CSCI 585- Database Systems
Homework Assignment 2
Description

Note: Look at discussion board for clarifications
NOTE: MOSSWILL BE USED FOR CHECKING CODE FROM ALL SEMESTERS.

The goal of this assignment is to design an application that queries a spatial database. This assignment will make you familiar with spatial data types using Oracle11*g*, Oracle Spatial features, and Java (JDBC).
You are required to write two Java programs to 1) store and 2) query your spatial database.

Scenario:
In case of fire happening in the campus, we need a system to keep track of all the fire hydrants, buildings and the buildings that are on fire. Each fire hydrant is represented as a point and each building is represented as a polygon.

Input Files:
You will be given the following files:
1. Image file: MAP - an 820x580 JPEG file that is an image of some area of USC.
2. Following input files:

a). building.xy. Each building is represented by a 2D polygon. Col 1: building ID. Col 2: building name. Col3: number of vertices on the polygon. The numbers after column 3 are the coordinates of the vertices. They are ordered as , , , , ..., , . For example, a row: b1, PHA, 4, 100, 120, 150, 130, 120, 200, 120, 220 represents a building with its building ID as "b1" and its name as "PHA". It has 4 vertices whose coordinates are (100, 120), (150, 130), (120, 200) and (120, 220) respectively.

b). firehydrant.xy. Col 1: firehydrantID Col2: x coordinate of the firehydrant location. Col3: y coordinate of the firehydrant location.

c). firebuilding.txt. Col 1: firebuilding represents the building name, which is on fire.

Required .sql files:
You are required to create two .sql files:
1. createdb.sql: This file should create all required tables. In addition, it should include constraints, indexes, and any other DDL statements you might need for your application.
2. dropdb.sql: This file should drop all tables and the other objects once created by your createdb.sql file.

Required Java Programs:
You are required to implement two Java programs:
1. populate.java: This program should get the names of the input files as command line parameters and populate them into your database. It should be executed as:
"> java populate building.xy firehydrant.xy firebuilding.txt".
Note that every time you run this program, it should remove the previous data in your tables; otherwise the tables will have redundant data.
2. hw2.java: This program should provide a GUI, similar to figure 1, to query your database. The GUI should include:
a) An 820x580 panel that shows the map when the application is started up.
b) The tile of the main window should display your full name and your student ID.
c) Text field (or Label) that shows the coordinates (x, y) of the current mouse location as it moves over the image. *Please notice that the coordinates given in .xy files are based on the origin (0, 0) at the upper left corner of the image and (820, 580) at its lower right corner.*
d) 3 Check boxes that specify the feature types that we are currently interested in. Multiple feature types can be checked at the same time.
They are called active feature types.
e) 4 Radio buttons that specify the kind of query we are going to do.
There are 4 kinds of queries: Whole Region, Range Query, Find neighbor Buildings Query, and Find closest fire hydrants query. Details are given later.
Only one radio button can be checked at any moment.
f) One button to submit the required query.
g) One text field to display the SQL statements for the queries that has been submitted so far. Use incremental counter for the queries, and print the counter along with the SQL statement (e.g., "Query 1: select * from restaurants;", "Query 2: select * from people where …").

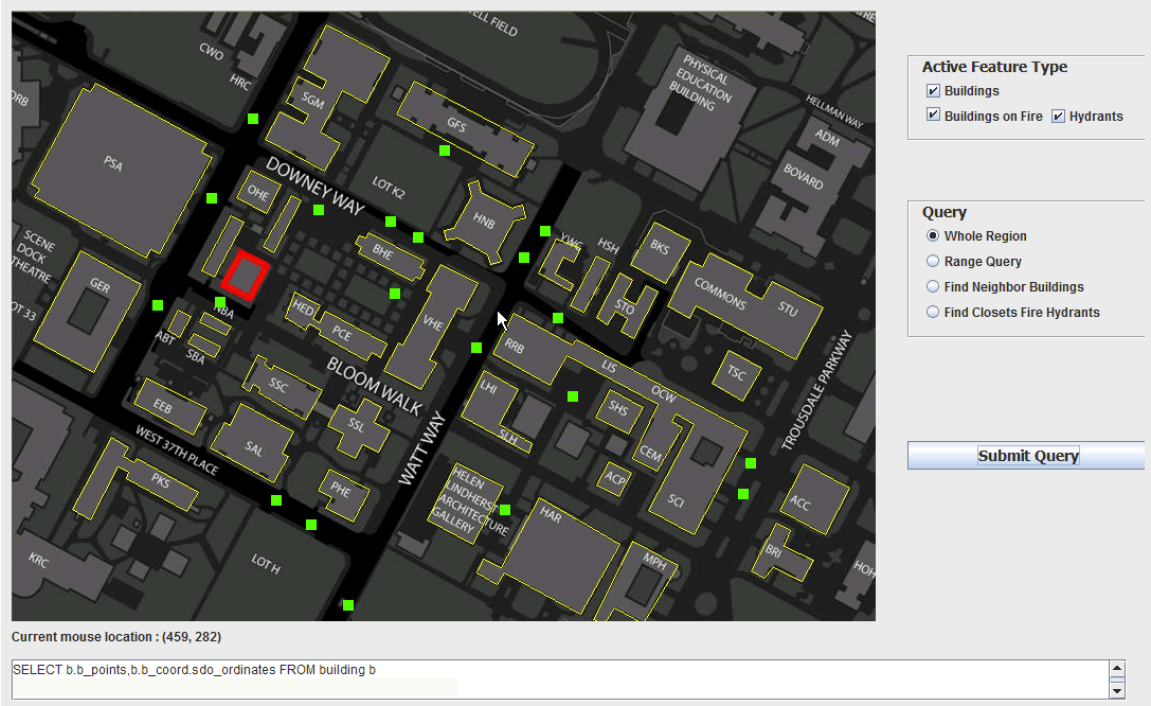

Figure 1: GUI demonstration

Queries:

1. Whole Region. This is to display all the features of the active feature types in the whole map. They should be displayed in the following way:

Table 1: Color Code for different types of features

Feature	Color	Shape
Buildings	Yellow	Polygon(outline, not solid region)
Fire Hydrant	Green	Square(15X15 pixels)
FireBuildings	Red	Polygon(outline, not solid region)

Graphical representation of Fire Hydrant, Buildings and Building on fire if checked should show up when we click the submit button.

2. Range Query. When this radio button is checked, the user can draw a polygon in the map. After pushing the Submit Query button, only the features of the active feature types that are inside (or intersect with) the polygons are displayed. These features should be displayed in the same way as specified in Table 1. The user draws the polygon by clicking the left mouse button to select its vertices sequentially and then clicking the right mouse button to close the polygon. Red line segments on the screen should connect the vertices as they are being selected. When the Range Query radio button is unchecked, the selected polygon should disappear.

3 Find neighbor Buildings. When this radio button is checked, the buildings on fire are displayed as Red polygons. You should also display the neighbor buildings in Yellow polygons. The neighbor buildings are the buildings that are located within 100 pixels. The people in those neighbor buildings also need to be evacuated to safe places.

4 Find Closest Fire Hydrants. When this radio button is checked, the user could use click to specify arbitrary number of buildings that are on fire. The buildings are shown as red polygons and the closest hydrants are displayed as Green Squares. If there are multiple buildings on fire, two buildings on fire might have the same closest fire hydrant. The closest fire hydrants shown on the GUI will provide the firefighter essential information to locate and secure the water sources.

Submission Guidelines

1. The links to the document for Oracle Spatial Reference are posted on the course website. The map image and the 3 input files are provided on the website.
2. Oracle JDBC Driver and Spatial Java APIs:
Oracle Spatial Java Library (ojdbc6.jar), is posted with this description. It is required to manipulate spatial objects of Oracle11g (or higher) in Java program.
You can compile your source as:
$ javac –classpath /path/to/your/ojdbc6.jar hw2.java
You run your application as:
$ java –classpath /path/to/your/ojdbc6.jar hw2
3. You need to have a readme.txt file that should include your name, student id, your user name on aludra.usc.edu, the list of the submitted files, resolution of your homework and how to compile/run them. There are 25 points penalty if this file or some of the required information is missing from your submission.
4. For the second Java program (i.e., hw2.java), you may develop your assignment using more than one Java program. It is recommended (but not required) to separate the GUI codes and database related codes into different files.
5. You must make a xxx.tar (or zip) file to include all of your files in one file (e.g., hw2.tar or zip) and the compressed file includes **hw2.java createdb.sql dropdb.sql readme.txt or**
Do NOT include the .class files, input files, or ojdbc6.jar in your .tar file.
We will compile your .java files.
6. You need to submit the assignment electronically to dropbox in den.usc.edu. Please make sure you pushed submit button.

7. You can write your Java programs on any machine you wish. You can use any Java Visual Programs (e.g., JBuilder, Visual Café, Visual J++) you wish to design your GUI, but make sure you can run your program from university computer which does not requires specific link or classpath before submitting it. Again please do not put absolute path for the linking and include files for grading. Otherwise it will be penalized.
8. Start working on your assignment early.
9. Grading guideline:

Points	
5	Creating/Dropping database tables
10	Populating database
20	GUI containing all of requirements mentioned for user interfaces. Hint: implement DB connectivity and queries first. If time left, move to GUI construction
10	Whole region
10	Range query
20	Find neighbor buildings
15	Find the closest hydrants

10. Again, please start early. Try to submit early. Any submission related issues should be tested/resolved before your submission. Do not try to submit the code right before the due. No late submissions will be accepted in any reason.