Database and Routing Homework

- Database: Create and Populate Database Tables, SQL Select Statements
- Routing: Achieve User Interface Reuse (JS code Injects Content on Click of Navigation Link)

CONTENTS

1. Overview ........................................................................................................................................... 1
2. Before You Start ............................................................................................................................... 2
3. Changing Your Mind About Web Application Functionality ............................................................. 2
4. Database Requirements .................................................................................................................. 3
   a) Create Tables ............................................................................................................................... 3
   b) Add Data ...................................................................................................................................... 4
   c) Run SQL Select Statements ......................................................................................................... 4
5. Routing for User Interface Reuse .................................................................................................... 5
6. Web Site Homework Requirements ................................................................................................ 5
7. Submission Requirements ................................................................................................................ 7
8. Homework Grading – Sample Deductions ..................................................................................... 7

1. Overview

In this homework, using MySQL Workbench, you will:

1. Create database tables: web_user and user_role (exactly as prescribed) and a third table designed as you propose (meeting the requirements listed in this document).
2. Populate all three tables with test data.
3. Generate several SQL select statements to extract data from the database.
4. Create a word document that shows your database work.

Next, you will also achieve user interface reuse (in your web site) by adding some JavaScript code that implements “routing”, meaning that when the user clicks on the “Blog” or “Home” links from your navigation bar, “Blog” or “Home” content gets injected into the content area. Then you will make sure that your home content aligns with the database table you propose and you will add some content into your Blog page.
2. Before You Start

Before starting on this Homework, make sure that you have installed MySQL Workbench and tested that you can connect to your database (if you added this course late, you may have to email your instructor to request a database be created for you ASAP). From the course web page (for this assignment) you’ll find links to the following documents:

- How to install MySQL Workbench
- How to connect to your Temple database (shows you how to find your database credentials, etc)
- MySQL Workbench Tutorial: How to create database tables and relationships (primary keys, foreign keys), enter data, and write SELECT statements

Create a word document to hold this week’s homework submission. Into this document put the following:

1. Your name and NetAccess user name
2. Your Web Site title
3. The (possibly updated) content of your home page (copy/paste), where you specify the functionality of your web site and entice users to visit your site.
4. In the next sections, you will be asked to copy/paste screens (into this word document) from MySQL Workbench – to show that you have done the prescribed database work.
   - To get a screen capture, click on Alt-PrtSc (copies active window into the clipboard), then paste into an image editor like MSPaint, then copy out just the part you want to show and paste that into the word document. If you paste from Alt-PrtSc directly into the word document, the screen capture shows too much, making things too small to read.

3. Changing Your Mind about Web Application Functionality

It is OK to change your mind (from last week’s homework) about the functionality that your web application will provide – as long as your database table meets the design requirements of this homework AND the web functionality you describe on your home page matches up with your database design. Basically, think of your web application as a place where users can contribute information (whatever info you designed in your “other” table) and browse information contributed by others.
4. Database Requirements

a) Create Tables

*Design tables web_user and user_role exactly as shown below* (this is so that sample code will work against your database). Then add a foreign key from web_user.user_role_id to user_role.user_role.id. The MySQL Workbench tutorial (linked from this week's homework) should provide you with everything you need to do this.

Then *create your own unique table* (which I will often refer to as your “other” table). You’ll name this table and design its fields according to what you proposed in your last homework. Make sure that what you propose for your database table aligns with what you said your website will allow users to do. Also be sure to meet these additional requirements:

- Id: primary key, auto-increment, name this field with table name followed by “_id”
- Name or some kind of character identifier (require it to be unique, click on “UQ”)
- image URL – a long varchar since fully qualified URLs can be pretty long
- long character (description)
- at least two nullable (user optional) non-character fields, e.g., integer, decimal (for money), date
- “web_user_id”, a foreign key that points to the user who contributed this information.

You can add more fields if you like, but don’t add too many extra fields because it will be more work for you with every homework assignment. *Don’t use any SQL keywords* as table names nor field names or else your web app will have trouble trying to run various SQL commands against your table. (Google to see list of SQL keywords, but you cannot use these for sure: user, role, password, state, date.)

*Paste a screen capture of the following table designs* into your word document:

- table design of user_role (the “columns” tab of what you see when you right click a table and select “alter table”).
- table design of web_user. Then also capture the “foreign keys” tab to demonstrate that you added the foreign key from web_user to user_role.
- table design of your “other” table then also capture the “foreign keys” tab to demonstrate that you added the foreign key from your other table to web_user.
b) Add Data

Add realistic looking data into your three database tables.

- Add 3-4 records into your user_role table.
  - Since your web_user table references data from user_role, you’ll need to enter records into user_role first.
  - Try to add a record with a duplicate primary key and notice that the database management system will not let that record be inserted.
- Add 5-7 records into your web_user table.
  - At least one of these records shall have null for all nullable non character fields. At least one of these records shall have all of its fields populated.
  - Since this table has an auto-increment primary key, you do not provide web_user_id when you insert – you let the database management system do that for you.
  - Try to add a record that has an invalid (non-existent) user_role_id and notice that the database management system will not let that record be inserted.
  - Then try to delete a user_role record that has been referenced by a web_user record. Notice the DBMS will not allow you to do that either. It is the “job” of the DBMS to maintain the integrity of the DB at all times, allowing no invalid foreign key references.
- Add about 15 records into your “other” table.
  - At least one of these records shall have null for all nullable non character fields. At least one of these records shall have all of its fields populated.

c) Run SQL Select Statements

Execute each of the following SELECT statements (and paste screen captures into word doc). Each screen capture should show the Select Statement and the result set like this:

a. Run a SELECT statement that lists all the columns of all the records of your “other” table, selecting out each column and presenting them in an order that you think users would like to view the data (don’t use SELECT * ). Sort the data by whichever column you decided to show first.

b. The SELECT statement above but adding a WHERE clause so that you see some of the records but not all.

c. A SELECT statement that shows all the records from the web_user table joined with the user_role table. Show the role name first, then email address then all the other columns (from both tables) in an order you think users would like to see the data – except don’t list the role id twice. Order the records of the result set by the first columns (primary sort role name, secondary sort email address). Note: There should be as many rows in your result set as you have records in your user table. If you have a lot more (and see duplication), you have forgotten the WHERE clause that joins the two tables together.

If you don’t know how to do the above, go back and read the MySQL tutorials mentioned at the start of this document.
5. Routing for User Interface Reuse

This week’s homework (in the course web page) provides is a zip file with various folders of sample code. The folders show you how to implement your own simple “Routing Framework” using JavaScript.

- It starts out showing you how you could copy index.html to blog.html, change the content area in blog.html and get a pretty good result where you can click back and forth between index.html and blog.html. The links change as you click and the user interface looks good too. The only problem is maintenance. With this simplistic approach, if you had 100 pages in your site and someone wanted to change the title or the footer or a nav link, a programmer would have to make that same change in 100 files. Not good!

- The next sample (Routing) shows routing where your index.html file becomes a template without any actual content. You then create an array of link/content pairs and set up an event handler that invokes a function whenever the active link changes. Based on the active link, you have code that injects the correct content based on the active link. This is a very simplistic example because “the content” being changed is just a character string, not complicated HTML code as would be more realistic.

- The following example (Routing Encapsulated) is the same but pulling out the JS code into a separate file and making it reusable (consumer/producer style) meaning that the code in the JS file does not reference anything in the HTML page, except what has been passed to it as a parameter.

- The next example (Routing Components) uses “components” whereby you have a separate js file, say home.js that runs whenever the home link is clicked. When home.js runs, it injects more realistic (more complex) content. Then there is an encapsulated version of this code also.

- The final example (Routing Components Encapsulated With DropDownMenus) just shows that there is no problem trying to use drop down menus with the encapsulated routing component code.

6. Web Site Homework Requirements

- Your home and blog links shall work when clicked, implemented using encapsulated routing components (as shown in the latest code examples mentioned above). This means you’ll have a JS file named home.js and another named blog.js that shall be invoked (upon click), injecting the proper content into the content area. When you click on home, you should just see your web site functionality overview enticing users to visit the site. When you click on blog you should see a page that has two blog entries (one for homework 1 and another for homework 2).

- Your site shall continue to meet last week’s requirements (drop down menus, images, fixed titleNav and footer, fluid layout, etc.). Check last week’s homework if you need to. Improve your layout if you can.

- Project Organization. Your project shall be organized as shown below:

```
- Web Pages
  - WEB-INF
  - Icons
  - Js
    - components
      - blog.js
      - home.js
      - dropdownFw.js
      - routeFw.js
  - pics
  - style
  - index.html
```
• **Blog Page.** Your project shall be organized as shown below:

  • **In the first blog entry**, put the bottom part of the content for your web page for your first homework (your web experience, what you learned, and what you found easy or hard about the first homework).

  • **In the second blog entry** (for this week’s homework), describe your database experience and link to your word document (that shows your Database/SQL work). To link to the word document, simply place it in your local web root folder, reference the file name in the href attribute of the `<a>` tag, (test it locally) and then include that file when you transfer the files to the web server for publishing. Then (still in the second blog entry) tell what you learned in this assignment (both the database part and the web part) and what you found easy and hard.

You may copy/paste from below to use as a template for your blog content if you like:

```html
<h3>Homework 1: Web Home Page</h3>
<p>
  My web experience consists of ...
</p>
<p>
  What I learned in this homework was … [even those with some web design experience should have dug in and learned something new]. I found … a little confusing/difficult, but … was pretty straightforward and easy.
</p>

<h3>Homework 2: Routing and DB</h3>
<p>
  My database experience consists of ...
</p>
<p>
  In the database part of this assignment, I learned ... I found [then talk about what was easy versus hard/confusing]. Click `<a href="db.docx">here</a>` to see my database work. [Remember to put the word document in the web root folder and publish it].
</p>
<p>
  In the web/routing part of this assignment, I learned [then say what was easy/confusing].
</p>
```
7. Submission Requirements

- Make sure your project is organized as shown in the diagram shown in a previous step.
- As usual, test your web site locally (including clicking from home to blog to home, then clicking from blog to the database document) – all links should work properly.
- Then publish, then test all links again.
- Then submit a zip file of your whole web project folder into Canvas.

8. Homework Grading – Sample Deductions

- **Up to -5 for Data Model:** We will check that your data model meets all of the requirements listed in this document (requirements such as data type, PK, FK, null-able, unique). This is very important, since you will lose points in many future homeworks and the project if your data model is not as specified.
- **Up to -4 for SELECT Statements** as prescribed in this document.
- **Up to -2 for Realistic Data:** You were asked to enter realistic data so that your web application looks good when it begins to display data on its pages.
- **Up to -3 for Home Page Content:** We are looking for quality of “marketing material” that would be acceptable by a “real company” that might be paying you to create their web site. The marketing material (what you say your web site will do) must be able to be supported by your choice of layout for your “other” database table.
- **Up to -8 for Lack of Originality of Project and/or Data Model:** As in most homework assignments, points will be deducted if your submission is too similar to the sample(s) provided or to another student in the class.
- **Up to -2 for Look and Feel:** Your layout should not render slowly (e.g., due to too large image files), should respond well when you narrow and expand the width of the browser, and should meet requirements for your first homework (e.g., all text legible, no text close to visible edges, not overly similar to the sample code).
- **Up to 2 for Syntax Errors:** When we View Source (in Firefox), we should not see any red syntax error messages.