Lab 8: Mini Project

Due date: Friday, December 9, midnight.

Assignment Preparation

This is the final major lab for the course. There may be one more mini-lab during the last two weeks of classes ("mini" meaning the lab will be assigned to be completed over the course of a single lab session).

This is a team lab. The three-person teams were either self-formed by you, or, for those who failed to form a team, were selected by the instructor.

Lab Assignment

The goal of this lab assignment is to **build a working database application.** It was traditional for us in the past to make everyone implement the same application, but this time around, we will do it differently.

Each team gets to come up with its own **database-driven application**, create the appropriate database, collect data for it, and develop the software to manipulate and query the data in the database.

The rest of this documents describes the rules of engagement.

Application. The goal of this project is to construct a complete database-driven application, and demonstrate its work. By database-driven application we mean the following:

- The software you build (the application) needs to provide useful information and/or services to its users. See below for the minimal list of features the software should have.
- The software shall be implemented in a host programming language (e.g., Java).

- The software should rely in large part on the data stored in a relational database. In fact, without the data stored in the database, the software should pretty much be meaningless. That is to say, the intent of your software shall be *data-centric*.

Database Requirements. Each team develops its own database to support the application it is building. The following requirements on the database are designed to allow you to create and maintain non-trivial databases.

1. **DB1.** The database shall contain information about at least three different categories of objects/notions (entities), together with all relevant relationships between those entities.
2. **DB2.** You shall create the database and populate it with a significant collection of data prior to submitting/demonstrating your application to the instructor. In general, to be significant, your data collection must either be (a) on the order of hundreds of objects/entities, and hundreds-to-thousands relationships between the objects, or (b) be complete, i.e., contain all the data *that specific database* can possibly contain[1].
3. **DB3.** You shall create a data collection to represent the *initial application state*, that is, the set of data that is the content of your application's database when the application is cold-started. (Essentially, I do not want to worry about populating tables when working with your application, I want it to load a reasonable database state so that I could proceed working with it).

It is left to each team to design the database schema and convert the raw data (from wherever it is coming) into the database tuples.

Software Requirements. In developing your application, please make certain the following requirements are satisfied.

1. **SR1.** Upon startup the software shall connect to the database. The database shall contain *at least* the initial application state. Further interaction with the database may, in some cases, modify that state. If one exists the program and restarts it, the new state shall persist. To make testing simpler, the software should implement a "reset" functionality, which restores the database to its *initial application state*.
2. **SR2.** Under most normal circumstances, your application shall, at a minimum, implement functionality that allows the user to *modify*

[1] As the example of the database that follows part (a) of this requirement, we can view `CSU` or `BAKERY`. As the example of the database that follows part (b) of this requirement we can view `KATZENJAMMER` — the database represents a (nearly) complete set of songs and song performances.

the database state, and to retrieve data from the current database state. In some special cases, permission may be granted (if appropriate thematically) to not implement insertion/modification functionality (for example if the application is a data warehouse where there is no such notion as modifying or inserting a single tuple).

3. **SR3.** Database state modification can involve creation of new data entries in the database (tuples in one or more tables), removal of data entries from the database, and an in-place update of some values in data entries in the database. If your application warrants any or all of these three types of database modification, your application shall include functionality to perform them.

4. **SR4.** Your application shall provide functionality for browsing the information stored in the database.

5. **SR5.** Your application shall provide some advanced querying facilities.

The requirements above are sufficiently vague, because the range of possible applications you can choose to build is large. Below are some comments about the different aspects of the lab.

What data? There are different ways in which you can approach creating the database itself. You can collect data yourselves and turn it into a database. You can find an existing dataset and convert it into a database. You can scrape the data from a web site and convert it into a database. You can choose a dataset for which you synthesize the dataset. The specific nature of your data and the means for collecting it are left to individual teams.

What application? Most database-driven applications are broken into CRUD (a.k.a., CReate, Update, Delete) applications, and analytical applications.

CRUD applications are designed to ingest data into a database. This can be done either automatically or interactively. Your application will have some CRUD code to make sure that you can load the initial dataset. IT can also contain CRUD code for insertion, deletion or modification of individual tuples.

Analytical applications usually stay away from modifying data, and instead present data stored in the database to the use in a variety of convenient and understandable ways. Your application may either be primarily analytical, or have an analytical component. You shall implement multiple queries to the database and display their answers in the reasonable ways in your application.

UI? Generally speaking, yes. You want to build a standalone application that is easy for the users to comminucate with. Therefore, the use of graphical UI is more than warranted. If you do NOT want to implement UI in your project, talk to me ASAP. I will determine how we can address all non-standard cases.

Submission Instructions

The first thing each team prepares is a short proposal. Your proposal is a text document called `README.proposal.txt`. The document shall contain the following information:

1. Name of your team
2. Names and email addresses of all team members
3. Name of your application/software
4. Short description of what your software is designed to do
5. Short description of the data your application is using, and the details database schema (this can be done by simply pasting your CREATE TABLE statements into the file). Discussion of the origin of your dataset (i.e., where is the data coming from).

The proposal shall be submitted by the end of the day **Monday, November 28** via the following command:

```
$ handin dekhtyar lab08 README.proposal.txt
```

The completed project is due midnight, **Firday, December 9**. Submit your code, data and documentation using the following command:

```
$ handin dekhtyar lab08 <files>
```

Documentation. You must submit a **README.txt** file with your final submission containing your team name, individual names of students on the team, and the instructions for how to compile and/or run your program on the CSL system.

Database Connectivity. Your software will use a `"dbconf.365"` file to obtain information about the database connection. The `"dbconf.365"` files shall have the following format:

```
host = <server>
userid = <userid>
password = <password>
```

Here, `<server>` if the name of the DB server on which your database resides (cslvm74.csc.calpoly.edu, in our example), `<userid>` is the name of a user connecting to the database, and `<password>` is the password. If you want to password-protect your work, feel free to not include password into the configuration file, and simply provide functionality for entering the password at the beginning of the work with your program.