Health and Fitness Club Management System

– Requirements –

Here is what my database needs to be able to do:

- Users need to be differentiated between admins, trainers, and members
- Users should be able to login and create accounts
- Trainers can conduct private sessions
- Members can be part of / partake in private sessions
- Sessions and group events need to be held in rooms
- There needs to be a loyalty program for members
- Members should be able to register to group events

– Creating an ER model –

Entities:

- Users
- Members
- Trainers
- Admin
- Events
- Sessions
- Loyalty Program
- Room

Note: The ER diagram is in the document but to see it more clearly you can open ERDiagramFinal.drawio.png

Explanations:

Admin, Trainer, Member are all weak entities because they are dependent on User. Loyalty program is a weak entity too because it's dependent on Member, which is dependent on User.

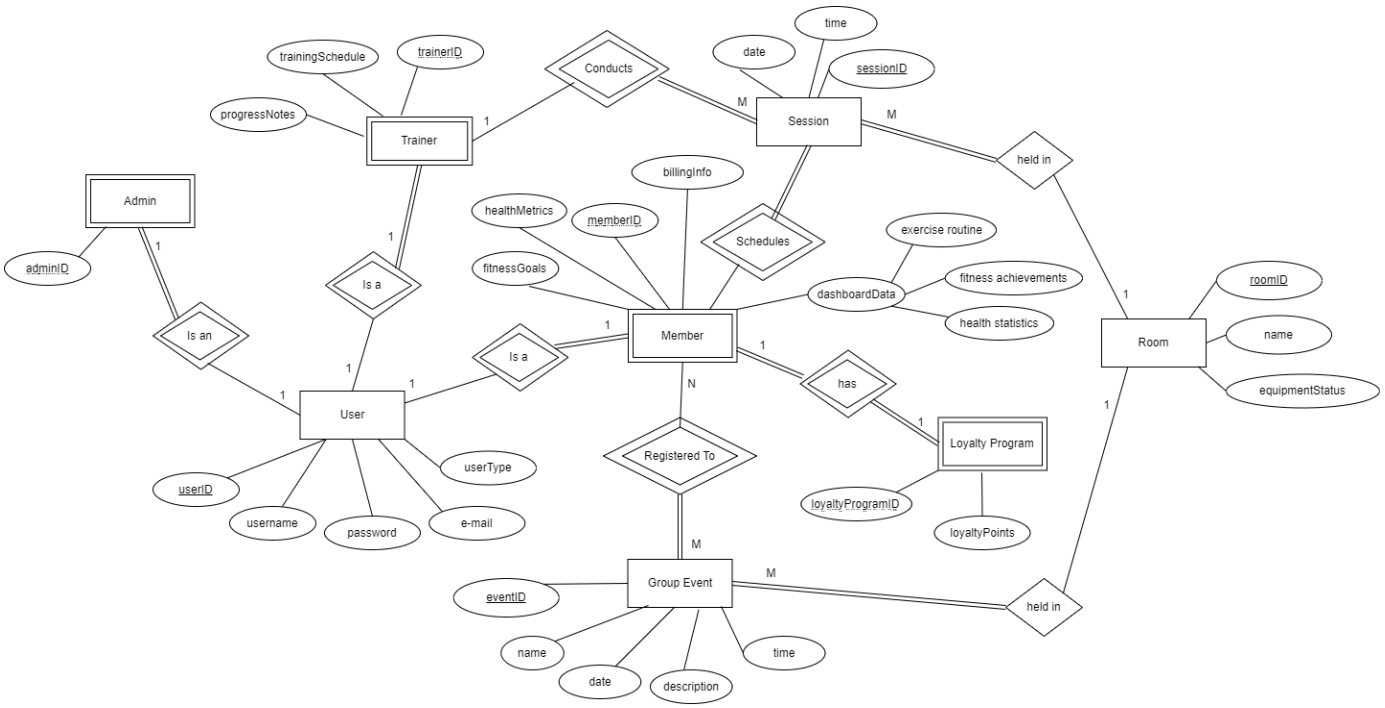
Primary keys are underlined, for composite primary keys I tried to do a dotted underline although it is not very easily visible. Any weak entity has a composite primary key.

User to Admin, Trainer, Member are all 1 to 1 relationships for obvious reasons, and full participation from admin/trainer/member side since every one of them is a user, but user may not be an admin so it's partial participation on that end.

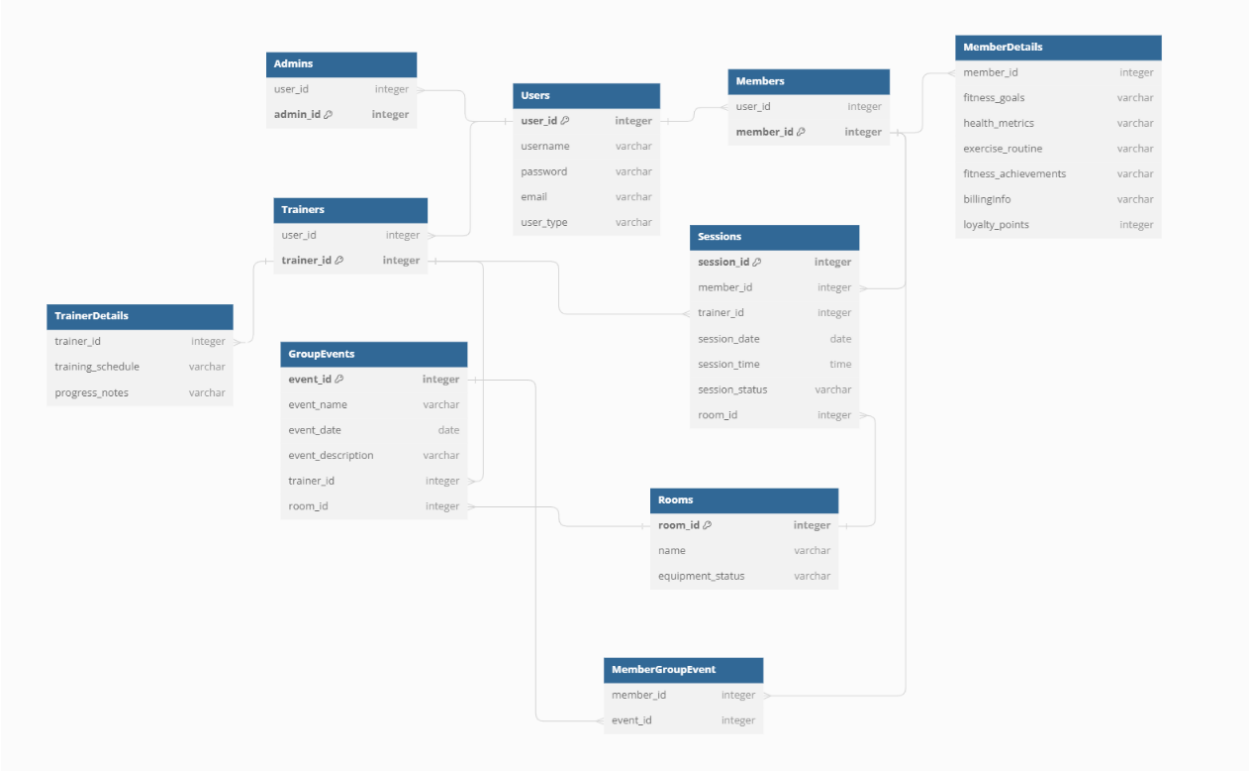
Trainer, Member and Room to session is 1 to M because there can only be one trainer / Member (it's a private session), and room for a session but a trainer/member/room can hold multiple sessions. It seems weird that a room can hold multiple sessions but probably bigger rooms like the gym can.

Session is full participation to Trainer, Member and Room because a session must have all of these things, although for trainer, member and room to session it's partial participation because none of these are required to hold private sessions. You might think surely a trainer needs to hold sessions always but maybe they are vacationing or maybe they only teach group events.

Member to Group Event is M to N relationship because a member can be in many group events and group events can have many members.



– Database schema diagram (pre-normalization) –



To get a better view of this table, open the “pre-normalization DB diagram.pdf”

Changes that I made going from ER to database diagram:

- For trainers and members, I split off the tables between trainers/trainerDetails and Members/MemberDetails. I wanted to have simple members/trainers/admins tables that are just the user_id and the admin_id and I also didn’t feel like dealing with composite keys.

- Since member and loyalty program is full participation on both ends and 1 to 1, I just merged them and now loyalty_points is under memberDetails

- Since member and group event is an N to M relationship I made another table called “MemberGroupEvent” to show which members are in which events

These are just the most notable changes but you can see my work more precisely in the pre-normalization DB diagram.pdf which showcases all my foreign keys as well.

– Normalization: Checking if it's 2NF and 3NF –

This is the portion that I check every table to see if they are 2NF and 3NF.

Right off the bat, any table with only two attributes is automatically following 2NF and 3NF so I won't be doing an evaluation on Admins, Trainers, Members, nor MemberGroupEvent.

For the following tables I will prove that they do not contain partial dependencies (2NF) nor transitive dependencies (3NF).

Users: user_id is the PK, and username, password, email and user_type are all completely dependent on user_id. None of the non-key vars can be guessed by seeing a different non-key variable.

MemberDetails: member_id is the FK and every other attribute depends on it. There is also no way you can guess the value of one non-key attribute with another non-key one. (Fitness goals and fitness achievements might seem similar but in practice they are pretty much two non-dependent variables that might be similar sometimes).

TrainerDetails: trainer_id is FK, and other variables depend on it alone. Progress_notes and training_schedule don't depend on each other at all.

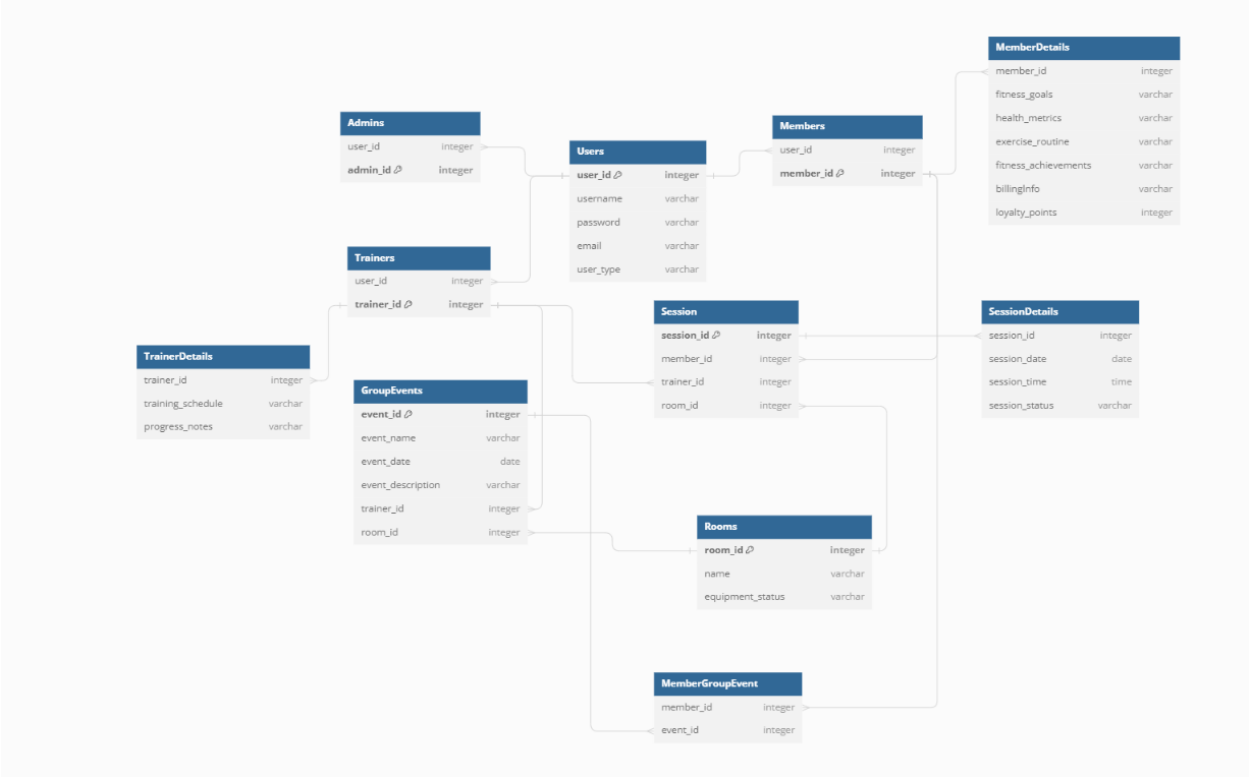
GroupEvents: event_id is PK, and all other values are dependent on it. You might think an event_name would be transitively dependent on event_description but it's actually not since it's possible that two events just happen to have the same description, but it's only event_id which is unique that can determine which one it is. Therefore this table also follows 2NF and 3NF.

Sessions: When I was first doing normalization for my tables I believed this to not be in 2NF because values such as session_date were dependent on session_id but not trainer_id. However I realized afterwards that session_id is the PK, and since all other attributes are dependent on it and only it, the table actually is in 2NF and 3NF! I already broke the "sessions" table into two different tables though and coded my program that way, so that's why you will notice a difference in the pre-normalization and post-normalization diagrams.

Rooms: This table has all its attributes dependent on PK room_id and has no transitive dependencies.

– Database schema diagram (post-normalization) –

Once again, for a better view I recommend opening my “post-normalization DB diagram.pdf” file. I realize now that I actually didn’t need to perform any normalization tasks as all my tables were in 2NF and 3NF anyways, but since I bothered to make the diagrams I figured I would just show them.



– Running the program –

To run the program follow the instructions in my Github’s readme file. I also have a link to a youtube video showcasing my final product in the Github readme!