COMP0178: Database Fundamentals

Database design exercises

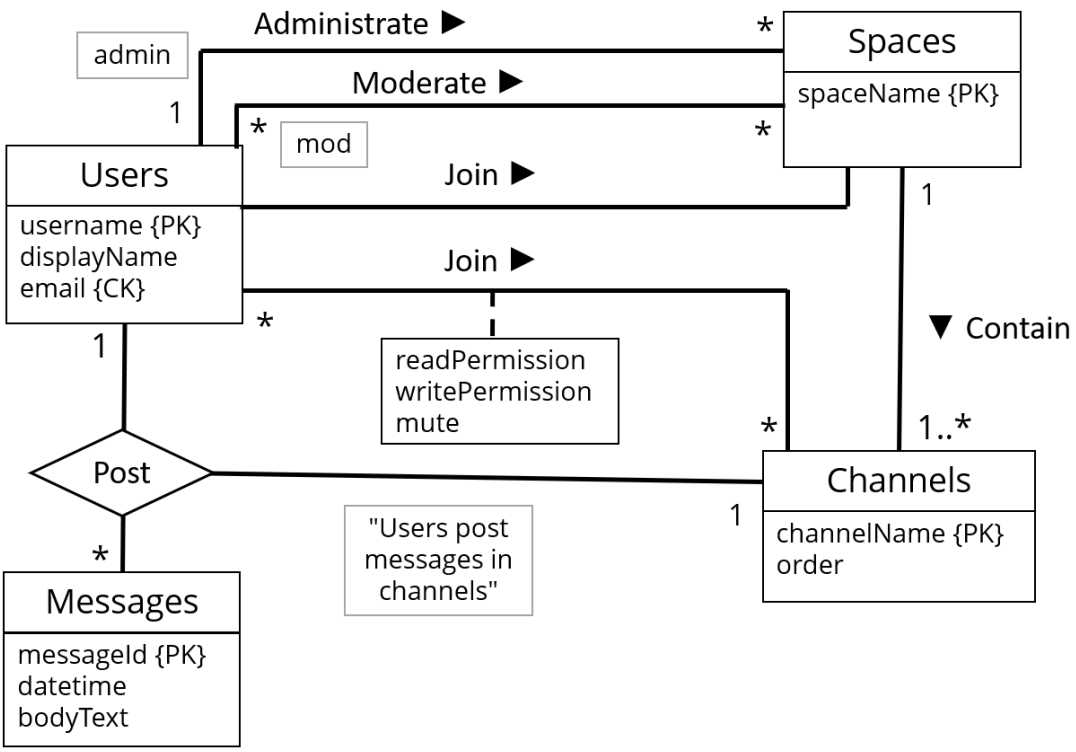
Conceptual design:

1. Design an ER diagram to model a system that facilitates a chatroom system.

   Some of the pieces of data that will be needed: Spaces, which have one admin/creator as well as a set of moderators, and which consist of a set of channels presented in a particular order; channels, which have a name and consist of a list of messages; users who belong to spaces and post messages in channels, and have various settings such as display name and channels they have joined or muted; messages, which have a creator, a date/time, and a body text.

   Some channels such as the “mod announcements” channel allow people to view but only allow moderators to post. Some channels such as the “moderator discussion” channel are only viewable by people with certain roles.

   Here is an example ER diagram. Some of the functionality is missing/simplified, but most of the information from above is incorporated here. Note that there are many possible answers. For example, you can choose to introduce more convenient primary key identifiers or to incorporate foreign keys into entities or to use inheritance to split up some of the User relationships with other entities. There is no one “right” way to do an ER diagram, as long as the ER diagram helps to clarify the problem.



Logical design:

1. For the following table, identify the functional dependencies:
   LibraryRental (transactionID, renterID, renterName, rentalDate, dueDate, loanItemID, loanItemMedia, homeLibrary)

   The main FDs of concern are:
   transactionID → (all other attributes in the table)
   renterID → renterName
   loanItemID → loanItemMedia, homeLibrary

   It's also possible that rentalDate determines dueDate (if the library has pre-defined policies about length of rentals, for example), or rentalDate combined with some other information can determine dueDate.

2. Come up with an example of a UNF schema and an update anomaly that can occur in that schema, which cannot occur in 1NF.

   This is a bit of a trick question as going from UNF to 1NF is more about putting tables into a standard form than preventing update anomalies. But there are possibly some update issues associated with UNF. For example, let's say we have a schema like this:

   Student (studentID, email, fName, lName, classList, programme, …)

   The classList here is a single cell containing a list of all the classes the student is taking, so it's a repeating group. If the student dropped a class, then this cell has to be updated in order to be accurate. That means all the other data in the cell has to be kept the same but the part that no longer applies has to be removed. There is a risk of introducing errors here to unrelated data (the other classes) by updating this cell. This risk would be lowered by using simpler data.

3. Come up with an example of a 1NF schema and an update anomaly that can occur in that schema, which cannot occur in 2NF.

   SessionSignup (sessionID, studentID, studentEmail, sessionTitle)

   Here, we have a composite primary key needed in order to distinguish separate sign-ups. However, the FDs studentID → studentEmail and sessionID → sessionTitle means that it's possible to update the sessionID and not the title and then your table is now inconsistent. Or update the studentEmail in one place but not another and then there is an inconsistency. This is just an example. There are many other possibilities.

4. Come up with an example of a 2NF schema and an update anomaly that can occur in that schema, which cannot occur in 3NF.

   The LibraryRental schema above contains transitive dependencies. Multiple rows

might (for example) involve the same loanItem and its homeLibrary. It's possible to alter the loanItem or the homeLibrary in one row, which will render it inconsistent with all the others. Or if this is the only place where these data are stored, you can lose loanItem/homeLibrary information by deleting rows. It would not be possible to do this if the table were restructured into 3NF.

5. Come up with an example of a 3NF schema and an update anomaly that can occur in that schema, which cannot occur in BCNF.

   Coming up with an example is a bit challenging because there are only a limited set of cases where something is in 3NF but not already in BCNF. For one thing, the primary key has to be a composite primary key. Then there are two possibilities: either the problematic FD has a simple determinant or it has a composite determinant. If simple, then the dependent attribute must be part of the primary key (but not all of the primary key, or this would function as a candidate key). The example given in lecture is of this type. If composite, then it must determine an attribute but not be a candidate key. Here's an example (a bit contrived): A store has a table with shipping data.

   OrderDelivery (<u>item</u>, <u>shippingType</u>, <u>orderDate</u>, shippingPrice, deliveryDate)

   The composite primary key determines the other attributes (item, shippingType, orderDate → shippingPrice, deliveryDate). Let's say that orderDate, shippingPrice → deliveryDate (if you know the order date and the price paid for shipping, you can infer the delivery date). This is a violation of BCNF but not 3NF, and the solution would be to split the tables like this:

   OrderShipping (<u>item</u>, <u>shippingType</u>, <u>orderDate</u>, shippingPrice)
   ShippingTimeline (<u>orderDate</u>, <u>shippingPrice</u>, deliveryDate)

   An update anomaly that could occur in the original table is that it's possible for the delivery dates to get out of synch with the order dates and shipping prices in ways that the functional dependency implies shouldn't happen. By separating them into their own table, this will not happen.