GIS – exercises

(Lecture 5 – Geospatial databases)

Data import

Download `croatia-latest.shp.zip` archive from the Exercises folder containing `shp` files, extract and upload to virtual machine as user ***postgres*** (e.g. using Filezilla FTP client)**.** You can use any folder you like –e.g. */tmp/geo* folder.

Connect to virtual machine (e.g. using Putty) as ***postgres*** anc change working folder to the one where the files are (*/tmp/geo*).

Using `shp2pgsql` (http://www.postgis.org/documentation/manual-1.3/ch04.html) tool convert all shape files to a series of SQL statements that create and insert data:

```
shp2pgsql buildings.shp > buildings.sql
shp2pgsql landuse.shp > landuse.sql
shp2pgsql natural.shp > natural.sql
shp2pgsql places.shp > places.sql
shp2pgsql points.shp > points.sql
shp2pgsql railways.shp > railways.sql
shp2pgsql roads.shp > roads.sql
shp2pgsql waterways.shp > waterways.sql
```

Execute those scripts using `psql` (http://www.postgresql.org/docs/9.2/static/app-psql.html) tool:

```
psql
\connect GISDB
\i buildings.sql
\i landuse.sql
\i natural.sql
\i places.sql
\i points.sql
\i railways.sql
\i roads.sql
\i waterways.sql
```

Beware: you must prefix psql statements with backslash „\“ (e.g. „`\connect`“, not „`connect`“).

Now you can delete the */tmp/geo* folder –it is no more needed.

The data is in the database and you can access it using pgAdmin tool or some alternative GUI/command line tool.

Working with geospatial data using a SQL interface

Example 0.

Find and delete invalid data:

```
SELECT 'buildings', gid FROM  buildings where st_isvalid(geom) != true
union all
SELECT 'landuse', gid FROM  landuse where st_isvalid(geom) != true
union all
SELECT 'natural', gid FROM  "natural" where st_isvalid(geom) != true
union all
SELECT 'places', gid FROM  places where st_isvalid(geom) != true
union all
SELECT 'points', gid FROM  points where st_isvalid(geom) != true
union all
SELECT 'railways', gid FROM  railways where st_isvalid(geom) != true
union all
SELECT 'roads', gid FROM  roads where st_isvalid(geom) != true
union all
SELECT 'waterways', gid from  waterways where st_isvalid(geom) != true
```

Example 1.

Print all data about roads having string „kralj“ in their title:

```
SELECT * FROM roads WHERE lower(name) LIKE '%kralj%'
```

Example 2.

Find the cumulative area of all schools (“škola” in Croatian):

```
SELECT SUM(st_area(geom))
  FROM buildings
 WHERE lower(name) like '%škola%'
```

The result is surprising. Find the SRID value for that attribute:

```
SELECT Find_SRID('public', 'buildings', 'geom');
```

It is not set up, zero means unknown. If you open .PRJ file for any of the .shp files that you downloaded from the Internet with a text editor, you will find out that these data in GCS_WGS_1984 format, which corresponds to 4326 SRID code in PostGIS. Let us assign all geospatial data with correct SRID:

```
SELECT UpdateGeometrySRID('buildings', 'geom', 4326);
SELECT UpdateGeometrySRID('landuse', 'geom', 4326);
SELECT UpdateGeometrySRID('natural', 'geom', 4326);
SELECT UpdateGeometrySRID('places', 'geom', 4326);
SELECT UpdateGeometrySRID('points', 'geom', 4326);
SELECT UpdateGeometrySRID('railways', 'geom', 4326);
SELECT UpdateGeometrySRID('roads', 'geom', 4326);
SELECT UpdateGeometrySRID('waterways', 'geom', 4326);
```

If you repeat the request, you still get "weird" results. However, now that projection is known, we can use the function to transform from one projection and system to another. Execute:

```
SELECT srid, srtext, proj4text FROM spatial_ref_sys WHERE srtext ILIKE '%croatia%'
```

Finally, for SRID 3765 the result is in m^2:

```
SELECT SUM(st_area(ST_Transform(geom, 3765)))
  FROM buildings
  WHERE lower(name) like '% škola %'
```

Example 3.

Print all the information about the landuses that are touching cemeteries::

```
SELECT *
  FROM landuse land1, landuse land2
 WHERE land1.name IS NOT NULL
   AND land1.type = 'cemetery'
   AND ST_Touches(land1.geom, land2.geom)
```

Example 4.

How many toilets are there at Bundek park?

```
SELECT points.* --, "natural".*
  FROM points, "natural"
  WHERE st_within(points.geom, "natural".geom)
    AND "natural".name = 'Bundek'
    AND points.type = 'toilets'
```

Explore the difference between st_within, st_intersects and st_overlaps.

Example 5.

Print the number and total length of all roads in Bundek::

```
SELECT count(*),sum( st_length(ST_Transform(roads.geom, 3765)))
  FROM roads, "natural"
  WHERE st_within(roads.geom, "natural".geom)
    AND "natural".name = 'Bundek'
```

The above solution is not entirely correct. Why?
Compare it to the next solution (try to do it with and without the last condition):

```
SELECT count(*),sum( st_length(ST_Transform(st_intersection( roads.geom, "natural".geom),
3765)))
  FROM roads, "natural"
 WHERE "natural".name = 'Bundek'
   AND st_intersects(roads.geom, "natural".geom) - with and w/out this - what's the diff?
```

Example 6.

Print landuses that have at least 10 cafes, descending by number of cafes:

```
SELECT landuse.gid, landuse.name, count(*) as noOfPoints
  FROM points, landuse
  WHERE st_within(points.geom, landuse.geom)
    AND points.type = 'cafe'
    group by landuse.gid, landuse.name
    having count(*) > 10
    order by noOfPoints desc
```

Where is the land with more than 100 cafes? (we will visualize it later in QGIS)

Example 7.

Print all landuses and natural geometries that are completely covered (equal). (this query will take significant time to complete ☺)

```
SELECT * FROM landuse
 WHERE EXISTS (SELECT * FROM "natural"
             WHERE ST_Equals(landuse.geom, "natural".geom))
```

This query prints data only from the landuse table - rewrite it so that the data is printed from both tables.

Example 8.

Print a building with a minimum area.

```
SELECT *, ST_Area(ST_Transform(buildings.geom, 3765)) AS area
  FROM buildings
 ORDER BY area ASC
 LIMIT 1
```

Example 9.

Print geometry type for entry "Zgrada D" in the buildings table:

```
SELECT DISTINCT st_geometrytype (geom) FROM buildings where name = 'Zgrada D'
```

Let's find out the types of attributes of geospatial tables:

```
SELECT f_table_name, type
  FROM geometry_columns
 WHERE f_table_schema = 'public'
   AND f_table_name in ( 'buildings', 'landuse', 'natural', 'places',
                         'points', 'railways', 'roads', 'waterways')
   AND f_geometry_column = 'geom';
```

Example 10.

Print distance from Murter to Šibenik:

```
SELECT ST_Distance(ST_Transform(p1.geom, 3765), ST_Transform(p2.geom, 3765))
  FROM places p1, places p2
 WHERE p1.name = 'Murter'
AND p2.name = 'Šibenik'
```

Example 11.
Print details of all buildings that are less than 500 meters from Murter place tag:

```
SELECT buildings.*
  FROM places p1, buildings
 WHERE p1.name = 'Murter'
   AND ST_Distance(ST_Transform(p1.geom, 3765), ST_Transform(buildings.geom, 3765))  < 500
```

Working with geospatial data through a graphical interface i.e. Quantum GIS tool

Install Quantum GIS, you can find it at: http://www.qgis.org , it is recommended to use the latest version.
Start QGIS Desktop.
Connect to the database GeoDB on a virtual machine and make available the data from all the tables:
Layer -> Add Layer -> Add a PostGIS Layer.
Familiarize yourself with the basic features Quantum GIS tools, see (at least the first 15 minutes where selections are discussed): https://www.youtube.com/watch?v=ppI6bd-TYyk

The same set of data can be made available in Quantum GIS in a number of different ways:

1. In the `landuse` layer select all that of type "grass" and "meadow" and save it as a new shp file, add it immediately to the project (as explained in the video) as "green1"
2. Copy the landuse layer (duplicate), set the duplicate name "green2" and set the filter, the same as before. Save Layer definition file, open the file with a text editor and search for the word "meadow".
3. Set the selection (again the same), copy the marked rows to the clipboard, and then: Edit-> Paste options As-> Memory New Vector Layer, name it "Green3"
4. In the database GeoDB make the appropriate view, name it "Green4" and add it to the project

You should know the answer to questions about what happens in these four different approaches, where is the data, what about refreshing data and so on.

Get familiar with all the options that exist when you right-click on the layer.

- Set the display color to green for all "green*" layers
- In places layer assign the star sign and print labels (the name of the place).

In addition to the data from the database, Quantum GIS to work with data stored in files according to the agreed format - for example in the shape files.

To try out shape files, download information about administrative areas from http://www.diva-gis.org/datadown (Administrative areas) for Croatia (HRV_adm.zip), unpack and add shape to the project file (Layer -> Add Layer -> Add Vector Layer -> File).

See details for HRV_adm1 layer (Layer -> Open Attribute table).
Are the text data properly displayed? Code page for the shape files is wrong (differs from default).

Try to find the correct code page. You can change it by means of dialogue options for the selected layers: Layer -> Properties.