WorkShop : OpenTripPlanner and MobilityDB

Author : Maazouz Mehdi

Mars 2022

1 Introduction

This workshop aims to show you how to connect OpenTripPlanner and MobilityDB and to display the generated trips on Qgis. The generated trips are multi-modal and focus on means of transport obtained via GTFS data.
In this document you will find the tools used for the realization of this workshop as well as the different steps for the creation and visualization of the trips.

2 Environment

This workshop was run on a Ubuntu 18.04 LTS Linux machine, 8GB Ram, Intel Core I5-5300U CPU and 512GB SSD. In addition, the following tools made this WorkShop possible :

- Java 11 (OpenTripPlanner requires Java 11 or later)
- PostgreSQL 13.6
- Postgis 2.5.5
- MobilityDB 1.0
- Python 3.6
  - psycopg2 (installed via PyPi)
- QGis 3.22 Białowieża and its module :
  - Move (please follow instructions on github)

Different versions can be used. Nevertheless, the requirements of MobilityDB must be respected (See here).

3 Procedure and Installation

3.1 GitHub

Please go to the official Github page of the project and download the various files there. Or simply clone the project.

Here is the link to go to the Github page.

This project contains several files :
- Workshop/ is the folder containing the file you are reading
- Code/ contains the sql and python files that will allow the generation of MobilityTrips.

— OTPData/ contains the files necessary for the WEB OTP server to function properly. These files are :
  — brussels.pbf corresponds to the brussels.osm file reduced
  — stib-gtfs.zip corresponds to the GTFS data of the Stib covering the months of March 2022 to April 2022
  — tec-gtfs.zip corresponds to the GTFS data of the TEC covering the months of March 2022 to May 2022
  — build-config.json contains options and parameters that are taken into account during the graph building process

3.2 Prepare Database

For the moment, we will use the OSM map of Brussels. The data was fetched via the OpenStreetMap API (Overpass API) and then reduced into a zip file. The result can be found in the root of the project in the file *./brussels.zip*. Please extract this file in order to get *./brussels.osm*. Then, open a console

```
-- in a console:
$ createdb -h localhost -p 5432 -U dbowner brusselsOTP
-- replace localhost with your database host, 5432 with your port,
-- and dbowner with your database user

-- create needed extensions
$ psql -h localhost -p 5432 -U dbowner -d brusselsOTP
 -c 'CREATE EXTENSION hstore'
-- adds the hstore extension needed by osm2pgsql

$ psql -h localhost -p 5432 -U dbowner -d brusselsOTP
 -c 'CREATE EXTENSION MobilityDB CASCADE'
-- adds the PostGIS and the MobilityDB extensions to the database

$ osm2pgrouting -W '<password>' -h localhost -p 5432 -U dbowner -f brussels.o
-- replace <password> by your dbowner's password
-- don't forget to extract brussels.zip
```

Osm2pgrouting brings us the ways_vertice_pgr table which we will need to choose the source and target nodes. During the conversion, osm2pgrouting transforms the data into WGS84 (SRID 4326), so we will need later to convert it back to SRID 3857.

3.3 Prepare OpenTripPlanner server

In order to contact the OpenTripPlanner API, we will set up a local web server (for now). The .jar file is located at the root of the project and corresponds to OpenTripPlanner 2.0 . When it is executed, it will use the files in the *OTPData/* folder to create the graph. By default, when the graph is created, the transit network will contain the GTFS data of the *Tec* and the *Stib*. If you want to add a GTFS file, please put it in the *OTPData/* folder and add its name in the *build-config.json* file like this.

"gtfs" : "tec-gtfs.zip|stib-gtfs.zip|**my-gtfs.zip**"

To remove a GTFS file from the graph creation (tec-gtfs.zip for example) :

"gtfs" : "stib-gtfs.zip"

Please be careful, for OpenTripPlanner 2 to detect a GTFS file, its name must end in .zip and must contain the letters 'gtfs'.

Now all we have to do is launch our server.

```
-- in a console in opentripplanner directory:
$ java -Xmx4G -jar otp-2.0.0-shaded.jar --build --save ./Datas
```

The parameter **-Xmx4G** means that you reserve 4GB of memory in order to build the graph containing our OSM and GTFS data from Stib and Tec. If you decide to use only OSM and Stib data, the parameter -Xmx2G will be sufficient. The paramter –**build** means the graph is build and –**save** means we store the graph onto disk.

To start the server

```
$ java -jar otp-2.0.0-shaded.jar --load --serve ./Datas
```

3.4 Prepare QGIS

To display the trips, the Qgis tool is used. Instructions for downloading and installing it can be found here.
Please make sure that the plugin is installed. When connecting Qgis with your database, please enable basic authentication. Otherwise, there is a risk that the *Move* module will not work properly.

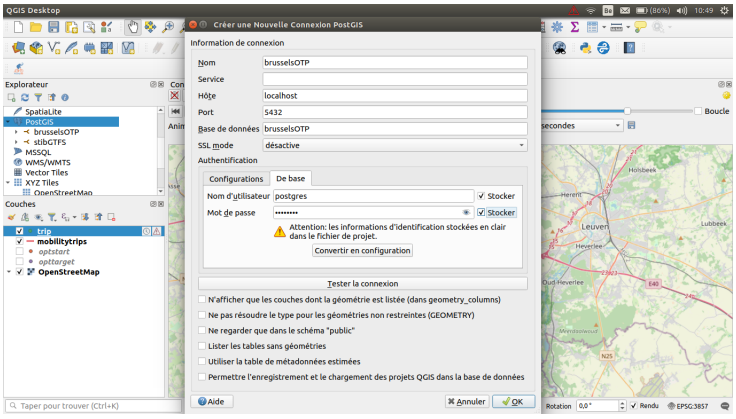

FIGURE 1: New connection in Qgis

4 Generating MobilityDB trips

Once the various tools have been installed and configured. We can start generating MobilityDB trips from trips obtained via OpenTripPlanner. To do this, we will randomly generate source and target nodes that will be used in the calculation of the trips.

```
-- in a console:
$ psql -h localhost -U postgres -d brusselsOTP
-f ./Code/combination.sql -v o=10

-- -v o=10 means we want to generate 10 source nodes and 10 target nodes
```

We will now contact the OpenTripPlanner API to generate trips, and convert them to be compatible with a PostgreSQL database. Please make sure you have started the OTP server before doing this.

```
-- in a console at the root of your project :
$ python3 ./Code/manageOTP.py localhost brusselsOTP dbowner password

-- as usual, replace dbowner by your database user
-- replace password by your database user's password
```

Running this command will prompt you to enter parameters to better define the trips you want to create.
Below are some examples of parameters you might want to enter. *OpenTripPlanner* is full of parameters, a full list can be found at this address.
the desired format is presented as follows : key=value
Here is an example you could enter in the console : numItineraries=2 optimize=QUICK

You could found below some parameters you put enter :

- date=2022-03-10
- time=13 :23
- numItineraries=2 (represents the number of itineraries generated by trip)
- mode=WALK,TRANSIT (represents transport modes to consider)
- wheelchair=true
- arriveBy=true (specifies that the given time is when we plan to arrive)
- maxWalkDistance=1000 (specifies the maximum distance in meters that you are willing to walk)

If you wish, you can press <Enter> directly. In this way, the default values will be taken into account.
Note that the default *date* represents the current day, the default *time* represents the current time.
Please note that the baseline GTFS data covers the months of March 2022. If you are doing this Workshop later, please enter a specific date that is taken into account by the GTFS data
For the sake of this WorkShop, we have changed the default value of the *mode* parameter. Originally *WALK*, it is now *WALK,TRANSIT*. The reason being that this Workshop deals with multi-modal routing, so there is no point in using only your feet to do the trips. Moreover, we changed the default value *numItineraries=1* by *numItineraries=2* mainly because the default value only takes account the *WALK* trip.

Now we just need to run one last sql script to generate the mobility trips.

```
-- in a console at the root of your project
$ psql -h localhost -U dbowner -d brusselsOTP
-f ./Code/generateMobility_Trips.sql

-- We generate MobilityDB trips
```

We now have the *mobilitytrips* table containing our tgeompoints. As well as a table *stibtrip* containing our tgeompoints which represents a trip by public transport, a table *waittrip* which represents people waiting for their transfer and a table *walktrip* which represents people walking. Now let's go to Qgis and open a connection to our BrusselsOTP Database. Once done, click on the *optstart* and *opttarget* tables to display the source nodes and taget nodes. The generated nodes are random, so there is a good chance that you will have other nodes.

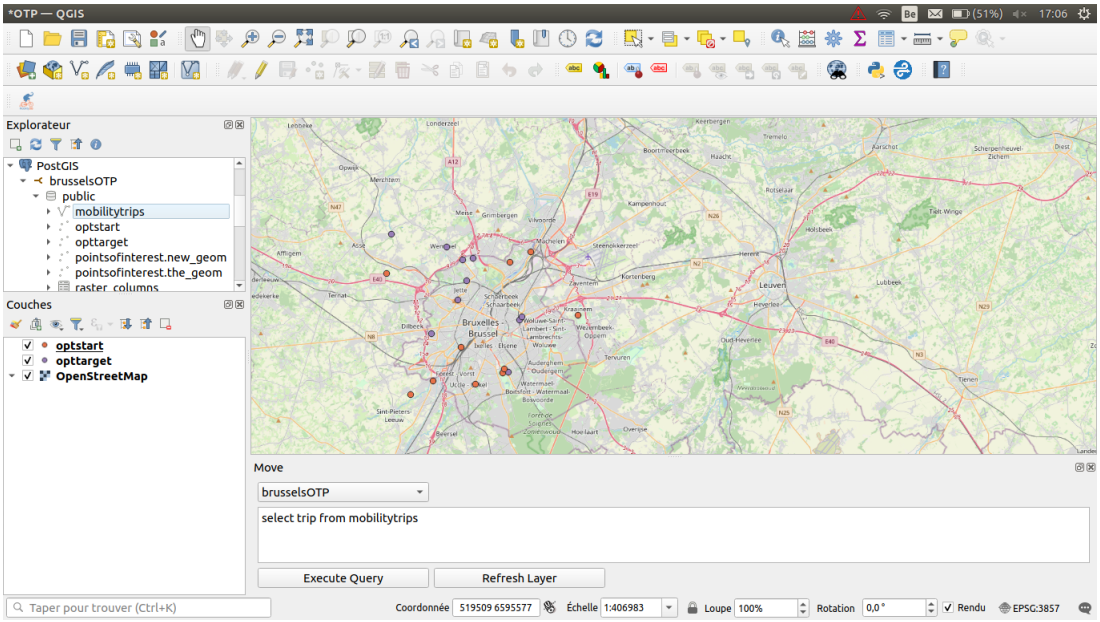

FIGURE 2: source nodes and taget nodes

We can display our *mobilitytrips* table on QGis. However, in order to be able to visualize our points (people) moving in time, we will use the *Move* module.

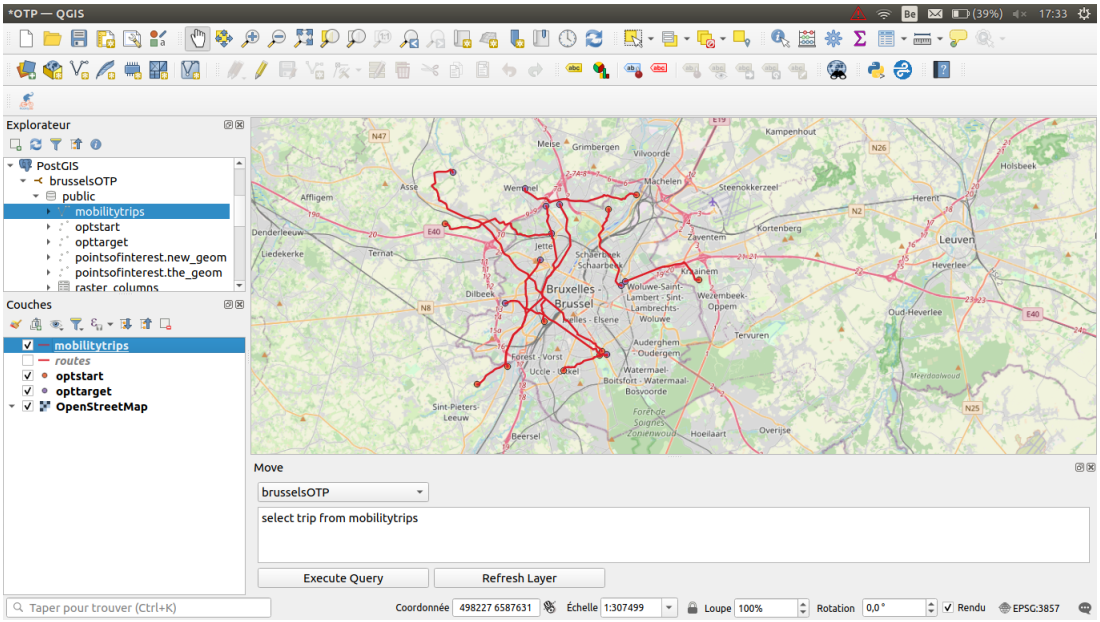

FIGURE 3: table mobilitytrips visualized in Qgis

In order to visualize the points in a temporal way, we just need to use the *Move* plugin (installed previously). Click on the plugin to display a window at the bottom where you insert the query *select trip from mobilitytrips* and click on the *execute query* button. This will create a temporal layer.
Open the controller panel (View->Panels->Temporal Controller Panel) and you can then see the moving points move over time.
Please note that in Belgium we are in UTC+1. I'm currently having trouble displaying the correct time on *Move*.
Therefore, if you generate trips between 04 :00 pm and 04 :30 pm for example, the display of the points on *Move* will be between 03 :00 pm and 03h30 pm.

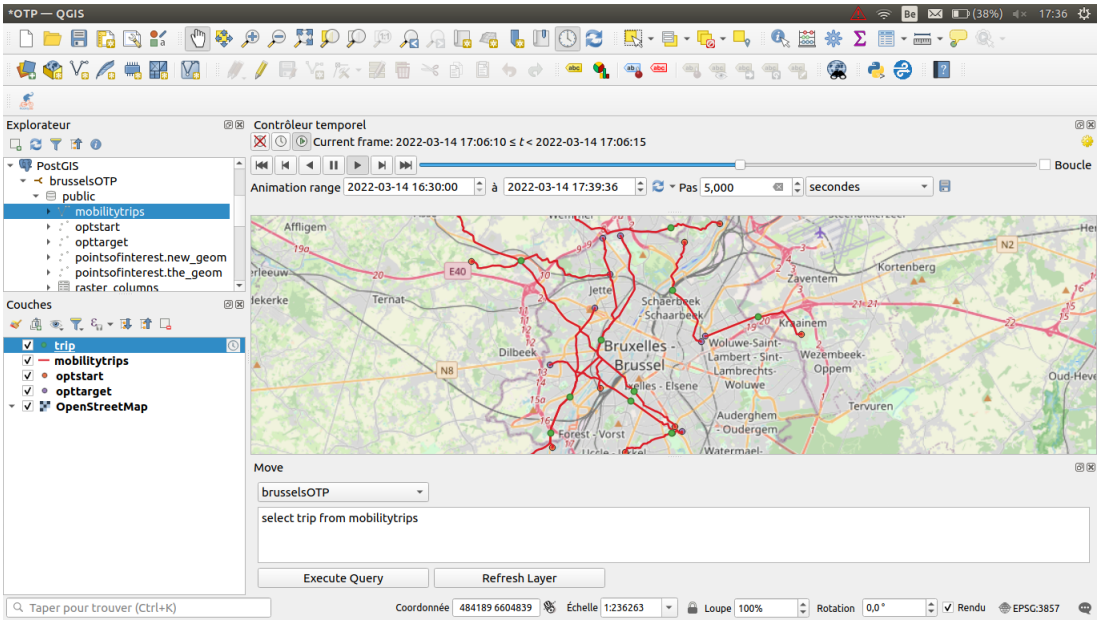

FIGURE 4: moving points visualized in Qgis

If you want a more advanced display to better differentiate between trips by foot and by public transport. Simply insert these 3 queries :
select trip as walktrip from walktrip
select trip as waittrip from waittrip
select trip as stibtrip from stibtrip

Here is one result you might get :

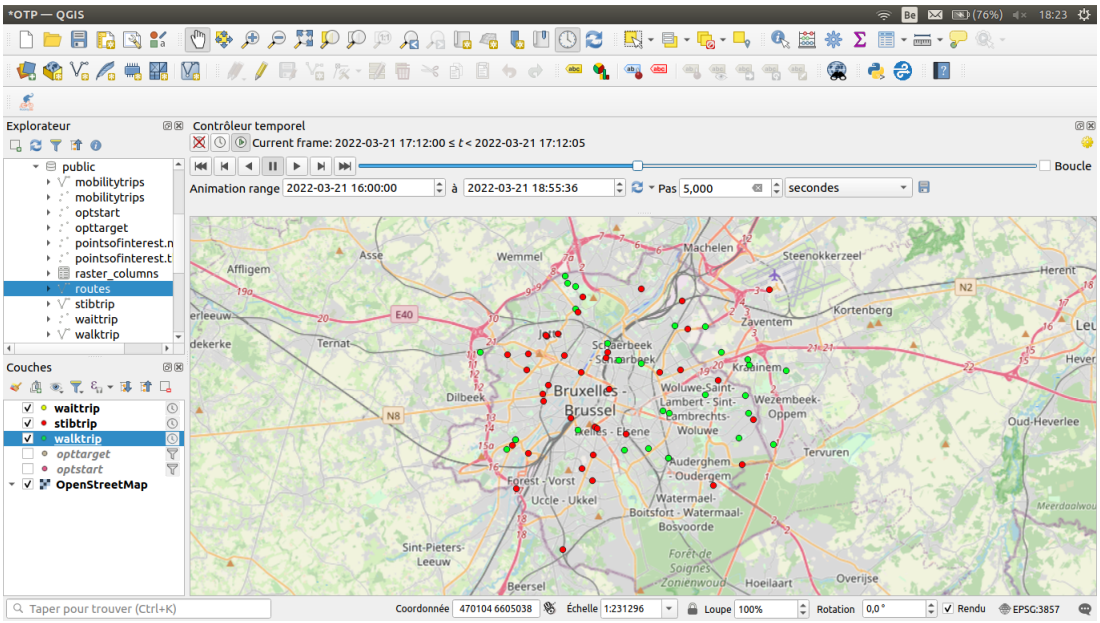

FIGURE 5: moving points visualized in Qgis