CS 336 -- Principles of Information and Data Management

Requirements Specification for the Database Programming Project

Introduction

In this project, you are asked to implement a Bar-Beer-Drinker PLUS database system.

More specifically you will use HTML and Javascript for the client side, Java for the server side, MySQL for the database server, and JDBC for connectivity between your application and the database server. (In case you are familiar with any other web development technology, you are free to use it.)

You will have to install your own web server that will host your web application as well as a MySQL server locally on your computer. More resources will be provided about how to do everything so don't worry. J

Bar-Beer-Drinker PLUS project

Individual project

We are extending the bar-beer-drinker schema, with information about specific transactions by customers (drinkers). We are now storing all "bills" of all drinkers, who may also order food items. For each bill we store the details of the transaction – what items are purchased (e.g. beer, food etc.), time when the bill was issued, name of the drinker and the transaction id. Each transaction also has a total amount paid, which is the sum of prices of all items on the bill, plus 7% tax. There may also be a tip for some transactions.

In addition, each bar (bar table) has information about opening and closing hours for each day. These hours vary from bar to bar.

The database tables with pre-populated data will be given in a .csv format.

Part 1 (50%)

In this project you will have to create a web interface possibly with plots (to better illustrate the results) with three different pages:

1) The Drinker page

In this page, given a drinker, a user should be able to:

a) see all his/her transactions ordered by time and grouped by bars.

b) see a graph with the beers s/he orders the most

c) see graphs with his/her spending in different bars, per days of the week (Monday, Tuesday etc.), and month (January, February etc.).

2) The Bar page

Given a bar, a user should be able to:

a) see the top 10 drinkers who are the largest spenders

b) see the top 10 beers which are the most popular

c) see the top 5 manufacturers who sell the most beers.

d) see the busiest periods of the day (more transactions) and week (2 graphs). The x-axis on the first graph should be time intervals (e.g. 6pm, 7pm etc) and for the second graph days of the week (Monday, Tuesday etc.)

3) The Beer page

Given a beer, a user should be able to:

a) see the 5 top bars where this beer sells the most

b) see all the drinkers who are the biggest consumers of this beer

c) see a time distribution of when this beer sells the most. (either per time of the day, or per day, or per month)

Part 2 (25%)

In addition, you are asked to write SQL queries to check if the following patterns are enforced in the database:

1) Transactions/bills cannot be issued at times when the given bar is closed

2) Drinkers cannot frequent bars in different state

3) For every two beers, b1 and b2, different bars may charge differently for b1

and b2 but b1 should either be less expensive than b2 in ALL bars or more expensive than b2 in ALL bars. For example, it cannot be the case that in one bar Corona is more expensive than Bud, and in another bar, Bud is more expensive than Corona. However, Corona may be more expensive than Bud in one bar, and have the same price as Bud in another.

You should write the SQL queries to verify the constraints and return TRUE or FALSE (in case constraint is not satisfied). Queries that don't return a boolean value won't be accepted.

Part 3 (25%)

Finally, you should enforce integrity constraints in your database, by specifying the primary keys in each table, as well as the foreign keys. Then, you should create a

"modification page" where end users should be able to modify any table in your database. If update is not accepted (e.g. because of a foreign key violation), you should provide the feedback message "Violates foreign key". If update is valid, you should show a successful message and execute the update.

Note: You don't have to do that for all the tables in the schema given. Just for the tables that pertain to the individual project (e.g. not about bartenders, inventory etc).

Clarification: This modification page, is a page on your web application (as your bar page is etc.). You should have a way, either through a box where the user enters a SQL query, or through UI elements (like dropdowns, textfields etc.), allowing users to make any update they want (insert, update or delete). It's up to you to design the page, as you would like.

Two-people project

We will extend the bar-beer-drinker scheme, with information about specific transactions by customers (drinkers). We will store all "bills" of all drinkers. Drinkers may order both food items and beers. In addition, we will have data about

bar inventory on each given day – how many beers of each type they keep in their inventory. We assume (unrealistically) that there is always enough food (Since otherwise we would have to measure recipes etc.). Finally, each bar will have different shifts (hours) covered by different bartenders. This data about which bartender is working which shift (i.e. hours from – to) is stored in your database.
For each bill, we will store the details of the transaction – what items were purchased (may be beer, may be food, soft drinks etc), time when the bill was issued, name of the drinker and the transaction id. Each transaction will also have the total amount paid, which will be the sum of prices of all items on the bill, plus 7% tax. There might also be a tip.
The database tables with pre-populated data will be given in a .csv format.

Part 1 (50%)

In this project you will have to create a web interface possibly with plots (to better illustrate the results) with five different pages:

1) The Drinker page

In this page, given a drinker, a user should be able to:

a) see all his/her transactions ordered by time and grouped by bars.

b) see a graph with the beers s/he orders the most

c) see a graph with his/her spending in different bars, per days of the week, and month.

2) The Bar page

Given a bar, a user should be able to:

a) see the top 10 drinkers who are the largest spenders

b) see the top 10 beers which are the most popular

c) see the manufacturers who sell the most beers.

d) see the busiest periods of the day (more transactions)

e) see the time distribution of sales

Bar Analytics: Find the top 10 bars with the biggest sales of each beer. A user should be able through a drop down menu to specify a brand of a beer and a specific day and see a ranking of bars by their sales of the specified beer and specified day.

3) The Beer page

Given a beer, a user should be able to:

a) see the 5 top bars where this beer sells the most

b) see all the drinkers who are the biggest consumers of this beer

c) see a time distribution of when this beer sells the most.

4) The Bartender page

Given a bartender and a bar a user should be able to:

a) see all shifts of this bartender in the past and how many beers of each brand s/he sold.

Bartender analytics: Given a bar, and a shift (say 8-10PM) and a day of the week, show a ranking of bartenders by total number of beers sold. (This will compare “apples with apples”, since we will only compare bartenders on the same shift and same day of the week).

5) The Manufacturer page

Given a manufacturer, a user should be able to:

a) see the regions (states) where their sales are the highest the . (Sales of manufacturers are total sales of all beers that they produce)

b) see the states where this manufacturer’s beers are liked the most (i.e. where do most of the drinkers who like their beers live).

Part 2 (25%)

In addition, you are asked to write SQL queries to check if the following patterns are enforced in the database:

1) Transactions/bills cannot be issued at times when the given bar is closed

2) Drinkers cannot frequent bars in different state

3) For every two beers, b1 and b2, different bars may charge differently for b1 and b2 but b1 should either be less expensive than b2 in ALL bars or more expensive than b2 in ALL bars. Cannot be the case that in one bar Corona is more expensive than Bud and in another Bud is more expensive than Corona.

But Corona may be more expensive than Bud in one bar, and have the same price as Bud in another.

4) A bar cannot sell more beers of a specific brand, than it has in its inventory

5) A bartender cannot work more than one shift a day.

You should write SQL queries to verify the constraints and return TRUE or FALSE (in case constraint is not satisfied). Queries that don't return a boolean value won't be accepted.

Part 3 (25%)

Finally, you should enforce integrity constraints in your database, by specifying the primary keys in each table, as well as the foreign keys. Then, you should create a "modification page" where end users should be able to modify (insert/update/delete) any table in your database. If update is not accepted (e.g. because of a foreign key violation), you should provide the feedback message "Violates foreign key". If update is valid, you can then show a successful message.

Submission Files

You should submit the following 5 files, the file names of which should be preceded by your lastname (e,g, Kalokyri_projectCode.zip or ImielinskiKalokyri_projectCode.zip, for groups of 2). Groups of 2 people should submit only once, under one person's sakai submission.

1) [lastname]_projectCode.zip: a tar/zip file of all the project code you have

written for part 1 of the project. Please submit all your eclipse project from your workspace (all your .java, .jsp, .html files etc.) NOT your .war file.

2) [lastname]_patterns.pdf: a .pdf file with the queries you have written for part 2 of the project.

3) [lastname]_constraints.pdf: a .pdf file with the primary keys and the foreign keys per table.

4) [lastname]_demo: A demo video where you show everything you have implemented (you can user any screen recording video application you like):

• For part 1, you should show all the pages of your web application, fully functional.

• For part 2, you should show what your queries return when you run them on MySQL workbench.

• For part 3, you should try making an insertion/update in the database through your UI, and show at least one example of an update that gets rejected because of a foreign key constraint violation along with the message, as well as one example that gets successfully updated.

IMPORTANT: In case things are unclear or we have concerns about your project, you should be ready to demo your application live for us.

5) [lastname]_README.txt: a .txt file mentioning anything you want us to know about your application. You can omit this file in case you have nothing to mention.

DEADLINE: Sunday, November 22 at 11:59pm

Good luck!