CS 166 Phase 3 Requirements

Introduction

In this phase you are provided a package including SQL schema, dataset, and a template user interface in Java. The dataset consists 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. The due date for this phase is **11:59pm, Wednesday, December 11, 2024**.

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 below steps to get started:

1. Download and unzip the folder cs166_project_phase3.zip
2. Use the scp command below to copy it to the server; make sure to replace the red text.

****You should not ssh into the cs166 server for this command.****

```
scp -r path/to/unzipped/folder <netid>@cs166.cs.ucr.edu:
```

For example:

```
rahulyenduri@RAHULs-MacBook-Air-2 ~ % scp -r Downloads/cs166_project_phase3 ryend001@cs166.cs.ucr.edu:
```

3. Log in to the cs166 server via SSH. You should see the directory cs166_project_phase3. This directory contains all the necessary files to get started. More specifically, it contains the following:
    - **cs166_project_phase3/data** - contains the data files which will be used to populate your database
    - **cs166_project_phase3/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 Phase 2 ER Diagram for simplicity. Check this .sql file before starting working on the project.
    - **cs166_project_phase3/sql/src/create_indexes.sql** - SQL script to create indexes. Initially empty, you should add all your indexes to this file.
    - **cs166_project_phase3/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 (instructions below).**
    - **cs166_project_phase3/sql/scripts/create_db.sh** - shell script, which you should use to setup your database.

- **cs166_project_phase3/java/src/PizzaStore.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 and runs your Java code.

4. Execute the following commands. The first command is to stop the database if you forgot to in the last session.

   `cs166_db_stop`

   `cs166_db_start`

   `cs166_db_status`

   Do not forget to replace <netid> with your actual netid

   `cs166_createdb <netid>_project_phase_3_DB`

5. Navigate to your project folder, then navigate to the data folder.

   `cd cs166_project_phase_3`

   `cd data`

6. Execute the following command, and copy the path that this command outputs. This is your absolute path to the data folder. Copy this path.

   `pwd`

7. Navigate back to your project folder.

   `cd ~/cs166_project_phase_3`

8. Modify each path inside the load_data.sql file to have the appropriate paths using the absolute path you retrieved from `pwd`.

   `vi sql/src/load_data.sql`

   Once you are done adding the absolute paths, write and quit out of the file. The file should look something like this:

```
COPY TrackingInfo
FROM '/home/csgrads/ryend001/cs166_project_phase3/data/trackinginfo.csv'
WITH DELIMITER ',' CSV HEADER;
```

9. Make sure you are still in cs166_project_phase3 folder. Execute the following shell script to load your project database.

   `source sql/scripts/create_db.sh`

10. Execute the following command to compile and run your Java code. We recommend using this to test your code after you finish each query.

    `source java/scripts/compile.sh`

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, Drivers, and Managers of the Pizza Store). You should therefore pay special attention to making the interface as intuitive as possible for regular user.

In order to run the template Java program you should follow the instructions above to start the Postgres database and execute shell script project/java/scripts/compile.sh.

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

Functionality of the Pizza Store

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. We initially planned to provide functionality in the following three groups:

Important: At least 1 index should be used to get credit for the indexing portion of the Phase 3 grade

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

1. Users/Menu

- **New User Registration**: when a new user comes to the system, he/she can setup a new account through your interface, by providing necessary information. The user will automatically be a customer. Their favorite item will be empty.
- **User Login/Logout**: user can use his/her login and password to login into the system. Once he/she logs in, a session will be maintained for him/her until logout (an example has been provided in the Java source code).
- **Browse Menu:** allows the user to view all the items on the menu. User should be able to filter their search based on type and price (they can search for food items under a certain type (such as “drinks” or “sides”), or search for food items under a certain price). User should also be able to sort the menu based on price from highest to lowest price and from lowest to highest price.
- **Profile**: Users should be able to view and update their favoriteItem. Users should also be

able to view their phoneNum. Users should be able to change their password & phoneNum. **Only managers can edit a user's login and role.**

- **Place Order**: user can order any item from the menu. User should first be asked which store they want to order from. User will be asked to input every itemName and quantity (the amount of each item they want) for each item they want to order. The total price of their order should be returned and output to the user. After placing the order, the order information needs to be inserted in the FoodOrder table with a unique orderID (and make sure you include the store they ordered at). Each itemName, orderID, and the corresponding quantity should be inserted into the ItemsInOrder table for every item in the order.
- **Update Food Item Information**: For **Managers**, they can update the information of any item in the menu given the itemName. **They should also be able to add new items.**
- **Managers**: Managers will be able view and update the information of all users (as well as change their role) and update menu information.
- **Drivers**: Drivers should be able to update the order status field of any given order. Managers should be able to do this as well.

2. Food Orders

- **See OrderID History**: Customers will be able to see **their** order history. They should be able to see a list of all their past orderIDs. **A <u>customer</u> is not allowed to see the order history of other customers. Managers & Drivers are allowed to see all orders from anyone.**
- **See Recent 5 OrderIDs**: Similar to the last section, customers can see their order history, but limit the output to the 5 most recent orders.
- **Lookup Specific Order**: Customers will be able to view details about a specific order. They should be able to see their orderTimestamp, totalPrice, orderStatus, and list of items in that order (along with the quantity). **A <u>customer</u> is not allowed to see the order information of orders that belong to other customers. Managers & Drivers are allowed to see all orders from anyone.**

3. Store

- **View Stores**: Customers should be able to view the list of all stores. They should see all information about the location of the store, the storeID, the review score, and whether or not it is open.

Extra Credit

1. Good User Interface

A good user interface will bring more users to your application, and also more points in this phase. A user interface is good if it is:

- Easy for users to explore features;
- Robust in exceptional situations, like unexpected inputs;
- 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 command and to execute these sequences on some table events (inserts, updates, etc). 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 show any of your ideas to improve your project! The only thing required will be clear documentation in your report!

Submission Guideline

1. Project Report: Write a project report using the following template: Project Report Template
2. Files - the following files should be submitted:
    a. Create Tables, Bulkload Data Scripts.
        If you have any schema or table modifications you should include your changes into the files and leave necessary comments for them.
    b. System Implementation
        Submit your source file(s). You should make sure that your code can be compiled and run successfully. **Any special requirement(s) for compiling and running should be stated clearly in your project report, or a README file 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 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 **11:59pm, Wednesday, December 11, 2024**.

There will be a **demonstration session** for each project group at a soon-to-be-determined date. We will post the schedule for the demos later, and you will be able to choose your slot. **Both of the group members must be present during the scheduled demo session.**