4CCS1DBS Database Systems
Coursework 2017
Part 2: Implementation

Overview

The purpose of this coursework is to create a database ER schema and relational schema based on specific domain based on the provided requirements. This coursework also involves implementing your relational schema in SQL and writing some queries in SQL's Data Manipulation Language.

This coursework is comprised of 2 Parts, **each with separate deadlines**. Be aware any late submissions will have the grade for that part capped at 40%. This is Part 1 of the Coursework.

The entire coursework is formally assessed and is worth **10%** of your final grade. Each Part of the Coursework is worth 5%. You will receive some feedback as part of the marking of the coursework.

Part 2: Implementation (5% of your final grade)

(2.1) Create and implement the design in appropriate SQL schema and table creation queries, including entities, relationships and constraints.

(2.2) Insert appropriate sample data using INSERT queries.

(2.3) Create and output the appropriate SELECT queries.

(2.4) Update data using the appropriate UPDATE query.

(2.5) Remove data using a DELETE query.

Part 2 Deadline: Sunday March 19th by 11:55 pm on KEATs.
(Part 2 Late Deadline: Monday March 20th by 11:55pm on KEATs)

Overall presentation including legibility and proper use of language (where applicable).

Part 2: SQL

Setup. In Part 2, you will implement your relational model for the Burrito Food Truck Database. On KEATs you will find a **.zip** containing template SQL files to edit for your Part 2 submission.

For Part 2, you may find that your original relational schema needs updating to accommodate the queries that you want to run. You are not locked into your original design for Part 1. If you do desire to change your design, submit an updated **PDF** file containing your updated design, highlighting your changes.

Remember that the original database requirements as described in Part 1 still apply, and regardless of the changes that you make for Part 2, **your original submitted Part 1 design will be assessed your Part 1 mark**.

2.1 **Schema Definition.** Based on your design from Part 1, write the required SQL DDL (Data Definition Language) statements (i.e. CREATE TABLE…) to create the schema and corresponding tables.

Ensure that:

- table and attribute names **do not** conflict with SQL reserved words
- attribute data types are **core primitive SQL data types** as described in the lectures (i.e. **do not** use the ENUM type for example)
- table columns have appropriate **key** and **entity** constraints properties
- every table has a **primary key** specified as it corresponds to your relational model
- **all foreign keys** are properly declared, and explicitly describe how they handle potential **referential integrity** constraint violations (i.e. it is up to you to decide the triggered action to the foreign key constraints)
- your schema enforces the **domain constraints** you identified in Part 1
- your schema enforces the **semantic domain constraints** you identified in Part 1

Note that you may not be able to enforce all of the semantic domain constraints in the `CREATE TABLE` statements and MySQL does **not** have Assertions in the manner that we discussed in lecture (i.e. using `CREATE ASSERTION`). If you are unable to enforce a semantic domain constraint **include a comment** in your schema explaining your constraint and the reason it is not implemented.

Write your schema in the provided template file: `schema.sql`

Assume that the database schema will already be created for you (i.e. **do not** include a `CREATE SCHEMA` statement in your file, **it will result in an error**). Also assume that your script will already be run within your database schema (i.e. do not include a `USE`…; statement in your file, **it will result in an error**).

2.2 **Populate Database with data.** Time to get creative! Populate your database with some data that you will come up with on your own. Since you only require a small test sample of data, you will use SQL INSERT statements to populate your database.

More precisely:

- Research 3 markets in London to include in your database.
- Chipp is starting out with 2 Food Trucks, and make sure that all of its information is included in the database.
- Schedule both Food Trucks to be active in the markets for 1 week at the end of May 2017 and the beginning of June 2017 (i.e. Sunday 28-May through Saturday 3-June).
- Pick **at least** 3 of your favourite celebrities to include as Customers, and give each of them a. Make up the DOB/Address/EmailAddress for these Customers.
- Implement the full burrito menu as described in the database requirements, including reasonable information about the beans, fillings, sizes and prices (Google "burrito menus" to help you discover the wonderful world of burritos).
- Include "Guacamole" as a topping with a price, and the other toppings mentioned in the database requirements.
- Include "Lemonade" and at least 2 other types of drinks on offer of various sizes.
- For **at least** 3 of your customers, have them each have at least **3 orders** spread out throughout the scheduled week mentioned above. Make sure that some orders occur in May and some occur in June.
- Include **at least** 3 orders through the scheduled week that are placed by people without a BurritoClub.
- Make your orders vary, some including drinks, some not, and a variety of sizes an options for the burritos.
- Make sure that at least one order includes at least a burrito with **at least** 2 toppings (one of which is guacamole) and a drink (you will use this order in part 2.3.4 below).
- **At least** 1 of your customers should order enough burritos to earn a free burrito.
- Be sure that all of your BurritoClub burrito count is current in the database.

Write you INSERT statements in the provided template file: `insert.sql`

You **may only** use the DML (Data Manipulation Language) commands covered in lecture to help you populate your database.

All of your data **must** be contained within the `insert.sql` file, do **not** load the data from separate data files (i.e. using a CSV file).

Do **not** use other SQL statements, such as FUNCTIONs, PROCEDURESs or other programmatic MySQL-specific commands.

Assume that your script will already be run within your database schema (i.e. do not include a `USE`...; statement in your file, **it will result in an error**).

2.3 **Query the Data.** Write the SELECT statements that to obtain the following queries:

1. **Total Sales.** Chipp would like to know his total for the month of May 2017 only. Write a SELECT query the gives the total sales for the month. Have your result return a **single scalar value** (i.e. *in total GBP*).

2. **Burrito Report.** For each filling type and size, list the total number of burritos sold the month of June 2017. In the listing, include the filling type / size combinations that did not sell (i.e. had 0 quantity sales).

3. **Top Customers.** Chipp would like to personally email his top customers by total purchases (i.e. it is up to you to decide what defines total purchases). Write a SELECT query that lists all of the customers Name and EmailAddress, and your total purchases metric. Sort the customers by purchases in decreasing order, showing the top-purchasing customer first.

4. **Guacamole Receipt.** Create an itemized receipt for the customer order that you created that included a burrito with guacamole and a drink.. To help you write the query, you can hard code the order number (i.e. refer to the order by id in the query). Write a SELECT statement that lists the burrito (or burritos if there were multiple burritos ordered) first, and the drinks second. For each item show a description of the item and the price for the item. Make sure you include the applicable information for each item (i.e. the type of bean for the burrito, the size of the drink…etc..).

   Hint: You may use the string concatenation function `CONCAT` (https://dev.mysql.com/doc/refman/5.5/en/string-functions.html#function_concat) to help construct the item description for a prettier receipt.

5. **Alcohol free!** Pick one of the markets frequented by the Food Trucks and we will assume that there is a regulation to only sell non-alcoholic drinks on the premises.

   Since MySQL does not support an assertion to check this constraint, write a SELECT statement that returns only a scalar Boolean value (i.e. either True or False). It should return True if there are **no violations** in the database of this regulation. If there is a violation, then the SELECT statement should return False.

   There is a violation if there is an order containing an alcoholic drink placed at the market you have picked. In your query, refer to the market by its name.

   Show that your SELECT statement works by placing an order that contains an alcoholic drink and then running your SELECT statement. Your asserting SELECT statement should yield False (which would cause the ASSERTION to fail).

Write all of these SELECT statements in the above order in the provided template file: `select.sql`

Assume that your script will already be run within your database schema (i.e. do not include a `USE`...; statement in your file, **it will result in an error**).

2.4 **One more thing...** Annoyingly the customer who placed the most recent order would like to add a few more items to the order. Instead of modifying the order:

1. Create a new order (using INSERT statements) with enough burritos so that one of the burritos they ordered will be free using their BurritoCard.
2. Write an UPDATE statement that updates a customer's burrito count for their BurritoCard.
3. Write a SELECT statement that calculates the final order total price taking into account the free burrito.
4. If you are keeping track of an order's total price in your database, write an UPDATE statement to keep it current.

Write all of these statements in the provided template file: `update.sql`

Assume that your script will already be run within your database schema (i.e. do not include a `USE`...; statement in your file, **it will result in an error**).

2.5 **Track me not!** Upon receiving an email from Chipp, his top-purchasing customer becames upset to find out that Chipp is tracking all of their Burrito Food Truck purchases. They have demanded to **have all of their data** to be removed from Chipp's Burrito Food Truck database.

Using this customer's email address as their identifying attribute in the query, write the DELETE statement(s) that removes this customer, and all their data, from the database. To avoid any future embarrassment in case of a data leak, make sure you remove all trace of the customer from the database.

Write all of these DELETE statements in the provided template file: `delete.sql`

Assume that your script will already be run within your database schema (i.e. do not include a `USE`...; statement in your file, **it will result in an error**).

What to turn in

For each SQL file that you turn in:

1. Include your **NAME** and **STUDENT NUMBER** at the top of every SQL file in a SQL line comment.
2. Edit these files as **text files, not Word files or propriety SQL software.**

3. Do **NOT** rename the files.
4. **ONLY** use the SQL line comment character (i.e. lines beginning with --) for comments.

Comment your SQL.

Just like program code, comments help outline, structure, and make clear what is written. You are expected and will be evaluated on your ability to provide comments to help provide structure, organization, and clarity to your SQL code. Make your comments useful, concise, and clear.

Submission

Put all of the SQL files (and your optional updated database design PDF) in a **ZIP** file (i.e. with a .zip file extension) and submit it on KEATs before the deadline.
Do **not** put the files in a RAR (i.e. rar file).
Do **not** put the files in a tar-gzipped file (i.e. tar.gz.).
Submit your files in a ZIP file (i.e. with a .zip file extension).

Any SQL file that is missing or renamed will result in 0 marks for that sub-part.

Evaluation

The SQL files you create will be evaluated using the NMS database server that you utilized in lab. Test all of you database SQL files on the NMS's database server in your own personal database to be absolutely sure that they work and do not have any errors.

Your files will be executed in following order:

```
1. schema.sql
2. insert.sql
3. select.sql
4. update.sql
5. delete.sql
```

Each file will be tested from the NMS UNIX command line, with this command:

```
mysql -u k123456 -p -h mysql2.nms.kcl.ac.uk -P 33306 yr_db < file.sql
```

Where `k123456` is the database user name (i.e. your k-number), `yr_db` is the database name, and `file.sql` is the sql file to be executed. Using your NMS databases, test your files, in this order, to make sure that they run, before you submit them. Test your files, even if you are "just adding comments".

If anyone of your submitted files do not run on the NMS database server, your Part 2 coursework will be capped to 60% of the Part 2 total marks.