CSCI 585 - Database Systems (Spring 2014)
Homework Assignment 2

Prof. Shawn Shamsian

Due: April 4, 11:59PM

The goal of this assignment is to develop a demo Java application that requires a spatial database as its back-end. In this assignment you will learn to: (1) design and query a spatial database; (2) programming with JDBC.

Project specification

The fire department of USC need an program for keeping track of all buildings, fire hydrants and the buildings that are on fire in USC. In this project, each fire hydrant is modeled as a spatial point, and each building is modeled as a polygon.

You are given 4 files for this project: *map.jpg*, *building.xy*, *firehydrant.xy* and *firebuilding.txt*. The first one is a 820×580 JPEG file that is the visualization of the map (Fig. 1). The other three files store the spatial data in the following format:

- *building.xy*. Each building is represented by a 2-dimensional polygon. Each line in this file represents a building and the meanings of the columns are: (1) Building ID; (2) Building name; (3) Number of vertices on the polygon (denoted n); (4) The following $2n$ columns are the coordinates of the vertices, respectively, where the x-coordinate and y-coordinate of each vertex is represented by to consecutive columns. For example, the line "b1, PHA, 4, 100, 120, 150, 130, 120, 200, 120, 220" represents a building with building ID "b1" and its name "PHA". It has 4 vertices whose coordinates are $(100, 120)$, $(150, 130)$, $(120, 200)$ and $(120, 220)$, respectively.
- *firehydrant.xy*. Each firehydrant is represented as a 2-dimensional point and each line represents a firehydrant. The columns are: (1) The firehydrant ID; (2) The x-coordinate of the firehydrant; (3) The y-coordinate of the firehydrant.
- *firebuilding.txt*. Each line contains the name of a building which is on fire.

Required SQL file(s)

You are required to submit two SQL files as follows:

- **(10pts)** *createdb.sql*. We will use this SQL file to create and populate the database that is used for this project on Oracle 11g. You should design the tables and assign datatypes to attributes such that the information of the buildings and firehydrants can be accessed and manipulated. It should include constraints, insertions and other necessary DDL statements. In addition, you must create spatial indices in the database that might be used for this project.
- **(10pts)** *dropdb.sql*. This file will be used to clean up all tables, functions, views and other objects that are created by *createdb.sql*.

Required Java source file(s)

You are required to submit a Java source file *HW2.java* which can be executed by the following command (assume the executable file that is generated by *HW2.java* is named *hw2*):

```
$java hw2 query_type [object_type other_parameters|demo_number]
```

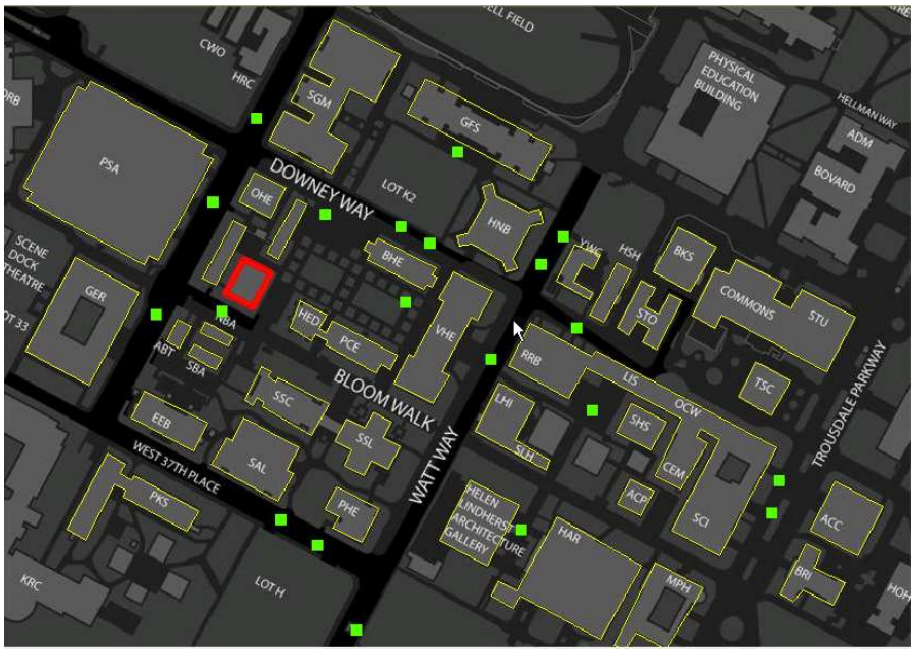

Figure 1: The map

The `Query type` parameter can only take one of the following values: `window`, `within`, `nn` or `demo`. In the former 3 cases, we need to specify `object_type` and `other parameters` as follows:

- **(10pts)** If `query_type = window`: we perform a window query in this case. That is, finding all objects that are completely inside the query window. The `object_type` parameter specifies what kind of objects we are looking for, which can only take one value from `building, firehydrant, firebuilding`, which stands for buildings that are not on fire, firehydrants and buildings that are on fire, respectively. `other_parameters` specifies the coordinates of the lower left and upper right vertices of query window, respectively. For example, the query

  ```
  $java hw2 window firebuilding 10 20 300 500
  ```

  lists the ID's of all buildings on fire that are completely inside the query window with lower left vertex $(10, 20)$ and upper right vertex $(300, 500)$.

- **(10pts)** If `query_type = within`: we do a within-distance query in this case. That is, finding all objects that are within the given distance to the given object(s). In this case, `object_type` is the same as above, and `other_parameters` specifies the name of the query building and the distance. For example, the query

  ```
  $java hw2 within firehydrant OHE 250
  ```

  lists the ID's of all firehydrants that are within distance 250 to all buildings named OHE.

- **(10pts)** If `query_type = nn`: we do a nearest neighbor query in this case. That is, find the nearest k objects to a given object. In this case, `object_type` is the same as above and `other_parameters` specifies the ID of a building and k, the number of nearest neighbors. For example, the query

  ```
  $java hw2 nn building b3 5
  ```

  lists the ID's of the 5 nearest buildings to b3.

- **(50pts/10pts each)** If `query_type = demo`: in this case we print the results of the following hard-coded demos on the screen. In this case the only supplementary input parameter is `demo_number`, which specifies the demo result to be displayed on the screen. For example,

  ```
  $java hw2 demo 3
  ```

  displays the result of demo 3 on the screen. The demos are as follows:

1. Find the names of the buildings with name initial 'S' that are NOT on fire.
2. For each building on fire, list its name and the ID's of the 5 nearest firehydrants.
3. We say a firehydrant *covers* a building if it is within distance 120 to that building. Find the ID's of the firehydrants that cover the most buildings.
4. We say a firehydrant is called a *reverse nearest neighbor* of a building if it is that building's nearest firehydrant. Find the ID's of the top 5 firehydrants that have the most reverse nearest neighbors together with their number of reverse nearest neighbors.
5. Find the coordinates of the lower left and upper right vertex of the MBR that fully contains all buildings whose names are of the form '%HE'. Note that you cannot manually figure out these buildings in your program.

Submission Guidelines

You must submit a compressed folder via DEN under the name $< your\_name > \_HW2.zip$ that contains the following files:

- **createdb.sql**
  This SQL file is used for creating tables, constraints, indices and inserting test data, as specified above.
- **dropdb.sql**
  This SQL file is used to drop all tables that are created by **createdb.sql**, as specified above.
- **HW2.java**
  This is the source of your Java program.
- **readme.txt**
  Include your full name, student ID, and email in the first three lines, and then describe how to compile and run your Java program.

* Feel free to use any network resources, make sure they are cited in your report.
* Feel free to discuss with each other, but please do acknowledge every partner in your report. But do not copy others' work. Violators will be reported to school.
* Please do not submit via e-mail.
* No late submission.