Introduction to Databases
Course Project
Fall 2018

Antonio Badia

Due Date: day of last lecture

This project should be done in MySQL. The project assumes that you have access to a computer in which you can download and install MySQL; if this is not the case, please contact the instructor immediately! Information on MySQL is widely available; see
`http://dev.mysql.com/doc/refman/8.0/en/`
for the 'official' page. For help downloading and install MySQL, check out
`http://dev.mysql.com/doc/refman/8.0/en/installing.html`
You can also watch
`https://www.youtube.com/watch?v=LnOnzNQnJMU`
an excellent video made by a former 535 student.

Please remember that this project should be your own work exclusively. You can ask anyone for help installing MySQL and troubleshooting the system, but what your turn in should be the result of your individual effort.

READ ALL THE INSTRUCTIONS CAREFULLY! Failure to follow instructions will result in a stiff penalty.

WHAT TO TURN IN: By 5 pm of the due date, send an email message with subject "535 Project", no content, and an SQL script as an attachment. This attachment should be called "username.sql" (i.e if your username is *Jjones01*, call the file *Jjones01.sql*). This file will be used as a script to be ran through the system, that is, it will be called with

```
shell> mysql db_name < your_file_name
```

or, within MySQL, with

```
mysql> source your_file_name
```

PLEASE NOTE: It is your responsibility to make sure the file can be run in this manner without problems. In particular, pay attention to the differences between running MySQL on Windows and on other systems; your work will be checked on a system running Linux. A few pieces of advice: MySQL is not case sensitive on Windows, but it is under Unix. Make sure that all your table names and other elements are named exactly the same in all statements. Also, generating an SQL script is much safer than cutting and pasting onto a text file. Finally, make sure that your script is self-contained, i.e. that it has all statements that it needs to run. Scripts that do not run without modification will receive a large penalty.

1. Create a database with name `usernameCECSProject` (i.e if your username is *Jjones01*, call the database `Jjones01CECS535Project`).

2. Inside this database, create the following tables:

   - `HOTEL`, with attributes `hotelid, number, street, city, zip, manager-name, number-rooms, has-pool, has-bar, has-restaurant`. The first one is a primary key and should be generated by the system. Attributes `number, street, city, zip` give the address of the hotel. Attributes `has-pool, has-bar, has-restaurant` are Booleans (i.e. 'yes' or 'no') to indicate whether the hotel has that facility.

- `ROOM` with attributes `type, occupancy, number-beds, type-beds, price`. The first attribute is one of 'regular', 'extra', 'suite', 'business', 'luxury', 'family', and a primary key. 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).
- `ROOMHOTEL` with attributes `hotelid, room-type, number`, where `hotelid` is a foreign key to `HOTEL` and `room-type` is a foreign key to `ROOM`. Each row tells how many rooms of a given type a hotel has. A hotel may have several types of rooms, and a type of room could be present in several hotels, so *you decide the primary key for this table.*
- `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 attributes `hotelid`, `cust-id`, `room-type`, `begin-date`, `end-date` , `credit-card-number` and `exp-date`. A tuple in this table states a customer with `cust-id` (so this attribute is a foreign key) has made a reservation of a room of type `room-type` (a foreign key also) in hotel `hotelid` (a foreign key too) 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.

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 (NOTE: a trigger is needed since MySQL does not support CHECK).
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. Create a table `CUSTPROFILE(cust-id,latest-stay,total)` that gives, for each customer, the start-date of its most recent stay and the total amount of money the customer has spent on our hotels. The latter is computed as follows: money-per-stay is the room price times number of days of the stay (end-date minus start-date). The total cost per customer is the sum of all his/her money-per-stays. To populate the table, do this:
   - create the table with a query that will obtain the necessary data from table `RESERVATION`.
   - create a trigger that will update the table each time there is an insertion into `RESERVATION`, as well as an update (you do not have to worry about deletions).
6. Create a view `FAVORITE(cust-id,hotel-id)` which yields, for each customer, her/his favorite hotel. The favorite hotel of a customer is the hotel where s/he has stayed the most days (not the one where s/he has made the most reservations; that is, a stay of 10 days in hotel A makes it favorite over 2 stays of 4 days in hotel B).