CS166: Project Phase 3

February 24, 2023

Introduction

In this phase you are provided a package including the SQL schema, datasets, and a template user interface in Java. The datasets consist of data records that should be loaded into your database system. The template user interface in Java shows you how to connect to the PostgreSQL database and how to execute SQL queries.

Additional extra points will be awarded to projects that use GUI interfaces and properly handle exceptional situations by providing user-friendly error messages.

Please follow the steps below to get started:

1. Your database should have been created in Lab 6 (Java and SQL). The first line is to stop the current PostgreSQL server if you forgot to stop the server from the last session. The second line is to start the server. The third line is to create the database.

   ```
   source ./lab5/stopPostgreDB.sh
   source ./lab5/startPostgreSQL.sh
   source ./lab5/createPostgreDB.sh
   ```

2. Download project.zip from elearn (Canvas).

3. In the download directory execute the following command:

   ```
   unzip project.zip
   ```

   You will see that a directory named “project” will be created. This directory contains all necessary files to get started. More specifically it contains the following:

   - **project/data** - contains the files, which will be used to populate your database

- **project/sql/src/create_tables.sql** - SQL script creating the database relational schema. It also includes the commands to drop these tables. **Some tables have been created in slightly different way from the Phase-2 ER Diagram for simplicity. Check this .sql file before starting working on the project.**
- **project/sql/src/create_indexes.sql** - SQL script that creates database indexes. Initially it is empty; you should add all your indexes to this file.
- **project/sql/src/load_data.sql** - SQL script for loading the data in your tables. The script loads each text file into the appropriate table. **Note that the file paths have to be changed to absolute paths in order to make it work**.
- **project/sql/scripts/create_db.sh** - shell script, which you should use to setup your database.
- **project/java/src/Hotel.java** - A basic java User Interface to your Postgres database. You should modify this program and write all your SQL-specific code there.
- **project/java/scripts/compile.sh** - compiles & runs your java code.

4. Change path to data files in **project/sql/src/load_data.sql**. Use absolute paths to avoid ambiguity. After that your load statements should look like this:

```
COPY USER_LIST
FROM '/home/user/project/data/usr_list.csv'
WITH DELIMITER ';';
```

5. Execute **project/sql/scripts/create_db.sh** to load your database

6. Execute **project/java/scripts/compile.sh** to compile and run your Java client.

Java console application

If this is the first time you work with Java there is no need to be worried. You are provided with a template client written in Java. You are expected to extend this basic client and add the functionality, required for a complete system. An Introduction to Java/JDBC can also be found in your Textbook (Sections 6.2 – 6.3 of Database Management Systems (Third edition)).

In this phase we basically want to create an interactive console for non-sql users (i.e. Customers and Managers of the Hotels). You should therefore pay special attention to making the interface as intuitive as possible for a regular user.

In order to run the template Java program you should first start the Postgres database as it was described in Lab 6. Then you should execute the shell script *project/java/scripts/compile.sh*

The script will compile and run Java source for the file *project/java/src/Hotel.java*. If you will add other Java source files to your project, please modify the script accordingly.

Functionality of the Online Hotel Management System

Below you will find the basic requirements for the functionality of the system. On top of them you may implement as many additional features as you want. Throughout the project you should keep in mind that the users are not SQL-aware. You should therefore make the user interface as intuitive as possible.

Below we provide the basic operations you must implement in your system.

1. Users
    - New User Registration: when users come to the system, they can setup a new account through your interface, by providing the necessary information.
    - User Login/Logout: users will use email and password to login into the system. Once a user logs in, a session will be maintained until logout (an example has been provided in the Java source code).
    - Browse Hotels: Allows the user to see the list of hotels within 30 units distance of the user's given input location (latitude and longitude). Thus, your interface should ask the user to input a latitude and a longitude value if this option is chosen. (**Note**: We have provided a user-defined SQL function calculateDistance() in **project/sql/src/create_tables.sql** file, which takes two latitude, longitude pairs and returns the euclidean distance between them and can be invoked from your SQL statements. We avoided original latitude, longitude distance calculation (haversine distance) for simplicity.)

- Browse Rooms: Allows the user to input a hotelID, and a date and it returns the list of rooms along with the price and the availability of the rooms on the given date.
- Book Room: This option will ask the user to input a hotelID, room number, and date. If the given room is available on that date, the user can book that room. Otherwise, the system should show some meaningful message mentioning the unavailability of the room. If booking is successful, the system should show the room price to the customer. Also the RoomBookings table will need to be updated accordingly if a booking is successful. [**Note**: You do not need to consider the days of stay. We are skipping days of stay for the simplicity.]
- Update Room Information: For Managers, they can update the information of any room. This option will ask from the manager to input a hotel ID, and a room number. Managers can only update the room information (price, image url) of the hotels they manage. The Rooms and RoomUpdatesLog tables will need to be updated accordingly if any updates take place. Managers can also view the information of the last 5 recent updates of their hotels.

2. RoomBookings
    - Browse booking history: Customers will be able to see the last 5 of their recent bookings from the RoomBookings table. They will be able to see hotelID, roomNumber, billing information, and date of booking. A customer is not allowed to see the booking history of other customers.
    - Managers can see all the booking information of the hotel(s) they manage. They will be able to see bookingID, customer name, hotelID, roomNumber, and date of booking for each booking. They will have the option to input a range of date and the system will show all the booking in that range.
    - Regular customer: This option is for the managers. If chosen, the system will ask for entering a hotelID. If the manager is managing that hotel, then the top 5 customers who made the most bookings in that hotel will be shown.
3. RoomRepairRequests
    - Place Room Repair Request: Managers can place room repair request for any room of their hotel(s). For that, they will need to

input hotelID, roomNumber, and companyID of the maintenance company which will handle the repair request. After placing the request, RoomRepairs and RoomRepairRequests table should be updated accordingly. Managers will have the option see all the room requests history (companyID, hotelID, roomNumber, repairDate) for the hotels they manage.

Extra Credit

1. Good User Interface.

   A good user interface will bring more users to your application, and also more points in this phase. We consider a good user interface one that is:

   - Easy for users to explore features;
   - Robust in exceptional situations, like unexpected inputs;
   - Has a graphic interface supporting all required features.

2. Triggers and Stored Procedures.

   Instead of processing the workflow step by step, triggers and stored procedures can be used to handle a sequence of SQL commands and to execute these sequences on specific table events (inserts, updates, etc). For example, you can write a trigger which will be adding the order information in the orders table after a customer places an order. Or you can write a trigger to add product update information on the ProductUpdates table when a manager updates a product. You can also find other opportunities for triggers and stored procedures.

   To submit your triggers and stored procedures with your project, please include them in the following location: *project/sql/src/triggers.sql*

3. Performance Tuning

   The performance can be improved if indexes are used properly. **You will receive points only if the index will actually be used in your queries**. You should defend why you have chosen to use an index at some particular table. For your submission, you should put index declarations in *project/sql/src/create_indexes.sql*

4. Any Other Fancy Stuff...

   Please feel free to include any of your ideas to improve your project! Make sure you documented them clearly!

Submission Guidelines

1. Project Report

   You should provide a high level description (1-2 pages) of your implementation. You should describe which part each person in your group was working on and any problems/findings, you found along the way.

2. Files

   The following files should be submitted:

   (a) Create Tables, Bulkload Data Scripts.

   If you have any schema or tables modifications you should include your changes into the files and leave necessary comments for them. Modify the following files: *project/sql/src/create_tables.sql*, *project/sql/src/load_data.sql*

   (b) System Implementation. Submit your source file(s). You should make sure that your code can be compiled and run successfully. **Any special requirement for compiling and running should be stated clearly in your project report which comes with your source code**. Please put all your source code within the *project/java/src* directory.

   (c) Other scripts, like triggers, stored procedures, and indexes.

   You should provide descriptions in your project report for these features and include all your scripts within the *project/sql/src* directory.

   You should submit a single zip archive (**1 submission per group!**) via elearn (Canvas) within the due date. The due date for this phase is **Monday, March 20, 2023, at 11:59PM.**

   NB: There will be demonstration sessions for each project group on **March 22-24, 2023**. We will post the schedule for the demo sessions later, and you can choose your slot from there. **Both of the group members must be present during their scheduled demo session.**