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

The goal of this assignment is to develop a 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:

USC Transportation Department wants to improve their transportation services for which they have selected certain buildings inside campus and some data of student coordinates. They have also kept track of tram stops on campus used by the students. The department is mulling over the idea of moving some tram stops so that they can cover maximum students and building.

You are given 6 files for this project: *map.jpg, building.txt, student.txt, tramstops.txt, sdoapi.zip and databaseDriver.jar*. 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.txt.* 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.

- *student.txt.* Each student is represented as a 2-dimensional point and each line represents a student. The columns are: (1) The student ID; (2) The x-coordinate of the student; (3) The y-coordinate of the student.

- *tramstops.txt*: Imagine a tram stop to be circular. Each tram stop is represented as a 2-dimensional point and each line in the file represents a tram station. The columns are: (1) tram station id; (2) The x-coordinate of the student; (3) The y-coordinate of the student; (4) radius it covers.

Assignment:

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 data types to attributes such that the information of the buildings and tramstops 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*.

Submit *Hw2.java* program which is required to support the following Command line Syntax:

$java hw2 query type [object type other parameters|fixed number]

The Query type parameter can only take one of the following values: window, within, nearest-neighbor or fixed. We need to specify object type and other parameters as follows:

1) (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, tramstops, student, which stands for campus buildings, tram stop and student, respectively. Other parameters specifies the coordinates of the lower left and upper right vertices of query window, respectively.

   Ex:
   Command:
   $java hw2 window student 100 100 300 300

   Output:
   Lists the ID's of all students on that are completely inside the query window with lower left vertex (100, 100) and upper right vertex (300, 300).

2) (10pts) If query type = within: we do a within-distance query in this case. That is, finding all the buildings and tram stops that are within the given distance to the given student id.

Ex:
Command:
$java hw2 within p1 300

Output:
lists the ids of building and tram stops that are within distance 300 to a p1 student id.

3) (10pts) if query type = nearest-neighbor: 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.

Ex:
Command:
$java hw2 nearest-neighbor building b3 5

Output:
Lists the ID's of the 5 nearest buildings to b3.

4) (50pts/10pts each) If query type = fixed: in this case we print the results of the following hard-coded fixed query on the screen. In this case the only supplementary input parameter is fixed number, which specifies the query number result to be displayed on the screen.

Ex:
Command:
$java hw2 fixed 3

Output:
Displays the result of fixed query 3 on the screen.

Following are the fixed queries:

1. Find the ids of all the students and buildings cover by tram stops: t2ohe and t6ssl.
2. For each student, list the ID's of the 2 nearest tram stops.
3. We say a tram stop covers a building if it is within distance 250 to that building. Find the ID's of the tram stop that cover the most buildings.
4. We say a student is called a reverse nearest neighbor of a building if it is that building's nearest student. Find the ID's of the top 5 students 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 'SS%'. Note that you cannot manually figure out these buildings in your program.

Submission Guidelines:

MOSS WILL BE USED FOR CHECKING CODE FROM ALL SEMESTERS.

- You must submit a compressed folder via D2L with name < your name > *HW2.zip* that contains the following files:
  1) *createdb.sql*: This SQL file is used for creating tables, constraints, indices and inserting test data, as specified above.
  2) *dropdb.sql* : This SQL file is used to drop all tables that are created by createdb.sql, as specified above.
  3) *HW2.java* : This is the source of your Java program.
  4) *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.

  Note: If not given in the above specified manner, there will be penalty of 10 points.
- 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.

Resources:

- **Oracle® Spatial Developer's Guide**
- Oracle Locator and Oracle Spatial 11g Best Practices

Steps to compile:

1) Set JAVA_HOME
2) Put sdoapi.zip, databaseDriver.jar and HW2.java into the folder containing yours Hw2.java.
3) Compile using
   javac –classpath . HW2.java
4) Run using : java –classpath . HW2