IS455 Database Design and Prototyping
Instructions for the Final Project Deliverables

Assignment Overview

In the Final Project, you are creating a MySQL-based prototype for a Web application based on a moderately-sized relational database. In the Final Project Proposal Assignment*,* you got this project started by creating a *System/Prototype Description* and a *Conceptual Data Model*. In this Final Project Deliverables Assignment*,* you are creating and delivering the prototype.

You may accomplish this by following the steps below:

1. Review the *System/Prototype Description* document that you created previously. Revise the document to reflect your best current understanding of the project requirements.

2. Review the *Conceptual Data Model* document that you created previously. Revise the document to reflect your best current understanding of the project requirements.

3. Reset the settings of the MySQL database product to make sure they are proper for this project.

4. Create a MySQL database schema for the prototype. Use the MySQL Workbench Modeler/Diagrammer to create an entity-relationship diagram (ERD) for the schema.

5. Use the graphical database editing tools available in MySQL Workbench to populate the database with an initial set of data that represent the state of the database at the beginning of a typical work week.

6. Use the MySQL Workbench database backup feature to create a database restore script that can be used to create a fresh, populated copy of the database whenever required.

7. Design, code, and test all Report SQL Scripts. These reports should address business questions that the users need answered to conduct their regular business processes.

8. Design, code, and test all Database Maintenance SQL Scripts. These scripts should implement changes in the database necessary to reflect data changes caused by typical business operations.

9. ~~If you wish to do the Challenge portion of the project, revise the Database Maintenance SQL Scripts to include code for transactions and locking.~~

10. Retest the entire system flow beginning with the database restore script and ending with the database maintenance scripts.

11. Package up the files in your prototype and submit your work.

More detail is presented below on each of these steps.

Assignment Details

Please Note: In the original version of these instructions, there was a challenge exercise (#9) in which you were invited to add transactions and locking to your database management scripts. Subsequently, I realized that this couldn't be accomplished without our having covered Chapter 13 of the text. So, I have canceled the challenge exercise and removed the distinction between regular and challenge exercises. The result is that the points available for the exercises in which you submit work sum to 100 points. So, meeting all expectations on these exercises will earn you a score of 100. I apologize for any confusion that this change might cause.

1. Review and Revise System/Prototype Description (~~Regular~~)

Review the *System/Prototype Description* document that you created previously. Revise the document to reflect your best current understanding of the project requirements. When you are ready to submit your work, check this document a final time to make sure that all revisions are incorporated. Submit this document as a .PDF file named system_prototype_description.pdf.

2. Review and Revise Conceptual Data Model (Not Submitted)

Review the *Conceptual Data Model* document that you created previously. Revise the document to reflect your best current understanding of the project requirements. This document is for your own use while completing your work and will not be submitted with this assignment.

3. Reset MySQL Settings (Not Submitted)

During the course, we have been running a script that I provided to set the SQL modes for MySQL Server. Another copy of this script is provided in the starter files for this assignment:

- set_sql_modes_for_database_course.sql

Please be sure to run this script again before starting your Final Project. We will be running this same script before grading your submission. Failing to run this script may make you vulnerable to overlooking mistakes in your work that will lower your grade.

4. Create a MySQL Database Schema (~~Regular~~)

Using your *Conceptual Data Model* document as a starting point, use the MySQL Workbench Modeler/Diagrammer to create a logical database model and diagram. Then, further develop the model until it represents a full physical database model and diagram. You will be using this physical database model and diagram to generate an SQL schema script.

While creating your database model/diagram, be sure to include details on the following:

a. Database encoding: make sure that your schema specifies encoding for your database as utf8mb4.
b. Table definitions

c. Column definitions with appropriate data types
d. Primary keys identified in all tables
e. Default values for columns whenever appropriate
f. Auto numbering for columns whenever appropriate
g. Uniqueness constraints that enforce uniqueness in columns other than the primary key whenever appropriate. This is particularly important for name values on lookup tables.
h. NOT NULL constraints that prevent rows being added to tables with important column values set to NULL
i. Foreign key constraints that implement the table-to-table relationships
j. Additional indexes created for any column or combination of columns that you believe require them.
k. Make sure that you use the Schema Editor Panel in the MySQL Workbench modeler/diagrammer to give your database schema a reasonable name other than *mydb*.

When your model/diagram is complete, use the **Database > Forward Engineer** menu options to display the schema script and to generate the database on the server.

Also, use the modeler/diagrammer to create a .PDF document that includes the entity-relationship diagram for your physical database design. Name this document:

database_schema_erd.pdf.

5. Populate the Database Using the Graphical Database Editing Features of MySQL Workbench (Regular)

MySQL Workbench has graphical database editing features that allow you to insert, update, and delete data rows without resorting to coding SQL scripts. Use these features to enter data into your tables in two phases:

a. Minimal Data Loading Phase
In this phase, load 3 to 5 rows of data into each of your tables. Your purpose in this phase is to test your physical database design. You are likely to discover problems during this phase that will need to be solved be going back to your physical design activities, revising your diagram, and repeating the forward engineering process. Each time you do this, you will lose the data previously entered into the tables. This is why you should enter only minimal data until you have fully tested your schema.

   When you are satisfied that your schema is working, take some further time to review the features to make sure that the database design meets the requirements of your application. You might want to add additional columns to tables. Or, you might want to add or delete tables and relationships. At this point, you can still do this without losing significant amounts of data.

When you are satisfied that the schema meets, your application requirements, then proceed to the next step.

b. Prototype Data Loading Phase
In this phase, load enough data into each table to support the Report Scripts and Database Maintenance Scripts that you will be creating (see below). The decision on how much data you will need is somewhat artful. When the system that you are prototyping is developed and in full production mode, it might easily have thousands of rows in the tables. It is not practical or necessary to enter that much data. Instead, try to enter enough data to provide meaningful results when running the reporting and database maintenance features (scripts) of the prototype. Imagine the data that you would want to have available in the system if you were to use the SQL scripts to demonstrate the prototype to a new user.

6. Create Database Restore Script (~~Regular~~)

This is the first deliverable script for the project. Before creating this script, make sure that you have an up-to-date copy of the database with a correct schema and all prototype data loaded. This copy of the database may have been the result of several iterations of steps 4 and 5.

You can create this script using the **Server > Data Export** menu option of the MySQL Workbench tool. Each time you create a script in this manner, **be sure to include the data as well as the structure**. Ignoring this advice can lead to accidentally submitting a project that contains no data.

The database restore script should be named:

```
10_restore_populated_database.sql
```

Make sure that your schema (database) has a meaningful name other than the default value of *mydb*. Also, make sure that this script will run equally well whether or not there is a copy of your database already exists on the server. To accomplish this, the script should begin with the following code:

```
DROP SCHEMA IF EXISTS `your_schema_name`;
CREATE SCHEMA IF NOT EXISTS `your_schema_name`;
```

As the first script for your project, it will be the first script that we will run when testing your project submission. It should be the first script that you run when you are testing your project submission. This script is also handy for restoring the database to a reliable state after fixing a problem found during testing.

7. Design, Code, and Test All Report SQL Scripts (~~Regular~~)

These reports should address business questions that the users need answered to conduct their regular business processes. Here are some guidelines on which report scripts to provide:

a. Simple table listings are not required.
   The graphical editor features of MySQL Workbench does a great job of generating listings for each table. You can assume that whoever is operating the prototype is familiar with these listing features. Nevertheless, you may want to create report scripts that show subsets of the data in a table that are tailored to a specific task. Users might not want to scan through the complete table listing to find exactly the columns in which they are interested.

b. Reports that combine data from more than one table using joins are required.
   Ideally, there should be a customized report available for every activity that the users need to perform. Users should not need to consult of series of reports to get their job done.

c. Summary reports are required.
   Users should not be expected to do further processing on data presented in reports. When users need summary values to make a decision, there should be a summary report available.

Each report script should have a descriptive name that includes a number prefix. Start numbering the series with 11. Here are some examples:

11_phone_list.sql
12_guide_roles_list.sql
13_employee_availability_list.sql

When testing these report scripts, we will run them before running any of the database maintenance scripts (see below). While we will test these report scripts in their sequence order. They should produce the same results when run in any order.

8. Design, Code, and Test All Database Maintenance SQL Scripts (~~Regular~~)

These scripts should implement changes in the database necessary to reflect data changes caused by typical business operations. Changes to simple “lookup” tables that include only a primary key and a description do not need maintenance scripts. You will have already populated these simple tables using the graphical database editing features of MySQL Workbench. So, that capability will continue to meet the maintenance need for those tables in the prototype.

By contrast, you should provide INSERT, UPDATE, AND DELETE scripts the more substantial tables in the database. Often, changes to these tables reflect more important parts of business operations. Think of the database maintenance scripts as the content that you would use to demonstrate the prototype to a new group of users. In that case, you would want a script available for your use as you demonstrate how to record each significant business event in the database.

Some scripts will implement scenarios that will require INSERTs, UPDATEs or DELETEs to multiple tables. Consider a scenario of a new employee joining organization or an employee leaving the organization. Each of these would likely require a series of changes that include several tables.

When authoring these database maintenance scripts, please remember that a properly defined database implements relations between tables with foreign keys. Foreign key relationships are implemented by foreign key constraints. Foreign key constraints assure that no transaction will leave the database in a state of damaged referential integrity. This means that when you create scripts that INSERT rows, you may need to insert rows into related tables before you can insert the intended row into the target table. Likewise, DELETE scripts can leave the database in a state of damaged referential integrity unless related rows in other tables are deleted or updated. In these cases, we are expecting many of your scripts to issue more than one SQL statement to achieve the job at hand while maintaining referential integrity. When creating a multi-statement script, remember to end each SQL statement with a semicolon.

When coding DELETE and UPDATE statements, you may identify the row that you want to DELETE or UPDATE using a hard-coded primary key value. When you use these hard-coded key values, your queries will only work properly if run in a particular order. When we test your project, we will be running the scripts according to the prefix numbers in their filename (see below). So, make sure that your scripts work properly when run in that order.

Each database maintenance script should have a descriptive name that includes a number prefix. Start numbering the series with 50. Here are some examples:

50_add_a_new_reservation.sql
51_transfer_a_reservation.sql
52_delete_a_reservation.sql
53_add_a_new_employee.sql
54_delete_an_existing_employee.sql
55_add_a_new_trip.sql
56_delete_an_existing_trip.sql

~~9. Add Transactions and Locking to Database Maintenance Scripts (Challenge)~~

~~If you wish to do the challenge portion of the assignment, revise the Database Maintenance scripts adding code to support transactions and locking. Be sure to re-test the revised code.~~

10. Retest the Entire System Flow (Not Submitted)

The scripts that you will be submitting (especially the database maintenance scripts) are very sensitive to being run in the correct order. To make sure that you will get full credit for your work, it is important to run a final test of the entire system flow. To accomplish this, drop the database, then run the scripts in order (based on their numerical prefixes). Check that you get the expected results.

11. Package Files and Submit Your Work (Part of *File Submitted* Grade)
Please note that the names of all files and directories submitted must conform to the naming schemes shown below including the use of lower-case letters and underscores. Deductions may be made for deviations from this scheme.

Your submission for this project will be a single .ZIP file. When unzipped, this .ZIP file should produce a properly named and organized structure of deliverable files.

Your .ZIP file should be named according to the following scheme:

surname_givenname_final_project_deliverables.zip

If this were my own project, I would name the .ZIP file as follows:

trainor_kevin_final_project_deliverables.zip

When the .ZIP file is expanded, it should produce a top-level directory that is similarly named. The top-level directory should be named according to the following scheme:

surname_givenname_final_project_deliverables

If this were my own project, I would name the top-level directory as follows:

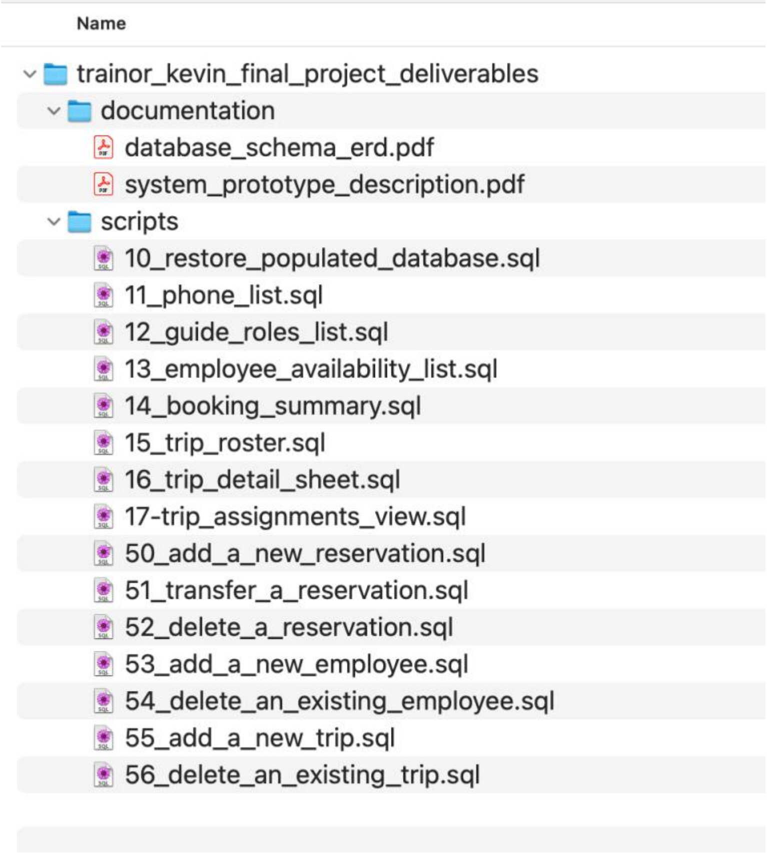
trainor_kevin_final_project_deliverables

Within the top-level directory, the files that make up the deliverable should be organized under two sub-directories:

- documentation
  Documentation files should be placed here.

- scripts
  SQL script files should be placed here

The following is a screenshot of the directory structure for a Final Project that I might submit:



```
Name

trainor_kevin_final_project_deliverables
  documentation
    database_schema_erd.pdf
    system_prototype_description.pdf
  scripts
    10_restore_populated_database.sql
    11_phone_list.sql
    12_guide_roles_list.sql
    13_employee_availability_list.sql
    14_booking_summary.sql
    15_trip_roster.sql
    16_trip_detail_sheet.sql
    17-trip_assignments_view.sql
    50_add_a_new_reservation.sql
    51_transfer_a_reservation.sql
    52_delete_a_reservation.sql
    53_add_a_new_employee.sql
    54_delete_an_existing_employee.sql
    55_add_a_new_trip.sql
    56_delete_an_existing_trip.sql
```

Tools

Use tools as indicated in the instructions above.

Submission Deadline

The submission deadline and submission activity will be indicated in the Weekly Schedule.

Grading

A separate grading rubric document will be posted to the Weekly Schedule.

Last Revised

2025-04-16