CSCI 585- Database Systems
Spring 2010
Homework Assignment 2
Due: 04/12/2010 @ 8:00pm
Description

As mentioned in Homework 1, Photogeek is an online photo sharing website which provides users with different functionalities and features. In this assignment, we are using two datasets. One is a set of geo-tagged photos of restaurants from Photogeek, and the other is a database of restaurants in Los Angeles. The goal of this assignment is to design an application that queries these two datasets. This assignment will make you familiar with the design and implementation of a real-world application (of spatial database), using Oracle10*g*, Oracle Spatial features, and Java (JDBC).

You are required to write two Java programs to 1) store and 2) query your spatial database.

Input Files:

You will be given four files:
1. Image file: la.jpg, a 600x500 JPEG file that is a map of part of Los Angeles.
2. restaurants.txt which contains the information of all restaurants.
3. photos.zip: contains a set of photos of restaurants.
4. photos.txt which contains the details of each photo in photos.zip.

You need to read three input files (items 2, 4, and 5) as follow:

a) Read the files line by line.
b) Each line represents a tuple, containing different columns separated by comma. (You can use StringTokenizer to extract the columns of each tuple.)
c) Item 2 (restaurants.txt) has 6 columns in each line which are ID, name, X & Y that represent the restaurant location, food type, and phone number.
d) Item 4 (photos.txt) has 5 columns in each line which are ID, name, X & Y that represent the photo location, and tags (separated by ;).
For the tags of a photo, the first tag is the name of the restaurant.

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 name of the input files as command line parameters and populate them into your database. It should be executed as: "> java populate restaurants.txt photos.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 the figure 1, to query your database. The GUI should include:
   a) A 600x500 (width x height) panel that shows the given map.
   b) A text field that shows the coordinates (X, Y) of the current mouse location as it moves over the image.
      Please notice that the coordinates given to you in .txt files are based on the origin (0,0) at the upper left corner of the image and (600,500) at its lower right corner.
   c) 2 Check boxes that specify the features we are interested in.
   d) 4 Radio buttons that specify the type of query we want to ask.
   e) A text field to give input to the program.
   f) A panel to show a photo when clicked on a photo in the map.
   g) A text field to show the restaurant information when clicked on a restaurant in the map
   h) A Submit button for submitting the query.
   i) A Clear button to clear everything on the map for doing a new query.
   j) A text field to show submitted SQL queries to the database.

Description of GUI :

The GUI should show the given map when the application is started up. The title of your main window should display your full name and your student ID.

a) Active Features: multiple features can be checked and those checked features will be retrieved and shown on the map. The following table shows how each point must be displayed.

Feature	Color	Shape
Restaurants	Red	Circle
Photos	Yellow	Triangle

b) In order to draw a box as an active region for range query, "Range" radio button should first be selected, then the first click (left mouse button) specifies the lower left corner and the next click specifies the upper right corner of the box. The box must be displayed as follows:

Selected Box	Blue	Show only 2 opposite corners specified by the left mouse clicks with blue crosses and 4 edges connecting the corners with blue lines.

Figure 1 shows an example of a rectangle as an active region.

c) Whenever your program is submitting a query to the database, the SQL statement for that query should be printed in a text field on the GUI (the text field at the bottom of Figure 1).
If you need to send more than one query (e.g., when more than one feature is selected), use an incremental counter for the queries, and print the counter along with the SQL statement (e.g., "Query 1: select * from restaurants where..;", "Query 2: select * from photos where …").

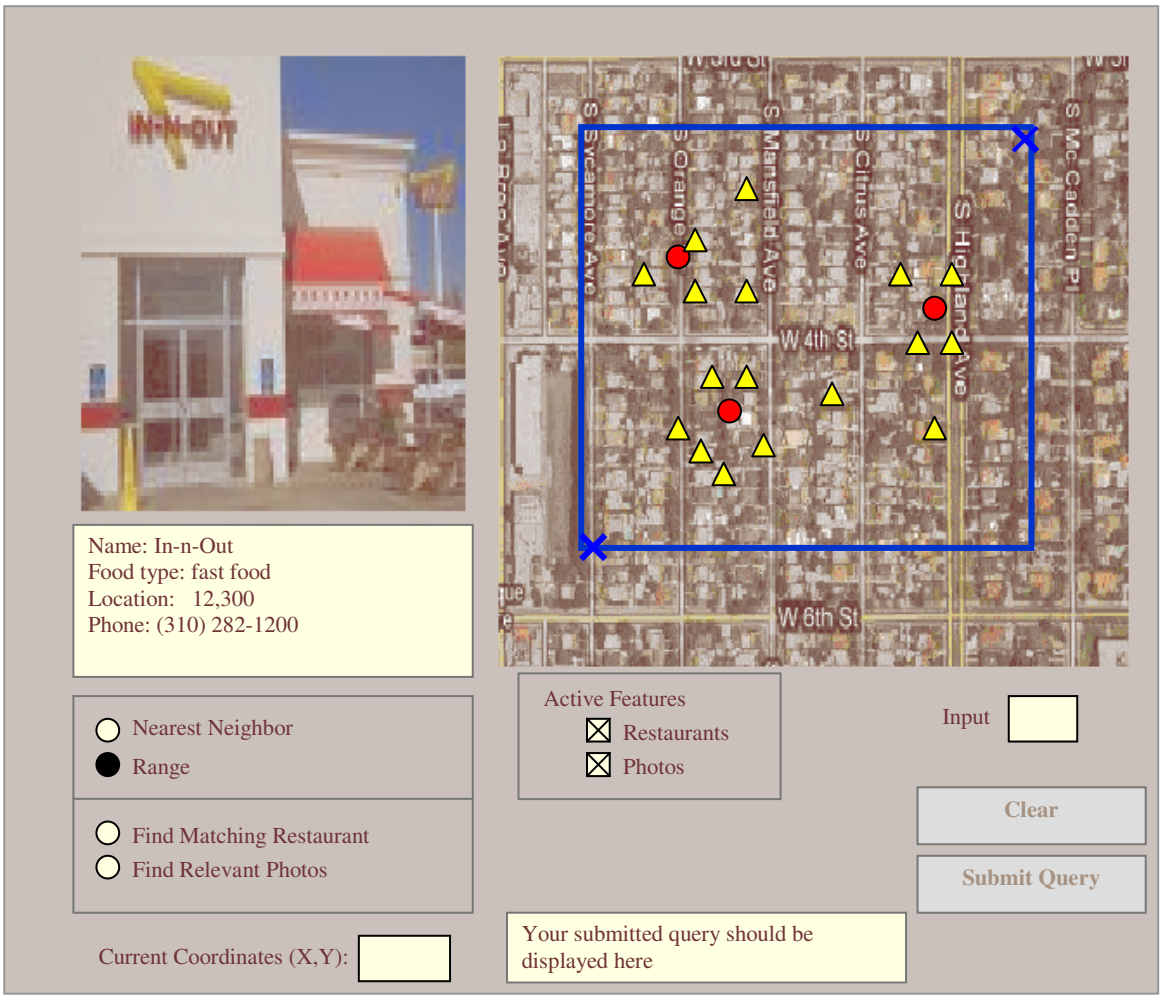

Figure 1 – Graphical User Interface of your application

Queries:

We will examine your homework as following:

1. ***For Nearest Neighbor query:***

    a. Suppose we're in a location, and we want to find our closest restaurant, or we want to see the photos that were taken close to our current location. To do so, we select the "Nearest Neighbor" radio button. We then use the right mouse button to specify a location on the map as the "origin point". Your program should display this point with a cross in red color (make the cross visible). Next, we select one or both of the features, (i.e., photos OR/AND restaurants). The input text field is used to input the k (number

of nearest objects we want to retrieve). For example, for k=2, your program should retrieve 2 objects per feature which are closest to the origin point. Finally, we click on the "Submit Query" button. Your program should submit separate queries for each feature. Once the objects are retrieved, clicking on each object should show its detailed information on the left side of the GUI. For restaurants, name, xy location, food type and phone will be displayed in the left text box. For photos, the photo should be displayed in the left panel.

Example: We first select "restaurants" and "photos" as the active features, and the "Nearest Neighbor" radio button as the query. We then select the origin point by a click on the right mouse button inside the map. Your program should draw the specified origin point. We also input k=1. Finally, we click on "Submit Query" button. Your program should find and display the nearest restaurant and nearest photo to the origin. By clicking on the restaurant icon, its info should be displayed in the left side. Similarly, by clicking on the photo icon, the photo should be displayed in the left panel.

2. ***For Range query:***

Assume we are interested in finding all the restaurants/ photos in a range. By selecting this radio button, the region selector gets activated. We then specify the corners of the box by clicking left mouse button on the map twice (your program should show a cross for each clicked corner), at this point your program should draw the box, then we select one or both of the features, and then click on the "Submit Query" button. Your program should submit separate queries for each feature. Each query returns all objects that are inside (or intersect with) the selected box. Your program should display these objects on the map. Similar to the previous query, clicking on each feature should display its info in the left side.
Figure 1. shows an example of range query asking for restaurants, and photos inside the box.

3. ***Find Matching Restaurant***

Suppose in the previous queries (1 or 2), we click on a photo, which is a photo of a restaurant, and we want to get some more details about that restaurant (e.g., name, phone,…). We select the "Find Matching

Restaurant" radio button. We then press the "Submit Query" button. Your program should use the photo's first tag (i.e., restaurant name) to find all the matching restaurants and rank them based on their (Euclidean) distance to the photo's location. The resulting restaurants should be drawn as green circles on the map. Also, a ranking number should be assigned to each restaurant, and displayed next to the icon. By clicking on the restaurant icon, we should see its information on the left text box.

4. *Find Relevant Photos*

Suppose in the previous queries (1 or 2), we click on a restaurant icon, and see the details of the restaurant in the left text box. We now want to find the k closest photos to the restaurant which have the same food type in their tags. We enter the number k in the input text field. We select the "Find Relevant Photos" radio button. We then press the "Submit Query" button. The resulting photos should appear on the map (No ranking is needed). By clicking on each, we will be able to see the photo on the left panel.

Submission Guidelines

1. **Oracle JDBC Driver and Spatial Java APIs:**
You can download both classes12.jar and Oracle Spatial Java Library (sdoapi.zip) to your working directory from the class web page. It is required to manipulate spatial objects of Oracle10g in Java program. Oracle Spatial Java API document is also available on the class web page. You can compile your source as:

```
javac -classpath classes12.jar;sdoapi.zip;. hw2.java
```

You run your application as:

```
java  -classpath classes12.jar;sdoapi.zip;. hw2
```

2. You need to have a readme.txt file that should include your name, student id, your Oracle user name, the list of the submitted files, and how to compile/run them. There is 25 points penalty if this file or some of the required information is missing from your submission.

3. 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.

4. Make a tar/zip file to include all of your files in one file. You must only send one file (either tar or zip), which includes all your files.

5. You need to submit your assignment electronically using the digital drop box on the DEN website. To do so, you login to DEN, and go to the course page. Click on the Tools option on the left panel and then click on the Digital Dropbox. There you can attach/send your file with the name "HW2-LastName-firstName". Make sure you see the confirmation that your homework has been successfully sent.

6. You need to develop your database tables in your machine using Oracle Client, and use JDBC connectivity to connect to your database, with the username/password you received for homework 1. You can see the examples on how to connect to database on the class webpage.

7. You can write your Java programs on any machine you wish. You can use any Java Visual Programs (e.g., netbeans, eclipse, JBuilder, Visual Café, Visual J++) you wish to design your GUI.

8. Start working on your assignment early.

Distribution of points:

Points	Contents	Comments
5	Creating/Dropping database tables.	If either of them does not work properly, you will probably lose points for other parts associated as well.
10	Populating database.	
15	GUI containing all of the requirements mentioned for user interfaces.	Hint: implement DB connectivity & queries first. If time left, move to GUI construction.
15	Nearest Neighbor Query	Database connectivity code will be counted here as well.
15	Range Query	
20	Find Matching Restaurants	
20	Find Relevant Photos	
100		