MPCS 53001 Databases

Final Project - Step 3 (13 points)

The University of Chicago - Department of Computer Science

Dr. Ahmed E. Khaled

In this step, you will implement the database you designed in the previous steps along with an interface allowing the users to use and query your database. Step 3 can be divided into three sub-steps:

A. Create Database and Tables:

Translate the relational schema you developed in Step 2 into an actual database. ***Use MySQL's Workbench or Shell and the SQL query language*** to create your database and the corresponding tables.

Place the constraints (e.g., default values, foreign keys) you find appropriate. Every relation must have primary key(s) and may have foreign key(s) based on how such relation is connected to other relations.

> **** Note:*** In Step 2, there were some requirements guiding the minimum number of entity sets and relationship sets for your database. These are the minimum requirements because of the limited time, however, as a database designer and developer, you may decide that your database requires more entity sets and relationship sets.

> **** Note:*** In Step 2, you submitted an ER diagram and corresponding relational schemas – as the design process for any database is an iterative one, you can add/remove/update the diagram and schemas to fulfill the implementation phase.

B. Prepare and Insert Records:

In step 2, you prepared data records (or data sets) to be used to populate the corresponding relations in the created database. In this sub-step, you will create either a ***Python or Java application*** to connect with the created database, read the data records, prepare them to fit the design relations (by preprocessing the data if necessary), and use ***the SQL insert commands*** to add such data records.

As mentioned before, in step 2, the data sets you prepared are expected to address the essential/important attributes (which can be few attributes per table). You may not find a data set that ***fully covers all attributes of all tables***.

If you don't have real data to use or you couldn't find a data set that fully covers all attributes of all tables (as discussed above), your app should generate random data for the uncovered attributes - The generated data should make sense as much as possible (e.g., don't generate negative ages for employees or students with GPA > 4.0). Take the following ***Football Games*** table from a football database as an example:

GameID	Team Name	Opponent Team	Match location	Match date	Result

To prepare data tuples/rows for this table, search the internet and find real data covering the Team Name, Opponent Team, and Match Date attributes. In this case, your application should then randomly generate data for the rest of the attributes (GameID, Match location, and Result). The GameID can be an integer that auto increments every time you insert a record; the Result attribute can be a random integer from 0 to 10, the Match location can be a random choice from a list like (Paris, London, Cairo, Moscow, Dubai).

If finding real data from external online resources or generating meaningful random data using will be a complex problem, you may use online tools to generate random data into files. Here is an example of an online tool to generate random data files: https://www.mockaroo.com/

**** Note:*** The application at this stage can be terminal-based that runs in the background and does not accept any input from users.

**** Note:*** This application should run once, otherwise you will get errors for trying to insert duplicate records.

**** Note:*** To keep the application manageable, your application will not accept user requests to insert/remove/update certain records.

C. Run Queries and Print Results:

You may extend the application you developed in the previous sub-step or create a new application dedicated to this sub-step.

In this sub-step, you will create either a ***Python or Java application*** with an interactive GUI that is able to handle user interactions and react accordingly. The GUI (that can be web-based or desktop based) is a friendly interface (or a frontend) to facilitate the interactions between users and your database that represents the backend of your system.

**** Note:*** In Step 1, you submitted a number of queries - as we discussed more SQL operators and techniques, you can update/modify the submitted queries to fulfill the implementation phase.

The GUI -given the time limit- is expected to be ***simple***, which means it should deliver the following essential functions:

1. The interface should display a name for your application, along with a short welcome message describing the goal of your database.

2. The interface should list the possible/supported queries. This interface is for users, so the queries must be listed in English, not in relational algebra or SQL.

3. The user should be able to read the available queries and select which query to run. This can be achieved by wide range of ways, for example:
    - you can display the queries as a set of radio buttons for the user to select one
    - the user may select the query to run by typing the number in an input box (e.g., the user may enter #4, to run the forth query: “display the top products”)

4. The interface may accept more user inputs, your query the user selects in (#3) may ask further inputs like:
    - the year or the type of product to filter the results accordingly
    - the fields to display in the result

5. The interface may have a button, when pressed, your application starts to execute the selected query and display results.

6. The interface should display the result (or part of the result) to the user in a readable and organized way

7. The user should be able to select another query to run, and so on.

You can design how your users may interact with your database the way you find appropriate, however, there are few important points:

- The user should not write SQL queries to be executed through your GUI.
- You should fetch the results from the database for each user query, your application ***should not cache/save results*** in case the user would like to run the same query again.

 # Resources for GUI:

In ***Python***, there is a wide range of GUI libraries, here are few examples that you may consider with respect to your design:

- PySimpleGUI: The Simple Way to Create a GUI With Python [https://realpython.com/pysimplegui-python/]
- How to Build a Python GUI Application With wxPython [https://realpython.com/python-gui-with-wxpython/]
- Want to create GUI applications with Python?
    - [https://www.pythonguis.com]
    - [https://www.pythonguis.com/tutorials/pyqt6-creating-your-first-window/]

In ***Java***, there are also GUI libraries, here are few examples that you may consider with respect to your design:

- Introduction to GUI Building [https://netbeans.apache.org/tutorial/main/kb/docs/java/gui-functionality/]
- Java Swing Tutorial: How to Create a GUI Application in Java [https://www.guru99.com/java-swing-gui.html?utm_campaign=click&utm_medium=referral&utm_source=relatedarticles]

You may also search for other libraries and tutorials

<u>Submission:</u>

A. Submit **ONE PDF file** to the folder titled ***Step 3*** under Assignments tab, structured as follows:

- On the first page, place your name and the names of your team members - ***Just one member should submit the PDF file***.
- On the second page, provide a description of the idea/topic along with the requirements/specifications your database addresses (this can be the version you submitted in step1 or an updated version).
- On the third page, the final version of the ER diagram along with the modifications/adjustments you made (if any)
- On the fourth page, the final version of the relational schema along with the modifications/adjustments you made (if any)
- On the fifth page, information about the data sets you used (this can be what you submitted in step2 or an updated version):
    a. The source(s) of your data sets (e.g., We will use the Energy Consumption Dataset provided by Kaggle, and here is the URL: http://…).
    b. The relations (tables) and attributes covered with such real dataset(s)
- On the following page(s), for each query, copy the SQL query you developed and take a clear screenshot(s) of the output (the result of running it).

B. Check the due date. Since this is the last step of the final project, the grace period is 1 hour after the due date for your submission to be graded out of 50% of the assignment's grade for the team. After this period, your late submission will not be accepted.

C. You will discuss your work and a demo with the TAs and myself during the lecture time in-class in the exams week. The discussion requires:

- The full team must attend and present the work
- Bring your laptop(s) to display the developed database and the developed application(s)
- Bring your final document to discuss the database design, no need to provide slides

The grading criteria will be as follows:

- The correctness of your design
- The quality of the queries
- The functionality of your application(s) and the developed GUI
- The type/size of data you used to populate your database
- The structure of the submitted document

The exact schedule for teams will be shared with the full class soon.