1. **otrouting documentation**

The pupose of this document is to present the scripts and the function used to perform routing on opentravellers:

it includes:

- a python script enabling to import openstreetmap data in pgsql database and create a routable database

- a sql script for shortest path calculation between a starting and ending point

- a second sql script enabling to get the route information from a list of point (route information based on osm tags of the closest way)

# References

1. [1] *01\_postgre+postgis installation and OSM data import\_v2.docx*

# pre-requisites

## Postgresql, postgis, osm2pgsql

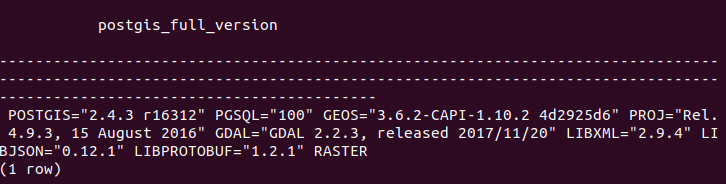
* 1. See [1]

pgpass shall also be installed

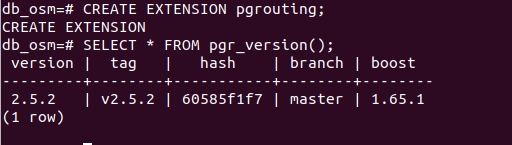
## Pgrouting extension

1. *apt update*
2. install pgrouting
3. *apt install postgresql-10-pgrouting*

## Create db\_osm\_routing postgresql data base with postgis and pgrouting extension

1. See ref [1] if needed to create super user *user*
2. Create database
3. *sudo -u postgres createdb -O user db\_osm\_routing*
4. Open db
5. *psql -h localhost -U user db\_osm*
6. Password for user user: xxxxx
7. *# CREATE EXTENSION postgis;*
8. *# SELECT postgis\_full\_version();*
9. 

press “q” to quit pgsql answer

1. *# CREATE EXTENSION pgrouting;*
2. *# SELECT \* FROM pgr\_version();*
3. 
4. press “ctrl+z” to quit sql db

## pgpass

1. .pgpass file shall be used so as to avoid to use password in the osm2pgsql line
2. <https://linuxandryan.wordpress.com/2013/03/07/creating-and-using-a-pgpass-file/>
3. *cd ~*
4. *touch .pgpass*
5. *nano .pgpass*
6. localhost:5432:db\_osm\_routing:user:xxxxx
7. *chmod 0600 .pgpass*

## Python

The following extension are used:

import sys

import psycopg2

import time

import math

import os

import subprocess

# OSM data import and formating

## Python script

Osm data are imported and formatted through the python script *otrouting\_py/otrouting\_v1.3.py*

## Input data

### Input data location

Input data shall be fill in the beginning of the ot\_routing\_v1.3.py script

Open python script to fill the input data

*nano ot\_routing\_v1.3.py*

### Database identification

Identification parameters to connect the database

*glob\_db = 'db\_osm\_routing'*

*glob\_user = 'user'*

*glob\_host = 'localhost'*

*glob\_port = '5432'*

*glob\_password = 'xxxxx'*

### osm input data to be imported

a table of osm input data to be imported, data will be imported from [http://download.geofabrik.de](http://download.geofabrik.de/)

osm input data shall be formatted as presented below:

http://download.geofabrik.de/europe/andorra.html **→**  ["europe", "andorra"]

*glob\_t\_input = []*

*#http://download.geofabrik.de*

*#MONACO - ANDORRA ######################*

*glob\_t\_input.append(["europe", "andorra"])*

*glob\_t\_input.append(["europe", "monaco"])*

*#FRANCE ###################################*

*#glob\_t\_input.append(["europe", "france", "alsace"])*

*#glob\_t\_input.append(["europe", "france", "aquitaine"])*

*#glob\_t\_input.append(["europe", "france", "auvergne"])*

*#glob\_t\_input.append(["europe", "france", "basse-normandie"])*

## Script run

### Execution command

Make otrouting file executable

*Chmod 777 ot\_routing\_v1.3.py*

Run python script

./*ot\_routing\_v1.3.py*

### script architecture overview

The script will basically:

1/ initialise database,

1.1/ Remove osm import temporary table from db:

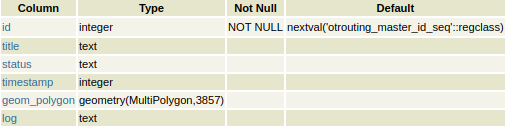
* + 1. § DROP TABLE IF EXISTS otrouting\_points

1. § DROP TABLE IF EXISTS planet\_osm\_\*
2. 1.2/ create extension postgis and pgrouting if not exists in db
   1. § CREATE EXTENSION IF NOT EXISTS postgis
3. § CREATE EXTENSION IF NOT EXISTS pgrouting
   * 1. 1.3/ Create table requested for otrouting requests in db:
4. § CREATE TABLE IF NOT EXISTS otrouting\_master
   1. § CREATE TABLE IF NOT EXISTS otrouting\_points
   2. § CREATE TABLE IF NOT EXISTS otrouting\_ways
5. § CREATE INDEX IF NOT EXISTS otrouting\_ways\_geom\_center\_gist
   1. § CREATE INDEX IF NOT EXISTS otrouting\_ways\_id
   2. 2/ upload data from download.geofabrik.de
6. > wget -O europe\_andorra-latest.osm.bz2 “http://download.geofabrik.de/europe/andorra-latest.osm.bz2”
7. > wget -O europe\_andorra.poly “http://download.geofabrik.de/europe/andorra.poly”
9. 3/ import osm data with osm2pgsql
10. > osm2pgsql --slim -C 100 -d db\_osm\_routing -U user -H localhost -P 5432 europe\_andorra-latest.osm.bz2
11. NOTE: .pgpass have to be created so as to avoid to have to fill manually the password
13. *4/ create routable table, otrouting\_points is populated with new summits, otrouting\_ways is populated with new vertex*

## Output data

### otrouting\_master pgsql table

This table enable to record all osm data import.



**status:**

**- *completed*** if data in otrouting\_ways and otrouting\_points tables corresponds to this import

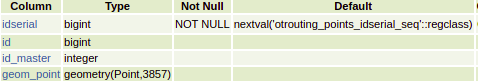
- ***obsolete*** if data in otrouting\_ways and otrouting\_points tables have been overseed by another import more recent

**geom\_polygon**: corresponds to the .poly file that describes the extent of the imported region from [http://download.geofabrik.de](http://download.geofabrik.de/)

**log: l**og with the versoin of the script used for this import, the number of point imported in otrouting\_points and otroutingways tables

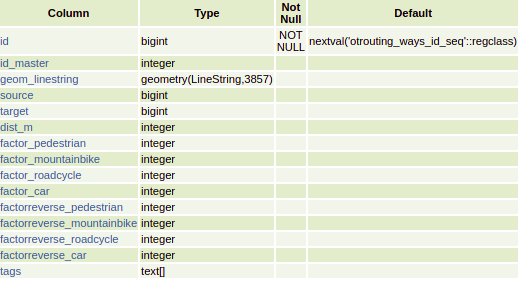
### otrouting\_points pgsql table

This table contains all the summit of the routable network generated from osm data



### otrouting\_ways pgsql table

This table contains all the edge of the routable network generated from osm database



## Examples of postgis call

### Create table with postgis entry and index

*CREATE TABLE IF NOT EXISTS otrouting\_master(id serial PRIMARY KEY, title text, status text, log text, timestamp int, geom\_polygon geometry(MultiPolygon,3857));*

*CREATE INDEX IF NOT EXISTS otrouting\_master\_geom\_polygon ON otrouting\_master USING GIST (geom\_polygon);*

*CREATE TABLE IF NOT EXISTS otrouting\_points(idserial bigserial PRIMARY KEY, id bigint, id\_master int, geom\_point geometry(Point,3857));*

*CREATE INDEX IF NOT EXISTS otrouting\_points\_geom\_point\_gist ON otrouting\_points USING GIST (geom\_point);*

*CREATE TABLE IF NOT EXISTS otrouting\_ways(id bigserial PRIMARY KEY, id\_master int, geom\_linestring geometry(Linestring,3857), source bigint, target bigint, dist\_m int, factor\_pedestrian int, factor\_mountainbike int, factor\_roadcycle int, factor\_car int, factorreverse\_pedestrian int, factorreverse\_mountainbike int, factorreverse\_roadcycle int, factorreverse\_car int, tags text[]);*

*CREATE INDEX IF NOT EXISTS otrouting\_ways\_geom\_linestring\_gist ON otrouting\_ways USING GIST (geom\_linestring);*

### Insert postgis multipolygon geometry

*INSERT INTO otrouting\_master (title, status, timestamp, geom\_polygon) VALUES ('europe\_andorra','started', 1647031730.99, ST\_Transform(ST\_GeomFromText('MULTIPOLYGON(((1.516233 42.4276,1.496092 42.43182,1.455187 42.43475,1.446745 42.43823)))', 4326), 3857));*

### Create, transform and insert postgis point geometry

create point from coordinate

*ST\_PointFromText('POINT(1.7359278 42.5436511)', 4326);*

transform coordinate from EPSG4326 to 3857

*ST\_Transform(ST\_PointFromText('POINT(1.7359278 42.5436511)', 4326), 3857);*

insert data into table

*INSERT INTO otrouting\_points (id, id\_master, geom\_point) VALUES(3655224911, 151, ST\_Transform(ST\_PointFromText('POINT(1.7359278 42.5436511)', 4326), 3857));*

### Create and insert postgis linestring geometry

*INSERT INTO otrouting\_ways (id\_master, geom\_linestring, source, target, dist\_m, factor\_pedestrian, factor\_mountainbike, factor\_roadcycle, factor\_car, factorreverse\_pedestrian, factorreverse\_mountainbike, factorreverse\_roadcycle, factorreverse\_car, tags) VALUES(152, ST\_GeomFromText('LINESTRING(193242.598749888 5242766.84650556,193224.531596532 5242753.24816413,193216.516593195 5242747.49153884,193205.763130383 5242740.13333786,193204.004282429 5242739.39298531)',3857), 3655224911, 4817024439, 34, 152, 152, 132, 97, 1, -1, -1, -1, ARRAY ['highway','primary','maxspeed','30','oneway','yes','ref','N 22','surface','asphalt'])*

## Clean working directory

Clean working directory and remove osm imported and log files

*chmod 777 clean.txt*

*./clean.txt*

the following function will be executed:

*rm -rf wget-log\**

*rm -rf \*.poly*

*rm -rf \*.osm.bz2*

# otrouting\_getrouting sql function

## Sql function

Import sql function from *otrouting\_sql/otrouting\_getrouting\_v1.5.sql* into db\_osm\_routing table

FUNCTION otrouting\_getrouting(

lon1 FLOAT,

lat1 FLOAT,

lon2 FLOAT,

lat2 FLOAT,

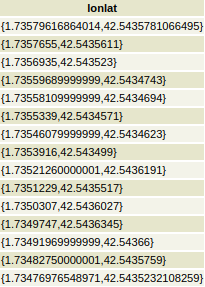
routingtype TEXT)

RETURNS TABLE (lonlat text[])

routingtype can have the following values: pedestrian / mountainbike / roadcycle / car

## Examples

*SELECT \* FROM otrouting\_getrouting(1.7358052810252, 42.54356918667, 1.73470359630932, 42.5435625682116, 'pedestrian');*



## Examples of postgis call

SELECT otrouting\_ways.id AS id,

ST\_Distance(otrouting\_ways.geom\_linestring, point1) AS dist,

otrouting\_ways.geom\_linestring,

otrouting\_ways.source,

otrouting\_ways.target,

otrouting\_ways.factor\_pedestrian,

otrouting\_ways.factor\_mountainbike,

otrouting\_ways.factor\_roadcycle,

otrouting\_ways.factor\_car,

otrouting\_ways.factorreverse\_pedestrian,

otrouting\_ways.factorreverse\_mountainbike,

otrouting\_ways.factorreverse\_roadcycle,

otrouting\_ways.factorreverse\_car

INTO tableway1 FROM otrouting\_ways WHERE ST\_Intersects(otrouting\_ways.geom\_linestring, ST\_MakeEnvelope(lonmin\_loc, latmin\_loc, lonmax\_loc, latmax\_loc, 3857)) AND factor\_car > 0 ORDER BY dist ASC LIMIT 1;

ST\_ClosestPoint(tableway1.geom\_linestring, point1)

ST\_AsText(ST\_Transform(linestring1,4326))

*ST\_GeomFromText('LINESTRING(' || text11 || ')',3857);*

*ST\_GeomFromText('MULTILINESTRING(('|| text11*

*|| '),('*

*|| text22*

*|| '))',3857);*

*ST\_AsText(ST\_CollectionExtract(ST\_Split(tableway1.geom\_linestring, proj1\_linestring),2));*

*ST\_Distance(\_temp\_otrouting\_points.geom\_point, proj1)*

# otrouting\_getrouteinfo sql function

## Sql function

Import sql function from *otrouting\_getrouteinfo\_v1.4sql* into db\_osm\_routing table

FUNCTION otrouting\_getrouteinfo(

arraysources TEXT[],

threeshold\_distProj\_m FLOAT)

RETURNS TABLE (

geom\_source TEXT,

geom\_proj TEXT,

ids\_way\_source\_target BIGINT[],

dist DOUBLE PRECISION,

tags text[]

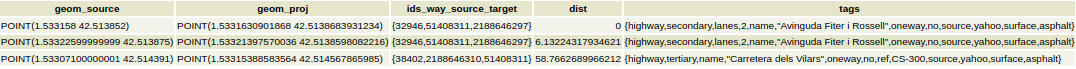
)

arraysources is a list of coordinate in EPSG4326, as example: ARRAY['POINT(1.533258 42.513852)','POINT(1.533126 42.513875)'

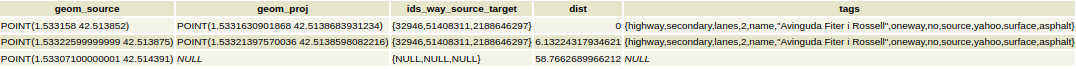
threeshold\_distProj\_m is a maximum distance to consider a projection between an input coordinate and a route

## Examples

*SELECT \* FROM otrouting\_getrouteinfo(ARRAY['POINT(1.533158 42.513852)','POINT(1.533226 42.513875)','POINT(1.533071 42.514391)'], 30.0);*



*SELECT \* FROM otrouting\_getrouteinfo(ARRAY['POINT(1.533158 42.513852)','POINT(1.533226 42.513875)','POINT(1.533071 42.514391)'], 10.0);*



## Examples of postgis call

*dist\_m := ST\_DistanceSphere(point1\_4326, point11\_4326);*

*dist\_3852 := ST\_Distance(point1, point11);*

*point1 := ST\_Transform(ST\_GeomFromText(arraysources[i],4326),3857);*

*lon1 := ST\_X(point1);*

*lat1 := ST\_Y(point1);*

*ST\_Intersects(otrouting\_ways.geom\_linestring, ST\_MakeEnvelope(lonmin\_loc, latmin\_loc, lonmax\_loc, latmax\_loc, 3857))*