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.
To do so, you will be guided through each step in order to correctly install the necessary tools

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
- QGis 3.22 Białowieża and its two modules :
  - OpenTripPlanner plugin (On Qgis go to plugins-> Manage and Install plugins -> All -> type OpenTripPlanner plugin on search bar)
  - Move is little bit harder to insall, 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 files that will allow the generation of MobilityTrips from the trips calculated via the OTP API

— 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.

```
$ osm2pgsql -c -H localhost -U postgres -P 5432
-d brusselsOTP Data/brussels.osm -W
-- loads all layers in the osm file, including the adminstrative regions
```

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|**MyGTFS.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 plugins are installed. When connecting Qgis with your database, please enable basic authentication. Otherwise, there is a risk that the Move module will not work properly.

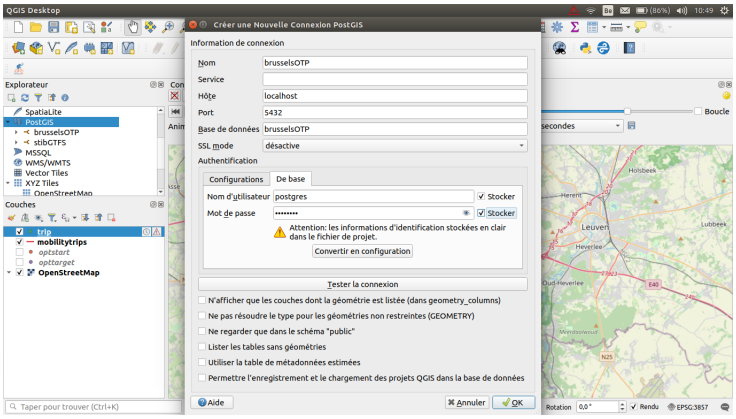

FIGURE 1: New connection in Qgis

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
```

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.

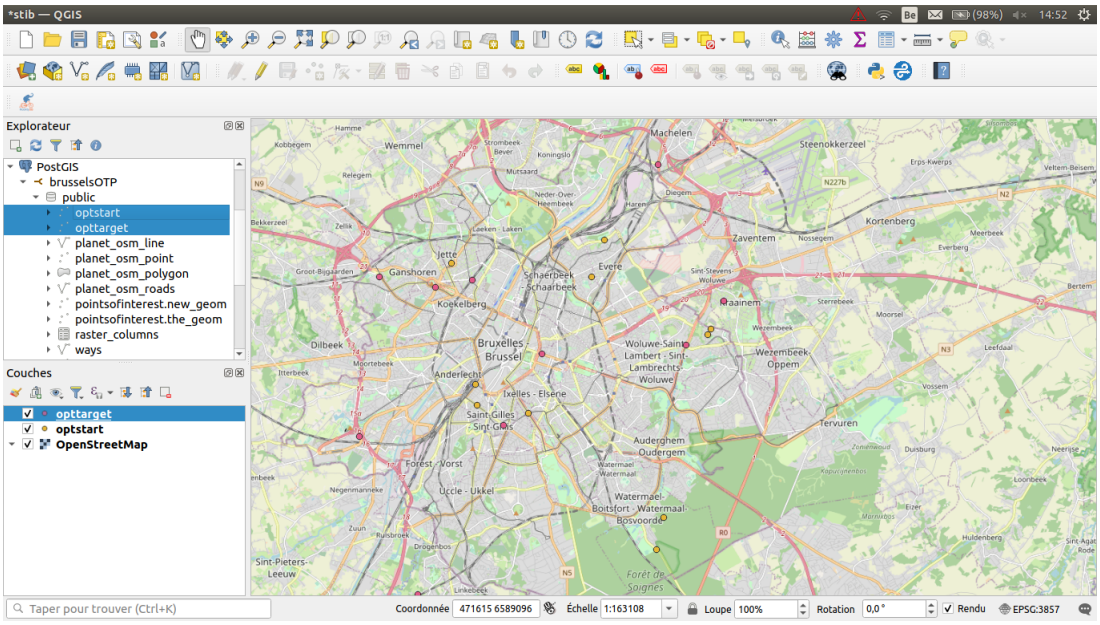

FIGURE 2: source nodes and taget nodes

Open the OpenTripPlanner plugin, click on the Create Routes tab and configure the query as follows :

- Select source layer : optstart
- Select target layer : opttarget
- Select Source Layer's Matching or ID Field : id
- Select Target Layer's Matching or ID Field : ein
- Check the Create only Routes for Matching fields box
- Transportation mode must be : WALK,TRANSIT
- The other parameters can be modified according to your preferences.

Please note that the dates chosen must be between 2022-03-08 and 2022-04-02 Before clicking on *Request Routes*, go to *General Settings* and click on *Check Server Status* to check that everything is ok (the OTP server must be running and the URL indicated must be http ://localhost :8080/otp/routers/default/).

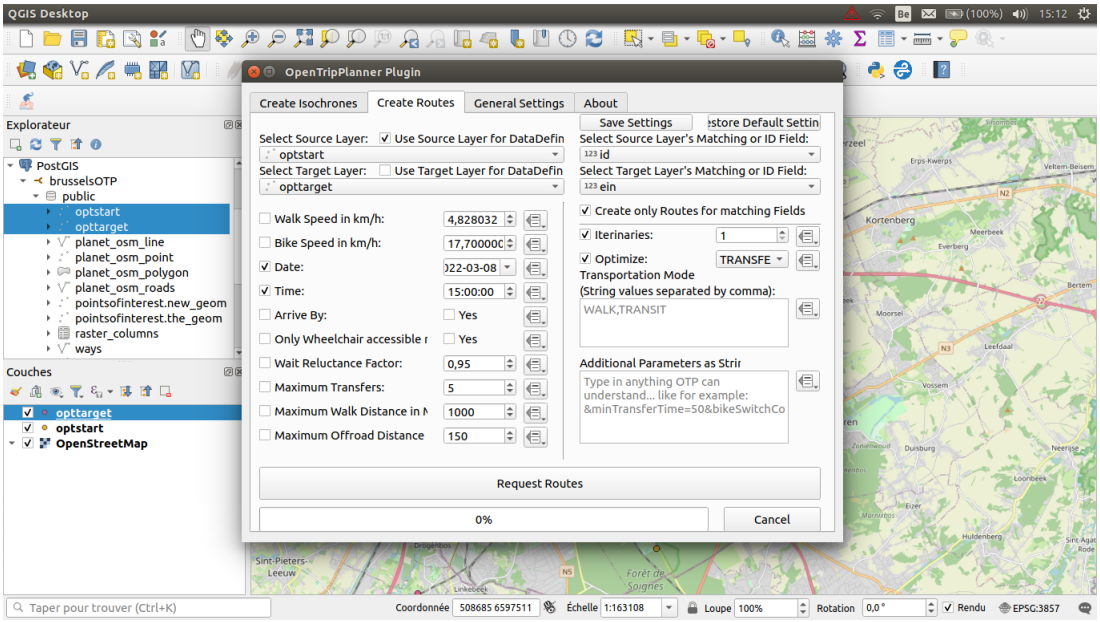

FIGURE 3: OpenTripPlanner Plugin

You get a temporary layer called *Routes* which you can easily view in Qgis.

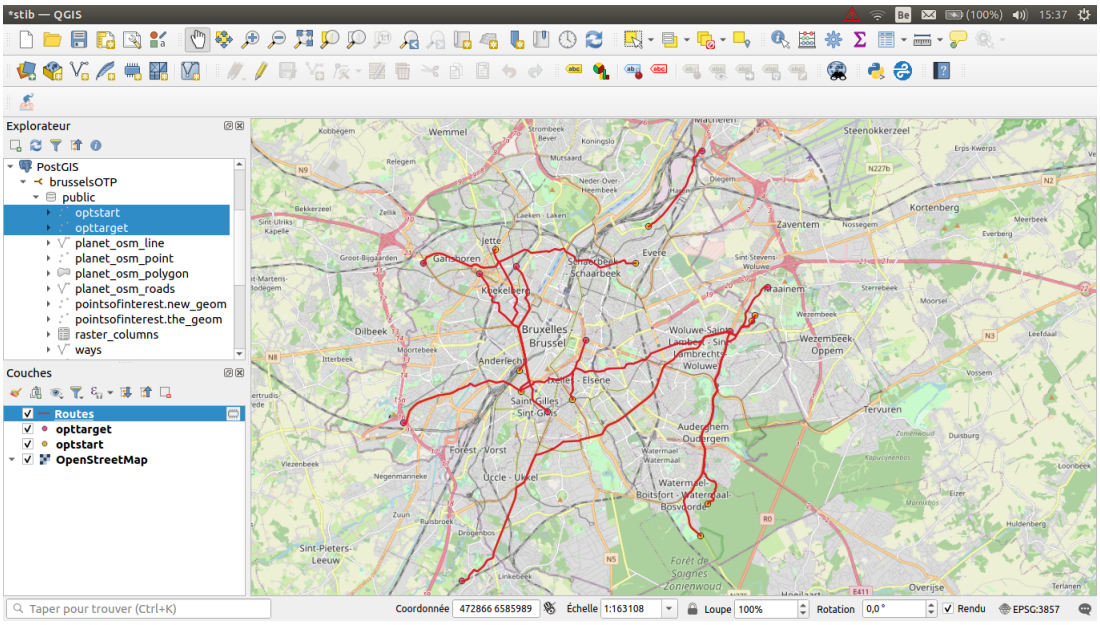

FIGURE 4: OpenTripPlanner trips visualized in Qgis

The route layer is a temporary layer, we need to convert it into a SQL table in order to work on it. To do this, we will contact the Qgis API via the Python console included in QGis.

Open the console (in QGis) and execute the following line :

```
>>> processing.run('qgis:importintopostgis',
{'INPUT':'Routes','DATABASE':'brusselsOTP'})
```

As a result, we transform our temporary layer into a table that we can view in QGis. Now we just need to run one last sql script to generate the mobility trips.

```
$ 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. However, if we display it in QGis, we do not see any change from our *routes* table

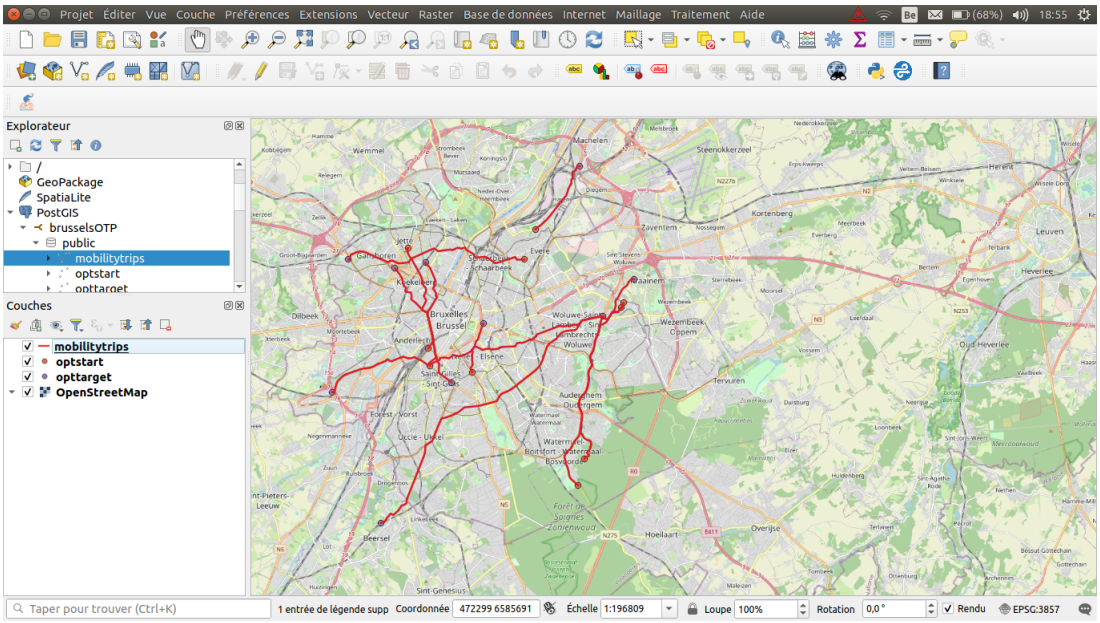

FIGURE 5: table mobilitytrips visualized in Qgis

In order to visualise 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.

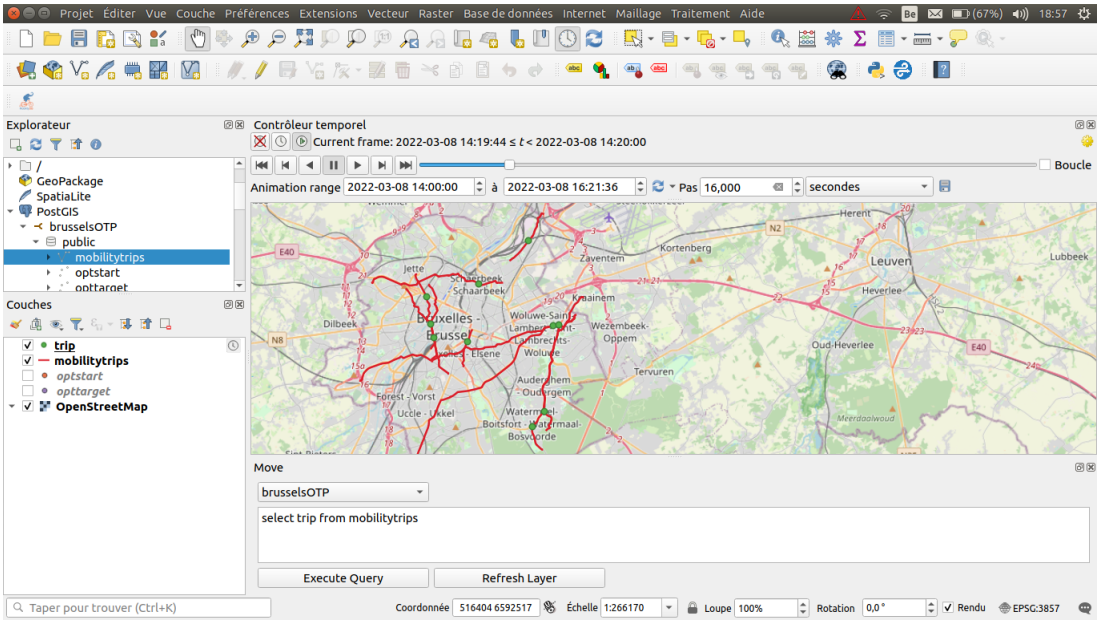

FIGURE 6: moving points visualized in Qgis