EECS118
Fall 2019
Mini Project 3 - Web and Database Programming
Assigned on: 10/17/2019

In this mini-project, you will learn how to create a web application that can interact with a database. The project is due on Thursday, October 31, 11:59PM.

Deliverables
A single .py file that is the webpage you write to interact with MySQL. A zip file if you have multiple .py files.

STEP 1 - Create a Database and Tables
The schemas below are assumed for the data in a gallery. There is a total of 4 tables. The name of the database should be “gallery”.

gallery (gallery_id, name, description)
primary key(gallery_id)

image (image_id, title, link, gallery_id, artist_id, detail_id)
primary key(image_id)

artist (artist_id, name, birth_year, country, description)
primary key(artist_id)

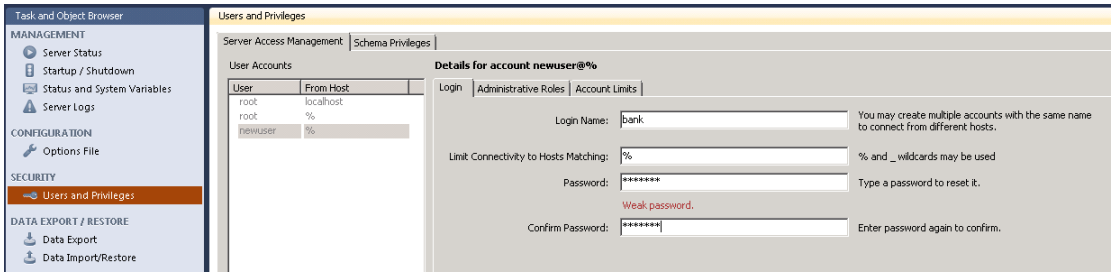
detail (detail_id, image_id, year, type, width, height, location, description)
primary key(detail_id)

This is a gallery database and you are going to write a management system. Each gallery contains a number of artworks that are stored as images. Each tuple in the image table stores the title of an artwork, the link of the picture of the artwork, and the ids that can connect the artwork to the gallery, the artist and details.

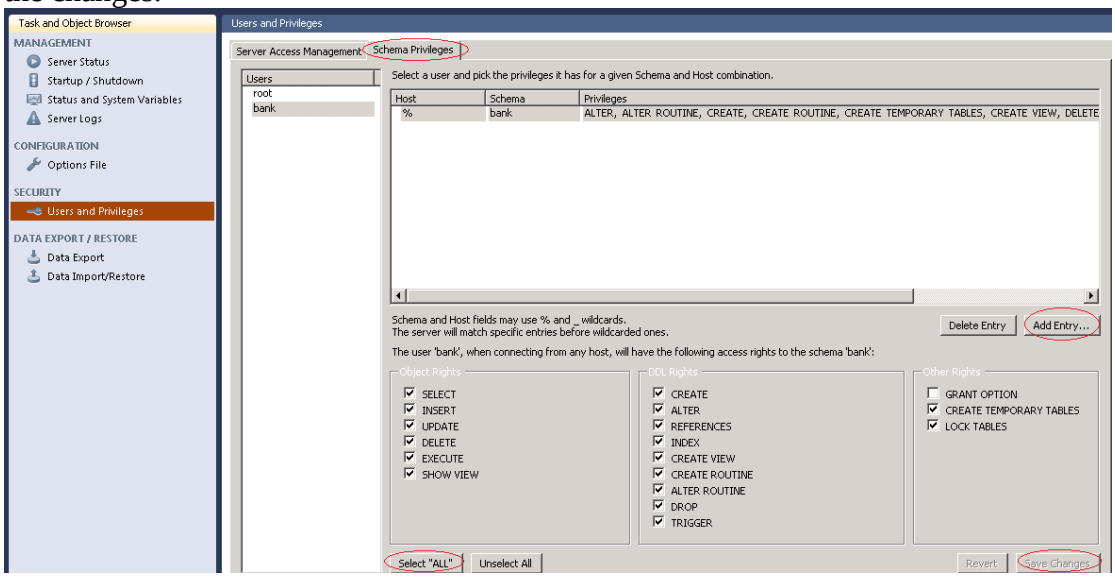
Each tuple in the artist table stores the details of an artist. Each tuple in the detail table stores the details of the artwork, including the year of creation, type of artwork (e.g., painting), the current location of the artwork (to be simple, just country), and descriptions.

First, we need to create a gallery schema in the management system. Open your MySQL Workbench, under the schemas tab on the left, right-click and select “create schema”. The name the new schema should be “gallery”. Click “apply”. Right click on the gallery schema you just created, select “Set as default schema”.

Now we are going to create a user account that can access this schema. click on the management tag on the left. Select “User and Privileges” and add a new account with the login name “gallery” and password “eecs118”.



Click on the “Schema Privileges” tag; click on the user “gallery” and then “Add Entry…”; and select the schema “gallery”. Click on the “Select All” button and save the changes.



Details of the tables

```
CREATE TABLE `artist` (
  `artist_id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(45) DEFAULT NULL,
  `birth_year` int(11) DEFAULT NULL,
  `country` varchar(45) DEFAULT NULL,
  `description` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`artist_id`)
) ;

CREATE TABLE `detail` (
  `detail_id` int(11) NOT NULL AUTO_INCREMENT,
  `image_id` int(11) DEFAULT NULL,
  `year` int(11) DEFAULT NULL,
  `type` varchar(45) DEFAULT NULL,
  `width` int(11) DEFAULT NULL,
  `height` int(11) DEFAULT NULL,
  `location` varchar(45) DEFAULT NULL,
  `description` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`detail_id`)
) ;

CREATE TABLE `gallery` (
```

```
  `gallery_id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(45) DEFAULT NULL,
  `description` varchar(2000) DEFAULT NULL,
  PRIMARY KEY (`gallery_id`)
) ;

CREATE TABLE `image` (
  `image_id` int(11) NOT NULL AUTO_INCREMENT,
  `title` varchar(45) DEFAULT NULL,
  `link` varchar(200) DEFAULT NULL,
  `gallery_id` int(11) DEFAULT NULL,
  `artist_id` int(11) DEFAULT NULL,
  `detail_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`image_id`)
) ;
```

No foreign keys or triggers are used in the schema.

You will have to maintain the data consistency of your application. Make sure that the names of the schema, tables, attributes and the account you create are exactly the same as what is in the descriptions.

STEP 2 – Design and Implement the Web Application

We have learned how to interact with MySQL from Python and how to set up and create a web application in previous mini-projects.

In this project, you are strongly encouraged to use CSS to design the layout of your website.

STEP 3 – Requirements

In this project, you are required to write the following functions in your web application:

1. List all the galleries (including names and descriptions).
2. List all the images and the number of images in a gallery on a new webpage (including title and link).
3. List the details of a given image (This function should be a link from the result of function 2. It should show the picture, the artist's name, and all the details of the artwork.)
4. List the details of an artist. (This function should be a link from the result of function 3.)
5. Create a new gallery.
6. Create a new artist.
7. Add a new image to a gallery. To simplify, you can just input the link of the picture instead of actually uploading it. This function should also input the artist_id, and input all the details of the image. Hint: max() function in SQL can be used to retrieve the latest auto-generated ID.

8. Delete an image from the gallery. You should also delete the details of the image but not the artist.
9. Modify the details of an image (including title and link).
10. Modify the details of an artist.
11. Modify the title and description of a gallery.
12. Find the images by type (“Find” means to list all the results.)
13. Find the images by a given range of creation year.
14. Find the images by artist name.
15. Find the images by location.
16. Find artists by country.
17. Find the artists by birth year.
18. Display variable greeting text (e.g. Good morning/afternoon/evening) depending on the time of the day.

Extra Credits:

(A) (7%) Assemble the above functions and design a modern album/gallery interface. For example, when listing the gallery, also show the first picture of that gallery, and you can just click on it to go into the gallery. When in the gallery, you can see the thumbnails of all the images. The search functions can be merged in a user-friendly search interface. The following picture is a good example:

(B) (3%) When adding an image, you can actually upload the image instead of just a url link. You should put the uploaded images under the /image subfolder, and, if necessary, write descriptions to explain how to make this function work if your program needs some other tools.

Please turn in your Python file in the EEE dropbox under mp3. The name of the file should be “xxxxxxxx_index.py”, xxxxxxxx being your student id. If you have more than one file to turn in, you should archive the files into "mp3-xxxxxxxx.zip", xxxxxxxx being your student id.

The procedure of our testing

We will put your web application on our own server. Make sure to use a relative URL for any link of local resources. The test will start with an empty database. Then we will try to add new galleries, new artists, and new images. If your application doesn’t

support uploading images, we will use some existing external URLs of images.

We will only input normal data and will not try to input a value that may exceed the column size. However, if you implement data validation that will be great, as it can be a part of extra credits in your UI.

After adding the items, we will then test the search functions, modification functions and delete functions. If you have any special requirements for your application to run correctly, please contact the TAs.