THE UNIVERSITY OF SYDNEY

School of Computer Science
Uwe Roehm

DATA2001/2901: Data Science, Big Data, and Data Diversity **Semester 1, 2019**

Tutorial Week 9: Geo-Data Processing and the Practical Assignment

In this week's tutorial, we will look at how to do integrate geo-location data into an existing data set in order to do a spatial join between two datasets. As part of this weeks tutorial we will also look at the assignment.

Assignment

The practical assignment of DATA2001/DATA2901 has been published in the week before the semester break. You can download the handout from Canvas under 'Modules - Practical Assignment'. There are also some data sets available which are the starting point for your project.

Exercise 1. Group Forming and Assignment Download

The practical assignment of DATA2001/2901 is group work for teams of normally two or three students. All team members should be in the same lab class. Please form teams by latest the start of the lab and let the tutor know about your allocation.

Exercise 2. Assignment Runthrough

You can download the assignment specification and the datasets from Canvas under 'Modules - Practical Assignment'. The tutors will briefly discuss with you the three main tasks which you have to work on in this assignment and the submission rules.

Exercise 3. Loading Geo-Spatial Data from Shapefiles Querying with PostGIS

In the first two exercises in the associated Jupyter notebook, we explore how to load geo-data from a file – specifically from a ESRI Shapefile – into Python, how to explore then that data, and hoe to store and query it in a database. We provided you here in Canvas with a `world.shp` and a `cities.shp` dataset which you have top load into Jupyter, then store in PostGIS.

The second part is then all about storing geo-data in PostgreSQL and querying that data there with spatial predicates, joins and index support.

IMPORTANT: Please note that PostGIS is installed on a different database server than we used so far: **soit-db-pro-1.ucc.usyd.edu.au**. The logins should be the same than you used so far. We provide corresponding directions in this week's Jupyter notebook.

Exercise 4. Collecting Geo-Data for Neighborhoods (Assignment preparation step)

In the practical assignment, you are asked to combine several datasets about *neighbourhoods*. While the statistical data from the ABS share the same neighbourhood ID, the example dataset about transport options - the bike-sharing pods - does not have

this ID, nor do the names match between bike-sharing pod locations and neighbourhoods. The bicycle pods do have a GPS location though...

Background: This is a simulated dataset. Assume bicycle pods are dedicated places where publicly shared bicycles and scooters can be found. Users can take a bicycle or scooter from one station, cycle around, and return them at the same or at another location - similar to the public bike-sharing facilities in Melbourne, London, or New York. Users pay for the time duration of the bicycle usage.

You have two options to associate bicycle pods with the other data on neighbourhoods:

1. You can collect some spatial data for the neighbourhoods too, such as their geographic boundaries. The Jupyter notebook for this week discusses two options on how to do this via the ABS website. You can access the corresponding notebook from our Canvas website.
2. Alternatively, you use a web API from a mapping service such as Open Street Map to translate the neighbourhood names to GPS locations too.
   Check back with last Week 7's tutorial on how to retrieve data from that Web API (careful! there is a 1-request-per-second limit on that service!).

In the first exercise, we shall collect geo-boundary-data for each neighbourhood – either as download from the webpage of the ABS (the Australian Bureau of Statistics) or as Open Street Map's Web API calls.

Exercise 5. Geo Data Storage

Create tables in PostgreSQL for the assignment data file which were provided in Canvas, in particular for the `Neighbourhoods.csv` file.

Then extend you table definitions with additional attribute(s) to store the boundary or geo-location information of the neighbourhoods too. Finally, store the data collected in the previous exercise in your Postgresql database.

Tips: PostgreSQL geometry datatypes documentation:
`https://www.postgresql.org/docs/9.5/static/datatype-geometric.html`
PostGIS documentation (postgis installed on soit-db-pro-1.ucc.usyd.edu.au):
`https://www.postgis.net/documentation`
We have pyshp installed on all Jupyter servers Parser for ESRI shapefiles (as can be downloaded from ABS): `https://pypi.org/project/pyshp/`

Exercise 6. Spatial Join using PostGIS

In the last exercise, we will be looking at how to perform a spatial join between the geo-locations of the bike-sharing pods (one of the provided datasets for the assignment) and the boundaries (or locations) of the statistical areas which you extracted in the previous exercises. We will use PostGIS for this (postgis is currently only installed on soit-db-pro-1.ucc.usyd.edu.au - logins tbd.). You need a local PostGIS installation.

Tips: PostGIS documentation:
`https://www.postgis.net/documentation`
PostgreSQL geometry functions documentation:
`https://www.postgresql.org/docs/9.5/static/functions-geometry.html`

PostGIS: Spatial database extension for Postgres supporting geographic objects (OGC):

- Geometry types for Points, LineStrings, Polygons, MultiPoints, etc. including import/export from standard formats such as GeoJSON or KML
- Support for multiple spatial reference systems and transformations between
- Spatial operators for determining geospatial measurements like area, distance, length and perimeter, and geospatial set operations, like union, difference etc.
- R-Tree indexing (via GiST)

References

Seppe van den Broucke and Bart Baesens: *"Practical Web Scraping for Data Science"*, Springer 2018. (available electronically via USYD library)