数不仅可以合并输入的多个几何图形，也可以合并空间数据表的几何图形字段里的所有图形。
  - 例5-44.合并图5-22(a)中多边形，语句如下：

```
SELECT POLYGON((0 0,5 0,5 4,10 4,10 6,5 6,5 10,0 10,0 0))'::geometry,'POLYGON((8 0,18 0,18 10,8 10,8 0))'::geometry));
```



5.4.2 多图形运算函数

- （六）ST_LineMerge函数
  - ST_LineMerge函数**拼接复合线串**几何图形并返回结果。语法如下：

    geometry ST_LineMerge(geometry amultilinestring);

    参数amultilinestring必须为复合线串（MultiLineString)类型的几何图形

  - 例5-45. 拼接输入复合线串并返回结果，语句如下：

    SELECT ST_LineMerge('MULTILINESTRING((-29 -27,-30 -29.7,-36 -31,-45 -33),(-45 -33,-46 -32))');

提示：当两条线存在端点相同时，合并成一条LineString，否则合并为MultiLineString.

5.4.3 线性参考函数

（一）ST_LineInterpolatePoint函数

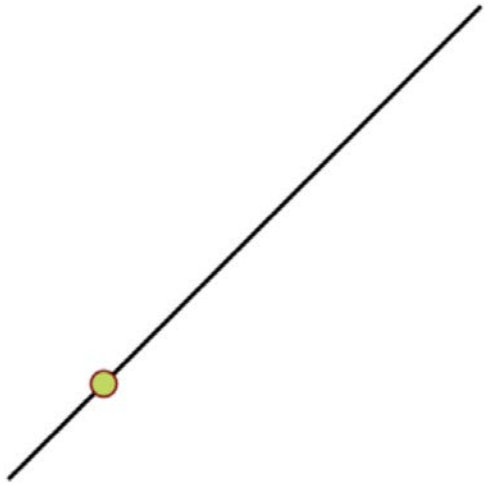
- ST_LineInterpolatePoint函数**计算线串上距离起点给定比例位置的坐标**。语法如下：

  geometry ST_LineInterpolatePoint(geometry a_linestring, float8 a_fraction);

  参数a_linestring是输入线串，a_fraction是线上的比例值，取值0~1

- 例5-46. 计算LineString(0 0,10 10)上0.2比例处的点坐标，语句如下：

  SELECT ST_LineInterpolatePoint('LineString(0 0,10 10)'::geometry,0.2);



5.4.3 线性参考函数

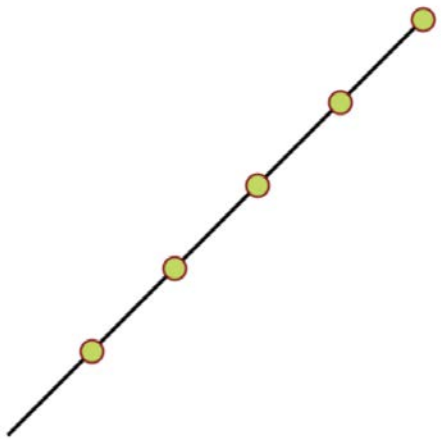
- （二）ST_LineInterpolatePoints函数
  - ST_LineInterpolatePoints按照给定的比例从输入的线串上获取若干个点，返回类型为复合点（MultiPoint）。语法如下：

    geometry ST_LineInterpolatePoints(geometry a_linestring, float8 a_fraction, boolean repeat);

    参数a_linestring是输入线串，a_fraction是线上的比例值，，repeat控制是否重复截取，默认True。

  - 例5-47. 使用ST_LineInterpolatePoints函数操作例5-46，语句如下：

    SELECT ST_LineInterpolatePoints ('LineString(0 0,10 10)'::geometry,0.2);



5.5 空间聚类

- 5.5.1 空间聚类概述
- 5.5.2 k-means聚类
- 5.5.3 DBSCAN聚类
- 5.5.4 k-means和DBSCAN算法对比

5.5.1 空间聚类概述

- **聚类**是把一个对象集合按照对象分布规则，分成若干个不同的组别（簇）或者子集（subset），从而保证同一个组别或子集中的对象都有**相似或相近的一些属性**，而组别之间的属性差别较为明显。
  - 传统聚类方法包括分裂和聚合两种类型。
- **空间聚类**是将空间要素集划分成具有一定意义的若干簇，使得每个簇内空间要素间具有**最大相似度**，簇间具有最大差别。
- 空间聚类方法大致分为：①基于划分的聚类方法、②基于层次的聚类方法（包括凝聚聚类和分裂聚类）、③局部聚类方法、④基于模糊集的聚类、⑤基于图理论的聚类、⑥混合聚类、⑦基于尺度空间的聚类等。

- PostGIS空间数据库实现了最为常用的两种经典空间聚类方法为**k-means**和**DBSCAN**。

5.5.2 K-MEANS聚类

- **k-means聚类**也称为k均值聚类算法，算法过程：首先随机选取k个观测对象为初始的聚类中心，每个聚类中心代表一个组；然后依次计算距离聚类中心最近的对象，并分配给该聚类中心所在组；在每次分配到一个对象时，聚类中心要根据当前对象集重新**计算当前对象集的均值中心**。上述过程不断重复迭代，直到满足终止条件。

- k-means聚类函数的语法如下：

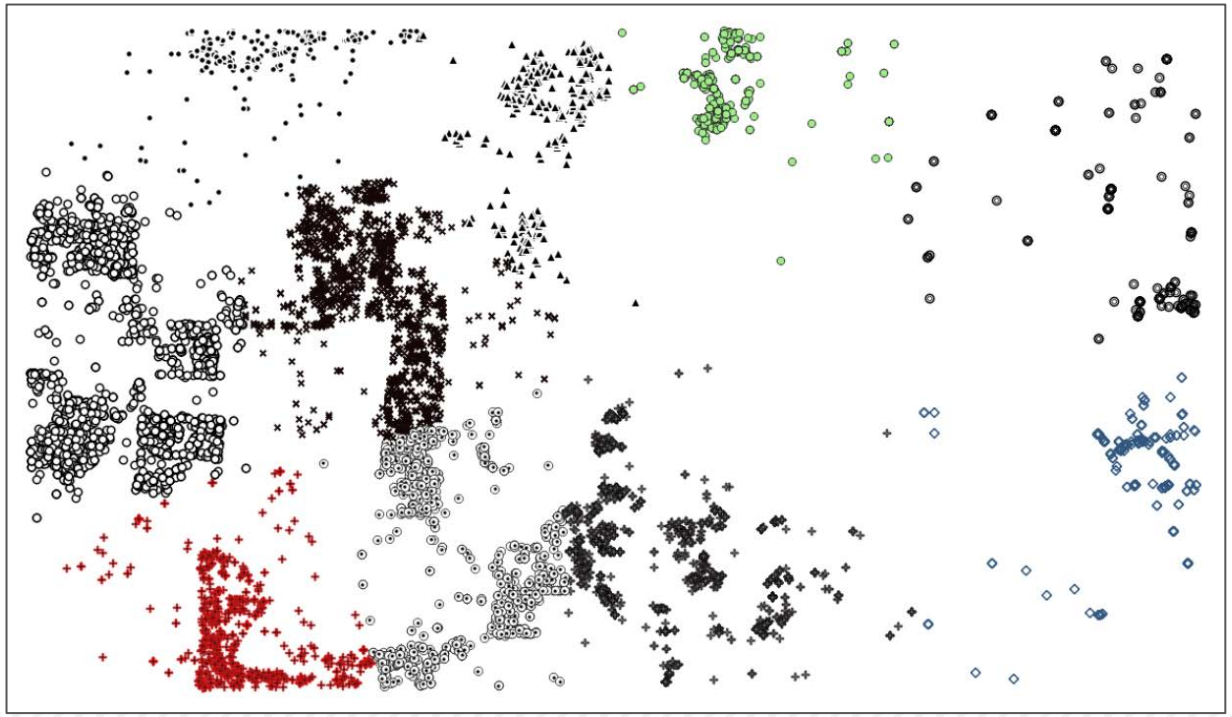
integer ST_ClusterKMeans(geometry set geom, integer number_of_clusters, float max_radius);

参数set geom是输入的几何图形集，当输入的要素包含M属性值时，此函数将M值作为权重值参与聚类计算；number_of_clusters是聚类数量；max_radius是每个聚类中心之间的最大距离，为可选项。

5.5.2 K-MEANS聚类

例5-48. 用k-means函数下图点集进行聚类，聚类数目k=10，语句如下：

```
① WITH kmeans AS
② (
③  SELECT
④    ST_ClusterKMeans ( geom, 10 ) OVER ( ) AS cid, ID
⑤  FROM
⑥    points
⑦ )
⑧ UPDATE points P
⑨  SET cid = K.cid
⑩ FROM
⑪   kmeans K
⑫  WHERE
⑬   P.ID = K.ID;
```



5.5.3 DBSCAN聚类

- **DBSCAN算法**将点集内所有邻接且点密度大于等于密度阈值的点聚为一类。
- DBSCAN算法设定了两个阈值：
  - eps表示点之间的距离，minpoints表示eps范围内的最小点数目。

- PostGIS中的DBSCAN函数语法如下：

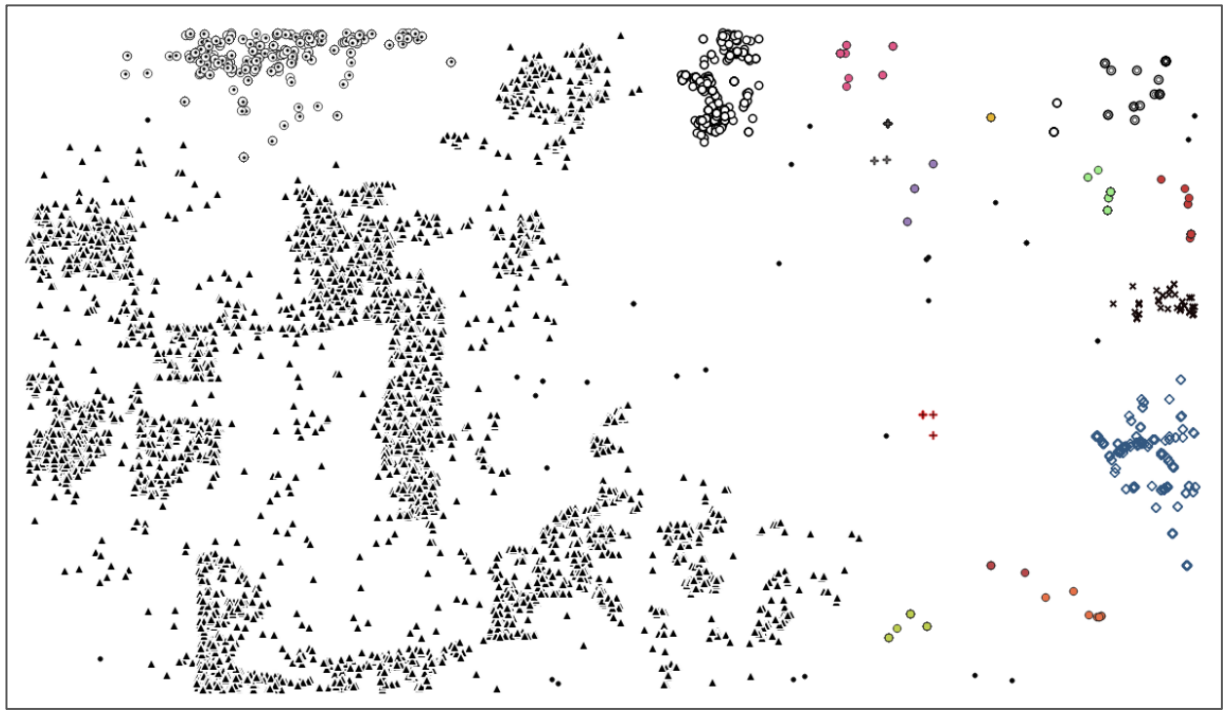
  integer ST_ClusterDBSCAN(geometry set geom, float8 eps, integer minpoints);

  参数set geom为要聚类的几何图形集合；eps为搜索半径；minpoints为eps范围内的点数量最小阈值。

5.5.3 DBSCAN聚类

- 例5-49. 使用DBSCAN方法对下图的点集进行聚类，语句如下：

```
① WITH dbscan AS
② (
③  SELECT
④   ST_ClusterDBSCAN(geom,eps:=0.005, minpoints:=5) OVER ( ) AS cid, ID
⑤  FROM
⑥                points
⑦ )
⑧ UPDATE points P
⑨    SET cid = D.cid
⑩ FROM
⑪   dbscan D
⑫  WHERE
⑬   P.ID = D.ID;
```



5.6 网络分析

- 5.6.1 网络分析概述
- 5.6.2 pgRouting

5.6.1 网络分析概述

- **网络分析**是借助图论和计算机算法、最优估计等理论，在地理网络数据集的基础上，对城市基础设施组成的网络进行地理分析和模型优化的过程。
- 通过研究网络状态，模拟和分析资源在地理网络上流动和分配情况，优化网络结构和资源分配。

- 根据链路资源的流动方向，网络链路可以分为有向链路和无向链路。
  - 有向链路是指资源流动具有一定的方向性，如河流的流向、电力的流向等
  - 无向链路是指资源流动没有明确的方向性，通常可以双向流动的链路

5.6.2 pgRouting

- pgRouting是一套开源程序，不仅可以增强空间数据库管理网络数据的功能，而且方便前端用户的共享与调用.
- pgRouting实现的算法：
  - Dijkstra算法
  - 约翰逊算法
  - 弗洛伊德-沃沙尔算法
  - A*算法
  - 双向Dijkstra*算法
- PostGIS里使用pgRouting:
  - 安装扩展：CREATE EXTENSION pgrouting;
  - 示例网站：https://docs.pgrouting.org/latest/en/sampledata.html

中南大學
CENTRAL SOUTH UNIVERSITY

感谢您的观看!