Project Course - Part One: Setup

Done by: Axel Facundo Preiti Tasat

01.01. Chosen city

For this project, I chose the city of **Milano**, Italy.

01.02. OSM file and `sed`

The coordinate box for the city of Milano are the following:

- Longitude: [9.0300, 9.3281]
- Latitude: [45.3367, 45.5527]

We download the OSM file for the city with the following command

```
wget -O 'milano.osm' \
  'http://www.overpass-api.de/api/xapi?*[bbox=9.0300,45.3367,9.3281,45.5527][@meta]'
```

The MobilityDB database will run in a Docker container, so we create a new container using the MobilityDB image

```
docker run -d --name mda-project -p 5433:5432 -e POSTGRES_DB=mobilitydb \
  -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres mobilitydb/mobilitydb
```

We copy the OSM file to the container, in the */tmp* directory

```
docker cp milano.osm mda-project:/tmp/
```

Now, we connect to the docker container and run the `sed` Linux command

```
docker exec -it mda-project bash
sed -r -i.org \
  "s/version=\"[0-9]+\" timestamp=\"[^\"]+\" changeset=\"[0-9]+\" uid=\"[0-9]+\" user=\"[^\"]+\"//g" \
   /tmp/milano.osm
```

Now we check the size of both original and filtered files

```
root@bbe187d26026:/tmp# ls -la
total 1826060
drwxrwxrwt 1 root root      4096 Apr  6 10:53 .
drwxr-xr-x 1 root root      4096 Apr  6 10:34 ..
-rwxr-xr-x 1 root root 302M Apr 13 17:15 milano.osm # Edited
-rwxr-xr-x 1 root root 528M Apr  6 21:10 milano.osm.org # Original
```

In summary:

- **Size of the original OSM file**: 528 Mb
- **Size of the compressed file**: 302 Mb
- `sed` **command to compress the original file**:

```
sed -r -i.org \
   "s/version=\"[0-9]+\" timestamp=\"[^\"]+\" changeset=\"[0-9]+\" uid=\"[0-9]+\" user=\"[^\"]+\"//g" \
    /tmp/milano.osm
```

01.03. Map config files

Given the *mapconfig_brussels.xml* file, we create two variants for this city:

- Eco-friendly approach
- Car-priority approach

Both files can be found as *mapconfig_milano_ecofriendly.xml* and *mapconfig_milano_carpriority.xml*, respectively, in the *mapconfigs* directory.

As we will need them for the next step, we copy those files to the container.

```
docker cp mapconfigs/mapconfig_milano_carpriority.xml mda-project:/tmp
docker cp mapconfigs/mapconfig_milano_ecofriendly.xml mda-project:/tmp
```

01.04. `osm2pgrouting` tool

We install the neccessary dependencies in the container.

```
apt update && apt upgrade
apt install postgresql-17-pgrouting
apt install osm2pgrouting
```

Next, we create the database and install the neccessary extensions. In order to do that, we connect to postgres using the `psql` tool.

```
psql -U postgres -d mobilitydb
```

```
CREATE DATABASE milano_traj;
\c milano_traj
CREATE EXTENSION IF NOT EXISTS MobilityDB CASCADE; -- Installs postgis automatically
CREATE EXTENSION IF NOT EXISTS PGROUTING CASCADE;
CREATE EXTENSION IF NOT EXISTS HSTORE CASCADE;
```

01.04.01. Run command

We run the `osm2pgrouting` tool accordingly.

 Replace `[VARIANT]` with one of the available config map files.

```
osm2pgrouting -h localhost -p 5432 -U postgres -W postgres \
   --f /tmp/milano.osm --dbname milano_traj -c /tmp/mapconfig_milano_[VARIANT].xml
```

01.04.02. Map config file in DB

The table that contains the map config file information is the **configuration** table.

This table has 16 rows

```
SELECT count(*)
FROM configuration;
-- 16
```

01.04.03. Created tables

If we want to know how many tuples were created for the **ways** and **ways_vertices_pgr** tables, we do the following two simple queries.

```
SELECT count(*)
FROM ways;
-- 151951

SELECT count(*)
FROM ways_vertices_pgr;
-- 127593
```

01.05. `osm2pgsql` tool

01.05.01. Download the tool

We install the tool inside the container.

```
apt install osm2pgsql
```

01.05.02. Run the tool

We run the following command in the container.

```
osm2pgsql -c -H localhost -P 5432 -U postgres -W -d milano_traj \
   --proj=3857 /tmp/milano.osm
```

01.05.03. *planet** tables

We the previous run command, four new tables were created. All of the have the same prefix, *planet*.

If we want to know how many tuples were created for each table, we run the following queries

```
SELECT count(*)
FROM planet_osm_line;
-- 197279

SELECT count(*)
FROM planet_osm_point;
-- 264876

SELECT count(*)
FROM planet_osm_polygon;
-- 167255

SELECT count(*)
FROM planet_osm_roads;
-- 19112
```

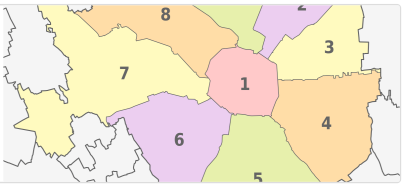
01.06. Prepared data SQL file

Now we must create the ***muncipalities*** table. For this, we use wikipedia as our source of information

Municipi di Milano

I municipi di Milano sono le nove circoscrizioni in cui è diviso il territorio comunale di Milano.

https://it.wikipedia.org/wiki/Municipi_di_Milano#I_nove_municipi



We copy the *brussels_preparedata.sql* file and modify the *Municipality* table with Milano data.

> 👁 The already edited file can be found in *scripts/milano_preparedata.sql*.

Replace the `insert into municipalities` with the following.

```
INSERT INTO Municipalities (MunicipalityId, MunicipalityName, Population, PopDensityKm2, NoEnterp) VALUES
  (1, 'Municipio 1', 99317, 10271, 6460),
  (2, 'Municipio 2', 163731, 13015, 2266),
  (3, 'Municipio 3', 145345, 10214, 1266),
  (4, 'Municipio 4', 165393, 7883, 14204),
  (5, 'Municipio 5', 126837, 4246, 3769),
  (6, 'Municipio 6', 152942, 8367, 1880),
  (7, 'Municipio 7', 176814, 5642, 3436),
  (8, 'Municipio 8 di Milano', 196562, 8287, 1170),
  (9, 'Municipio 9', 190656, 9027, 9304);
```

```
UPDATE municipalities SET
   PercPop = round((population::float / (SELECT SUM(population) FROM municipalities))::numeric, 2),
   PercEnterp = round((NoEnterp::float / (SELECT SUM(NoEnterp) FROM municipalities))::numeric, 2);
```

Then we copy the modified file to the docker container.

```
docker cp scripts/milano_preparedata.sql mda-project:/tmp
```

And we run it.

```
psql -d milano_traj -U postgres -p 5432 -h localhost -f /tmp/milano_preparedata.sql
```

01.07. Import BerlinMOD Generator

We copy the berlinmod_datagenerator.sql file to the container.

```
docker cp scripts/berlinmod_datagenerator.sql mda-project:/tmp
```

And we use `psql` to execute the script and add the functions to the database.

```
psql -d milano_traj -U postgres -p 5432 -h localhost -f /tmp/berlinmod_datagenerator.sql
```

01.08. Run BerlinMOD Generator

We run the function in the database, indicating the `scaleFactor`.

```
SELECT berlinmod_datagenerator(scaleFactor := 0.005);
```

01.09. Dump database

We dump the database to a SQL file.

```
pg_dump -U postgres -p 5432 -h localhost milano_traj > milano_traj_db_dump_[VARIANT].sql
```

Then, we copy the file from the container to our local disk.

```
docker cp mda-project:/tmp/milano_traj_db_dump.sql .
```

 The dump for both variants are already in the repository in the *dumps* directory.