Introduction to Databases
Course Project
Spring 2022

Antonio Badia

Due Date: day of last lecture

1 INSTRUCTIONS

This project should be done in MySQL. The project assumes that you have a laptop or desktop where you can download and install MySQL; if this is not the case, please contact the instructor immediately!

Information on MySQL is widely available; see `https://dev.mysql.com/doc/refman/8.0/en/` for the 'official' page. There is a short tutorial at `https://dev.mysql.com/doc/refman/8.0/en/tutorial.html`. For help downloading and install MySQL, check out `https://www.youtube.com/watch?v=LnOnzNQnJMU`, an excellent video made by a former 535 student. If this is not enough, type "install MySQL on Windows" (or "Mac" or "Linux") in YouTube and you will get plenty of help.

Please remember that this project should be your own work exclusively. You can ask anyone for help installing MySQL and troubleshooting the system, but you should work on the project questions individually; what your turn in should be the result of your individual work. This is not a group project.

- **Deliverable:** A plain SQL script. You **MUST NOT** create a full-stack app or anything of the sort. Students are expected to send a ".sql" file (which is nothing more than a text file) that creates the requested user, database, triggers, stored procedures, etc (you must ensure the user is given the right permissions to modify the database). **DO NOT** send Microsoft word (.docx, doc) files or anything similar; only SQL scripts in plain text.

- **Submit your file through the "Project" entry in blackboard's "Assignments" tab by 4/26/2022 at 5 pm EST**. The script should be named "535ProjectSpring21-[YourUofLUsername].sql". For instance, if your username is 'JoJones01', the script should be named "535ProjectSpring21-JoJones01.sql".

- **Recall that late projects are not accepted**; if you have not finished the project by the deadline, submit what you have done for partial credit. If you have any questions, send the instructor an email with the subject "CECS 535 Project." Note: emails sent during the exam week may take longer than usual to be answered. Do not procrastinate!

- **SQL script:**

  - **Read** all instructions carefully to see what you are required to do. If you need any clarifications, email the instructor ASAP.
  - **DO NOT** include any INSERT or UPDATE statements other than the ones you may need within TRIGGERS and STORED PROCEDURES. You can always insert data to test your script works as expected, but **DO NOT** include such data in the script you submit.

- **Attributes with the dash (-) character:** Round them with backtick (`) symbols to escape them (both at creation and querying). For example: SELECT t.`credit-card`FROM Table t.
- **Errors:** Use SIGNAL to raise errors.
- **Case sensitivity:** Scripts will be run on a linux environment, so **students MUST ACCOUNT FOR CASE SENSITIVITY (points will be taken off if you do not)**. That is, ALWAYS (not only at CREATE TABLE statements, but throughout the whole script) use the name of the tables and attributes **EXACTLY** as they are given in the project's description.
- **Using `mysqldump`:** The dump **WILL NOT** include any database user, so you **MUST** create one manually (see Section 2 for details about the username and other requirements).
- **Testing:** If you do not have access to a linux machine, then you can test your script using a virtual machine (look at `https://www.virtualbox.org/` or `https://www.vmware.com/products/workstation-player.html`, both are free) or a container software like docker (look at `https://www.docker.com/`, it is also free). If you use docker, look at the file `docker-compose.yml` included in the assignment page in blackboard (you can also see it in Annex B of this document). It is a docker-compose file that will download an already configured container with MySQL 8.0. You can run it using `docker-compose up -d` and access MySQL using `docker-compose exec db mysql -u user -p password` (See Annex A for more information). Naturally, you can modify the settings to your liking by editing the file. **Students are encouraged to use docker-compose because it replicates the environment that will be used to grade the projects (if your script runs in a MySQL docker container, it will run in the TA's testing environment), but the instructor and TA WILL NOT provide support for testing**. You will not be given any points by using docker, nor taken any points off by not using it.

Grading Rubric: The grading is based on: i) running and testing of your script by the TA; and ii) inspection of your code by the TA. Each project starts with 100 points; points will be deducted following these rules:

- if the script fails to execute and it must be modified to make it work (this includes case-sensitivity issues that prevent it from running without modifications), you can only receive a maximum of 60 points for the project. Your first priority should be to make sure your script runs when invoked.
- If your script requires extensive editing, it will not be fixed, and automated tests will not be run. In this case, you can only receive a maximum of 50 points.
- whether it runs or not, your script will be inspected. If you have not declared the database schema as required (including keys), you will lose up to 25 points (depending on the extent of the failure).
- triggers will be tested with a variety of data. If a triggers do not behave as required, or have not been implemented, you will lose up to 15 points per trigger (depending on the severity of the failure). Note that raising errors when appropriate is also considered as expected behaviour, thus points will be taken off if a trigger does not raise errors when expected.

2 PROJECT

1. Create a database with name `CECSProject`.
2. Inside this database, create the following tables:
   - `HOTEL`, with attributes `hotelid, address, manager-name, number-rooms, amenities`. The first one is a primary key and should be generated by the system. Attribute `amenities` gives the amenities available at the hotel, separated by commas (for instance, "pool,restaurant,gym").

- `ROOM` with attributes `number, type, occupancy, number-beds, type-beds, price, hotel-id`. The first attribute and the last one together are the primary key; the last one is a foreign key to `HOTEL`. `type` is one of 'regular', 'extra', 'suite', 'business', 'luxury', 'family'. The `occupancy` is the number of people (always 1 to 5) who can sleep on a room. `Number of beds` is always 1, 2 or 3 and `type-beds` is one of 'queen', 'king', 'full'. `price` is a number (always a positive number, of course).
- `CUSTOMER` with attributes `cust-id,name,street,city,zip,status`. The first one is a primary key and should be generated by the system. Attributes `number,street,city,zip` give the address of the customer. `status` is one 'gold', 'silver', 'business'.
- `RESERVATION` with attribute `hotelid`, `cust-id`, `room-number`, `begin-date`, `end-date` , `credit-card-number`, `exp-date` that states a customer `cust-id` has made a reservation of room `room-number` in hotel `hotelid` from `begin-date` to `end-date`, paying with credit card `credit-card-number` with expiration date `exp-date`. You have to decide on a primary key for this table, but `hotel-id` and `room-number` together must be a foreign key to `ROOM`.

You will have to pick adequate data types for each attribute. Make sure to declare all primary key and all foreign keys in order to have integrity constraints.

3. Write a trigger to make sure that on every insertion into `ROOM`, all values in the tuple are as described. If not, the insertion should be rejected.

4. Insert at least five tuples into each relation (you can make up the values, but they should be valid data, i.e. respecting all constraints).

5. Write a procedure call `Occupancy` that, given the number-id of a hotel and a date, will return the number of rooms reserved on that hotel on that date. Note that for a reservation r to be valid on a date d, it must be the case that `r.start-date` $\leq d \leq$ `r.end-date`.

6. Create a table `REVENUE(hotelid,total)` which adds up all the money that a hotel has generated so far. To populate this table initially, for each hotel multiply the rooms reserved by their price and number of days in reservation (note that `price` in `ROOM` is the daily price). Then, write a trigger that will update the `total` after each insertion into `RESERVATION`. You do not have to worry about cancellations on this project. However, you must make sure that the update to `REVENUE` is done without recomputing the table, taking into account only the new reservation and not touching any data that does not change.

A Useful docker-compose commands

To start the container run the command below in the same folder where your `docker-compose.yml` is located at:

```
docker-compose up -d
```

To import your SQL script into MySQL running on the container run the following command (MacOS and Linux only):

```
docker exec -i 'docker-compose ps -q db' mysql -ppassword < YourSQLScript.sql
```

To enter mysql shell on the container run the following command:

```
docker-compose exec db mysql -ppassword
```

or you can also run:

```
docker-compose exec db mysql -u user -p password
```

Windows users would benefit from reading this: `https://4sysops.com/archives/using-docker-compose-on-windows/`

B docker-compose.yml

```
version: '3'
services:
  db:
    image: "mysql"
    restart: always
    environment:
      MYSQL_DATABASE: 'db'
      # So you don't have to use root, but you can if you like
      MYSQL_USER: 'user'
      # You can use whatever password you like
      MYSQL_PASSWORD: 'password'
      # Password for root access
      MYSQL_ROOT_PASSWORD: 'password'
    ports:
      # <Port exposed> : < MySQL Port running inside container>
      - '3306:3306'
    expose:
      # Opens port 3306 on the container
      - '3306'
      # Where our data will be persisted
    volumes:
      - my-db:/var/lib/mysql
      # Here you also can share folders in
      # your host machine with the container. For instance, if
      # you want to share the folder /home/user/scripts with the
      # container and access it within the container at location
      # /scripts, you must add the following line:
      #- /home/user/scripts:/scripts

# Names our volume
volumes:
  my-db:
```