Lab Project: OpenStreetMap

Zhenfeng Shi, Hongru Zhu, Chang Zhou

jack.shi2013@gmail.com

Abstract

Keywords: OSM, Database

1. Usage

1.1. Environment

Python 3 + pymysql

1.2. Install

Enter the root path of this project, run the following command in the shell:

```
python SZZ_install.py [-h] [-c host] [-u user] [-p passwd] [-n dbname] [-i input]

                     -c:  host connect, for instance 'localhost'
                     -u:  username for mysql, for instance 'root'
                     -p:  password for mysql, ignore this if no password
                     -n:  name for the new database
                     -i:  inputfile path, for instance '../shanghai_dump.osm'
```

For instance,

```
python SZZ_install -c localhost -u root -n OSM -i data/shanghai_dump.osm
```

1.3. Queries

2. Database Design

2.1. XML Parsing

2.2. E-R Model

2.3. SQL For Table Creation

```
CREATE TABLE ways(
           wayID VARCHAR(12),
                    LineString LINESTRING,
                    name VARCHAR(100), INDEX(name),
                    isRoad VARCHAR(100),
                    otherInfo TEXT,
                    PRIMARY KEY(wayID)
```

```
                ) ENGINE=MyISAM

CREATE TABLE nodes(
                    nodeID VARCHAR(12),
                    version TINYINT(1), INDEX(version),
                    version BOOLEAN,
                    PRIMARY KEY(nodeID)
                ) ENGINE=MyISAM

CREATE TABLE POIs(
                    nodeID VARCHAR(12),
                    position POINT NOT NULL, SPATIAL INDEX(position),
                    planaxy POINT NOT NULL, SPATIAL INDEX(planaxy),
                    name VARCHAR(100), INDEX(name),
                    poitype VARCHAR(100), INDEX(poitype),
                    otherInfo TEXT,
                    PRIMARY KEY(nodeID)
                ) ENGINE=MyISAM

create table nonPOIs(
                    nodeID VARCHAR(12),
                    position POINT NOT NULL, SPATIAL INDEX(position),
                    planaxy POINT NOT NULL, SPATIAL INDEX(planaxy),
                    otherInfo TEXT,
                    PRIMARY KEY(nodeID)
                ) ENGINE=MyISAM

create table WayNode(
                     wayID VARCHAR(12), INDEX(wayID),
                     nodeID VARCHAR(12), INDEX(nodeID),
                     node_order INT(2),
                     FOREIGN KEY (nodeID) REFERENCES nodes(nodeID),
                     FOREIGN KEY (wayID) REFERENCES ways(wayID)
                ) ENGINE=MyISAM
```

2.4. Data Insertion

For the data we parsed from XML, we inserted them into corresponding fields of our created tables.

Notably, if we insert the data directly into the table, the insertion time complexity would be $O(log(N))$, where N is the entries already existed in the table, due to the index (primary key) building process.

Therefore, in order to speed up the insertion process, we disable all the keys before the insertion, and enable them after the insertion. This will ensure every row is inserted in time complexity $O(N)$.

The SQL code is as follows:

```
              LOCK TABLE ‘nodes‘, ‘pois‘, ‘nonpois‘ WRITE;
```

```
ALTER TABLE ‘nodes‘ DISABLE KEYS;
ALTER TABLE ‘pois‘ DISABLE KEYS;
ALTER TABLE ‘nonpois‘ DISABLE KEYS;
/*...insertion...*/
ALTER TABLE ‘nodes‘ ENABLE KEYS;
ALTER TABLE ‘pois‘ ENABLE KEYS;
ALTER TABLE ‘nonpois‘ ENABLE KEYS;
UNLOCK TABLES;
```

The **LOCK TABLE** is to make sure no other users are writing at the same time.

2.5. *Index*

Besides index for primary keys, we built 8 indexes to accelerate the queries. Especially, in order to speed up the spatial queries, we applied Spatial Index in MySQL. For *MyISAM* tables, Spatial Index creates an R-tree index. The key idea of the R-tree is to group nearby objects and represent them with their minimum bounding rectangle in the next higher level of the tree. For storage engines that support non-spatial indexing of spatial columns, the engine creates a B-tree index. A B-tree index on spatial values is useful for exact-value lookups, but not for range scans. In our cases, the R-tree is more suitable because required query 4, 5, 6 all include range scans.

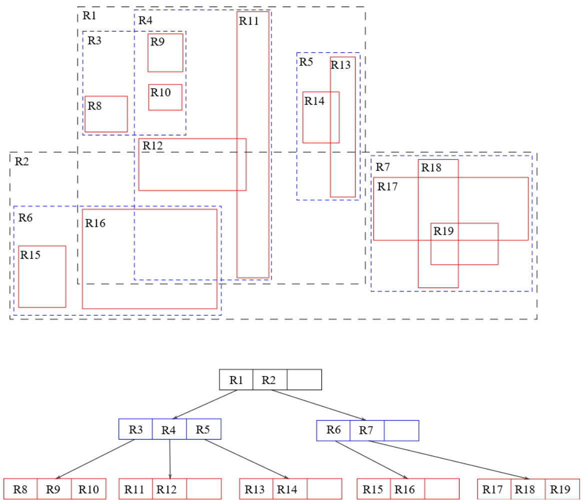

Figure 1: R-tree in 2 dimention

3. Point Mapping

The longtitude and latitude are used as the absolute coordinates. However, when calculating the distance between two points of given longtitude and latitude, we have to take spherical properties into consideration.

For instance, the distance between (30.4, 122.1) and (30.4, 122.6) is 48.0073 Km. The distance between (32.4, 122.1) and (32.4, 122.6) is 46.995 Km. There would be a error about 1 Km if we ignore the spherical properties of the Earth.

3.1. *Different Ways of Calculating Distances Between Two Given Coordinates*

a) Vincenty's formulae are two related iterative methods used in geodesy to calculate the distance between two points on the surface of a spheroid, developed by Thaddeus Vincenty (1975). They are based on the assumption that the figure of the Earth is an oblate spheroid, and hence are more accurate than methods that assume a spherical Earth. For simplicity and focus on the course related work, here we only provide the link to the Vincentys paper without further explanation. (`http://www.ngs.noaa.gov/PUBS_LIB/inverse.pdf`)

b) Another approach is to map latitude-longitude coordinates to plana coordinates and then calculate the distance in between. We used the definition of Millers cylindrical projection, which is more accurate near the equator. Further about the derivation please see the original paper. (`http://www.jstor.org/stable/210384`)

3.2. *Implementation*

We randomly sampled the start and the destination and got the distribution of the distance error derived using two methods. We concluded that the Vincenty distance, which is more accurate, was 0.66 1.15 times of the Miller distance. This is further explored in our Query 4 and Query 5 design.

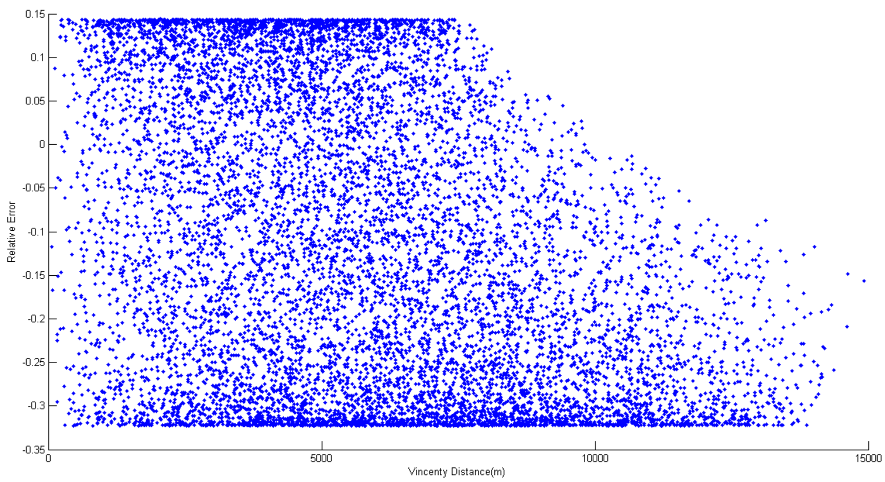

Figure 2: Relative error between miller distance and vincenty distance

4. Solution to Required Queries

1. Give a node, return all ways taht contain it, and infer whether the node is an intersection of roads, i.e., a crossroad.
2. Given a way, return all the nodes along the way.
3. Search the name of the road and return information of those matched.
4. Query the POIs within a radius of a given location (Longtitude-latitude coordinates).

5. Find the closest road to a given GPS coordinate.
6. Implement an API to return the XML in osm format defined in the wiki page, given a rectangular area bounding box (x1, y1, x2, y2) as parameters.

5. Extended Queries

6. Human Computer Interaction