OpenStreetMap Sample Project

Data Wrangling using MongoDB

Area of interest aarhus – Denmark

1- Area of interest:

Aarhus is the second-largest city in Denmark and the seat of Aarhus municipality. It is located on the east coast of the Jutland peninsula, in the geographical centre of Denmark, 187 kilometres (116 mi) northwest of Copenhagen and 289 kilometres (180 mi) north of Hamburg, Germany. The inner urban area contains 264,716 inhabitants (as of 1 January 2016) and the municipal population is 330,639 (as of 2016). Aarhus is the central city in the East Jutland metropolitan area, which had a total population of 1.378 million in 2016

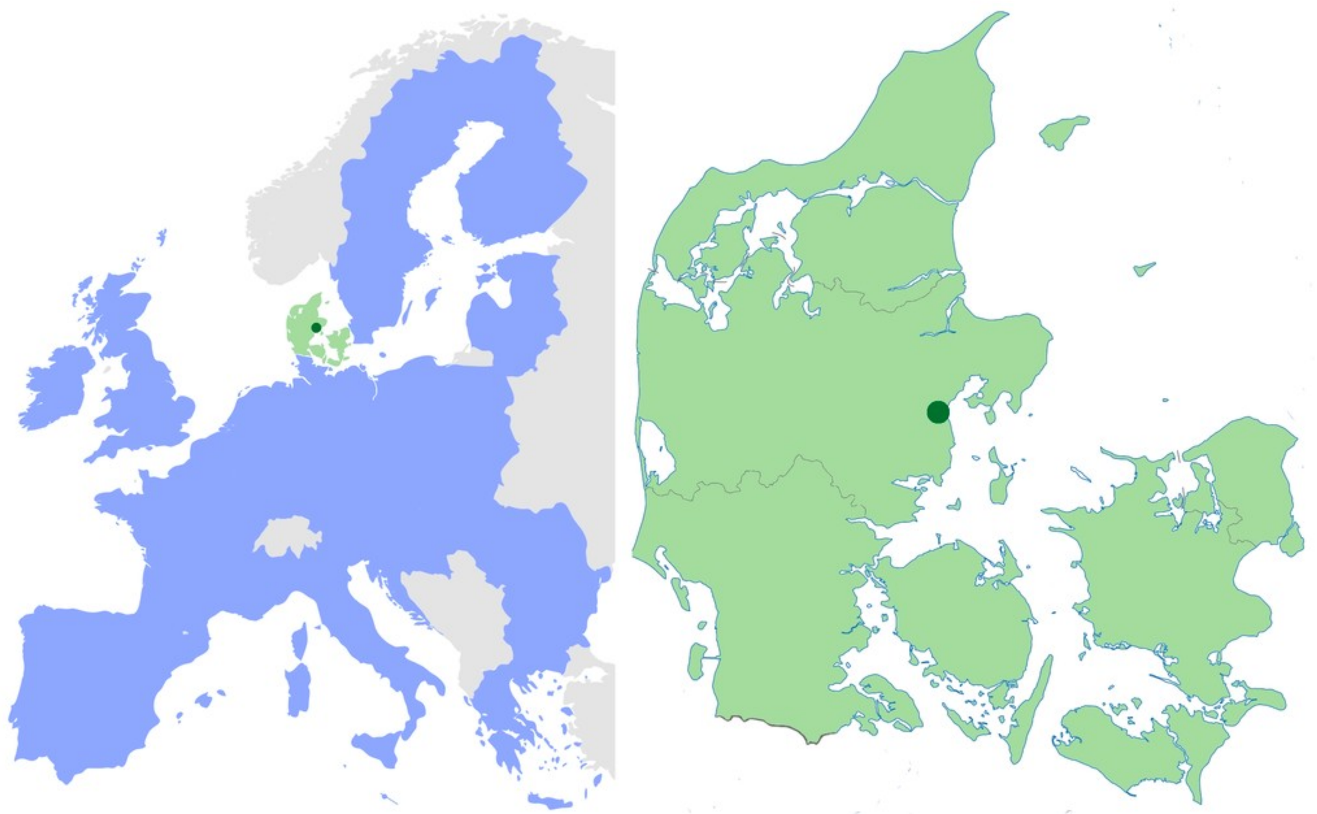

Illustration 1: Position of Aarhus in Denmark and in the E.U

I downloaded the OSM data of Aarhus from Map Zen https://mapzen.com/data/metro-extracts/metro/aarhus_denmark/ with a size of:

aarhus-denmark.osm   ------- 135.6 MB

2- Problems encountered with the data:

the raw data needs to be extracted in the JSON format so that it can be later inserted in MongoDB so for that purpose the data will need to be cleaned from problematic characters like [=+/&<>;'"?%#$@\,\. \t\r\n] and since the name of the streets and nodes are in Danish I had to make sure note only the 26 alphabetical characters are taken in consideration but also the additional characters in the Danish Alphabet.

Another problem is that some tags included in address appear in this way : <tag k="addr:street" v="Bjørnholt"/> , this has been taken care of in the wrangling part to shape the elements in the JSON format.

Some tags have duplicate information in addr key and in osak which stands for *Officielle Standard Adresser og Koordinater* (Official Standard Addresses and Coordinates) in Denmark. So for this project I only consider data inside the addr tags

3- Wrangling Procedure:

The goal of this part is to explain what has been done to get the ready to insert data in mongoDB.

The input is the XML file from the OSM data of Aarhus the first outputs would be two JSON files: aarhus_nodes.json with the elements with the node tag are shaped in JSON format, and aarhus_ways.json which contains elements with way tag shaped into the same JSON format.

This first part of the data wrangling is done automatically in the file XMLtoJSON.py, here is an overview of what happens in it:

- the script opens the aarhus-denmark.osm file
- iterates over its element using xml.etree.cElementTree:
    - creates a dictionary for that element
    - checks if the element is a node or a way (by its first tag)
    - inserts the available properties as entries of the dictionary
        - loop through the addr: tags of the elements
        - filter the problematic characters from the address tags
        - group the elements of the address as a dictionary and add it as an entry to the element's dictionary
    - if the element is of type way group the references of its nodes “nd” in an array and add it as an entry to the element's dictionary
- writes the element in the JSON file for its type (nodes/ways)

the JSON output should be in this form :

```
{
```

```
"id": "2406124091",
"type: "node",
"visible":"true",
"created": {
          "version":"2",
          "changeset":"17206049",
          "timestamp":"2013-08-03T16:43:42Z",
          "user":"linuxUser16",
          "uid":"1219059"
        },
"pos": [41.9757030, -87.6921867],
"address": {
          "housenumber": "5157",
          "postcode": "60625",
          "street": "North Lincoln Ave"
        },
"amenity": "restaurant",
"cuisine": "mexican",
"name": "La Cabana De Don Luis",
"phone": "1 (773)-271-5176"
}
```

after generating the 2 json files, the script JSONtoMONGO.py inserts their content in the mongo Database.

4- Data Overview:

in this part I will present some statistics about the data, first the size of the files generated in the first part of the wrangling process:

```
aarhus_nodes.json → 119.9 MB
aarhus_ways.json → 18.4 MB
```

* Number of documents:

the Database named DAND3 contains 2 collections aarhus_nodes and aarhus_ways, here are the sizes of each:

```
db.aarhus_ways.find().count() → 55432
db.aarhus_nodes.find().count() → 441696
```

* Number of unique users :

for both Collections how many users there are :

```
db.aarhus_nodes.distinct(“created.user”) → 335
db.aarhus_ways.distinct(“created.user”) → 285
```

* Top contributing user:
db.aarhus_nodes.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":1}]) → { "_id" : "AWSbot", "count" : 82996 }

I looked for it in the OSM wikis, this user is a bot used by the council of the city aarhus in Denmark to load data from different servers to OSM, it stands for Adresse Web Services

however for the ways another user takes the lead :

db.aarhus_ways.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}}, {"$sort":{"count":-1}}, {"$limit":1}]) → { "_id" : "Flare", "count" : 11580 }

* Number of users appearing only once:
db.aarhus_nodes.aggregate([{"$group":{"_id":"$created.user", "count":{"$sum":1}}}, {"$group":{"_id":"$count", "num_users":{"$sum":1}}}, {"$sort":{"_id":1}}, {"$limit":1}]) → 73

In this project I am more interested in nodes since they are the elements that have a geographic information

5 – Additional Ideas :

since the ways are represented as a group of nodes, and they have no geographic information, and as a GIS engineer I would like to consider extraction the position from the nodes and generate a geometry attribute for the ways.

This pipeline does the work for now, it uses $lookup to join the 2 collections on the nodes id and outputs the result in a new collection called nodes:

db.aarhus_ways.aggregate([{$lookup:{from: "aarhus_nodes",localField: "node_refs",foreignField: "id",as: "nodes"}},{"$project":{"id":1,"name":1,"highway":1,"nodes.pos":1}},{"$out":"nodes"}])

* Note that I ignored some of the data since here I'm only interested in the [id,name,highway] of the ways and the position from the nodes

There is another way to do it using a combination of $unwind , $lookup and $group stages of a pipeline:

first I unwind the ways collection on the node_refs table then I do a lookup to join it with the nodes collection and then group by the id of the way to get all nodes belonging to each way.

db.aarhus_ways.aggregate([{"$unwind":"$node_refs"},{$lookup:{from: "aarhus_nodes",localField: "node_refs",foreignField: "id",as: "nodes"}},{"$project":{"id":1,"name":1,"highway":1,"nodes.pos":1}},{"$group":{"_id":"$id"}}]).pretty()