INFS4205/7205 Advanced Techniques for High Dimensional Data

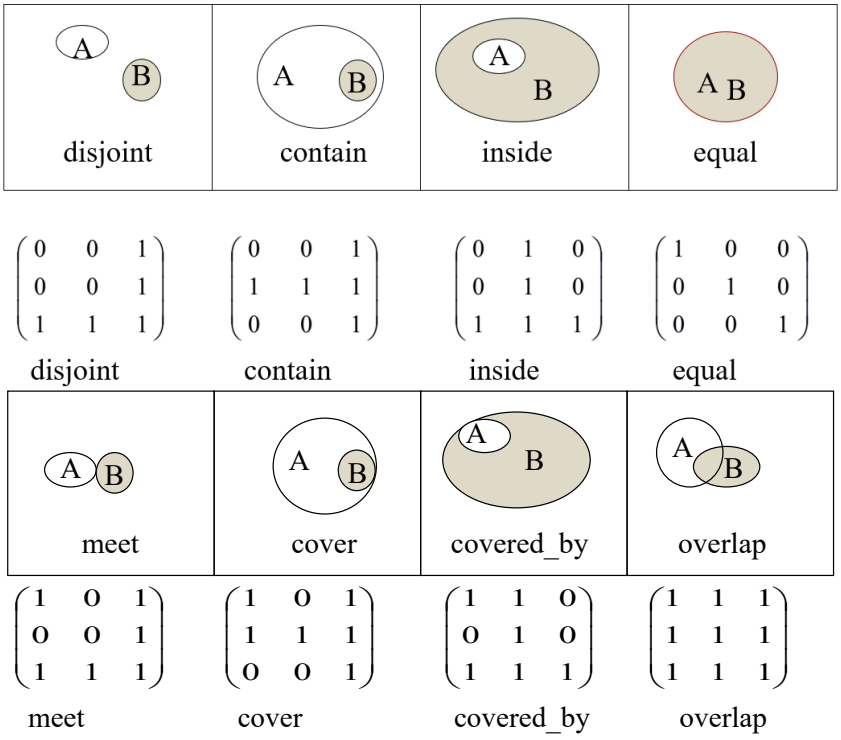
Tutorial 1: Spatial Database Introduction

Semester 1, 2021

Question 1: The 9-Intersectin Matrix is a topological model and a standard used to describe the spatial relations of two polygons (although it can be extended to line, we only discuss polygon here). Each polygon has three parts: A boundary, an interior, and an exterior. Therefore, for any two polygons, their topological relationships can be characterized using nine possible intersections between their interiors/boundaries/exteriors.

$$\begin{pmatrix} A_b \cap B_b & A_b \cap B_i & A_b \cap B_e \\ A_i \cap B_b & A_i \cap B_i & A_i \cap B_e \\ A_e \cap B_b & A_e \cap B_i & A_e \cap B_e \end{pmatrix}$$

Depending on the *True/False* of each intersection, eight and only eight topological relations can be identified. Please discuss and understand them.



Question 2: A demonstration of spatial database using PostGIS

PostGIS is a spatial extension that adds support for geographic objects to the PostgreSQL (Just like Oracle Spatial to Oracle). This tutorial works as a demonstration of some basic spatial operations. Some of the useful tools and link:

- PostGIS, you can download and try it on your computer : https://postgis.net
- The complete tutorial can here: https://postgis.net/workshops/postgis-intro/
- To view the spatial data, you can use the free QGIS: http://qgis.org
- The New York dataset used in this tutorial: http://s3.cleverelephant.ca/postgis-workshop-2018.zip

Spatial Database Creation

After the installation, open the pgsql command line to interact with PostgreSQL

- Create a database
  CREATE DATABASE sdbms;
- Connect to the new database
  \c sdbms
- Load the PostGIS spatial extension
  Create EXTENSION postgis;
- Confirm that PostGIS is installed
  SELECT postgis_full_version();

```
jinfm=# CREATE DATABASE sdbms;
CREATE DATABASE
jinfm=# \c sdbms
You are now connected to database "sdbms" as user "jinfm".
sdbms=# CREATE EXTENSION postgis;
ERROR:  extension "postgis" already exists
sdbms=# SELECT postgis_full_version();

             postgis_full_version

-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------
 POSTGIS="3.1.1 aaf4c79" [EXTENSION] PGSQL="130" GEOS="3.8.1-CAPI-1.13.3" PROJ="7.1.1"
GDAL="GDAL 3.1.4, released 2020/10/20" LIBXML="2.9.10" LIBJSON="0.13.1" LIBPROTOBUF="1.
3.3" WAGYU="0.5.0 (Internal)" TOPOLOGY RASTER
(1 row)
```

Data Import

The data used in this tutorial is the format of "shapefile", which commonly refers to a collection of files with .shp, .shx, .dbf that have the same prefix name. A shapefile is a simple, nontopological format for storing the geometric location and attribute information of geographic features.

Geographic features in a shapefile can be represented by points, lines, or polygons (areas). The workspace containing shapefiles may also contain dBASE tables, which can store additional attributes that can be joined to a shapefile's features.

POINT(0 0)
MULTIPOINT((0 0),(1 2))

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

GEOMETRYCOLLECTION(POINT(2 3),LINESTRING(2 3,3 4))

LINESTRING(0 0,1 1,1 2)
MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4)))

For example, the three mandatory files of the nyc_street shapefile are:

- **nyc_street.shp**: The main shape file, store the feature geometry like points, lines, and polygons (areas). The area features are represented as closed loop, double-digitized polygons. All the geometries are stored as a set of vector coordinates.
- **nyc_street.shx**: The shape index file, stores the offset of the corresponding main file record from the beginning of the main file.
- **nyc_street.dbf**: The attribute table file, stores the columnar attributes for each shape in dBASE format. It can be joined to a shapefile's features.

More details of the shapefile can be found here:
http://downloads.esri.com/support/whitepapers/mo_/shapefile.pdf
To import a shapefile into the database, we first use shp2pgsql command line tool that shipped with PostGIS to convert it into SQL script. For example:

- shp2pgsql -s 26918 nyc_streets.shp nyc_streets > nyc_streets.sql

The -s parameter set the *Spatial Reference Identifier* (SRID), which defines all the parameters of the data's geographic coordinate system and projection. All the other shapefiles can be converted similarly.

```
jinfm@d-i184-53-246 ~ % cd Desktop/INFS_4205_7205_tutorial/week1/postgis-workshop-2018/data
jinfm@d-i184-53-246 data % shp2pgsql -s 26918 nyc_streets.shp nyc_streets > nyc_streets.sql
Field id is an FTDouble with width 11 and precision 0
Shapefile type: Arc
Postgis type: MULTILINESTRING[2]
```

After that, in our sdmbs, we import these SQL scripts using command \i

- \i path\to\nyc_streets.sql

```
sdbms=# \i /Users/jinfm/Desktop/INFS_4205_7205_tutorial/week1/postgis-workshop-2018/data/nyc_streets.sql
SET
SET
BEGIN
CREATE TABLE
ALTER TABLE
                         addgeometrycolumn
-------------------------------------------------------------
 public.nyc_streets.geom SRID:26918 TYPE:MULTILINESTRING DIMS:2
(1 row)

INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
```

- \d nyc_streets

```
sdbms=# \d nyc_streets
                                        Table "public.nyc_streets"
 Column |             Type              | Collation | Nullable |                  Default
--------+-------------------------------+-----------+----------+-------------------------------------------
 gid    | integer                       |           | not null | nextval('nyc_streets_gid_seq'::regclass)
 id     | double precision              |           |          |
 name   | character varying(200)        |           |          |
 oneway | character varying(10)         |           |          |
 type   | character varying(50)         |           |          |
 geom   | geometry(MultiLineString,26918) |           |          |
Indexes:
    "nyc_streets_pkey" PRIMARY KEY, btree (gid)
```

Now you can use QGIS to connect to your database and see these geometries.

Data Description

Before we query into the data, let's first describe their attributes:

nyc_census_blocks

A census block is the smallest geography for which census data is reported. All higher level census geographies (block groups, tracts, metro areas, counties, etc) can be built from unions of census blocks.

blkid	A 15-digit code that uniquely identifies every census **block**. Eg: 360050001009000
popn_total	Total number of people in the census block
popn_white	Number of people self-identifying as "White" in the block
popn_black	Number of people self-identifying as "Black" in the block
popn_nativ	Number of people self-identifying as "Native American" in the block
popn_asian	Number of people self-identifying as "Asian" in the block
popn_other	Number of people self-identifying with other categories in the block
boroname	Name of the New York borough. Manhattan, The Bronx, Brooklyn, Staten Island, Queens
geom	Polygon boundary of the block

nyc_neighborhoods

name	Name of the neighborhood
boroname	Name of the New York borough. Manhattan, The Bronx, Brooklyn, Staten Island, Queens
geom	Polygon boundary of the neighborhood

nyc_streets

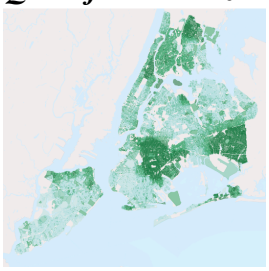
name	Name of the street
oneway	Is the street one-way? "yes" = yes, "" = no
type	Road type (primary, secondary, residential, motorway)
geom	Linear centerline of the street

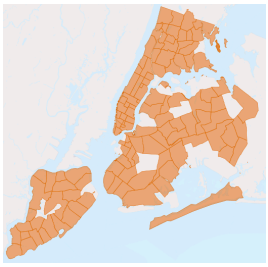
nyc_subway_stations

name	Name of the station
borough	Name of the New York borough. Manhattan, The Bronx, Brooklyn, Staten Island, Queens
routes	Subway lines that run through this station
transfers	Lines you can transfer to via this station
express	Stations where express trains stop, "express" = yes, "" = no
geom	Point location of the station

```
sdbms=# \dt
              List of relations
 Schema |        Name          | Type  | Owner
--------+----------------------+-------+-------
 public | nyc_census_blocks    | table | jinfm
 public | nyc_neighborhoods    | table | jinfm
 public | nyc_streets          | table | jinfm
 public | nyc_subway_stations  | table | jinfm
 public | spatial_ref_sys      | table | jinfm
(5 rows)
```

QGIS for visualization


nyc_census_blocks


nyc_neighborhoods


nyc_streets


nyc_subway_station

Simple Spatial SQL

1. What is the population of the City of New York?
   SELECT Sum(popn_total) **AS** population
   FROM nyc_census_blocks;

```
sdbms=# Select Sum(popn_total) As population
sdbms-# From nyc_census_blocks;
 population
------------
    8175032
(1 row)
```

2. What is the population of the Bronx?
   SELECT Sum(popn_total) **AS** population
   FROM nyc_census_blocks
   WHERE boroname = 'The Bronx

```
sdbms=# Select Sum(popn_total) As population
sdbms-# From nyc_census_blocks
sdbms-# Where boroname='The Bronx';
 population
------------
    1385108
(1 row)
```

3. For each borough, what percentage of the population is white?
   SELECT boroname, 100 * **Sum**(popn_white)/**Sum**(popn_total) **AS** white_pct
   FROM nyc_census_blocks
   GROUP BY boroname;

```
sdbms=# Select boroname, 100*Sum(popn_white)/Sum(popn_total) As white_pct
sdbms-# From nyc_census_blocks
sdbms-# Group By boroname;
   boroname    |     white_pct
---------------+--------------------
 Queens        |  39.72207739459101
 Brooklyn      |  42.80117379326865
 The Bronx     | 27.903744689944755
 Manhattan     |  57.44930394804628
 Staten Island |   72.8942034860154
(5 rows)
```

Geometry Query

1. What is the area of the 'West Village' neighborhood?
   SELECT ST_Area(geom)
   FROM nyc_neighborhoods
   WHERE name = 'West Village';

ST_Area(): Returns the area of a polygonal geometry.

```
sdbms=# Select ST_Area(geom)
sdbms-# From nyc_neighborhoods
sdbms-# Where name='West Village';
      st_area
--------------------
 1044614.5296485956
(1 row)
```

2. What is the area of Manhattan in acres?
   SELECT Sum(ST_Area(geom)) / 4047
   FROM nyc_neighborhoods
   WHERE boroname = 'Manhattan';

```
sdbms=# Select Sum(ST_Area(geom))/4047
sdbms-# From nyc_neighborhoods
sdbms-# Where boroname='Manhattan';
      ?column?
-------------------
 13965.32012239119
(1 row)
```

ST_GeometryN(geom,n):
Returns the n-th geometry within a geometry object.

3. How many census blocks in New York City have a hole in them?

   SELECT Count(*)
   FROM nyc_census_blocks
   WHERE ST_NumInteriorRings(**ST_GeometryN**(geom,1)) > 0;

Here, we convert the MultiPolygon geometries of the blocks into simple polygons, so we extract the first polygon from each collection using ST_GeometryN().

```
sdbms=# Select Count(*)
sdbms-# From nyc_census_blocks
sdbms-# Where ST_NumInteriorRings(ST_GeometryN(geom,1))>0;
 count
-------
    43
(1 row)
```

ST_NumInteriorRings(geom):

Return the number of interior rings of the first polygon in the geometry.
This will work with both POLYGON and MULTIPOLYGON types but only looks at the first polygon.
Return NULL if there is no polygon in the geometry.

4. What is the total length of streets (in kilometers) in New York City?

   SELECT Sum(ST_Length(geom)) / 1000
   FROM nyc_streets;

```
sdbms=# Select Sum(ST_Length(geom))/1000
sdbms-# From nyc_streets;
      ?column?
--------------------
 10418.904717199996
(1 row)
```

5. How long is 'Columbus Cir' (Columbus Circle)?

   SELECT ST_Length(geom)
   FROM nyc_streets
   WHERE name = 'Columbus Cir';

```
sdbms=# Select ST_Length(geom)
sdbms-# From nyc_streets
sdbms-# Where name='Columbus Cir';
     st_length
-------------------
 308.3419936909855
(1 row)
```

6. How many polygons are in the 'West Village' multipolygon?

   SELECT ST_NumGeometries(geom)
   FROM nyc_neighborhoods
   WHERE name = 'West Village';

```
sdbms=# Select ST_NumGeometries(geom)
sdbms-# From nyc_neighborhoods
sdbms-# Where name='West Village';
 st_numgeometries
------------------
                1
(1 row)
```

7. What is the length of streets in New York City, summarized by type?
   SELECT type, **Sum(ST_Length**(geom)) **AS** length
   FROM nyc_streets
   GROUP BY type
   ORDER BY length DESC;

```
sdbms=# Select type,Sum(ST_Length(geom)) As length
sdbms-# from nyc_streets
sdbms-# group by type
sdbms-# order by length DESC;
                      type                       |       length
-------------------------------------------------+--------------------
 residential                                     |   8629870.33786606
 motorway                                        |  403622.4781263628
 tertiary                                        |  360394.8790513027
 motorway_link                                   |   294261.419479668
 secondary                                       |  276264.3038979258
 unclassified                                    | 166936.37160445828
 primary                                         |  135034.2330179469
 footway                                         |  71798.48783780965
 service                                         | 28337.635038596007
 trunk                                           |  20353.58198260764
 cycleway                                        |  8863.751448259294
 pedestrian                                      |  4867.050328250257
 construction                                    |  4803.081621035617
 residential; motorway_link                      | 3661.5750629374543
 trunk_link                                      | 3202.1898124020076
 primary_link                                    | 2492.5745708353616
 living_street                                   | 1894.6390545733245
 primary; residential; motorway_link; residential | 1367.7657694133486
 undefined                                       |  380.5386191034604
 steps                                           | 282.74522134212725
 motorway_link; residential                      | 215.07778911517033
(21 rows)
```

Spatial Relationship Function

1. **ST_Equals(geometry A, geometry B**) tests the spatial equality of two geometries. It returns TRUE if two geometries of the same type have identical x,y coordinate values, i.e. if the second shape is equal (identical) to the first shape.

   The ordering of points can be different but represent the same geometry structure

Example:

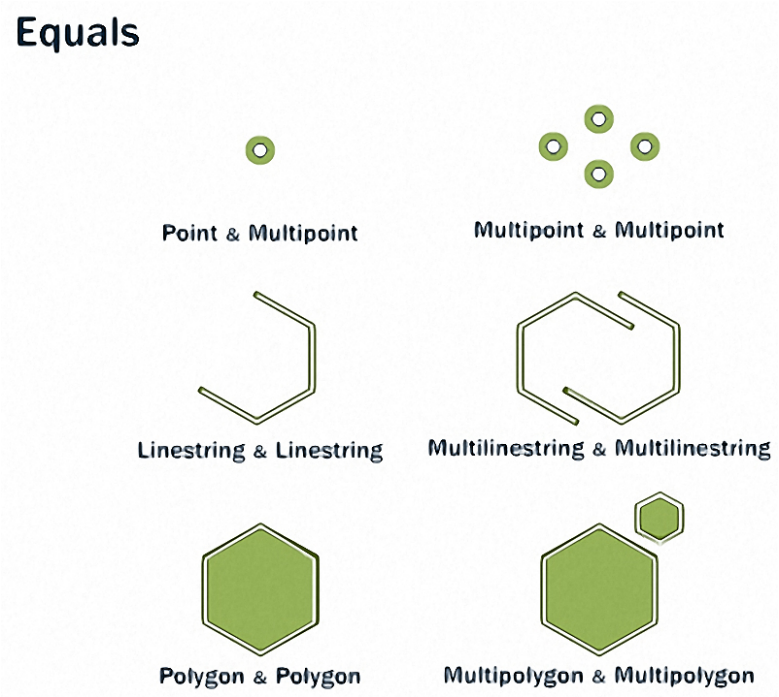
SELECT name, geom, ST_AsText(geom)
FROM nyc_subway_stations
WHERE name = 'Broad St';

```
sdbms=# Select name,geom,ST_AsText(geom)
sdbms-# From nyc_subway_stations
sdbms-# Where name='Broad St';
   name   |                        geom                        |                 st_astext
----------+----------------------------------------------------+-------------------------------------------
 Broad St | 0101000020266900000EEBD4CF27CF2141BC17D69516315141 | POINT(583571.9059213118 4506714.341192182)
(1 row)
```

This code is an exact representation of a specific point. It is not very human readable. But for a test like equality, using the exact coordinates is necessary.

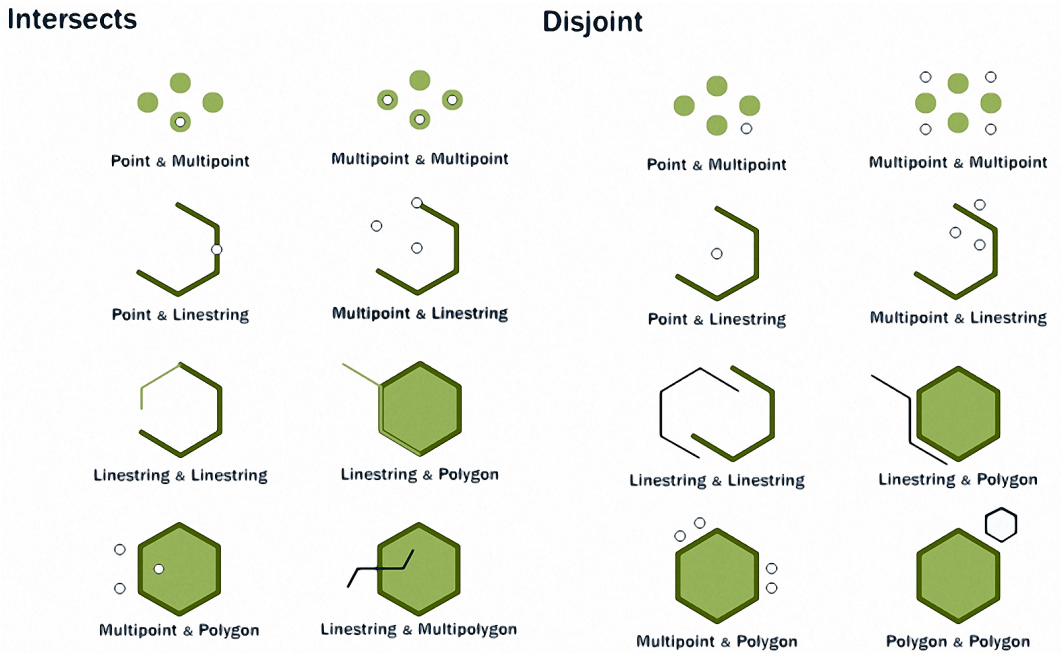
SELECT name
FROM nyc_subway_stations
WHERE ST_Equals(geom,
'0101000020266900000EEBD4CF27CF2141BC17D69516315141');

```
sdbms=# Select name
sdbms-# From nyc_subway_stations
sdbms-# Where ST_Equals(geom, '0101000020266900000EEBD4CF27CF2141BC17D69516315141');
   name
----------
 Broad St
(1 row)
```



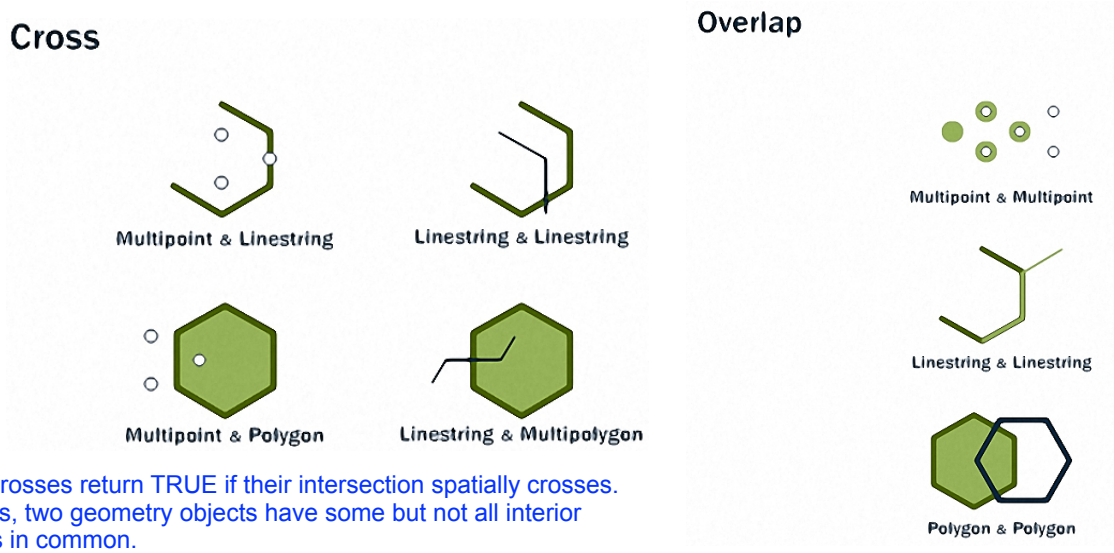
2. **ST_Intersects(geometry A, geometry B)** returns t (TRUE) if the two shapes have any space in common, i.e., if their boundaries or interiors intersect.

3. **ST_Disjoint(geometry A , geometry B)** is the Opposite of ST_INTERSECTS. If two geometries are disjoint, they do not intersect, and vice-versa. In fact, it is often more efficient to test "not intersects" than to test "disjoint" because the intersects tests can be spatially indexed, while the disjoint test cannot.



4. **ST_Crosses(geometry A, geometry B)** returns t (TRUE) if the intersection results in a geometry whose dimension is one less than the maximum dimension of the two source geometries and the intersection set is interior to both source geometries.

5. **ST_Overlaps(geometry A, geometry B)** compares two geometries of the same dimension and returns TRUE if their intersection set results in a geometry different from both but of the same dimension.



ST_Crosses return TRUE if their intersection spatially crosses. That is, two geometry objects have some but not all interior points in common.

Example:

```
SELECT  name, ST_AsText(geom)
FROM    nyc_subway_stations
WHERE   name = 'Broad St';
```

```
sdbms=# Select name,ST_AsText(geom)
sdbms-# From nyc_subway_stations
sdbms-# Where name='Broad St';
   name   |                 st_astext
----------+-------------------------------------------
 Broad St | POINT(583571.9059213118 4506714.341192182)
(1 row)
```

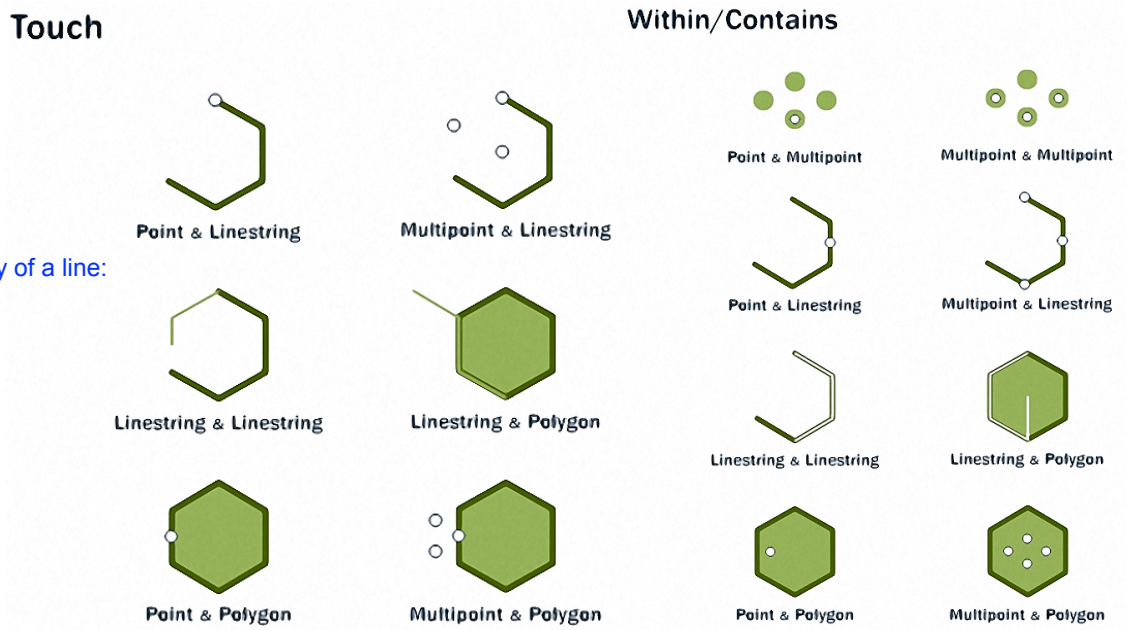
SELECT name, boroname
FROM nyc_neighborhoods
WHERE ST_Intersects(geom, ST_GeomFromText('POINT(5835714506714)', 26918));

```
sdbms=# Select name,boroname
sdbms-# From nyc_neighborhoods
sdbms-# Where ST_Intersects(geom,ST_GeomFromText('POINT(583571 4506714)',26918));
        name        | boroname
--------------------+-----------
 Financial District | Manhattan
(1 row)
```

This query returns the name and boroname of every neighborhood that intersects with the Broad St subway station.

6. **ST_Touches** tests whether two geometries touch at their boundaries, but do not intersect in their interiors. **ST_Touches(geometry A, geometry B)** returns TRUE if either of the geometries' boundaries intersect or if only one of the geometry's interiors intersects the other's boundary.
7. **ST_Within(geometry A , geometry B)** returns TRUE if the first geometry is completely within the second geometry. ST_Within tests for the exact opposite result of ST_Contains.
8. **ST_Contains(geometry A, geometry B)** returns TRUE if the second geometry is completely contained by the first geometry.

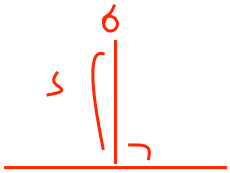
The boundary of a line: its endpoints



9. The **ST_Distance(geometry A, geometry B)** calculates the *shortest* distance between two geometries and returns it as a float. This is useful for actually reporting back the distance between objects.

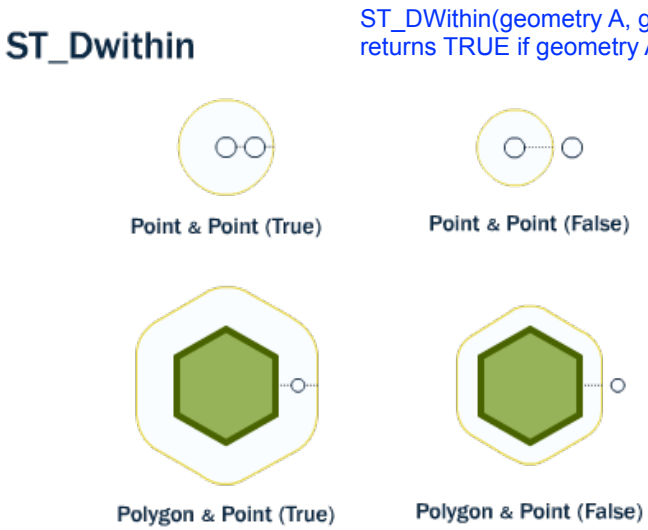
Example:

```
SELECT ST_Distance(
        ST_GeometryFromText('POINT(0 5)'),
        ST_GeometryFromText('LINESTRING(-2 2, 2 2)'));
```



```
sdbms=# Select ST_Distance(
sdbms(# ST_GeometryFromText('POINT(0 5)'),
sdbms(# ST_GeometryFromText('LINESTRING(-2 2,2 2)'));
 st_distance
-------------
           3
(1 row)
```

10. For testing whether two objects are within a distance of one another, the **ST_DWithin** function provides an index-accelerated true/false test. This is useful for questions like "how many trees are within a 500 meter buffer of the road?". You don't have to calculate an actual buffer, you just have to test the distance relationship.



Example:

```
SELECT  name
FROM    nyc_streets
WHERE   ST_DWithin(geom,  ST_GeomFromText('POINT(583571  4506714)',
26918), 10);
```

```
sdbms=# Select name
sdbms-# From nyc_streets
sdbms-# Where ST_DWithin(geom,ST_GeomFromText('POINT(583571 4506714)', 26918), 10);
   name
-----------
 Wall St
 Broad St
 Nassau St
(3 rows)
```

Spatial Relationship Query

1. "What is the geometry value for the street named 'Atlantic Commons'?"

```
SELECT ST_AsText(geom)
FROM   nyc_streets
WHERE  name = 'Atlantic Commons';
```

```
sdbms=# SELECT ST_AsText(geom)
sdbms-# From nyc_streets
sdbms-# Where name='Atlantic Commons';
                                                st_astext
-----------------------------------------------------------------------------------------------------
 MULTILINESTRING((586781.7015777241 4504202.153143394,586863.5196448397 4504215.988170098))
(1 row)
```

2. "What neighborhood and borough is Atlantic Commons in?"

Is ST_Touches() okay for this query?

```
SELECT name, boroname
FROM   nyc_neighborhoods
WHERE ST_Intersects(geom, ST_GeomFromText('LINESTRING(586782 4504202,586864 4504216)', 26918));
```

```
sdbms=# Select name,boroname
sdbms-# From nyc_neighborhoods
sdbms-# Where ST_Intersects(geom,ST_GeomFromText('LINESTRING(586782 4504202,586864 4504216)', 26918));
    name    | boroname
------------+----------
 Fort Green | Brooklyn
(1 row)
```

3. "What streets does Atlantic Commons join with?"

distance threshold = 0.1 ?
Again, the coordinates of the linestring have been rounded rather than the exact values. So it is not proper to use ST_Crosses() here.

```
SELECT  name
FROM    nyc_streets
WHERE  ST_DWithin(geom,ST_GeomFromText('LINESTRING(586782 4504202,586864 4504216)', 26918), 0.1);
```

```
sdbms=# Select name
sdbms-# From nyc_streets
sdbms-# Where ST_DWithin(geom,ST_GeomFromText('LINESTRING(586782 4504202,586864 4504216)', 26918), 0.1);
       name
------------------
 Atlantic Commons
 Cumberland St
(2 rows)
```

4. "Approximately how many people live on (within 50 meters of) Atlantic Commons?"

```
SELECT Sum(popn_total)
FROM nyc_census_blocks
WHERE ST_DWithin(geom, ST_GeomFromText('LINESTRING(586782 4504202,586864 4504216)', 26918),50);
```

```
sdbms=# Select Sum(popn_total)
sdbms-# From nyc_census_blocks
sdbms-# Where ST_DWithin(geom,ST_GeomFromText('LINESTRING(586782 4504202,586864 4504216)', 26918),50);
 sum
------
 1438
(1 row)
```

Spatial Join Query

By default, we are using an INNER JOIN.

In the previous section, we explored spatial relationships using a two-step process: first we extracted a subway station point for 'Broad St'; then, we used that point to ask further questions such as "what neighborhood is the 'Broad St' station in?"

Using a spatial join, we can answer the question in one step, retrieving information about the subway station and the neighborhood that contains it:

Any function that provides a true/false relationship between two tables can be used to drive a spatial join, but the most commonly used ones are: ST_Intersects, ST_Contains, and ST_DWithin.

```
SELECT    subways.name AS subway_name,
          neighborhoods.name AS neighborhood_name,
          neighborhoods.boroname AS borough
FROM      nyc_neighborhoods AS neighborhoods
JOIN      nyc_subway_stations AS subways
ON        ST_Contains(neighborhoods.geom, subways.geom)
WHERE     subways.name = 'Broad St';
```

```
sdbms=# Select subways.name As subway_name,
sdbms-# neighborhoods.name As neighborhood_name,
sdbms-# neighborhoods.boroname As borough
sdbms-# From nyc_neighborhoods As neighborhoods
sdbms-# Join nyc_subway_stations As subways
sdbms-# On ST_Contains(neighborhoods.geom, subways.geom)
sdbms-# Where subways.name='Broad St';
 subway_name | neighborhood_name  |  borough
-------------+--------------------+-----------
 Broad St    | Financial District | Manhattan
(1 row)
```

1. "What is the population and racial make-up of the neighborhoods of Manhattan?"

```
SELECT    neighborhoods.name AS neighborhood_name,
          Sum(census.popn_total) AS population,
          100.0 * Sum(census.popn_white)/Sum(census.popn_total) AS white_pct,
          100.0 * Sum(census.popn_black) / Sum(census.popn_total) AS black_pct
FROM      nyc_neighborhoods AS neighborhoods
JOIN      nyc_census_blocks AS census
ON        ST_Intersects(neighborhoods.geom, census.geom)
WHERE     neighborhoods.boroname = 'Manhattan'
GROUP BY  neighborhoods.name
ORDER BY  white_pct DESC;
```

1. The JOIN clause creates a virtual table that includes columns from both the neighborhoods and census tables.
2. The WHERE clause filters our virtual table to just rows in Manhattan.
3. The ON clause helps to remain all the geometry areas that interact between two tables
4. Sum up the population and format (e.g., GROUP BY, ORDER BY) on the final numbers

```
sdbms=# SELECT  neighborhoods.name AS neighborhood_name,
sdbms-#         Sum(census.popn_total) AS population,
sdbms-#         100.0 * Sum(census.popn_white)/Sum(census.popn_total) AS white_pct,
sdbms-#         100.0 * Sum(census.popn_black) / Sum(census.popn_total) AS black_pct
sdbms-# FROM    nyc_neighborhoods AS neighborhoods
sdbms-# JOIN    nyc_census_blocks AS census
sdbms-# ON      ST_Intersects(neighborhoods.geom, census.geom)
sdbms-# WHERE   neighborhoods.boroname = 'Manhattan'
sdbms-# GROUP BY neighborhoods.name
sdbms-# ORDER BY white_pct DESC;
  neighborhood_name   | population |     white_pct      |     black_pct
----------------------+------------+--------------------+--------------------
 Carnegie Hill        |      18763 |  90.08154346319886 | 1.4123541011565315
 West Village         |      26718 |  87.59637697432443 | 2.1745639643685903
 North Sutton Area    |      22460 |  87.56455921638468 |  1.553873552983081
 Upper East Side      |     203741 |  85.02216048807064 | 2.7220834294520984
 Soho                 |      15436 |  84.64628142005701 |  2.235034983156258
 Greenwich Village    |      57224 |  81.97609394659584 |   2.43778834055641
 Central Park         |      46600 |  79.49356223175965 |  7.967811158798283
 Tribeca              |      20908 |   79.1180409412665 |  3.548880811172757
 Gramercy             |     104876 |   75.4586368663946 |  4.720813150768526
 Murray Hill          |      29655 |  75.01264542235711 | 2.5122239082785365
 Chelsea              |      61340 |  74.81578089338116 |  6.372676882947506
 Upper West Side      |     214761 |    74.561489283436 |  9.188353565125885
 Midtown              |      76840 |  72.60150963040083 |  5.166579906298803
 Battery Park         |      17153 |   71.8300005829884 |  3.375502827493733
 Financial District   |      34807 |  69.92846266555578 |  3.829689430287011
 Clinton              |      32201 |   65.2743703611689 |  7.943852675382752
 East Village         |      82266 |  63.26550458269516 |  8.771545960664186
 Garment District     |      10539 |  55.18550147072777 |   7.08795900939368
 Morningside Heights  |      42844 | 52.719167211278126 | 19.370273550555503
 Little Italy         |      12568 | 49.029280712921704 | 1.8220878421387652
 Yorkville            |      58450 |   35.5551753635586 |  29.72284003421728
 Inwood               |      50047 |  35.21489799588387 | 16.846164605271046
 Washington Heights   |     169013 | 34.873057102116405 | 16.827107973942834
 Lower East Side      |      96156 |  33.50908939639752 |  9.075876700361912
 East Harlem          |      60576 |  26.43621236133122 |  40.38398045430534
 Hamilton Heights     |      67432 | 23.889251393996915 | 35.772333610155414
 Chinatown            |      16209 |  15.16441483126658 |  3.757171941513974
 Harlem               |     134955 | 15.094661183357415 |  67.06161313030269
(28 rows)
```

2. "What subway station is in 'Little Italy'? What subway route is it on?"

```
SELECT    s.name, s.routes
FROM      nyc_subway_stations AS s
JOIN      nyc_neighborhoods AS n
ON        ST_Contains(n.geom, s.geom)
WHERE     n.name = 'Little Italy';
```

Find the stations contained by the neiborhood called 'Little Italy'

```
sdbms=# SELECT  s.name, s.routes
sdbms-# FROM    nyc_subway_stations AS s
sdbms-# JOIN    nyc_neighborhoods As n
sdbms-# ON      ST_Contains(n.geom, s.geom)
sdbms-# WHERE   n.name = 'Little Italy';
   name     | routes
------------+--------
 Spring St | 6
(1 row)
```

3. “What are all the neighborhoods served by the 6-train?”

```
SELECT    DISTINCT n.name, n.boroname
FROM      nyc_subway_stations AS s
JOIN      nyc_neighborhoods AS n
ON     ST_Contains(n.geom, s.geom)
WHERE     strpos(s.routes,'6') > 0;
```

the DISTINCT keyword is to remove duplicate values from result set where there were more than one subway station in a neighborhood.

strpos(str, substr): find the position from where the substring is being matched within the string

```
sdbms=# SELECT  DISTINCT n.name, n.boroname
sdbms-# FROM    nyc_subway_stations AS s
sdbms-# JOIN    nyc_neighborhoods AS n
sdbms-# ON      ST_Contains(n.geom,s.geom)
sdbms-# WHERE   strpos(s.routes,'6')>0;
        name         | boroname
---------------------+-----------
 Chinatown           | Manhattan
 East Harlem         | Manhattan
 Financial District  | Manhattan
 Gramercy            | Manhattan
 Greenwich Village   | Manhattan
 Hunts Point         | The Bronx
 Little Italy        | Manhattan
 Midtown             | Manhattan
 Mott Haven          | The Bronx
 Murray Hill         | Manhattan
 Parkchester         | The Bronx
 Soundview           | The Bronx
 South Bronx         | The Bronx
 Upper East Side     | Manhattan
 Yorkville           | Manhattan
(15 rows)
```

4. “After 9/11, the ‘Battery Park’ neighborhood was off limits for several days. How many people had to be evacuated?”

```
SELECT    Sum(popn_total)
FROM      nyc_neighborhoods AS n
JOIN      nyc_census_blocks AS c
ON        ST_Intersects(n.geom, c.geom)
WHERE     n.name = 'Battery Park';
```

```
sdbms=# SELECT  Sum(popn_total)
sdbms-# FROM    nyc_neighborhoods AS n
sdbms-# JOIN    nyc_census_blocks AS c
sdbms-# ON      ST_Intersects(n.geom, c.geom)
sdbms-# WHERE   n.name = 'Battery Park';
  sum
-------
 17153
(1 row)
```

5. “What are the population density (people / km^2) of the ‘Upper West Side’ and ‘Upper East Side’?”

```
SELECT    n.name,
          Sum(c.popn_total) / (ST_Area(n.geom) / 1000000.0) AS popn_per_sqkm
FROM      nyc_census_blocks AS c
JOIN      nyc_neighborhoods As n
ON        ST_Intersects(c.geom,n.geom)
WHERE     n.name = 'Upper West Side'
OR        n.name = 'Upper East Side'
GROUP BY n.name, n.geom;
```

```
sdbms=# SELECT   n.name,
sdbms-#          Sum(c.popn_total) / (ST_Area(n.geom) / 1000000.0) AS popn_per_sqkm
sdbms-# FROM    nyc_census_blocks AS c
sdbms-# JOIN    nyc_neighborhoods As n
sdbms-# ON      ST_Intersects(c.geom,n.geom)
sdbms-# WHERE   n.name = 'Upper West Side'
sdbms-# OR      n.name = 'Upper East Side'
sdbms-# GROUP BY n.name,n.geom;
      name       |   popn_per_sqkm
-----------------+-------------------
 Upper East Side | 48524.48774898572
 Upper West Side | 40152.48960800237
(2 rows)
```