Practice Session N° 2: Databases and Data Preparation

This is the initial version of the document, shared in beginning of the lab session

In this tutorial 1st you will learn:

- How to setup the database server
- How to manage the database using admin-client
- How to manage the database using command line interface
- How to store/access the data using command line interface
- How to setup python database client
- How to store/access the data using python database client

In 2nd half of the lab we will deal with preprocessing of the mobility data

1st Part

PostgreSQL

The PostgreSQL is an advanced open source relational database management system (RDBMS) originating from Berkeley, it supports classics DB transactions following ACID properties as well as many features seen in commercial databases (triggers, views, indexes etc.). It supports most popular operations systems and can be used standalone as well as in cluster deployment. Multiple well known open source projects rely on PostgreSQL for managing the data, for us most notably: OpenStreetMap, PostGIS, CartoDB, GRASS GIS, QGIS, Mapnik. The current upstream of Postgre is 13.2 which we are going to demonstrate in this tutorial.

Installing on Linux

Most of the modern Linux distributions include software repositories simplifying the installation of specific packages. Postgre SQL is included along with it's dependencies in APT based (Debian, Ubuntu, Mintu); YUM based (Fedora, RedHat, CentOS); YaST based SUSE Linux. One notable disadvantage of repository based distributions - the repo may not contain the latest version of the software.
The up-to-date version of the software in this case is installed through a custom repository (if exists); or manually compiled. In the following sections we list the install routines for most popular Linux distributions.

**Installing in general**

The PostgreSQL documentation is well organized and provides detailed instructions along with command line (CLI) examples. The corresponding downloads section contains pre-compiled packages as well as the source code archives.

https://www.postgresql.org/download/ (https://www.postgresql.org/download/)

**Task 001:** install the PostgreSQL related packages specific to your platform

**PgAdmin graphical administration utility**

The PostgreSQL comes with nice administration GUI (desktop or web-based application). The PgAdmin includes an educational sandbox setup, which allows to get the basic knowledge without damaging the local PostgreSQL server setup.

https://www.pgadmin.org/try/ (https://www.pgadmin.org/try/)

**Optional Task:** Follow the online demo, make yourself familiar to PgAdmin

**Question:** How many tables are there in the pagila schema?

**Managing PostgreSQL server using command line**

**Starting/Stopping/Reloading the server daemon process**

**Creating the user role and associating to a schema**
Accessing the created schema using created role and password

Managing the tables in the schema

Exercises:

Refer to official documentation:

https://www.postgresql.org/docs/13/tutorial-sql.html

Task 002:

Using either CLI or PgAdmin create the required tables and relations in your local PostgreSQL

See the entity-relationship diagram below

```
In [7]: from matplotlib import pylab as plt
import matplotlib.image as mpimg
image = mpimg.imread("image.png")
plt.imshow(image)
```

Out[7]: <matplotlib.image.AxesImage at 0x7f240c448d30>

Task 003:

Fill in the created tables with the data
Accessing the database using Python

Basic access using `psycopg2` module

[https://www.psycopg.org/docs/install.html#quick-install](https://www.psycopg.org/docs/install.html#quick-install)

**Task 004:**

Install the `psycopg2` module

Fetch the data from your previously created schema, following the previously shared link

Reading the SQL query into Pandas dataframe

**Task 005:**

Fetch the data from your previously created schema into a Pandas dataframe

Use the following code, modify it if needed

```python
In [ ]:
import pandas as pd
import psycopg2

# host =
# port =
# dbname =
# username =
# pwd =

with psycopg2.connect("host='{}' port={} dbname='{}' user={} password={}".format(host, port, dbname, user, password)) as conn:
    sql = "select count(*) from table;"
    dat = pd.read_sql_query(sql, conn)
```
Second Part

Preprocessing of the mobility data

We will refer to the and exiting tutorial of scikit-mobility You can copy paste commands directly to your Jupyter notebooks

https://github.com/scikit-mobility/scikit-mobility#preprocessing

Since we start with the *preprocessing* section, make sure the corresponding data is loaded first

In [ ]: