Project 1: Setting Up Container and MySQL Database

Overview

In Project 1, you are asked to perform the following tasks:

1. You will have to set up our project development environment on your machine, which is based on Docker.
2. You will have to brush up with Unix Command-Line Interface (CLI).
3. You will have to learn how to create tables in MySQL and populate them with data.

After setting up the development environment, you will create a few tables to contain information about movies and actors, load them with (real) data we are providing, and run a few queries to get familiar with MySQL.

Part A: Setup Your Development Environment

Docker is a popular software development and deployment tool that implements *container technology*. It allows allows running multiple isolated OS environments on a single machine. As the first task of Project 1, you **must** go over the following tutorial and follow its step-by-step instruction:

- Docker Setup and Basic Use

The page will guide you setting up the Docker App on your machine and teach the most important concepts and commands for using Docker. For the first three projects, we will use the `mysql-apache` container that we prepared for this class. As described in the Docker tutorial, download the container image and run it on your machine through the following command (if not done yet) replacing `{your_shared_dir}` with your directory of choice, say, `/Users/cho/cs143/`):

```
$ docker run -it -v {your_shared_dir}:/home/cs143/shared -p 8888:80 --name mysql-a
```

The Container image has MySQL, Apache2, PHP, Python, and Node installed and the underlying operating system is Linux, a variant of Unix. For the most part, therefore, you will be using a Unix shell through a terminal app. If you are not familiar with Unix shell commands, read the Unix Tutorial for Beginners and learn the basic Unix commands. Your username inside the container is "cs143" with password "password".

Part B: MySQL Database

In this part, you get yourself familiar with the basic MySQL commands by creating and loading a table and issuing a few queries. Start with reading our Introduction to MySQL. Try the examples in the document, and experiment with the help command to get familiar with the MySQL command-line interface.

Description of Our Dataset and Table Schema

The dataset needed for Homework 1 can be downloaded from data.zip. Download the file, and unzip it in the shared directory.

Once unzipped, browse the data files with your favorite text editor (or the Unix command `less [filename]` inside the container) to see how fields are enclosed and delimited, how NULL values are represented, and so on.

Here is a brief description of the tables that you will have to create for this project and the data files that contain the tuples for the tables.

The Movie table: This table describes information regarding movies in the database. It specifies an identification number unique to each movie, the title of the movie, the year the movie was released, the MPAA rating given to the movie, and the production company that produced the movie. The schema of the Movie table is given as follows:

```
Movie(id, title, year, rating, company)
```

Name	Type	Description
id	INT	Movie ID
title	VARCHAR(100)	Movie title
year	INT	Release year
rating	VARCHAR(10)	MPAA rating

company	VARCHAR(50)	Production company

In all our table description, the primary keys are underlined unless the primary key is not defined for the table. The load file for the table is `movie.del`.

The Actor table: This table describes information regarding actors and actresses of movies. It specifies an identification number unique to all people, the last name of the person, the first name of the person, the sex of the person, the date of birth of the person, and the date of death of the person if applicable. The schema of the Actor table is given as follow:

```
Actor(id, last, first, sex, dob, dod)
```

Name	Type	Description
id	INT	Actor ID
last	VARCHAR(20)	Last name
first	VARCHAR(20)	First name
sex	VARCHAR(6)	Sex of the actor
dob	DATE	Date of birth
dod	DATE	Date of death

There are three load files for the table: `actor1.del`, `actor2.del`, and `actor3.del`. You will have to load each file only once to the table.

The MovieGenre table: It describes information regarding the genre of movies. It specifies the identification number of a movie, and the genre of that movie. The schema of the MovieGenre table is given as follow:

```
MovieGenre(mid, genre)
```

Name	Type	Description
mid	INT	Movie ID
genre	VARCHAR(20)	Movie genre

The load file for the table is `moviegenre.del`.

The MovieActor table: It describes information regarding the movie and the actor/actress of that movie. It specifies the identification number of a movie, and the identification number of the actor/actress of that movie. The schema of the MovieActor table is given as follow:

```
MovieActor(mid, aid, role)
```

Name	Type	Description
mid	INT	Movie ID
aid	INT	Actor ID
role	VARCHAR(50)	Actor role in movie

The load files for the table are `movieactor1.del` and `movieactor2.del`. You will have to load each file only once.

The Review table: In Project 2, you will create a Web interface where the users of your system can add reviews on a movie (similarly to Amazon product reviews). The Review table stores the reviews added in by the users in the following schema:

```
Review(name, time, mid, rating, comment)
```

Name	Type	Description
name	VARCHAR(20)	Reviewer name
time	DATETIME	Review time
mid	INT	Movie ID
rating	INT	Review rating
comment	TEXT	Reviewer comment

Each tuple specifies the name of the reviewer, the timestamp of the review, the movie id, the rating that the reviewer gave the movie (i.e., x out of 5), and additional comments the reviewer gave about the movie.

Since this data will be added by your users, there is no load file.

Loading and Querying the Dataset

Now that you understand our provided dataset, it is time to create the tables in MySQL. Write a SQL script named `create.sql` that creates all tables above. If you are not sure how to create a table in MySQL or how to write and run a SQL script, please read our Introduction to MySQL again. In creating the tables, please make sure that their schemas are ***exactly as we specified above including their cases***, because having the same schema among all students is essential for the correct functioning of later project.

Note that in our container, we have created database `class_db` (note these database names are case sensitive) in MySQL. Use this database to create tables and load data. We also have created the user `cs143` in MySQL as well (with no password) that has the full permission to the cs143 database.

Once you the finish writing `create.sql`, create the tables in the `class_db` database, using a command like

```
$ mysql class_db < create.sql
```

inside the container (or using `SOURCE` command in `mysql`).

Notes on CR/LF issue: If your host OS is Windows, you need to pay special attention to how each line of a text file ends. By convention, Windows uses a pair of CR (carriage return) and LF (line feed) characters to terminate lines. On the other hand, Unix (including Linux and Mac OS X) uses only a LF character. Therefore, problems arise when you are feeding a text file generated from a Windows program to a program running in our guest OS (such as `mysql`). Since the end of the line of the input file is different from what the tools expect, you may encounter unexpected behavior from these tools. If you encounter this problem, you may want to run the `dos2unix` command from the guest OS on your Windows-generated text file. This command converts CR and LF at the end of each line in the input file to just LF. Type `dos2unix --help` to learn how to use this command.

Now that you have created tables, it is time to load data. Create a MySQL script, named `load.sql`, that loads all our provided data into the created tables. Again, if you are not sure how to load data into tables, read our Introduction to MySQL. Read MySQL Reference on LOAD DATA as well to learn more detailed options available for MySQL LOAD command. Assume that all data files are available at the current directory, like `'./movie.del'` and write your script accordingly. Make sure that the double quotes enclosing string fields in the load files are removed when inserted into a table. Load all your data into the tables by running the command

```
$ mysql class_db < load.sql
```

in the container. Remember to use the `class_db` database when you load the data. Otherwise, you will get error messages. Pay attention to CR/LF issue and run `dos2unix` command on your script files if necessary.

Now explore the dataset by running the SELECT statements that answer the following questions from your database:

- Q1: Give me the birth date of the actor with id 15914
- Q2: Give me the last names of all the actors in the movie 'Die Another Day'.

Create two files, `q1.sql` and `q2.sql` that the above Q1 and Q2 queries, respectively. That is, when we run

```
$ mysql class_db < q1.sql
```

the result of query Q1 should be printed. Feel free to run a few other SELECT queries to explore the dataset and play with our MySQL database.

Notes on editors for development: You can choose whatever editors you like for development. Options include:

- Unix text editors in the container: You may use any text editor in the container (`vi` and `nano` are available) to edit text files directly.
- Your favorite text editor on the host: You may use your favorite text editor from your host OS (e.g., Sublime Text or Visual Studio Code) and transfer the edited file to the container through the shared directory. Remember that the directory that you specified as the shared directory from the host (e.g., `/Users/cho/cs143` from Mac) is available at `$HOME/shared` in the container. Again, be careful with the CR/LF issue if you use this option.

Your Final Submission

Your project must be submitted electronically before thYour project must be submitted electronically before the deadline through GradeScope. You can submit your work an unlimited number of times. In case of multiple submissions, the grade from the latest submission will be used.

What to Submit

You must submit a file named `project1.zip`, created using a zip compression utility (like using `zip` command in the container), which has the following packaging structure.

```
project1.zip
 +- create.sql
 +- load.sql
 +- q1.sql
 +- q2.sql
 +- README.txt (Optional)
```

Here is more detailed description of each file to be included in the zip file:

- `create.sql`: The MySQL script that you used to generate the tables. Please make sure **the created tables follow the schema exactly as we specified**. Be extra careful with case sensitivity. Make sure your script runs without any error when you execute the command `mysql class_db < create.sql`.
- `load.sql`: The MySQL script that loads all tuples into the tables. Please make sure that **the data files are loaded from the current directory** like `'./movie.del'`, so that your script is able to access the files during our grading. Otherwise, your script is likely to generate error when we run it for grading.
- `q1.sql` and `q2.sql`: There are the two SQL query files that you created.
- `README.txt`: This file is optional in case you you want to include any information regarding your submission.

Please ensure that your submission is packaged correctly with all required files. Make sure that each file is correctly named (including its case) and `project1.zip` contains all files directly, not within a subdirectory. In other words, unzipping project1.zip should produce the files in the same directory as `project1.zip`.

To help you package your submission zip file, you can download and use our packaging script p1_package in our container. After downloading the script in the directory where your sql script files reside and set its executable permission, you can run it like the following inside the container:

```
$ wget http://oak.cs.ucla.edu/classes/cs143/project1/p1_package
$ chmod +x ./p1_package
$ ./p1_package
zip project1.zip create.sql load.sql q1.sql q2.sql
  adding: create.sql (deflated 63%)
  adding: load.sql (deflated 81%)
  adding: q1.sql (stored 0%)
  adding: q2.sql (deflated 20%)
[SUCCESS] Created '/home/cs143/p1/project1.zip'
```

When executed, our packaging script will collect all necessary (and optional) files located in the current directory and create the `project1.zip` file according to our specification that can be submitted to GradeScope.

Submitting Your Zip File

Visit GradeScope to submit your zip file electronically by the deadline. In order to accommodate the last minute snafu during submission, you will have 1-hour window after the deadline to finish your submission process. That is, as long as you start your submission before the deadline and complete within 1 hour after the deadline, you are OK.