Workshop WPS

Network analyses
with PostGIS/pgRouting

ubuntu

Preparation

- Create database
- Add extension
- Create network topology

Extensions

```
CREATE EXTENSION postgis;
CREATE EXTENSION postgis_topology;
CREATE EXTENSION pgrouting;
```

Data

```
shp2pgsql roads.shp roads > roads.sql
\i roads.sql
```

Topologie

```
ALTER TABLE roads ADD COLUMN "source"
  integer;

ALTER TABLE roads ADD COLUMN "target"
  integer;


SELECT pgr_createTopology('roads',
  0.00001, 'geom', 'gid');
```

Dijkstra

```
SELECT seq, id1 AS node, id2 AS edge, cost
  FROM pgr_dijkstra('
   SELECT gid AS id,
            source::integer,
            target::integer,
            length::double precision AS cost
    FROM roads',
           30, 60, false, false);
```

Output

```
 seq | node  | edge  | cost
-----+-------+-------+-----
   0 |    30 | 22330 |  145
   1 |    29 | 22332 |  627
   2 | 17736 | 22333 | 2429
   3 | 17737 | 22334 |  424
. . .
  86 |  8704 | 10687 |  253
  87 |  8706 | 10797 | 3054
  88 |    60 |    -1 |    0
(89 rows)
```

Connect edges geometry

```
SELECT seq, route.id1 AS node, route.id2 AS edge,
   route.cost, edges.geom FROM pgr_dijkstra('
   SELECT gid AS id,
        source::integer,
        target::integer,
        length::double precision AS cost
   FROM roads',
        30, 60, false, false)
AS route, roads AS edges
WHERE route.id2 = edges.gid;
```

Result

```
 seq | node  | edge  | cost | geom
-----+-------+-------+------+-----
   0 |    30 | 22330 |  145 | 01050000...
   1 |    29 | 22332 |  627 | 01050000...
   2 | 17736 | 22333 | 2429 | 01050000...
   3 | 17737 | 22334 |  424 | 01050000...
. . .
  86 |  8704 | 10687 |  253 | 01050000...
  87 |  8706 | 10797 | 3054 | 01050000...
(88 rows)
```

ubuntu

Steps

- Seachr path between two municipalities
- Find nodes close to selected municipalities
- Find path between these nodes
- Save results as a new table in PostgreSQL
- Publish table from PostgreSQL as WMS (WFS) service

Find node

```
SELECT a.id gid, b.id nodeid,
  ST_distance(a.geom,b.the_geom) AS
  distance FROM municipalities AS a,
  roads_vertices_pgr AS b

WHERE
  ST_dwithin(a.geom,b.the_geom,2000)
  AND

a.name = 'Ostrava'

ORDER BY distance

LIMIT 1;
```

Find node

```
ResultSet r = s.executeQuery("SELECT a.id
  gid, b.id nodeid,
  ST_distance(a.geom,b.the_geom) AS
  distance FROM municipalities AS a,
  roads_vertices_pgr AS b WHERE
  ST_dwithin(a.geom,b.the_geom,2000)
  AND a.name = '" + from + "' ORDER BY
  distance LIMIT 1");

while (r.next()) {

    nodefrom = r.getInt(2);

}
```

Find path

```
s.execute("CREATE TABLE path AS SELECT seq, route.id1 AS
    node, route.id2 AS edge, route.cost, edges.geom FROM
    pgr_dijkstra('\n"
                + "   SELECT gid AS id,\n"
                + "          source::integer,\n"
                + "          target::integer,\n"
                + "          length::double precision AS cost\n"
                + "   FROM roads',\n"
                + nodefrom + ", " + nodeto
                + ", false, false) \n"
                + "AS route, roads AS edges \n"
                + "WHERE route.id2 = edges.gid ORDER BY seq;");
```

Publish on WMS

```
String strURL =
    "http://localhost:8080/geoserver/rest/workspaces/crwgs84/da
    tastores/user/featuretypes";

PostMethod post = new PostMethod(strURL);



post.setRequestHeader("Content-type", "text/xml");

post.setRequestEntity(new StringRequestEntity("<?xml
    version=\"1.0\"?
    ><featureType><name>path</name></featureType>"));
```

Add to service

```
@DescribeProcess(title = "networkingWPS", description =
   "Networking example")

public class NetworkingWPS implements
   GeoServerProcess {

 ...

}
```

Add to service / 2

```
@DescribeResult(name = "result", description =
"WMS where the path is published")

  public String execute(@DescribeParameter(name
= "from", description = "Municipality from")
String from, @DescribeParameter(name = "to",
description = "Municipality to") String to) {

      Examples e = new Examples();

      return e.networking(from, to);

  }
```

Input WPS

Constructeur de requête WPS

Constructeur pas à pas de requête WPS.

Choisir process

gs:NetworkingWPS

Networking example (WPS DescribeProcess)

Entrées du process

from* - String

Municipality from

Ostrava

to* - String

Municipality to

Fulnek

Sorties du process

result* - String

WMS where the path is published

☑ Generate

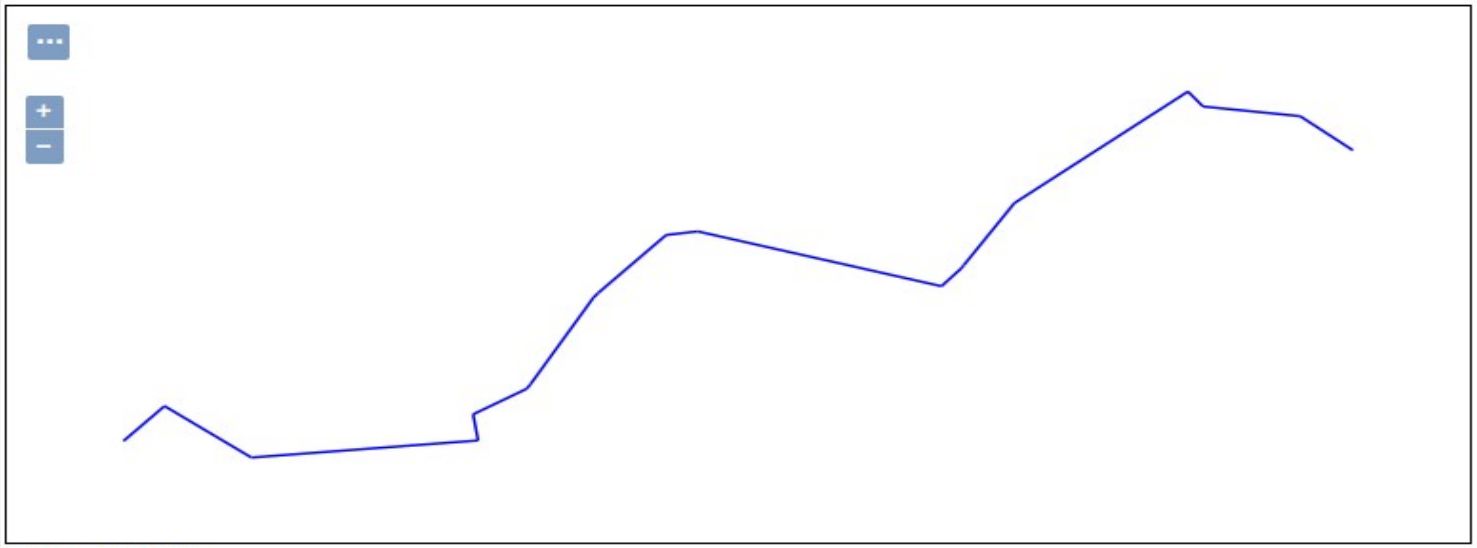
ubuntu

Input WPS

```
<?xml version="1.0" encoding="UTF-8"?><wps:Execute versi
  <ows:Identifier>gs:NetworkingWPS</ows:Identifier>
  <wps:DataInputs>
    <wps:Input>
      <ows:Identifier>from</ows:Identifier>
      <wps:Data>
        <wps:LiteralData>Ostrava</wps:LiteralData>
      </wps:Data>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>to</ows:Identifier>
      <wps:Data>
        <wps:LiteralData>Fulnek</wps:LiteralData>
      </wps:Data>
    </wps:Input>
  </wps:DataInputs>
  <wps:ResponseForm>
    <wps:RawDataOutput>
      <ows:Identifier>result</ows:Identifier>
    </wps:RawDataOutput>
  </wps:ResponseForm>
</wps:Execute>
```

Output WPS

```
Name from to:Ostrava Fulnek
Id from to:10817 22077
http://localhost:8080/geoserver/cr/wms?servic
Layer name:path1444205667463
```



Scale = 1 : 136K
Click on the map to get feature info

ubuntu