Database Systems, CSCI 4380-01
Homework # 3 Answers

Homework Statement. This homework is worth 3% of your total grade. If you choose to skip it, Midterm #1 will be worth 3% more.

Question 1. We are living history right now and we know that the data that we generate now will be used by many in the future generations to draw conclusions. Data we keep is data we learn from.

There is a lot of attention being given to two types of data right now: the massive amount of misinformation circulating and how the disease is progressing in different parts of the world. There are also many other types of data that might be overlooked, but it may be equally important to learn from. Sociologists talk about how in times of extreme difficulty like this, people show extreme kindness towards each other. These are rarely covered or systematically documented.

One such effort is the mutual aid networks that are popping up in different communities. Groups of people come together to support each other, by pooling their resources in terms of time and skills, communicating with each other about their constraints such as risk tolerance, etc. These networks work on a completely altrustic manner, people volunteer to help others. While technology cannot create the social trust that underlies these networks, may be we can help out with the logistics.

Create an ER diagram for a system that will be used by groups of individuals to create trusted pods, advertise for certain services to offer to the others in the pods.

The database for this system will have a number of users, each user has a unique email, password, preferred name, preferred pronouns (text field), address and a personal statement as text. There are pods, each pod has a unique id, a name, creation date, a statement of values, status (open/closed), private/public status and city, state and neighborhood. The database has a number of service types. Each servicetype is identified with an id, a name, description, unit time.

Users belong to zero or more pods, pods can have one to many users. For each user and pod, there is joined date. Pods have one or more users who serve as administrators. When users want to join a pod, they make a request. Once the request is approved (by outside software), then they become member of pods. So you should keep track of members of pods and people who applied for pods in the database.

Users offer services of a given service type, on a specific day, time, a given number of units and a capacity (number of people allowed to participate). Each service is offered to a single pod only and pods have many such services. A person can offer multiple services of the same type on the same day, but at different times. Users attend specific services by signing up for them. Users also leave kudos and thank you notes for other users for offering services. For each kudos, there is a sender and a receiver, a text and timestamp. One person can send multiple kudos to the same person, even on the same date.

In your ER diagram, make sure all entities have keys and your relationships have participation constraints, all well-marked. If an entity does not have a natural key, you can add an id attribute. If you cannot represent something in the specifications, put a short note as something that the ER diagram does not capture.

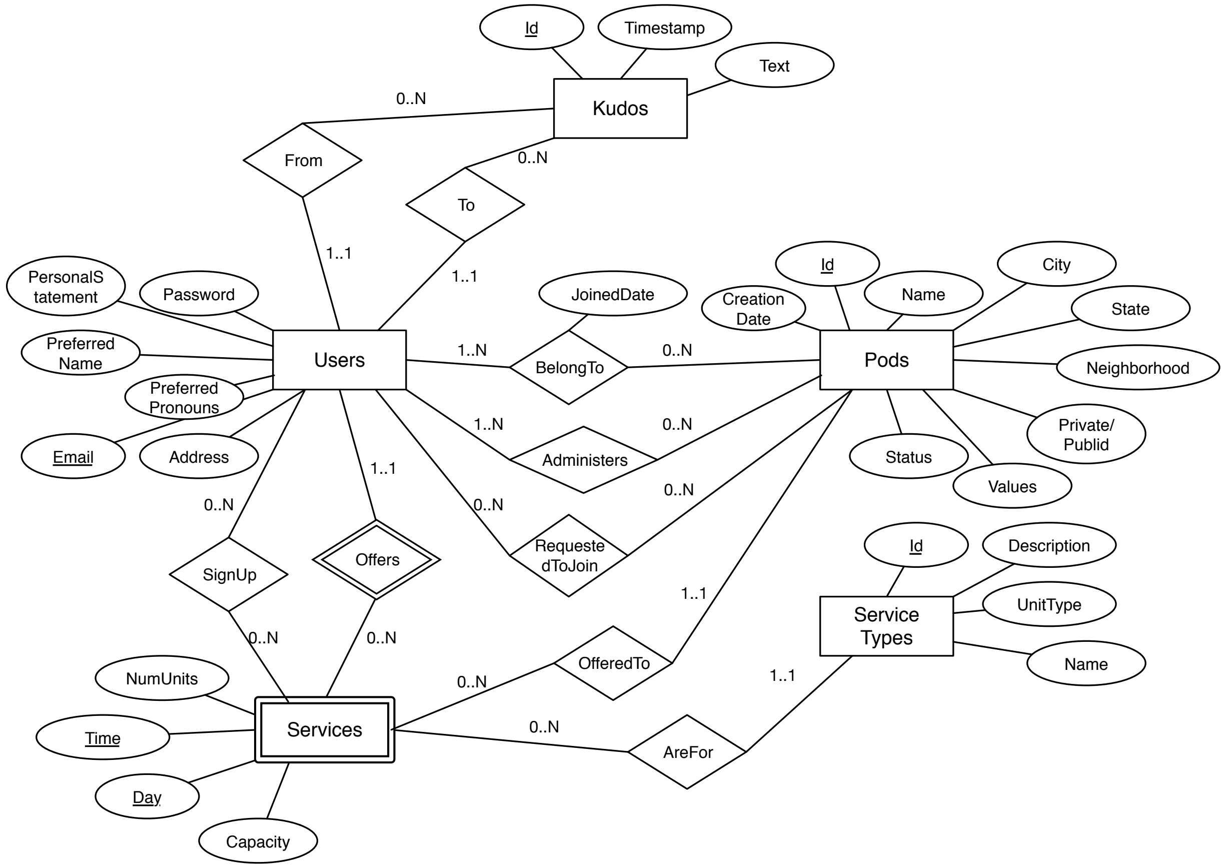

Figure 1: ER Diagram for Question 1

Answer. Note that if Administrators are required to be members of the pod they are administering, we can join Administers relationship with BelongTo and add an attribute "isAdmin" for this. Both are valid models.

Question 2. You are given the below ER data model for a competitor to Lyft/Uber we are developing. Convert this model to relational data model and list all your tables, as well as keys for each relation. Again, use the ER diagram as your guide, do not change things that are not on the diagram based on your intuitions.

Answer.

`Drivers(DriverId, Name, DayJoined, LicenseNo)`

`RatingComments(RCId, Text)`

`Vehicles(VehicleId, Make, LicensePlate, Color, Year, ownerDriverId)`

`Riders(RideId, Name, DayJoined, PhoneNo, CCNo)`

`RatingsTypes(RatingTypeName, Scale, Description)`

`Rides(RideId, StartTime, Price, PriceTier, Tolls, Date, DriverId, VehicleId)`

`RouteSegments(RideId, SegmentOrder, StartPoint, EndPoint, StartTim, EndTime)`

`Drives(DriverId, VehicleId)`

`ParticipateIn(RiderId, RideId)`

`GivesComment(RideId, RiderId, RCId)`

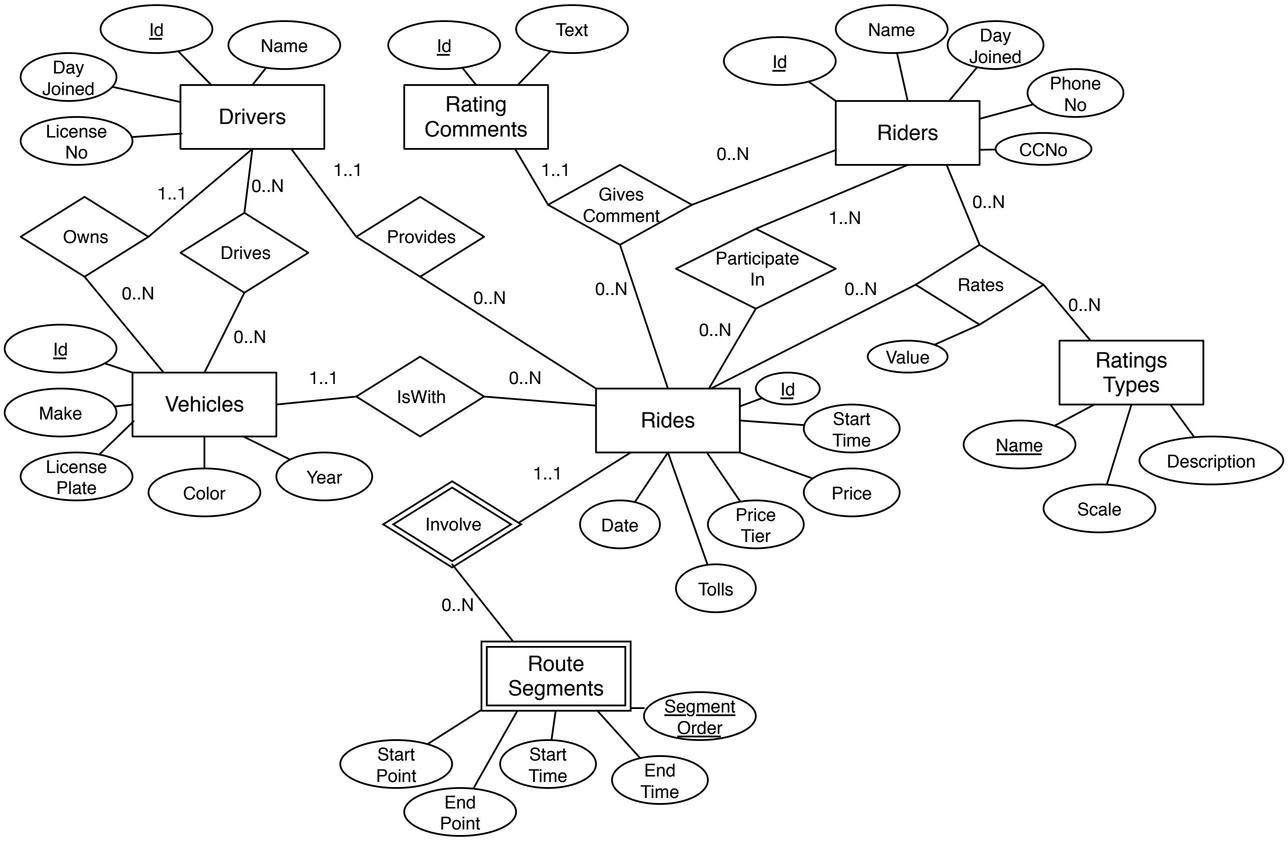

Figure 2: ER Diagram for Question 2

`Rates(RideId, RiderId, RatingTypeName, Value)`

A few notes: When you convert your ER data model to the relational data model, you can revisit and see if it makes sense. If you did your job well, it should be at least in 3NF. However, you can see whether the model serves your purposes. For example, it seems that two relations `RatingComments` and `GivesComment` can be combined to one:

`RatingComments(RCId, Text, RideId, RiderId)`

`RiderId RideId` → `RCId, RCId` → `Text RideId RiderId`

The first functional dependency comes from the ternary GivesComment relationship. Basically this relation has two keys. In this model, if it is important that there can only be one comment for a given rider and ride, then you would have to implement that as a constraint.

Another option would be to get rid of the ID for the RatingComments completely to get:

`RatingComments(RideId, RiderId, Text)`

In this case, what we are doing is changing the model by removing the RatingComments entity altogether in the ER model and adding an attribute for comment in the GivesComment relationship which has now become binary.

Always do sanity check for relations found after the conversion to see if they implement what you wanted.