Spatial Databases

Spatial Data Analysis and Simulation modelling,
2020, Dr. Zhiyong Wang

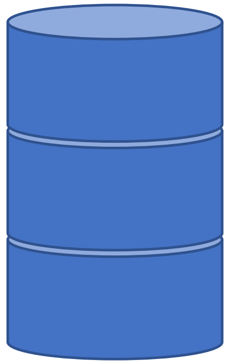
Learning goals

- The basic concepts of spatial databases
- Learn to create your own spatial database
- Learn to import geo-data into the spatial database
- Learn to perform simple and complex spatial queries.

Outline

- Spatial Databases
- PostGIS
    - Install and set up PostGIS
    - Spatial data types and geometric types
    - Spatial reference ID
    - Spatial functions
- Practical examples
    - Non-spatial query
    - Spatial query

Spatial Databases



What is a spatial database?

A spatial database is a database that is optimized for storing and querying data that represents objects defined in a geometric space.

extensions to store and manage spacial data

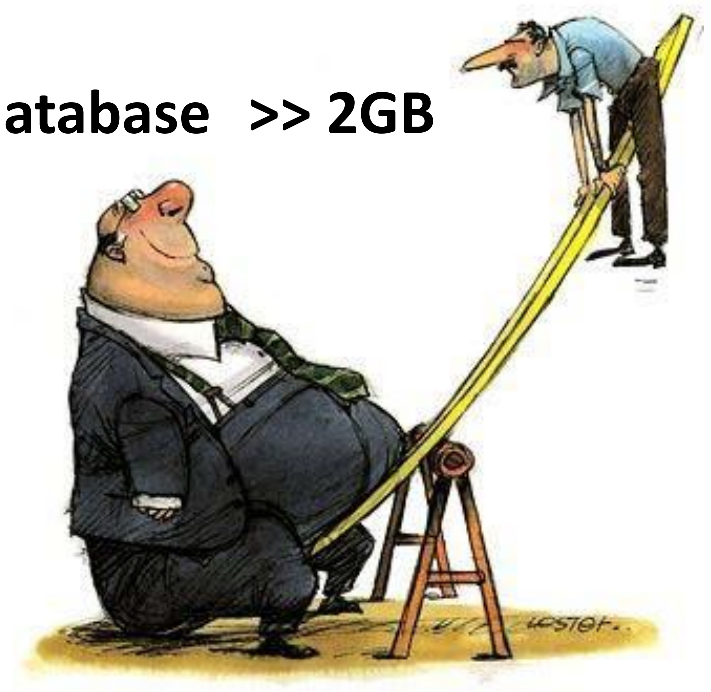
From https://en.wikipedia.org/wiki/Spatial_database

Why spatial databases?

Spatial databases VS Shapefiles

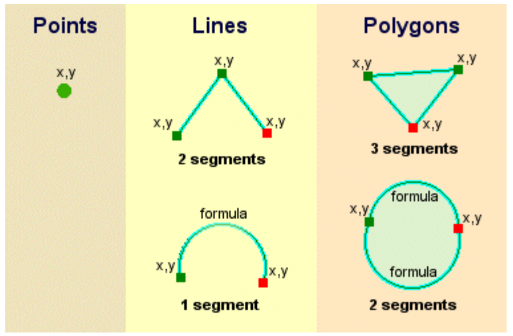
Shapefile < 2GB limitations because only less than 2GB data

Spatial database >> 2GB



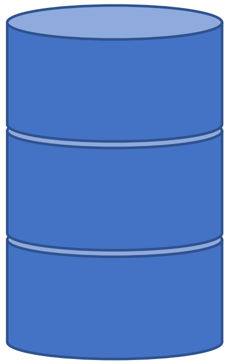
Why spatial databases?

Vector data



different geometrics

Spatial databases



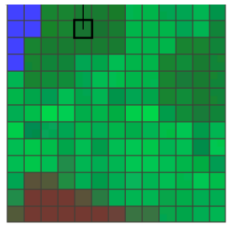
complex spatial analysis

Spatial analysis

- Spatial join
- Spatial aggregation
- Spatial computation

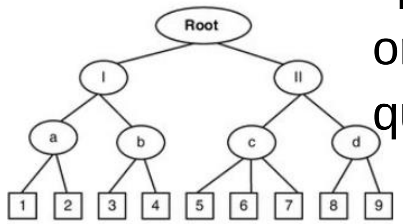
  …….

Raster data



dif. resolutions

Spatial indexing



spatial indexing to organize for faster queries

Examples of spatial databases

PostgreSQL / PostGIS (open source)

Oracle / Oracle Spatial and Graph

DB2 spatial extender

SpatiaLite (dateibasiert)

<u>NoSQL:</u>

PostGIS

Postgresql

Postgresql is a free and open-source relational database management system (RDBMS).

-- over 30 years of active development

-- provides a variety of datatypes

-- available for many operating systems

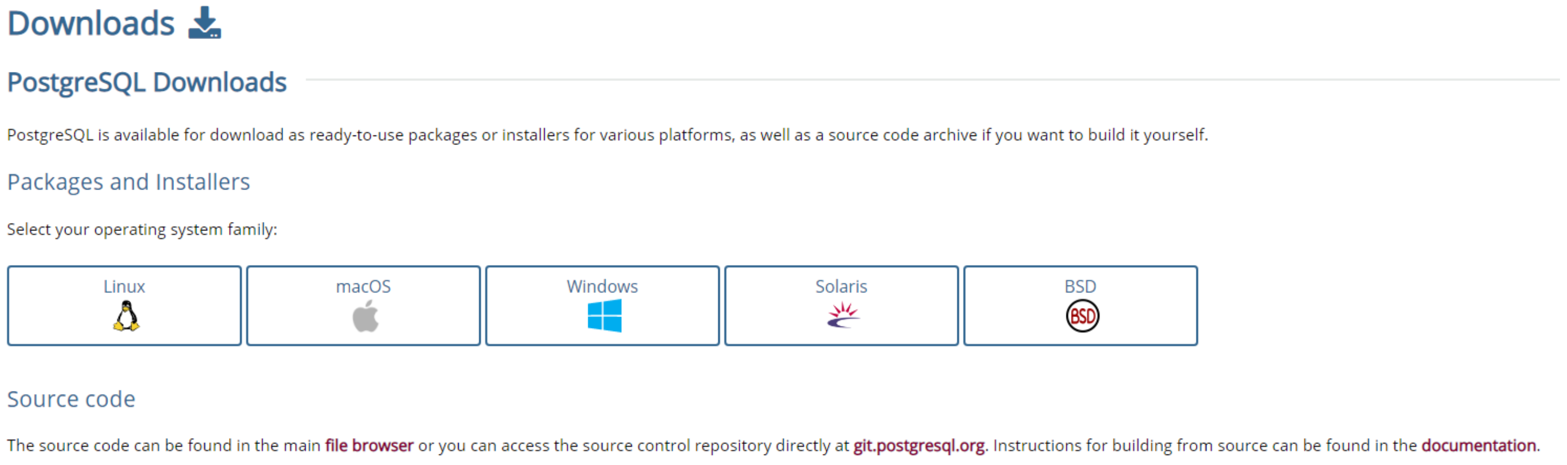
PostGIS

PostGIS adds spatial capabilities to the PostgreSQL relational database. It is an extension of PostgreSQL, and enables it to store, query, and manipulate spatial data.

build on top of PostgreSQL - extension for spatial functions and datatypes

Install PostgreSQL /PostGIS

- Install PostgreSQL



Downloads

PostgreSQL Downloads

PostgreSQL is available for download as ready-to-use packages or installers for various platforms, as well as a source code archive if you want to build it yourself.

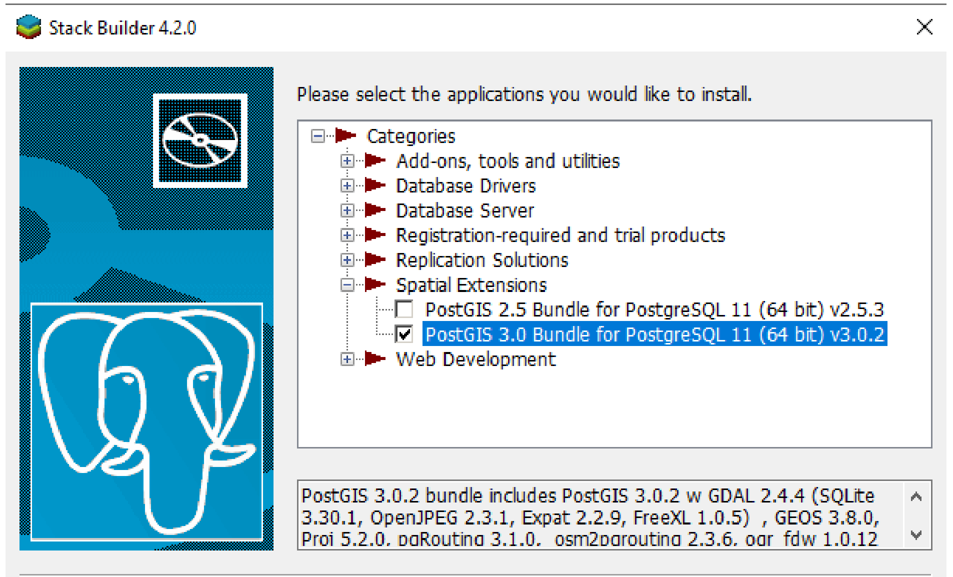
Packages and Installers

Select your operating system family:

Source code

The source code can be found in the main file browser or you can access the source control repository directly at git.postgresql.org. Instructions for building from source can be found in the documentation.

- Install PostGIS



Stack Builder 4.2.0

Please select the applications you would like to install.

```
Categories
  Add-ons, tools and utilities
  Database Drivers
  Database Server
  Registration-required and trial products
  Replication Solutions
  Spatial Extensions
    [ ] PostGIS 2.5 Bundle for PostgreSQL 11 (64 bit) v2.5.3
    [✓] PostGIS 3.0 Bundle for PostgreSQL 11 (64 bit) v3.0.2
  Web Development
```

PostGIS 3.0.2 bundle includes PostGIS 3.0.2 w GDAL 2.4.4 (SQLite 3.30.1, OpenJPEG 2.3.1, Expat 2.2.9, FreeXL 1.0.5) , GEOS 3.8.0, Proj 5.2.0, pgRouting 3.1.0, osm2pgrouting 2.3.6, ogr_fdw 1.0.12

Set up PostGIS

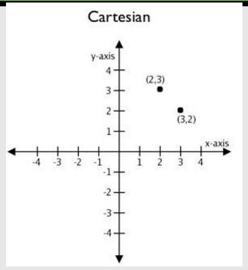
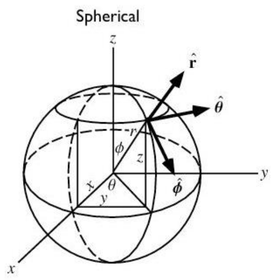
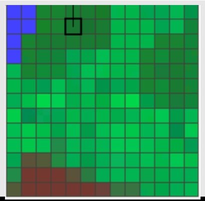
- Activate the PostGIS extension in PostgreSQL

```
CREATE DATABASE ads_db;
CREATE EXTENSION postgis;
```

activation of PostGIS

This will install the corresponding PostGIS functions in the public schema.
In addition, a table "spatial_ref_sys" and four views "geometry_columns", "geography_columns", "raster_columns" and "raster_overviews" are created.

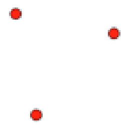
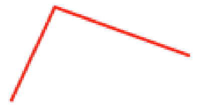
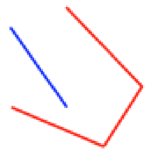
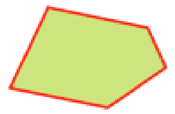
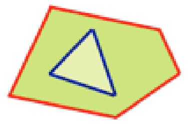
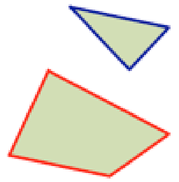
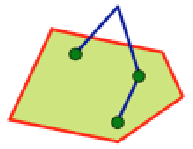
PostGIS: Spatial datatypes

Spatial Data type	Description	
Geometry	Represent a feature in planar (Euclidean) coordinate systems.	
Geography	Represent a feature in geodetic coordinate systems. Geodetic coordinates are spherical coordinates expressed in angular units (degrees).	
Raster	Represent raster data (imported from TIFFs, PNGs)	

PostGIS – Geometric types

supports different geometrics like:

one can express them in text representation which can be used as input and output

Geometry Type	*WKT representation*
Point	`POINT(3 7)`
Multipoint 	`MULTIPOINT(3 7, 4 2, 8 6)`
LineString 	`LINESTRING(1 2, 3 6, 9 4)`
MultiLineString 	`MULTILINESTRING((1 8, 4 4), (4 9, 8 5, 6 2, 1 4))`
Polygon 	`POLYGON((1 2, 6 1, 9 3, 8 5, 3 6, 1 2))`
Polygon (*with hole*) 	`POLYGON((1 2, 6 1, 9 3, 8 5, 3 6, 1 2), (3 3, 5 5, 6 2, 3 3))`
MultiPolygon 	`MULTIPOLYGON(((1 2, 6 1, 9 3, 3 6, 1 2)), ((4 9, 7 6, 9 8, 4 9)))`
GeometryCollection 	`GEOMETRYCOLLECTION( POINT(4 5), POINT(7 4), POINT(6 2), LINESTRING(4 5, 6 7, 7 4, 6 2), POLYGON((1 2, 6 1, 9 3, 8 5, 3 6, 1 2)) )`

PostGIS: Spatial Reference ID

- A Spatial Reference Identifier (SRID) is an identifier associated with a specific coordinate system.
- All existing projections are stored in the **spatial_ref_sys** table.

SRID:28992 --- Amersfoort / RD New

SRID:4326 --- WGS 84 (World Geodetic System)

- Search for suitable SRID or EPSG codes:
  - http://www.epsg-registry.org
  - http://spatialreference.org/ref/epsg/

PostGIS: spatial functions

Creation, access, editing and output of geometries

- Creation of points, lines and polygons:
  - `ST_MakePoint`, `ST_MakeLine`, `ST_MakePolygon`
- Access to geometries:
  - `ST_StartPoint`, `ST_EndPoint`, `ST_X`, `ST_Y`

all start with ST=spatial type

- Access to properties:
  - `ST_IsValid`, `ST_IsClosed`, `ST_Npoints`, `ST_IsSimple`
- Edit geometries:
  - `ST_AddPoint`, `ST_Multi`, `ST_Translate`
- Ouput geometries:
  - `ST_AsText`, `ST_AsKML`, `ST_AsGML`

PostGIS : create spatial tables

creating table hospitals with the attributes ID and geometry

Name of the spatial table

```
CREATE TABLE hospitals (
  id integer,
  geom geometry(POINT, 4326));
```

Name of the geometry column

Data type (alternatively geography)

WKT (well-known text) representation of geometries

Spatial Reference ID / Coordinate system (**EPSG-Code**)

PostGIS : Create and import geometries

- Insert geodata (vector) into tables, using **INSERT INTO .....**

name table   columns

keywords

```
INSERT INTO hospitals (id, geom)
VALUES (1,
  ST_GeomFromText('POINT(…)', 4326), -- WKT representation of geometries + SRID
);
-- other formats possible, e.g. ST_GeomFromGML
```

  - via GUI (e.g., pgAdmin (www.pgadmin.org))

  - via command line (ogr2ogr, shp2pgsql…)

  e.g., ogr2ogr -f “PostgreSQL” PG:”host=<hostname> dbname=<dbname> user=<yourusername> password=<yourpassword>” <dir>\yourdatafile.shp

  - and many more

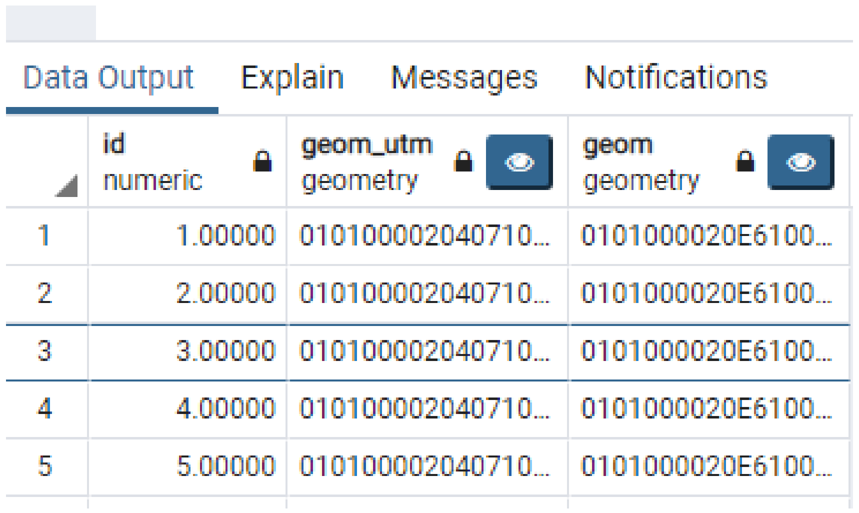
  http://postgis.net/workshops/postgis-intro/loading_data.html
  https://techcommunity.microsoft.com/t5/azure-database-for-postgresql/importing-spatial-data-to-postgis/ba-p/1255421

PostGIS: Transformation

- Coordinate transformation of a geometry column:

```
ALTER TABLE hospitals
 ADD COLUMN geom_utm geometry(POINT, 28992);

UPDATE hospitals SET geom_utm = ST_Transform(geom, 28992);
```



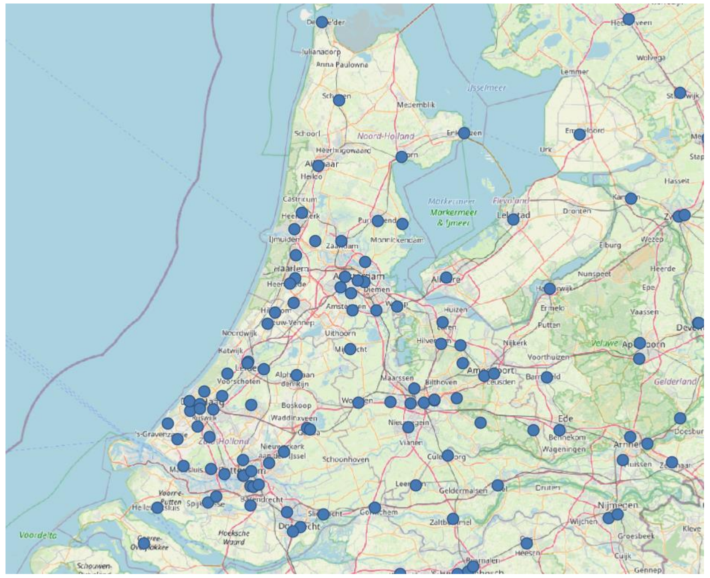
Data Output | Explain | Messages | Notifications

	id numeric	geom_utm geometry	geom geometry
1	1.00000	010100002040710...	0101000020E6100...
2	2.00000	010100002040710...	0101000020E6100...
3	3.00000	010100002040710...	0101000020E6100...
4	4.00000	010100002040710...	0101000020E6100...
5	5.00000	010100002040710...	0101000020E6100...

adding a new geometry column based on the dutch coordinate system

two geometry columns

Use case



Hospitals in the Netherlands

Practical examples: non-spatial query

- **SELECT** count (*) **FROM** hospitals

nonspatial query
count the number of hospitals

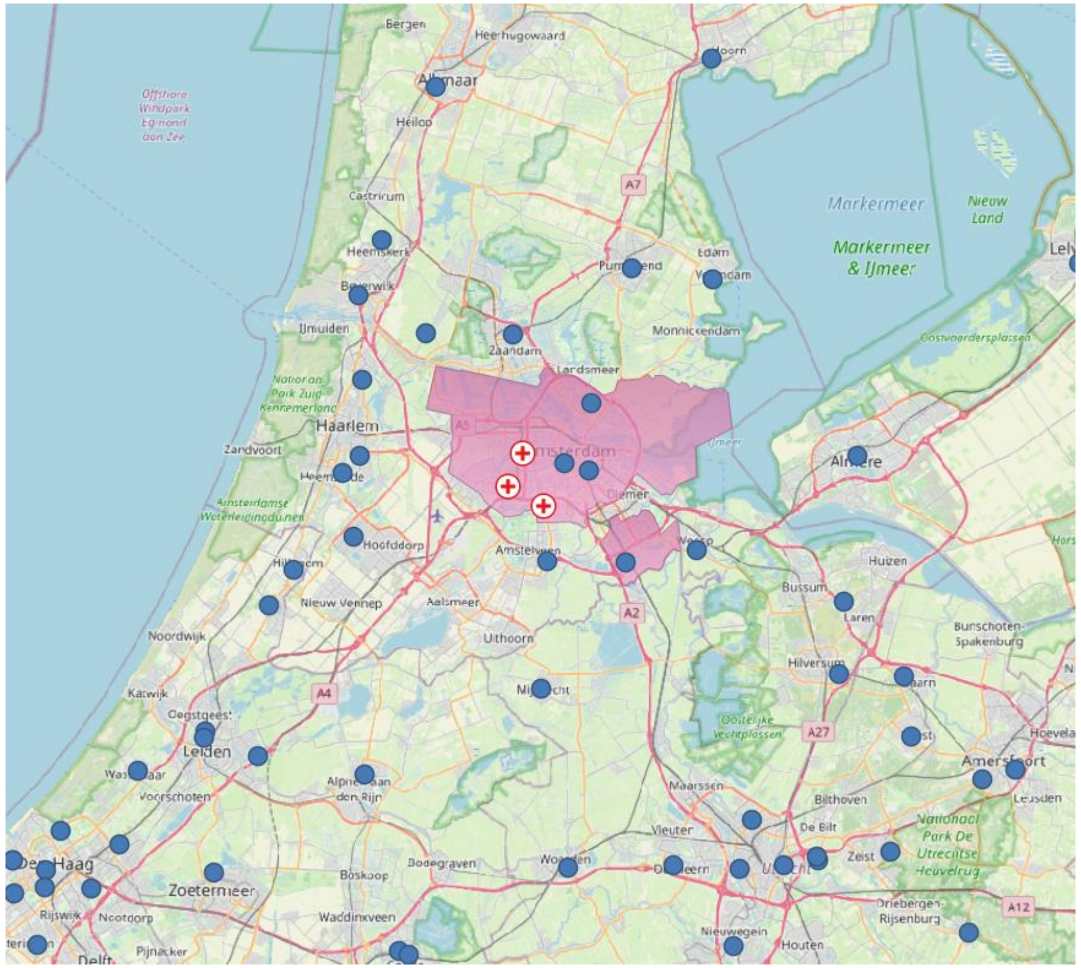
Result:
186

Practical examples: spatial query

- **SELECT count(*) from** hospitals hp **JOIN** municipality_ams ma **on st_contains(**ga.geom, hp.geom**)**

spatial query: joining two tables (municipality and hospitals)

Result:
7



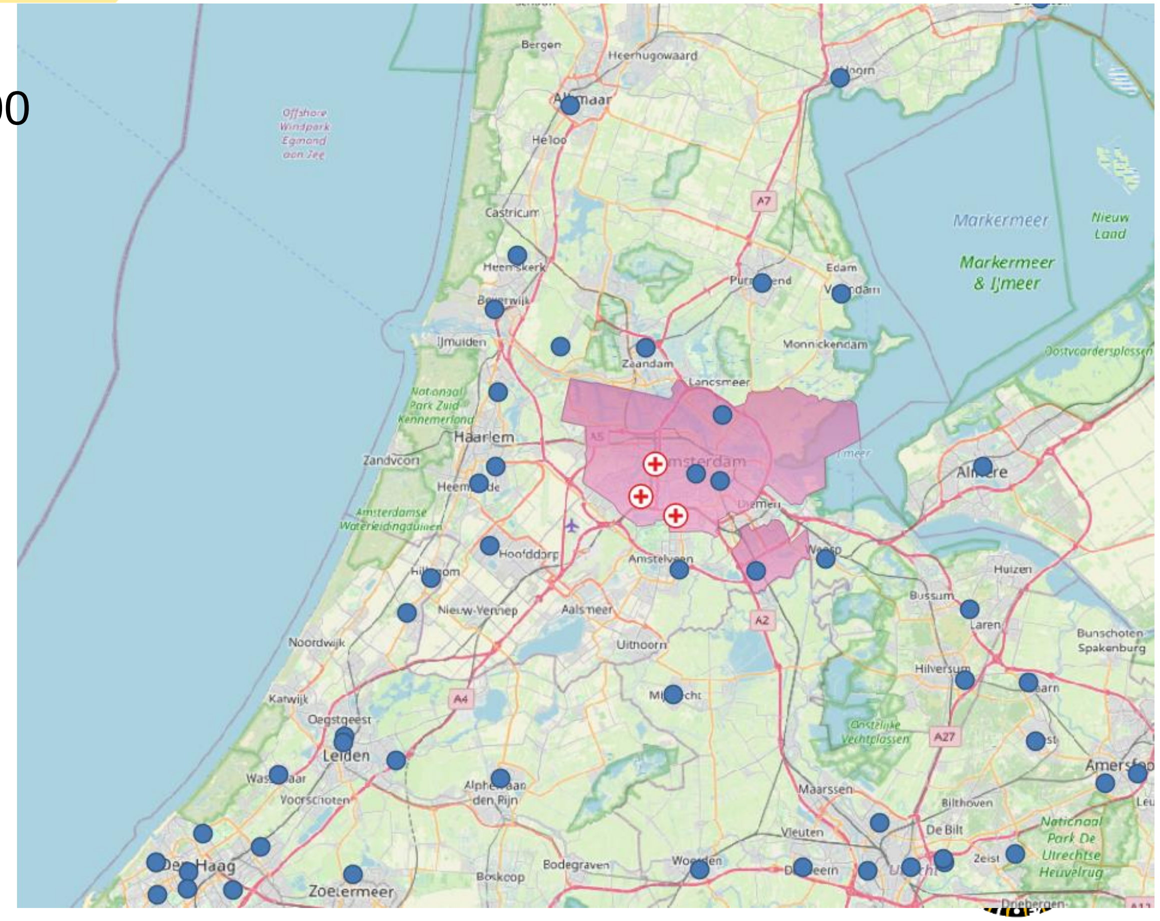
https://postgis.net/docs/ST_Contains.html

Practical examples: spatial query

- **SELECT count(*) from** hospitals hp **JOIN** gemeente_ams ga **on st_contains(**ga.geom, hp.geom**) where** hp.id > 100

constrain the query so only hospitals with IDs larger than 100 are included

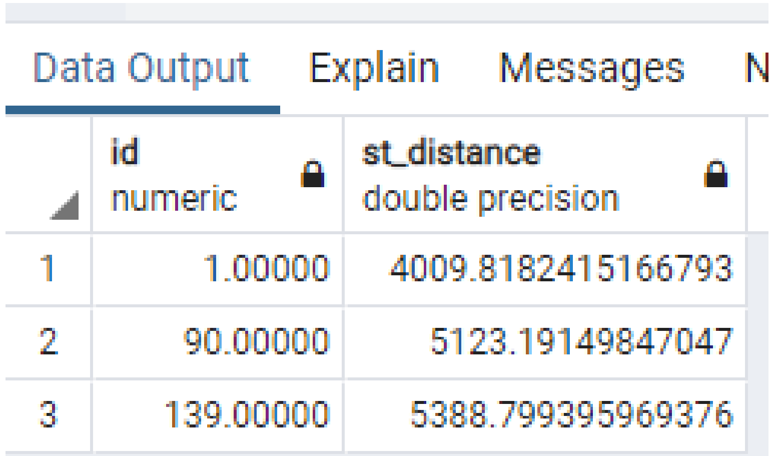
Result:
3



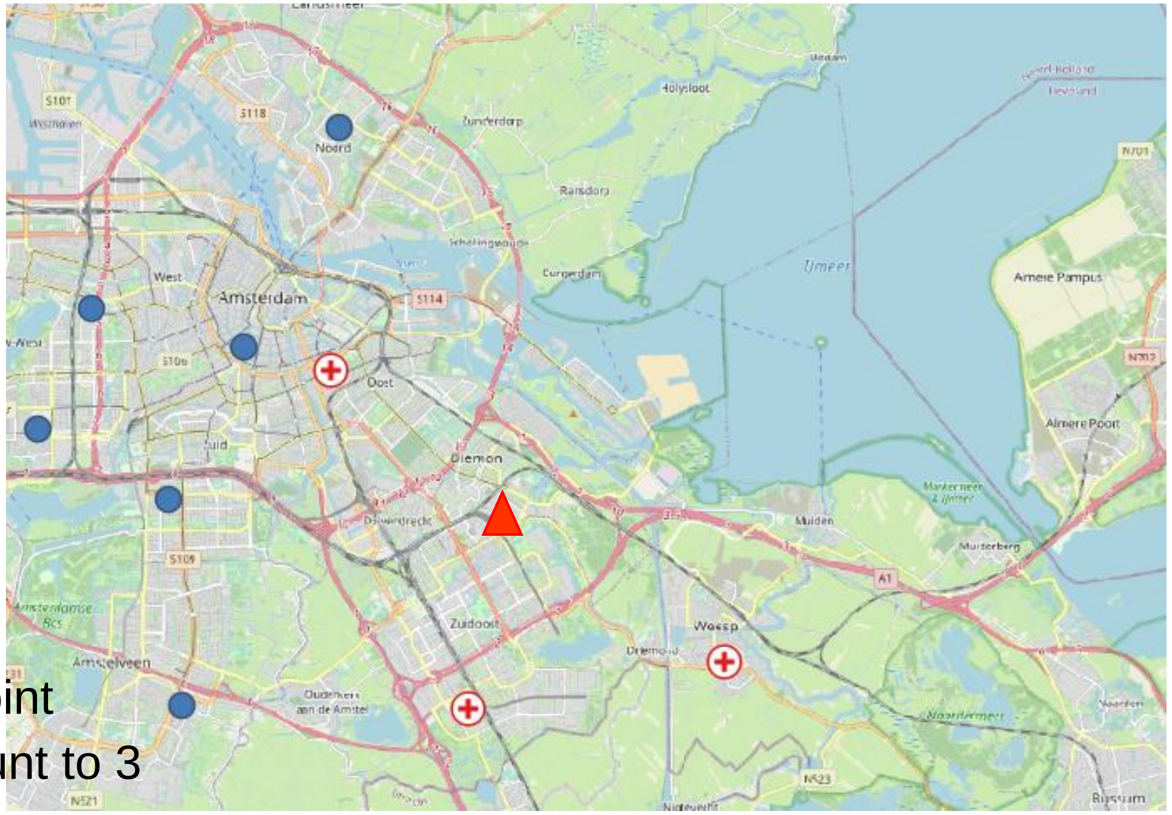
returns ID and distances of first 3 hospitals that are closest to point

Practical examples: spatial query

- **SELECT** id, st_distance(ST_SetSRID(ST_MakePoint(126706.3, 482433.8),28992), geom) **from** hospitals **ORDER BY** st_distance(ST_SetSRID(ST_MakePoint(126706.3, 482433.8),28992), geom) **ASC limit** 3;



	id numeric	st_distance double precision
1	1.00000	4009.8182415166793
2	90.00000	5123.19149847047
3	139.00000	5388.799395969376



create point in database based on coordinates and set SR
then st_distance to calculate the distance from hospitals to point
then order the hospitals in ascending order and limit the amount to 3

Reference

- Introduction to PostGIS https://postgis.net/workshops/postgis-intro/
- PostGIS 2.4.5dev Manual: http://postgis.net/docs/manual-2.4/
- Open Geospatial Consortium Inc. (2011) OpenGIS Implementation Stadnard for Geographic information - Simple feature access - Part 1: Common architecture. & Part 2: SQL option.
- Rigaux, P., Scholl, M. & Voisard, A. (2002) Spatial databases with application to GIS. San Francisco, CA.

Questions (Q&A session)

z.wang2@uu.nl