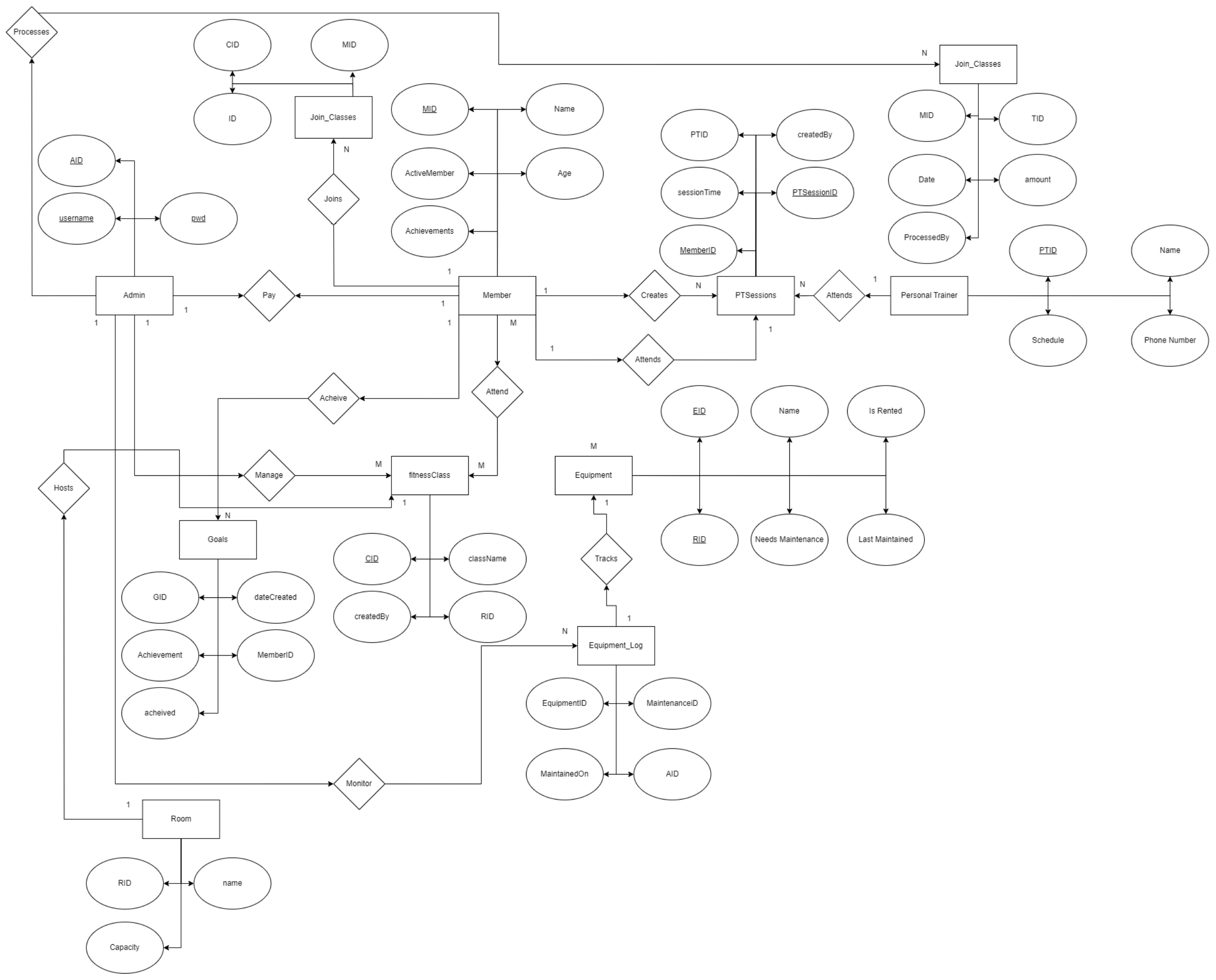
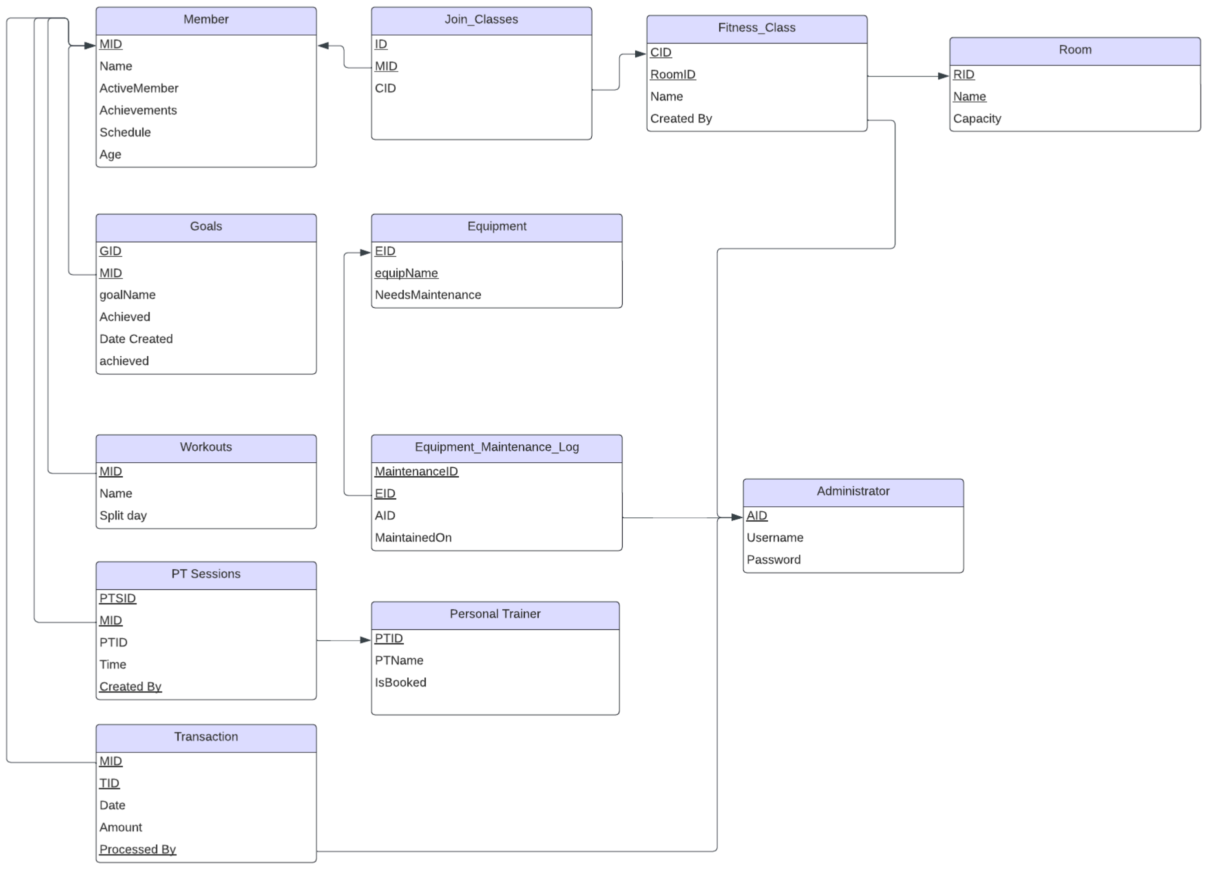
2.1



Processes
N
Join_Classes
CID
MID
ID
Join_Classes
N
Joins
MID
Name
ActiveMember
Age
Achievements
AID
username
pwd
PTID
createdBy
sessionTime
PTSessionID
MemberID
MID
TID
Date
amount
ProcessedBy
PTID
Name
Admin
Pay
Member
Creates
PTSessions
Attends
Personal Trainer
Schedule
Phone Number
1
M
N
Attends
Attend
Acheive
Manage
fitnessClass
Hosts
Goals
EID
Name
Is Rented
Equipment
RID
Needs Maintenance
Last Maintained
CID
className
createdBy
RID
Tracks
GID
dateCreated
Achievement
MemberID
acheived
Equipment_Log
EquipmentID
MaintenanceID
MaintainedOn
AID
Monitor
Room
RID
name
Capacity

2.2



Design Choices

In our design, we chose to execute a design that uses the least amount of tables possible. This means that our database design prioritized efficiency employing a straightforward approach to the database schema to minimize complexity and enhance the performance of each query. We tried to use fewer foreign keys to establish any relationship between different tables ensuring data integrity by only using the foreign keys that are necessary to get the required information. We made sure that the overall design of our program was compact and simple to ensure reliability and efficiency. Looking at the relational schema diagram, mostly all the tables have at most 1--2 foreign keys with most of the tables referencing the member table’s MemberID. We also kept two tables for logging/history, one for the equipment maintenance that tracks all of the maintenance that has been done to each item and a table for the classes that have been joined by each member which tracks all group_fitness class activity. This is essentially similar to a browser history log that tracks what you do. We also separated personal trainer sessions, goals, and workouts to each of their tables which tracks all of the members' personal information. With the member ID being referenced to each of the tables, we can query anything that is associated with that certain member's identification number. We also did a transaction table which tracks all of the purchases made by each of the members handled by the Administrator. To conclude, with all of these design principles in place, we made the program very efficient and it can query all of the information efficiently and effectively. By keeping the program compact, it saves a lot of space and doesn't confuse the user which makes the program very usable to any person.

Additional Comments

Something we should have spent more time on was how the logic in the terminal UI worked as around half the time was spent debugging the call stack where the scanner would cause an error. The logic for building queries and running them all went smoothly overall with the queries being straightforward.

We're happy overall with how things were structured and organized with the main handling main menus and SQL execution, submenus handling heavier loads of input, and query classes for admin, personal trainer, and member. This made it easier to navigate and stay organized while working.

We're also pleased about how the database was designed and mapped out as it was very simplified and so minimal making it easy to perform CRUD actions and keep track of where things went when creating entities. Using the CLI interface made developing the program straightforward with every option making a call do a specific order of functions. When making all the options perform the same order of functions it makes it clear where everything is handled.

Here's my dog