Lab: Project Proposal & Data Model

1. Overview

In this lab, you will:

1. Learn the basics about databases: designing tables and relationships, primary keys, foreign keys, database constraints, and SQL SELECT statements that can join data from more than one table.
2. Come up with a project idea for your web application (that meets project requirements).
3. Use MySQL Workbench (an open source GUI for MySql, an open source Database Management System) to design a database that supports your web application.

Before starting on this lab, read the following help documents and tutorials that are associated with this lab. I will be lecturing on concepts that are presented in these documents and you are responsible for this material.

- How to install MySQLWorkbench
- How to connect to your Temple database (how to discover your database credentials, etc)
- MySQL Workbench Tutorial 1: How to create database tables (with primary keys), enter data, and write single table select statements
- MySQL Workbench Tutorial 2 (continuation of previous tutorial): How to reverse engineer a database, create database relationships (with foreign keys), and how to write select statements that join data from more than one database table

2. Requirements for Project Proposal

Think up a creative idea for a web application project that you will be implementing for the rest of the semester. Your web application functionality will depend on the design of your database and the database must meet certain criteria so that your database works with the labs throughout the rest of the semester. Your labs will culminate in your own individual web application (by the end of the semester).

Here are the criteria for your data model. Each student's database must have four tables: a user table, a role table, "another table", and an associative table that implements a "many to many" relationship between the user table and the "other" table. While the prescribed database "shape" does limit the choices that a student can select for web application functionality, there are a lot of options that fit the requirements, as you can see below.

Possible Project Ideas

	"User table"	"Other table"	"Associative Table"		
What is the topic of the web app?	**Who are the users that contribute content?**	**What "other table" do your users interact with?**	**What do users contribute to the database?**	**What can viewers of the site do?**	**Why is this a many-to-many relationship (between the user table and the "other table")?**
Music Concerts	music band	concert hall	**Concert**: music band schedules a concert at a concert hall	search for concerts	A band can play at many different concert halls. A concert hall can host many different bands.
Wrestling Tournament Registration	wrestler	tournament	**Registration**: wrestler registers for a tournament	see who's registered for what tournaments	A wrestler can register for many tournaments. A tournament has many wrestlers registered.
Home Improvement Quotes	home owner	contractor	**Quote**: home owner requests quote from a contractor	see what quotes have been given to whom and for how much	A home owner can request quotes from several contractors. A contractor can provide quotes for several home owners.
course registration system	student	course	**Registration**: student registers for a course	generate class lists, create student schedule.	A student can take many courses. A course has many students enrolled.
Library Borrowing System	library patron	book	**Borrow**: library patron borrows a book	see what books have been borrowed and by whom	A library patron can borrow many books. A book can be (over time) borrowed by many library patrons.
Ecommerce	customer	product	**Purchase**: customer purchases a product	see who's bought which products.	A customer can purchase many products. A product can be purchased by many customers.
Sports Blog	sports fan	sports team	**Comment**: sports fan comments on a sports team	see other people's comments.	A fan can comment on many teams. A team can be commented on by many fans.
eCookbook	cook	food item (e.g., brownie)	**Recipe**: cook contributes a recipe for a food item	search for recipes	A cook can contribute recipes for many food items. A food item can have many different recipes.
Employee Benefit System	employee	benefit	Selected **Benefit**: employee selects a benefit	see who's signed up for which benefits	An employee can select many benefits. A benefit can be selected by many employees.
Travel Blog	traveler	country	**Trip**: traveler shares experiences about a trip to a country	search for travel experiences	A traveler can visit many countries. A country can be visited by many travelers.

So that you have a better understanding of your web application will do by the end of the semester, here is a list of the things that you will have done to your web application, once you have completed all the labs.

1. Home page with navigation bar and content that describes what your web application has to offer.
2. List of users (joined with user role table), ability to insert (register), edit, and delete users.
3. List of “other” records, ability to insert, edit, and delete.
4. List of associative records, joined with user data and “other” data.
   - ➔ Ability to search for records associated with just a particular user and/or just a particular “other” record and/or attributes from associative table (e.g., date range).
5. Log on and log off functionality. Restricted access to a page within your web site.
6. (You will also have a "labs" page, but this is more related to lab and project grading, than it is a page you would see on a real web site.)

If you would like to see some examples of web applications that have been implemented in previous semesters, visit my web site (google “sallyk temple”) and click on the “Teaching” tab. Look for links the web applications that have been done by students in previous semesters.

Once you have come up with a topic for your web application (such as those listed on the previous page), **answer the following questions about your proposed web application**. These answers will go into a document that you will submit for this lab. Most of these questions have been already answered for the project suggestions on the previous page, so please compare your answers with those – to make sure that you fully understand the basic concept of a “many to many” database relationship.

1. What is the title of your web application?
2. Who will be users of your web application?
   - What attributes will you store for each of your users?
3. What “other” table will be part of your database?
   - What attributes will be stored for each record of this table?
4. What information will be stored in your associative table (the table that implements a "many to many" relationship between your "other table" and your user table)?
5. Explain how there is a many to many relationship between your user table and your “other” table.
6. What functionality will your web application provide? Your application will provide the following at a very "mechanical" level, but functionally, how will users benefit from your web app?
   - Users will be able to register, log on, and log off.
   - Users will be able to insert (edit and delete) "other" records (whatever you decided to keep in your "other" table.
   - You can pretend that users will be able to insert (edit/delete) associative records (even if you don't choose to do the "insert associative" challenge).
7. Write 1-2 paragraphs of “marketing material” that attempts to entice people to become registered users and/or viewers of your web site. This will become the content area of your home page. In your paragraphs, do not explain to me what your web site will be able to do, write text that will entice users to visit your site.

3. Requirements for Data Model

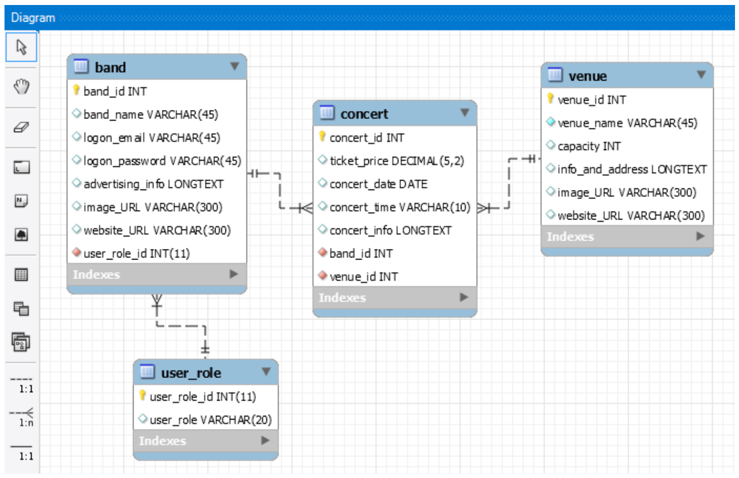
Your database that you model shall consist of 4 related database tables (as specified below).

IMPORTANT: ***Do not use any SQL KEYWORDS*** as table names or field names. Google "SQL keywords" to see what to avoid, but you definitely cannot use these: user, role, password, grant, date… You will get lab deductions, but more importantly, you will have problems as you attempt to implement your web application.

1. A **user role table.** Your user role table shall have at least these two fields:
    - role id (PK, not auto-increment, integer or small varChar storing values like "ADMIN", "VIEW", etc.)
    - role name (must be unique - add database constraint).
2. Some kind of **user table** (named appropriately, depending on who your users are, like customer, or traveler). This table shall include these fields:
    - auto-increment primary key (to uniquely identify a particular user record),
    - logon name (typically it is an email address, must be unique – add database constraint),
    - logon password,
    - screen name,
    - foreign key that references the user role table,
    - ***At least one more field*** (you choice) that is a ***null-able non-character field*** (e.g., date, decimal, or integer). Null-able means that it is OK for the user to not put something into that field - it is optional for the user.
        - Decimal is a good choice for money type fields.
        - Pick Date over DateTime or else you will get an unintelligible real number that stores milliseconds form the beginning of time.
3. An **"other" table** named according to what you will store in it (can't be named "other") and including these fields:
    - auto-increment primary key (to uniquely identify a particular "other" record),
    - descriptive character field (must be unique – add database constraint),
    - ***At least two more fields*** (you choose), ***one of which must be a null-able non-character field*** (e.g., date, decimal, or integer). Null-able means that it is OK for the user to not put something into that field (optional for the user).
4. An **associative table** that implements a "many to many" relationship between your user table and your "other" table. This table shall be named according to what you will store in it (can't be named "associative"). If you think of your user table as the subject of a sentence and the "other" table as the object of the sentence, then your associative table describes the verb within the sentence. Attributes might be something like "number of items purchased", "when purchased", "discount amount", etc.
    - auto-increment primary key (to uniquely identify a particular associative record),
    - foreign key that references the user table,
    - foreign key that references your "other" table,
    - **At least two more fields** (you choose), ***one of which is a null-able non-character field*** (e.g., date, decimal, or integer). Null-able means that it is OK for the user to not put something into that field (optional for the user).
5. **Naming conventions**:
    - Every table shall have a **PK** that is named: tableName_id
    - All FKs shall be named tableName_id (referring to the tableName that they are pointing to)
6. ***Somewhere in your user table and/or your "other" table*** (didn't say associative table) shall be the following:
    - **at least one optional Date type field**, (e.g., you didn't click "Not Null" in the Table Design Screen).
    - **at least one optional Decimal** (usually a money amount , don't click "Not Null") **or an optional integer**.
    - **at least one LongText**.

4. Examples of Data Model Screen Captures

Here is an example of how your **data model** might look: **band** is my *user table*, **venue** is my *"other" table*, and **concert** is my *associative table*. Remember that you can't promise user functionality unless your database design can support that functionality. Also, remember to **save your data model** so you don't have to keep creating it over and over – it will be stored as a ".mwb" file. If you later double click on it, the data model will open up in MySQL Workbench.



This is a screen capture of **"table design"** (right click on table name and select "Alter Table").

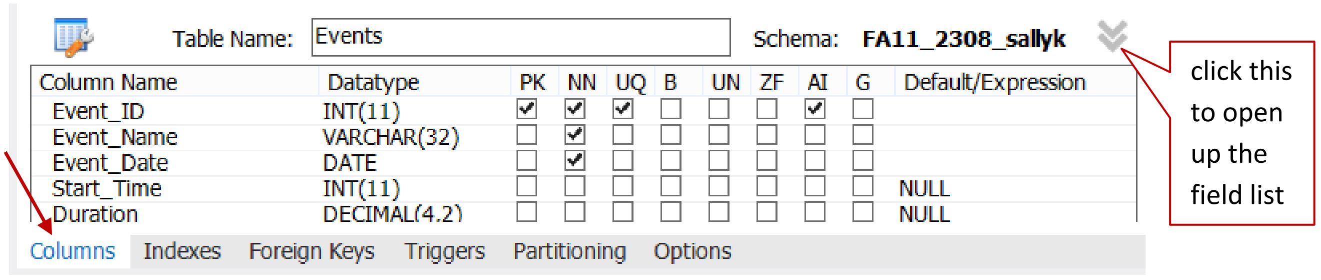


Table Name: Events   Schema: **FA11_2308_sallyk**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Event_ID	INT(11)	✓	✓	✓				✓		
Event_Name	VARCHAR(32)		✓							
Event_Date	DATE		✓							
Start_Time	INT(11)									NULL
Duration	DECIMAL(4,2)									NULL

Columns Indexes Foreign Keys Triggers Partitioning Options

click this to open up the field list

This is a screen capture of **"foreign keys"** within table design (click on "Foreign Keys Tab: from "Alter Table").

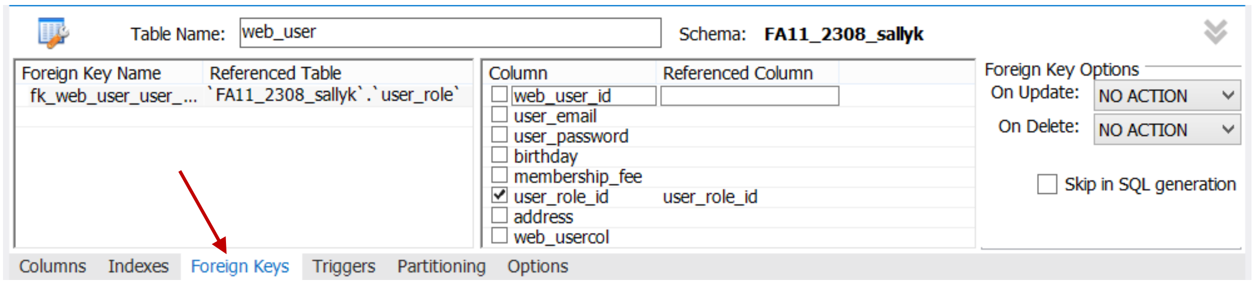


Table Name: web_user   Schema: **FA11_2308_sallyk**

Foreign Key Name	Referenced Table
fk_web_user_user_...	\`FA11_2308_sallyk\`.\`user_role\`

Column	Referenced Column
☐ web_user_id	
☐ user_email	
☐ user_password	
☐ birthday	
☐ membership_fee	
✓ user_role_id	user_role_id
☐ address	
☐ web_usercol	

Foreign Key Options
On Update: NO ACTION
On Delete: NO ACTION
☐ Skip in SQL generation

Columns Indexes Foreign Keys Triggers Partitioning Options

5. Submission Requirements

There are two deliverables for this lab (and they are both to be submitted into blackboard).

1. A word or rtf document (named with your last name in it please) that contains:
    - The (red) questions about your proposed web application (copied from section 2 of this document) and your answers to these questions.
    - A screen capture of your data model (created in MySQL Workbench) that meets the requirements of this document.
    - For each of the four tables you designed: a screen capture of its **table design** and (if it has a foreign key) a screen capture of the **foreign key** tab of its table design.

    To get a screen capture, you can do an Alt-PrtSc (copies active window into the clipboard), paste into MsPaint (or any other image editor), then copy just the part needed into the document.

2. The actual data model file (".mwb" file) that you saved from MySQL Workbench when you created your data model. Name it as you wish.

Attach the document and data model file to blackboard (click on the title of the lab in blackboard, scroll down, click on "browse my computer", attach the more files, then click submit.

For this lab (to facilitate faster feedback), I also ask that you hand in a printout of your word document in lecture (not lab).

6. Lab Grading

- **Originality of Project and Data Model:** As in most lab assignments, points will be deducted if your answer is too similar to the sample or sample(s) provided. In this lab, I gave a lot of project ideas, so it's OK if your web application proposal is similar to one of them. However, your data model cannot be overly similar to the one I provided as an example.
- **Check list:** We will check that your data model meets all of the requirements listed in section 3 "Data Model". This is very important, since all of your future labs depend on your database design. You had to have three null-able non-character fields (1 a date, 1 an integer or decimal
- **Consistency:** In this particular assignment, your data model has to be able to support the functionality that you say your web application will offer and this functionality has to align with the "eventual functionality of your web application" (as described above).
- **Professionalism:** When you answer the questions about your proposed web application, especially the last question which asks for "marketing material", we are looking for quality that would be acceptable by a "real company" that might be paying you to create their web site.
- **Timeliness:** To avoid late penalty, be sure to submit your work into blackboard by the due date and hand in a print out of the word document (in lecture).

7. Suggested Approach

1. Install MySQL Workbench on your laptop or visit a CIS dept lab PC. Connect to your database and practice with MySQL Workbench while you read the two tutorials that are associated with this lab. You need to understand basic database concepts before you attempt data modeling.

2. After reading the "Possible Project Ideas" (above), come up with your project idea, draw your data model on paper and make sure it meets all the requirements of section 3 - especially the requirements about the null-able non-character fields, etc. Then answer the questions (the red ones in this document).

3. When you are ready to use MySQL Workbench to create your data model, use the two tutorials (from this lab) for help. Create all of your tables (without entering any foreign key fields), then reverse engineer (an empty data model). Add the foreign key relationships (from data modeling tool), then synchronize the changes from the data model to your database. Save the data model into a file (will be submitted for the homework), get all the screen captures you need from MySql Workbench.