University of California, Los Angeles
Department of Computer Science

Computer Science 143 **Prof. Ryan Rosario**

Homework 1
Solutions

For Part 1, use the following scenario:

Bay Area Rapid Transit (BART) is a subway system that stops at various places on the Bay Area Peninsula, City of San Francisco, underwater to Oakland and the East Bay. We have two relations in this dataset: `Station` and `RideCount`. `Station` has data about BART stations and `RideCount` contains counts of passengers that travel between two stations per hour, per date.

```
Station(Abbreviation, Location, Name)
# Abbreviation is a 3 letter code given to the station, just like an airport code.
# Location is the name of the city where the station is located. There may be multiple
#    stations in one city.
# Name is the name of the station and each station has a different name, though the name can change.

RideCount(Origin, Destination, Throughput, Date, Hour)
# Origin is a 3 letter code (like Abbreviation) that specifies where a trip started.
# Destination is similar to Origin and Abbreviation but specifies where a trip ended.
# Together, Origin and Destination form a "route" or "ride."
# Throughput is the number of people that traveled that "route" during a particular hour on a
# particular date.
# Date is the calendar date corresponding to the count in Throughput.
# Hour is the hour of the day (from 0 to 23) corresponding to the count in Throughput.
```

In your work, you may wish to abbreviate the names of the relations and attributes to save space. For `Station`, you can use `S(A, L, N)`, and for `RideCount` you can use `R(O, De, T, Da, H)`.

Part 1. Keys, Keys, Keys

Exercises

(a) Find all superkeys of `Station`.

We can generate all $2^3 - 1 = 7$ subsets of attributes, excluding the empty set.

$$\{(A), (L), (N), (A, L), (A, N), (L, N), (A, L, N\}$$

Then we will eliminate any that violate the definition of superkey – that is, we eliminate subsets that would not be able to uniquely identify a tuple in that `Station` relation:

Based on the context of the problem, `Abbreviation` (A) and `Name` (N) are both attributes that can be used to uniquely identify stations. We will get back to that.

$$\{A, \cancel{L}, N, (A, L), (A, N), (L, N), (A, L, N)\}$$

So there are 6 superkeys:

$$\{A, N, (A, L), (A, N), (L, N), (A, L, N)\}$$

and (A, L, N) is a trivial superkey.

Note that some students included $\varnothing$. The problem with this is that if the empty set is a superkey, then the candidate key must also be the empty set, because it has the fewest possible number of attributes. According to O'Neill, in *Database: Principles Programming Performance*, "an empty set of columns cannot distinguish two rows."

(b) Which superkey(s) of `Station` is/are candidate keys? Explain why.

A candidate key is the minimal superkey, the superkey that has the fewest number of attributes. Out of all 6 superkeys, the smallest order is 1, and two of the superkeys have order 1. The candidate keys are A and N individually.

(c) Which candidate key would you choose to be the primary key of `Station` and why?

The primary key is a candidate key that is chosen by the database designer. It should be a candidate key that does not change in value over time. According the to problem, the `Name` (N) may change, for example, if a politician dies and the station is renamed in their honor. `Abbreviation` (A) does not change. The problem gave a hint that the abbreviation works the same as an airport code, which uniquely identify airports.

We would choose `Abbreviation` (A) to be the primary key.

Some may believe that the composite $\{A, N\}$ would make a good primary key. There are two reasons why this is not the case:

(a) A primary key is a candidate key, and a candidate key is the minimal superkey. Our candidate keys contain 1 attribute, so we cannot have a composite of 2 attributes be the primary key.

(b) Using both attributes in a composite makes a false assumption about the data. It would mean it would be possible to have an `Abbreviation` refer to multiple stations and/or a `Name` to refer to multiple stations. This follows from #1 above.

(d) Which candidate key for `RideCount` would you choose to be the primary key of `RideCount`? Try to infer the PK by the context rather than constructing all superkeys.

We could start by listing out all 31 combinations of attributes, but my hope is that with practice we can identify candidate keys without having to enumerate all possible subsets of attributes.

This relation measures throughput as a function of route (`Origin` and `Destination` pairs), `Date` and `Hour` of day. These four attributes uniquely identify a tuple, and no smaller subset of attributes does, so the candidate key is $\{O, De, Da, H\}$. It's the only candidate key, so it is also the primary key.

(e) Are there are any foreign keys in `RideCount` or `Station`? What are they?

In order for `Station` to have a foreign key, it would have to refer to ALL of the primary key in `RideCount`, but that isn't possible! The PK in `RideCount` has more attributes than are even in `Station`, so there is no way for that to work! **`Station` has no foreign keys/**

In `RideCount`, the attribute `Origin` is related to `Abbreviation` in `Station`, and `Abbreviation` is the primary key of `Station`, so `Origin` is a foreign key. `Destination` is also a foreign key.

Thus, there are two foreign keys in `RideCount`: `Origin` and `Destination`.

Part 2: Schema Diagram

Suppose you are working for the data team at *Bird Scooter*, a Santa Monica based startup that aims to revolutionize how people get around on wheels. How the Bird Scooter service works: a user installs an app on their phone and enters their credit card information. The app shows a map of deployed scooters nearby. The user scans a QR code on the scooter using the app, which activates the scooter for use and begins the clock for per-minute and per-use billing. The user rides the scooter for a distance, for a certain number of minutes. When the user is done with the scooter, he/she leaves it somewhere, and marks the trip as complete in the app.

We have the following relations for managing scooters and users:

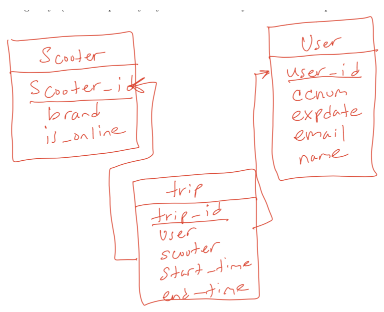
```
Scooter(scooter_id, brand, is_online)
# scooter_id is some value that identifies a scooter.
# brand is a code that represents what type of scooter it is.
# is_online just indicates whether or not the scooter is able to be used.

User(user_id, ccnum, expdate, email, name)
# user_id is some value that identifies the user
# ccnum is the user's credit card number (hashed, hopefully!) and may be null, and may change.
# expdate is the credit card expiration date and may also be null, and may change.
# email is the user's email address, and may change.
# name is the user's name

Trip(trip_id, user, scooter, start_time, end_time)
# trip_id is some value that identifies the trip.
# user represents the user that took the trip.
# scooter represents the scooter that the user rode for this trip.
# start_time and end_time represent the start and end times of the trip.
```

Exercise Using these relations, draw the schema diagram. For an example of a schema diagram, see Figure 2.8 in the text (page 47), or the end of the Intro to the Relational Model section in the lecture slides for Lecture 3. Clearly denote primary and foreign keys (note that primary keys were intentionally withheld in the specification above).

Based on the information provided, `scooter_id` is a primary key for `Scooter`, `user_id` is a primary key for `User` and `trip_id` is a primary key for `Trip`. The relations `Scooter` and `User` are only related to each other when considering a `Trip`.

In the `Trip` relation, `scooter` refers to a scooter identifier, `scooter_id` that appears in `Scooter` and is the primary key there, so `scooter` is a foreign key. `user` refers to a user identifier, `user_id` that appears in `User` and is the primary key there, so `user` is a foreign key.

Therefore, the schema diagram is as follows.


Scooter
Scooter_id
brand
is_online
User
user_id
ccnum
expdate
email
name
trip
trip_id
user
scooter
start_time
end_time