NYU Tandon School of Engineering
Computer Science and Engineering
CS6083, Fall 2019

October 14, 2019

Problem Set #2 (due October 25)

Problem 1:

In this problem, you have to write SQL and RA queries for a database modeling a system of weather stations that measure temperature, humidity, and precipitation (rainfall) data, given by the following very simple relational schema:

Station (sid, scity, sstate, slatitude, slongitude);
Measurement (sid, mtimestamp, mtemp, mhumid, mprecip);

For each station we have a unique sid, a city and state, and longitude and latitude coordinates. Each station performs measurements at some (regular or irregular) time intervals, and for each measurement, stores a timestamp, the current temperature (in Fahrenheit) and humidity (in percent), and the amount of precipitation (in inches) since the last measurement. In the following, when we talk about the average temperature or humidity for one station (for some period in time), this means we average over all measurements by that station during that period. We can also talk about the average temperature in a city, say Chicago, by ***averaging the average temperatures reported by all stations located in Chicago*** during the period we are interested in. For rainfall, we can get total rainfall in inches at a station in a month by adding all rainfall measurements during that month. (You should assume there are many measurements during each day, so you do not have to worry much about measurement intervals going across monthly or even daily boundaries.)

(a) Draw an ER diagram that is consistent with the above schema. Identify any weak entities.

(b) Create the above original schema in a database system, choose appropriate attributes types, and define primary keys, foreign keys, and other constraints. Data for this schema will be made available on NYU Classes; please use that data. Load the data into the database using either insert statements or the bulk-loading facilities. You may use any mainstream relational database system.

(c) Write the following SQL queries and execute them in your database system. **Show the queries and the results**:

(i) For each city, output the average temperature in the city during March 2018.
(ii) For each state, output the highest temperature ever measured during February, and the city or cities where it occurred.
(iii) Try to write a query to answer the following question: When the temperature rises in a place, does the humidity also usually rise? Or is it more likely to fall?
(iv) Output the distances between all pairs of stations located in Rhode Island. (Output the sids of the stations together with their distance, using the simplified formula for distances between close-by points at http://jonisalonen.com/2014/computing-distance-between-coordinates-can-be-simple-and-fast/.)
(v) Define a stored function distance() that computes the distance between two longitude-latitude coordinates using the above formula. Check how this is done in your database system.
(vi) Using this function, write a query to output all pairs of stations that are at most 10 miles apart and that have very different levels of precipitation. In particular, we mean stations where the average total annual rainfall differs by at least 50 inches.

(d) Write expressions in Relational Algebra for queries (i) to (iv).

(e) Write SQL statements to perform the following updates to the database:
(i) Insert a new station into the database.
(ii) Delete the station with sid 1234 from the database, together with all its measurements.
(iii) For all temperature measurements higher than 120 degree or lower than -60 degrees, replace them with NULL values as they cannot be correct.

Problem 2:

In this problem, you have to create views and triggers for the schema in Problem 1. Execute everything on your database system, design and run suitable tests, and report the results.

(a) Define a view that stores, for each city and each date, the average temperature and humidity and the total rainfall in that city on that day.

(b) Using this view, output for each day in 2018 the city (or cities, if there is a tie) with the highest temperature on that day.

(c) Write a trigger that rejects any insertion of a new measurement that is more than 10 degrees different from the previous measurement and also more than 10 degrees different from the average temperature at this station on that day over the last 10 years.

(d) Create an additional table "tempRecords", containing attributes city, day, year, and temperature, that will be used to store for each city and each day of the year, the highest temperature ever recorded in that city, and the year it occurred. Write a query to initialize this table on the data you are given. Then write a trigger that keeps this table up to date as new measurements are inserted into the database. (In the case of a tie, you may keep the older data.)

Problem 3:

In this problem, you need to design a relational database for a chain of fitness clubs that keeps track of memberships and members, and their use of the facilities, and of different locations of the chain and their hours. The chain has a number of locations identified by a unique number, each with an address and a phone number. There are also a number of different membership types that you need to keep track of. Memberships could be individual or couple or family membership type, and there are monthly and yearly membership terms. (You may assume that all memberships start at the beginning of the month or year.) Also, there could be different classes of memberships such as basic, premium, or gold. The cost of a membership would of course depend on the type, term, and class of the membership, and you need to store the current costs of such memberships, as well as information about when a membership was purchased and at what price. Note that terms such as "basic", "premium", or "gold", and even "couple" and "family", are names that could be changed or expanded in the future, so you should not hardcode them as attribute names into your relational design. For simplicity, you may assume that the cost of a membership depends on the type but not on the number of members associated with it; i.e., the cost of a family membership does not depend on the size of the family (but there might be a limit on the number of family members who are allowed to use the facilities).

For each membership, you also need to store all people (members) that are part of the membership, i.e., for a family membership the names of all family members that are allowed to use the facilities. The chain will then issue each member an ID card that they have to use to swipe in and out of the locations, and you need to store sufficient information for each swipe. Swipes may happen at any time during the opening hours, not just at full hours. Members are allowed to use any location of the chain. However, different classes of membership may entitle you to different amounts of usage. In particular, each location may split their opening hours into peak and standard hours. Different classes of membership may impose different limits on usage of the facilities. For example, a basic membership might give people access for up to 20 hours per

month during standard hours but no access during peak hours, and a premium membership might allow unlimited standard hours and up to 10 hours of peak usage, while a gold membership might give completely unlimited usage. (These are examples, so you should not hardcode these exact limitations, but store them in the database. However, you may assume in your design that there are only two types of hours, standard and peak hours, though the schedule of these might vary between locations and you need to store this schedule as part of the opening hours for each location.)

There are a couple of issues that you DO NOT have to model in your design. First, when a membership is purchased, the company may check if the members qualify; e.g., for a family membership it might be required that all people live at the same address, and there might be a limit such as at most six family members being able to use the facilities. There might be cases where a member checks in and stays until their usage limits are exceeded (e.g., a basic member uses up all their 20 hour or stays after peak hours start). We assume the company has some way of dealing with this, say by notifying the member that they may have to leave or pay extra. You also do not have to model cases such as members paying for extra hours beyond their membership class, or paying for special fitness classes, even though a real business would probably offer these as choices.

(a) Design an ER diagram that models the above scenario. Identify suitable keys and the cardinalities of the relationships. Also identify any weak entities. Discuss any assumptions that you are making in your design.

(b) Convert your ER diagram into a relational schema. Show primary keys and foreign key constraints.

(c) Write statements in SQL for the following queries. Note that if your schema does not allow you to answer a query, you may have to go back and change your design.

(i) Output the average cost that was paid for monthly memberships valid for April 2016. (Note that these memberships may have been bought before April 2016.)

(ii) For each location, output the number of people that were using the facilities at 7:30pm on September 23, 2018.

(iii) Output the IDs of all locations where 4-5pm falls into the standard hours.

(iv) For each membership that was valid in August 2019, output the membership ID, the number of people included in the membership, and the number of standard hours used during August 2019.

(d) Create the tables in your database system, and insert some sample data (maybe 5-10 tuples per table). Choose an interesting and meaningful data set. Then execute the queries from part (c). Submit your sample data, and screenshots or logs of the executed queries and the relative outputs.