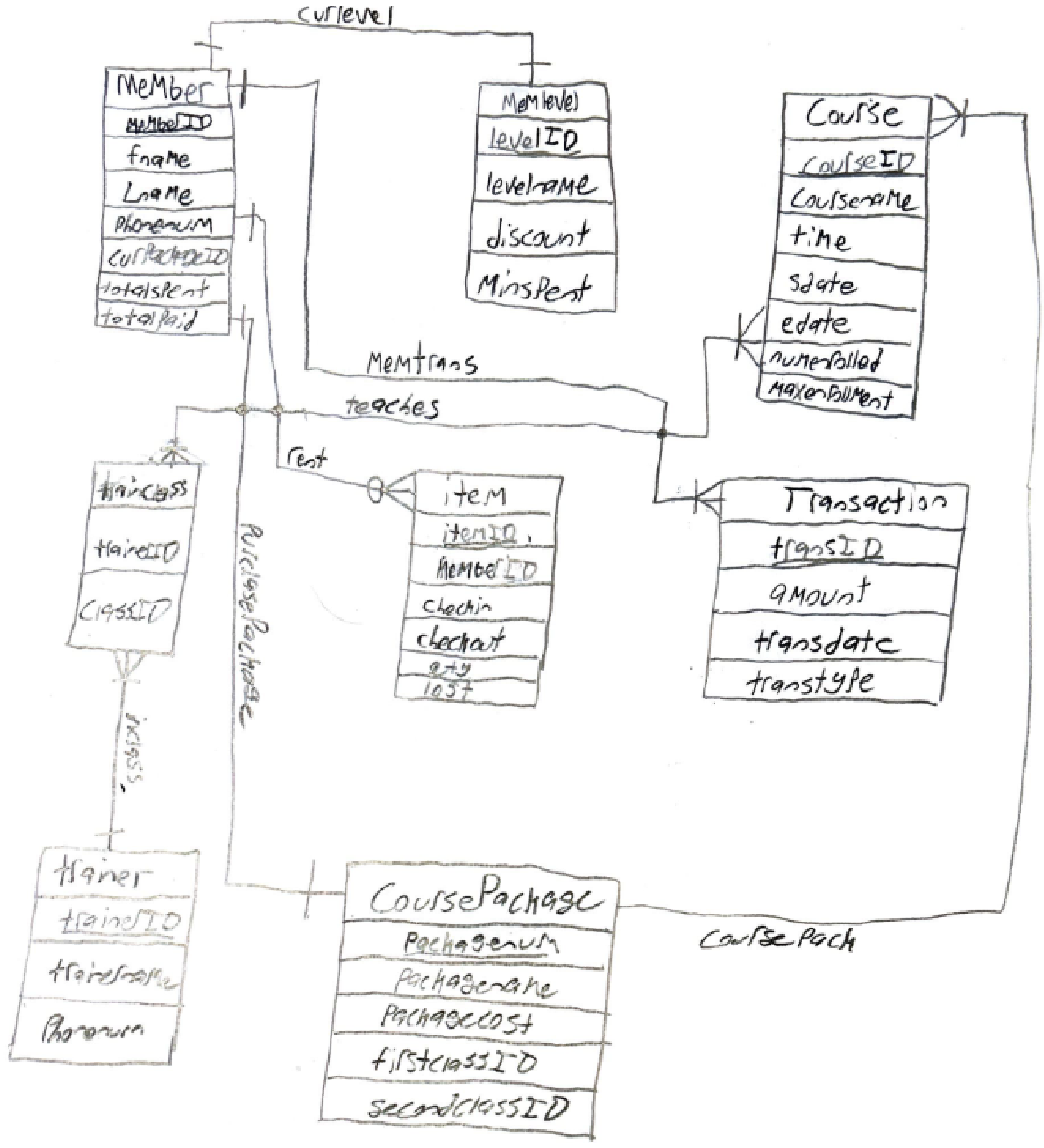
(i) Final ER Diagram



Rationale for design:
Our main goal when approaching this series of tables was to be able to have them work together and have interconnected data. Naturally, the user will be the one interacting with the JDBC application, so most of the relationships stem from the user such as the Member->Course Package->Course relationship. There is one constraint that can be seen in the ER diagram that must be elaborated on slightly: Essentially there is a 2 class limit to a package. The spec provided information for 3 different packages with 2 different classes each, and because of this, we decided to limit the package to contain a maximum of 2 classes per package, with any individual class being able to be removed and updated. But otherwise, our database schema was constructed with the intention of being able to update a series of tables' data, and have that data be able to work with interconnectivity in order to produce the queries.

(ii) Logical database design
As you can probably tell from the above ER diagram, it already shows the relationships between each table, so I believe that fulfills this section's requirements, but I will still display the creation of our tables in an SQL script to show what the actual data looks like, which is in this link.

(iii) Normalization analysis
In order to show that the data is in 3NF, we must show that each individual table is in 3NF. To do that, we can use the definition in the class slides: "A relation R is in 3NF if for every non-trivial FD X->A in R, either X is a superkey of R, or A is a prime attribute of R.

Where:

- A superkey is a set of attributes that includes a candidate key
- A prime attribute is an attribute that is a member of any candidate key of the relation

Member:

memberID->{fname, lname, phonenum, curpackageID, memlevelID, totalSpent, totalPaid}

Within this table alone, this is going to be the only functional dependency. You could say {fname,lname} could act as a functional dependency for the other attributes, but there is no constraint specifying that no member can have the same first name and last name. The same goes for phone numbers; yes phone numbers are unique to an individual, but a member could use a home phone number where multiple people use it to make an account. Because of this, memberID is the only real functional dependency.

Due to this, memberID will be a superkey, which fulfills the definition of 3NF. There is no redundancy, meaning there is no transitivity or the like.

The following tables will have a similar rationale

Memlevel:

levelID -> {levelname, discount, minspent}

Levelname can't identify everything in the table uniquely since there could possibly be outdated versions of the level, where by nature of a primary key, there cannot be 2 of the same IDs, so there will be the most updated version of the level.

LevelID is a superkey and also a prime attribute of memlevel, so this table is in 3NF.

Course:
courseID->{courseName, time, sdate, edate, numenrolled, maxenrollment}

For similar reasons listed above, courseID is the only uniquely identifying attribute. Because of this, courseID is a superkey as well as a prime attribute of Course since it is a member of a candidate key, meaning this table is in 3NF.

Trainer:
trainerID -> {trainername, phonenum}

Just like with members, there is no constraint telling us that a trainer cannot share a phone number with another trainer. An example of this could be if a couple live together and both work as trainers; they both use a home phone number, and both work at the gym, so the phone number cannot be used to uniquely identify trainers. The name is self explanatory since trainers may have the same name.

Because of this, trainerID is a superkey as well as a prime attribute of Trainer, and thus is in 3NF.

TrainClass:
trainerID -> classID
classID -> trainerID

There must be some trainer which teaches a class, and there must be some class that a trainer must teach (for them to qualify as a trainer). Neither of these are primary keys, and thus are candidate keys. That makes trainerID and classID superkeys and also fulfill the definition of a prime attribute.

Because of this, TrainClass fulfills the parameters of being in 3NF.

Item:

itemID->{memberID, checkin, checkout, qty, lost}

An itemID marks who checked it out using their memberID and other information regarding the checkout. MemberID cannot functionally determine anything in the table because a member can check out multiple items where the itemIDs are different. Because of this, itemID is both a superkey and prime attribute of Item.

Because of this, Item fulfills the definition of 3NF.

Transaction:

transID -> {amount, transdate, transtype}

This is very straightforward. The amount, date, or type of the transaction can’t identify anything in the table. TransID is a superkey (because it is the only attribute on the LHS of the FD for the table) and also a prime attribute for Transaction.

Because of this, Transaction is in 3NF.

CoursePackage:

packagenum -> {packagename, packagecost, firstclassID, secondclassID}
{firstClassID, secondClassID} -> {packagenum, packagecost, packagename}

Packagenum is the primary key, but the classes included in the package can also act as an identifier. A package is essentially a combination of classes with additional information, so those classes the packagenum represents could also identify what the packagenum and the other attributes since the packagenum simply represents a couple of classes. Packagenum, similar to the previous tables, is both a prime attribute and superkey, so it fulfills the definition of 3NF. {firstClassID, secondClassID} fulfills the definition of 3NF because they are both candidate keys which could be used to determine which package a course belongs to.

Because of this, CoursePackage is in 3NF.

(iv) Query description:
Our query is the following: List all members of a class taught by a specific trainer.

Purpose: It could be used to, for instance, mark attendance of some kind, or be able to get the amount of students a trainer has to see if their workload is too high or not.