NYU School of Engineering February 11, 2019
Computer Science and Engineering
CS 6083, Spring 2019

Problem Set #1 (due 2/22)

Note: In this homework, you do not need to create tables and execute queries using an actual DBMS. A written solution is sufficient. Also, for the first homework, you can use informal expressions such as year(pd) = '2017', where pd is a timestamp, to check if the year is 2017.

Problem 1: Suppose you have a database modeling a very simple car rental company, given by the following schema:

Customer (cID, cName, cCity, cCCN, cLicense)
Car(carID, carMake, carModel, carModelNum, carColor, carYear)
CarType(carMake, carModel, carModelNum, carSize, carSeats)
Branch(bid, bCity, bStreet, bState, bPhone)
Rental(rID, cID, carID, pickupTD, pickupBid, returnTD, returnBid, cost)

In this schema, each customer is identified by a cID, and we also store the name, city, credit card number, and drivers license number of the customer. Each car has a make (e.g., "Toyota"), a model (e.g., "Corolla"), a model number (e.g., "400XL"), and a year when it was manufactured. For each make and model, we also store the size of the car (e.g., "compact", "mid size", "luxury") and the number of passengers it can hold. The car rental company has many branches, and for each car rental, we store which car was rented and by which customer, the total cost, and when and at what branch it was picked up and returned.

(a) Identify suitable foreign keys for this schema.

(b) Write statements in SQL for the following queries.

I. List the cID and name of any passenger who picked up a blue Toyota with 6 seats in 2017.
II. List the cID of any passenger who has rented a Toyota but who has never rented an Audi.
III. For each car, output its carID and how often it has been rented.
IV. Output the cID of any passenger who has picked up a car in every state that has at least one branch.
V. Output the cID and name of the passenger who spent the most money overall on rentals picked up during 2017.
VI. Output the cID and name of the customer(s) who made the most expensive rental picked up during 2017.

(c) Write expressions in Relational Algebra for the above queries.

(d) Write either DRC or TRC queries for the above queries. Or explain the reason why you think a particular query cannot be done in DRC or TRC.

Problem 2: In this problem, you need to design a relational schema for a company called National Parcel Service (NPS), similar to UPS or FedEx, that delivers packages within the United States. Basically, your database need to keep track of all packages and who sent them, how much it cost to sent them, and where the packages currently are on their way to the recipient, etc.

Now more details. Customers sign up for an account, and supply basic information such as a name (which could be a company name), a phone number, and a credit card number. When a package is sent, it is sent from an source location to a destination location, and the cost of the package is charged to a customer account (usually of the sender, but this is not required). Note that some of the customers are larger companies that may use one account to send packages from many different locations, say their various offices. So you cannot assume that each customer has one location from which packages are sent. A location is basically an address, with a name (personal or business name), street address, city, zip code, state, and a contact phone number. Note that while each package has to be billed to a customer, the source and destination locations are not required to "belong" to a registered customer.

For each package, you need to keep track of which customer paid for it, its source and destination, its weight, and its cost. The cost of sending a package depends on its weight and its source and destination. In particular, the weight falls into one of several categories, e.g., "up to 16 ounces", "up to 48 ounces", "up to 96 ounces", etc., up to some maximum weight. Zip codes of locations are divided into three categories, "central", "rural", and "remote", and the price of a package depends on the category of the source and destination zip code, and on whether the source and destination are in the same state or different states. Thus, a package sent from a remote location in Alaska to a remote location in Oregon might be more expensive than one of equal weight sent between two remote locations in the same state, or between two central locations in different states, etc. Your database should keep track of the current price table. (In addition, for each package you also need to store its cost, since prices in the price table may have changed since it was sent.)

Customers are also divided into different classes that may give them certain discounts. For example, a larger company might have a "Gold" account that gives them 5% discount on all shipping costs, while "Silver" accounts may get 2% off. These classes might be based on negotiations, or based on the customer spending at least one million $ in the previous quarter, but the details of how an account is assigned to a class are not your concern.

Finally, you should support tracking of packages. You should assume that packages are scanned in various places along the route. There will be a source scan when the package is picked up and a delivery scan when/if it is finally delivered. Also, whenever a package is moved to a new truck, or unloaded from an airplane, etc., there will be a scan. For each scan, you should store a time and date, a location in terms of longitude, latitude, city and zip code, and a short label/description such as "arrival scan, Memphis airport" or "loaded onto truck #44, NPS warehouse Cincinatti", etc.

(a) Design a relational database schema that supports the above functionality. Specify all primary and foreign key constraints, and state any assumptions you are making. You can decide which exact attributes make sense for this schema.

(b) Write SQL statements for the following queries. If your schema does not support these, you need to modify it appropriately.

I. For each customer, output their ID, name, and the total amount of money they spend on shipments during 2017.

II. Output the IDs of any packages that were last scanned with label “Arrival scan, Miami airport” more than 5 days ago, and that were never scanned afterwards. (In other words, look for packages that were last scanned at the Miami airport but then got lost for the next 5+ days.)

III. Output the IDs of all packages paid for by customer “ACME Global” during October 2018 that were more expensive than what ACME Global would pay under the current price table.