Computing and Data Analytics 5540 -- Team Project

==========

Date posted: Feb. 19 Submission & Deadlines: see below

==========

INTRODUCTION

The Halifax Science Library (HSL) maintains an SQL database containing tables about various publications. HSL now has the following immediate plans:

a. to sell library items and record information about these sales (transactions) in their database;
b. to record, for each scientific journal (***magazine***), data about all articles in that magazine;
c. to record data about monthly expenses.

You have been hired to realize these immediate plans.

HSL personnel has already extracted and recorded in the SQL database some names and contact information of authors from available magazines--see about file "existing_tables.sql" further below. Moreover, a set of articles has been extracted from these publications in json format (see about file "articles.json" further below).

DATA REQUIREMENTS

1. New library items

-

HSL has existing tables about books, magazines and authors. A magazine (journal) publishes several volumes each year. HSL now needs to record information about the articles contained in magazine (journal) volumes. For each article the following information need to be recorded: a unique id, title, the list of author names, the magazine volume number as well as the page numbers where the article appears, the year of publication, and the magazine in which the article is contained.

-

The following constraints must be satisfied:

- The name of a magazine cannot be empty or NULL.
- No two different volumes of any magazine have the same volume number.
- All articles in the same magazine volume have the same publication year.

2. Customers and Transactions

-

Information will be kept about (loyal) customers who have purchased library items in the last five years. In particular, for each customer, the following information is required: a unique customer ID number (CID), the customer's first and last names, telephone number, mailing address, discount code, and a list of transactions within the last five years. Each transaction of the customer involves a transaction number, the date of the transaction, the total purchase price, and the list of id numbers and prices of the items that were purchased by the customer in that transaction. The following facts should be noted:

- No two transactions (of the same or different customers) have the same transaction number.
- The total purchase price of a transaction is equal to Sum*(1-2.5*DC/100), where Sum is the sum of the prices of the items in the transaction and DC is the discount code of the customer. The discount code is computed based on the transactions of the customer as follows: It is equal to 5 if, in the last five years, the customer has purchased items whose total purchase value exceeds $500. It is equal to i, where i=0,..,4, if, in the last five years, the customer has purchased items whose total purchase value is greater than 100*i, and less than or equal to 100*(i+1).

3. Monthly Expenses

-

HSL wants to record information about its expenses, for each month of the last twenty years. A monthly expense record involves the following: cost of electricity, heat, water, and the rent---the rent is the same for all months of a year. Moreover, for each monthly expense record, it is required to know the list of hours each employee worked in each day of the month. Employees are identified by their unique SIN.

YOUR TASKS

1. **Form teams**: This is a team project. Initial expectations: (a) make teams of five; (b) each team should submit in Brightspace, by the deadline specified below, a text file called "team_members.txt" stating the names and A-numbers of the team members, one per line. It is fine if your team has less than 5 memebrs. Thus, only **one** member of the team will submit the file "team_members.txt". If you cannot find a team then again you are expected to submit the file "team_members.txt" before the deadline stating that you cannot find a team.

2. **Design a GOOD relational database schema**: Use the ideas of the relational data model to design a good database schema according to the above requirements. Make sure to indicate primary and foreign keys. Your design is to be submitted as specified below. ***Some hints:*** It is a good idea to first design an EER diagram according to the data requirements. Then map your EER diagram to a relational database schema. You might need to modify a few schemas to make sure that all schemas are in 3NF. Note: The only required information about employees in this design is the employee SIN.

3. **Create new tables in MySQL**: You are required to implement (in MySQL) only the parts of the database that pertain to customers, their transactions, and the items involved in those transactions, as well as magazine volumes, the articles in those volumes, and the authorship information about these articles. Note that already available for you in the Linux Server is the MySQL script

   `/home/course/u00/sk_public/existing_tables.sql`

   containing definitions and data for the schemas pertaining to authors, magazines, and prices of library items. You are expected to run this script in your MySQL account before creating new tables for the present task. Then, make a MySQL script file, called "new_tables.sql", containing the definitions of your new tables. In your script, each statement creating a new table X, say, must be preceded by a statement to drop table X --see how this is done in the given file "existing_tables.sql".

4. **Create Mongo collections from given data**: You are given in the Linux Server the file

   `/home/course/u00/sk_public/articles.json`

   containing data about magazine (journal) articles. Note that this file has been produced by a program that read article data from files in various formats, so each article object includes fields that are useless for this application. Write a Bash script, called "data2mongo.sh", that does the following:
   a. Connects to MySQL and executes your script "new_tables.sql".
   b. Uses the MySQL table AUTHOR, as defined in "existing_tables.sql", to make a corresponding Mongo collection AUTHOR.
   c. Makes a Mongo collection ARTICLE out of the useful data in the given file "articles.json". In particular, ARTICLE must have **no** articles in which the field **author** is missing.

   Make sure that your Bash script drops the Mongo collections AUTHOR and ARTICLE before it makes them as specified above (this is in case these collections already contain some data before executing the script).

   You are free to do this task as you wish. One approach you could take is as follows: First make an initial Mongo collection from "articles.json" where any articles with no author field are removed. Then, write a javascript that reads the initial collection and creates the collection ARTICLE in which each object corresponds to an article with the required fields: _id (taking the same type of values as the id attribute in your SQL article table), title, name of magazine (journal), volume, year, pages, authors, where authors is

an array of author names.

5. **Update MySQL tables from the Mongo collections**: Write a Bash script, called "mongo2sql.sh", that will use the two Mongo collections AUTHOR and ARTICLE to insert data into the corresponding new MySQL tables you created above, and to the existing MySQL tables AUTHOR and MAGAZINE---this is because the given file "articles.json" contains magazines or authors that do not exist in the given tables AUTHOR and MAGAZINE.

   Again you are free to do this task as you wish. One approach is to write some other supporting script (e.g., javacript) that (i) it would process the two collections AUTHOR and ARTICLE to produce intermediate results into appropriate text files (e.g., .tsv or .csv files); (ii) it would insert data from these text files into the MySQL tables.

6. **Write and Present a PHP web application:** that allows the user to perform the following operations:
   a. *Show table:* the PHP app shows a list of all tables in the database, allows the user to enter the name of a table, and prints the contents of that table.
   b. *Add new article:* the user enters the attribute values pertaining to an article (title, magazine id, volume number, pages, and list of authors) and the PHP app inserts rows in the appropriate MySQL tables. In case any integrity constraints are violated with the given attribute values, the app should provide an appropriate message.
   c. *Add new customer:* the user enters the attribute values pertaining to a customer and the program inserts the corresponding row in the database. In case there is an existing customer with the same last name and the same first name, the PHP app should ask the user whether this is indeed a new customer.
   d. *Add new transaction:* the user enters the ID of the customer and the list of id numbers and prices of the items involved in the transaction. Based on this input, the program will update the database.
   e. *Cancel transaction:* the user enters the transaction number and the program removes from the database everything pertaining to that transaction, provided the transaction occurred no more than 30 days before the current day.

   The PHP application will be presented and tested in class. In addition you are required to prepare a file **"report.pdf"**, containing a short description of how your PHP application is structured (what the main files are and how they work together). In case you can achieve the above functionality using a language other than PHP you are welcome to do so.

7. **Confidential peer evaluation**: Each team member will write a short confidential peer evaluation file "peers.txt" stating which parts of the project they contributed and what were the contributions of each of their team mates--see deadlines below.

ABOUT YOUR SUBMITTED SCRIPTS: When your scripts attempt to connect to MySQL or Mongo, use the technique used in your first assignment; that is, instead of writing your actual database name, username and password, use the variables $db, $user and $pass, whose values are initialized at the top of the script as $1, $2, $3 (these correspond to the command line parameters of the script). All scripts will be tested on the Linux server for this course.

SUBMISSIONS & DEADLINES (strict)

You are supposed to submit the following:

1. In Brightspace, the file "team_members.txt" (see above the task Form Teams).
2. In Brightspace, the file "design.pdf" containing the design of the relational database schema specified above. **One** submission per team.
3. In the Linux Server, a zip folder, called "scripts.zip", containing the files "new_tables.sql", "data2mongo.sh", "mongo2sql.sh", and any other supporting programs/scripts that will enable one to execute the required scripts data2mongo.sh and mongo2sql.sh. **One** submission per team. The full path

name to your zip folder should be specified in your submission of "report.pdf" (see below). I will use in the server the command

```
unzip scripts.zip
```

to unzip your zip folder, so you are expected to use the server command `zip` to make scripts.zip.

4. In Brightspace, the file "report.pdf" about your web app. **One** submission per team. Add as a comment to this submission the full path name of the server folder containing your zip folder "scripts.zip"
5. In Brightspace, the file "peers.txt" containing the peer evaluation report. **Every** team **member** will submit their own file.

The following *strict* deadlines must be observed:

- the team membership file "team_members.txt" --> **Feb. 22**
- the DB design file "design.pdf" ----------> **Mar. 16**
- "report.pdf" and "scripts.zip" ----------> **Apr. 14 at 5pm**
- the team presentation of the web app ----------> **Apr. 15 at 8:30am**
- the confidential file "peers.txt" ----------> **Apr. 16 at 8:30am**

MARKING SCHEMA

- 02% "team_members.txt"
- 26% "design.pdf"
- 20% The script "data2mongo.sh" and any companions.
- 20% The script "mongo2sql.sh" and any companions
- 32% The in-class presentation and the file "report.pdf"
- Marks will be deducted if there is evidence that your contribution to the project was not satisfactory and/or you fail to submit the file "team_members.txt" as specified above.