NYU Tandon School of Engineering **March 1, 2020**
Computer Science and Engineering
CS6083, Spring 2020

Problem Set #2 (due March 13)

Problem 1:

In this problem, you have to write SQL and RA queries for the bakery database already seen in class. Here are the tables:

Customer (custid, custname, ccn, cphoneno, address, city, zip);
Cake (cakeid, cakename, slices, status, price);
Ingredient (ingredid, iname, price, available);
Contain (cakeid, ingredid, qty);
Orders (custid, cakeid, ordertime, pickuptime, pricepaid);

The bakery offers different types of cakes. A cake has a name, current price, and number of slices (servings) per cake. Customers need to make orders one day or more in advance, and then pick them up on the agreed day. Each cake contains a number of ingredients (e.g., flour, sugar, eggs) that are needed to make it. The *Contain* table states how much of each ingredient is needed for each cake, while the *Ingredient* table stores the cost per pound, and the number of pounds available, of each ingredient. Finally, table *Cake* has a field *status* that can be set to either *available* or *discontinued*, depending on whether a cake is still being offered, or has been discontinued. The *ordertime* and *pickuptime* attributes should store both time and date information.

(a) Draw an ER diagram that models this relational schema. Identify any weak entities, and the cardinalities of all the relationships.

(b) Create the above 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, so use that data. Load the data into the database using either insert statements or the bulk load facilities. You may use any relational database system, as long as it supports basic SQL, views, and some sort of triggers (needed below).

(c) Write the following SQL queries and execute them on your database. Show the queries and the results:
(i) For each cake, list the number of orders for this cake made during 2014.
(ii) Output the cake that is currently the most expensive per slice.
(iii) List any cake that was ordered less than ten times in 2014.
(iv) Output the names of any cakes whose current price is less than twice the total cost of its ingredients.
(v) List the ID and name of any customer from Chicago who has ordered a cake containing peanuts.
(vi) Output the names of any pairs of cakes that share at least three ingredients (even if they use different amounts of these ingredients).
(vii) For each ingredient, list how many pounds were used for cakes ordered in May 2015.
(viii) Output the name of any customer who has ordered every currently available cake at least once.

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

(d) Write SQL statements to perform the following updates to the database:
(i) For any cake currently costing less than twice the cost of its ingredients, raise its current price to twice the cost of ingredients.

(ii) For every costumer who spent at least $100 on cakes ordered during 2015, give them a free *apple pie* by inserting a new tuple into the *Orders* table with a price of $0 and a pickup date of October 10.

(iii) For any cake that has not been ordered for 6 months, set the status of the cake to *discontinued*.

(e) Consider query (ii) of part (d) again. Can you implement these tasks via triggers, so that an order for a free apple pie is automatically generated as soon as a customer reaches $100 of orders during 2015? How about query (iii)? Can you get the system to automatically change the cake status somehow? Implement and execute these tasks, or discuss why you cannot do this.

Problems 2:

In this problem, you have to create views and then write queries on the views, for a Video Rental Chain database that will also be discussed in class. The tables will be made available on NYU Classes, and you have to execute them on your database system.

Customer (cid, came, caddress, cphone, ccn);
Movie (mid, title, genre, year);
Branch (bid, bname, baddress);
Copy (copyid, mid, bid);
Rental (cid, copyid, outdate, returndate, cost);

(a) Define a view that contains for each branch and each movie the information about how many copies of the movie are currently available in that branch (not checked out). The view should have the attributed *bid, mid, title,* and *numcopies*. Using this view, answer the following queries:

(i) Output the *bid* of the branch that has the most copies of the movie *Godfather* available.

(ii) For each movie, output the total number of copies available overall (across all branches).

(b) Define a view on top of the Movie table that contains only the *mid, title,* and *year,* but not the *genre*. Then write queries *using only this view*, or explain if you cannot do so:

(i) Add a record to the underlying *Movie* table with title *Minions*, *mid 6672*, and *year 2015*.

(ii) Delete all movies made in *1966* from the *Movie* table.

(iii) Delete all movies from the genre *science fiction* from the *Movie* table.

(iv) For each year, list the number of movies from that year.

Problem 3:

In this problem you have to design a database for the NYC Department of Health and Mental Hygiene (DOH) that models restaurant inspections and the grades that restaurants get when they are inspected. Here is the somewhat simplified scenario you have to model: DOH employs inspectors that visit restaurants unannounced to check if the restaurant is clean and if the food is handled safely. Each inspector has an ID, a name, and a date when she was hired. Each restaurant has a unique ID, a name, an address, a phone number, and an owner (a person or company responsible for it). Owners have IDs, names, and a contact phone number. An owner may of course own a number of restaurants.

When an inspection occurs, the inspector will check for certain *violations*, for example “restroom dirty”, “meat is stored outside refrigerator”, “rats in kitchen”, or “cook did not wear gloves”. Each type of violation comes with a certain number of penalty points, e.g., 2 points for a dirty restroom but 5 points for meat stored outside the refrigerator (since this is more dangerous in terms of health). Each type of violation has a code (ID), a description (e.g., “rats in kitchen”), and a number of penalty points for it. (For simplicity, we assume that violations either occur

or not – in real life there might be different numbers of penalty points, say, for a slightly dirty versus very dirty restroom.) Your database needs to store the time and date of an inspection, the inspector, the restaurant, and also which violations were discovered during the inspection. Finally, for each restaurant there is a *health grade*, say, A for less than 4 penalty points, B for 5 to 10 points, C for 11 to 15, and FAIL for more than 15. This information about the grades and points required for a grade should also be stored in the database, so that the current grade of a restaurant can be computed directly from the data about the last inspection.

(a) Design a database for the above scenario using the ER model. Draw the ER diagram, show the cardinalities of all relationships, and identify primary keys and any weak entities.

(b) Convert your ER diagram into a relational schema. Identify all tables, attributes, primary keys, and foreign keys.

(c) Write SQL queries for the following questions. If you cannot answer a query using your schema, then you have to modify your solutions in (a) and (b) appropriately.

(i) For each inspector, output her ID and the average number of penalty points she gave for inspections in 2014.

(ii) For each restaurant, output the name and address and the health grade based on the last inspection. (Note that this health grade should be computed from the violation data, and not precomputed and stored in the tables.)

(iii) List the name and address of the dirtiest restaurant in NYC, i.e., the one with the most penalty points in the most recent inspection

(iv) Output the names of any owners that own at least 2 restaurants and where all their restaurants had rats in the kitchen during the last inspection.

(d) Create tables in the database system, and insert some sample data (say, 5-10 tuples per table, but choose an interesting data set, so that queries do not output empty results.). Then execute the queries in (c) and submit a log.