NYU Tandon School of Engineering **February 23, 2017**
Computer Science and Engineering
CS6083, Spring 2017

Problem Set #2 (due March 6)

Problem 1:

In this problem, you have to write SQL and RA queries for a database modeling trips in a system similar to Uber. Here are the tables:

Customer (cusid, cusname, cusphone, cuscity);
Driver (did, dname, dphone, dcity);
CarOwnership (did, carid);
Car (carid, carbrand, carsize);
Trips (cusid, carid, did, getontime, getofftime, price, distance);

Customers are identified by a cusid, and we also store their name, phone number, and the city they live in. Drivers are identified by a did, and have a name, phone number, and city. One driver could own multiple cars, and a car could have multiple owners. Cars are identified by a carid, along with car brand and car size (e.g., compact, midsize, large). Trips are identified by cusid, carid, did, getontime, and we also store getofftime, price, and distance. The getontime is considered as the time when a trip takes place (which will be used in the queries). The getontime and getofftime attributes should store both time and date information.

(a) Suppose the *Trips* table has only *cusid, carid* as primary key. Would that be a good idea? How about using *cusid, did, getontime* as primary key?

(b) Identify suitable foreign-key relationships for the above tables.

(c) 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; 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.

(d) Write the following SQL queries and execute them in your database. Show the queries and the results:
(i) List the cusid and cusname of all customers who made trips in Jan 2017.
(ii) List the cusid of any customer living in "Brooklyn" who has made a trips with a driver living in "Queens".
(iii) Output the cusid and cusname of the customer(s) who took the most expensive trip(s).
(iv) Output the car brand that was used in trips by the largest number of distinct customers.
(v) Output the did and dname of the driver who earned the most money (sum of prices) during Jan 2017.
(vi) For each cuscity, output the cusid of customer living in that city who travelled the longest distance overall.
(vii) For each month in 2017, output the customer city that had the largest number of trips during the month.
(viii) Output the average duration of trips (getofftime - getontime) for drivers living in Brooklyn, for 2017. The unit of average time should be minutes.

(e) Write expressions in Relational Algebra for queries (iii) to (viii).

(f) Write SQL statements to perform the following updates to the database:
(i). For every car of which the size is "small", change it to "compact".

(ii) Delete tuples of any drivers who do not own a car from the *Driver* table.

(iii) For any trip longer than 30 min, add $5 to the price as a special charge. (We admit that this may not make sense in practice, since you cannot retroactively charge people extra money.)

Problems 2:

In this problem, you have to create views and then write queries on the views, and create triggers, for the schemas in Problem 1. Execute everything on your database system, and report the results.

(a) Define a view that contains for each trip the cusid, cusname, cuscity, carid, did, and getontime. Using this view, try to answer the following queries:

(i) For each customer city, output the customer with the most trips.

(ii) Change the name of any customer with name "Taylor Swift" to "Katy Perry".

(b) Consider the last query in Problem 1(f). Can you write a trigger that automatically adds the special charge when a trip longer than 30min is inserted?

(c) Write a trigger that disables additional trips if a driver has already done 5 trips in the last 60 minutes.

Problem 3:

In this problem, you are asked to design the relational backend for an e-commerce platform for garden/outdoor design. There are three main types of parties in this platform: customers, outdoor designers, and gardening/outdoor stores and companies. Customers are (usually wealthy) people who own gardens or terraces, and who need help in designing them. To do so, they sign up with an outdoor designer that specializes in how to best set up outdoor spaces, i.e., how to organize the space, what plants to grow in different locations, where to build a deck or terrace, etc. Gardening/outdoor stores and companies sell products such as plants, seeds, soil, wood, and various equipment, or services such as planting a lawn or building an entire deck.

Customers are identified by a customer ID, and the database also stores their real name, address, credit card number, email, and phone number. Designers are identified by an ID, a name, and a phone number. Customers can have one or several designers that do work for them, although there can be only one designer for each project they do. For example, a customer could work with designer A for a project titled "frontyard design", and designer B for a project titled "backyard design", and then, say, a few years later use designer A again for a project titled "backyard redesign and expansion". Stores are identified by a unique store ID, a name, and a phone number. Products and services have a unique ID and a name. Products and services have a global product ID that is unique across stores; thus, if two stores offer a product with the same ID it is in fact the same product or service.

Note that in this scenario, customers will usually first start working with a designer offline, and this designer will propose an initial design for the garden to the customer, for a fee. This interaction between customer and designer happens outside the database, and is not modeled here. However, once customer and designer agree on a basic design, the designer will start ordering plants, materials, and services via the e-commerce platform. To do so, the designer will create a new project in the system that has a unique ID number, a name, a starting date, and the IDs of the customer and the designer. So the database is just about the purchases needed for the project, not the actual design process.

An important aspect of this business is that registered designers usually get discounts from stores. More precisely, we assume that there are contracts between designers and stores that specify the amount of discount (say, 5% or 10% off) that the store gives to this designer. These contracts also need to be stored. (If no contract exists, that

means that there is no discount.) When a designer creates a project, she can decide if the discount is passed on to the customer (i.e., the customer pays less), or if the discount should be given to the designer as a commission (i.e., the customer pays the full amount). The designer can then search for products and services, and make purchase orders that will be charged to the customer's credit card and delivered to his address. Note that customers cannot make their own purchases, but they can sign up and see what has been ordered for their various projects at what price. Each purchase order has to keep track of the product ordered, the project, the store, the time and date, and the price before discount.

(a) Design an ER diagram that can model 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 constrains.

(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) For each designer, output their ID and the total amount of commission they earned from orders during 2016.

(ii) For product ID 481787 and project 82347, output the lowest price and the name of the store offering that price. Take into account any discounts and whether they are passed on to the customer for this project!

(iii) For each customer, output the names of their projects and the total amount spent for each project.

(iv) For store "Garden Paradise", output the ID of the designer who gets the highest discount rate in this store.

(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.