Health and Fitness Club Management Database

Quentin Wolkensperg [101157870]

COMP 3005 Fall 2023

Design

This section will go over the design intentions of the database as well as covering some basic assumptions from table to table.

General Design

The basic design of this model is sort of 'Member' focused. The member table is the main sort of entity that will be interfacing with the rest of the tables and has three main attributes to consider.

- Member type. Are they a trainer or an admin.
- Member attributes. From user entered info to calculated stats as well as transactions committed. These are things that can been seen as defining a particular member and are member specific values.
- Bookings. This just holds what training sessions and group events specific members are a part of and can be seen as their schedule.

Member Type Tables

Member

Keeps track of username and unique member ID. As well as health metrics for this example case it is just simple stuff like height and weight.

Trainer

This table just defines that the member is actually a trainer. Trainers also have different relationships to group events and training sessions and are treated differently in those tables to regular members.

Admin

Really just to separate out admin users to the rest of the program. More on why this exists in assumptions.

Member Attributes

Transactions

This database holds members and their transactions. This is to keep track of member spending as well as to apply loyalty points. This was separated out to abstract from the booking side of things to make thing more logical when working on a front-end. Transactions also have a sub table Service that defines a service name (what a member is paying for) the base cost and the amount of loyalty points awarded for purchasing that service. A very simple implementation of a transaction service as there is not even a transaction number, who made the transaction, what was the type and at what time is what defines the transaction.

Loyalty Points

Defines how much loyalty points a given user has.

Achievement

Assumed managed by staff members. I.e staff members define achievements in which the members can strive to complete. Keeps track of what achievements there are to complete. Members can also "complete" these achievements.

Routine

Assumed members make their own fitness routines. Keeps track of fitness routines and their steps as well as calories burned.

Fitness Goal

Assumed members create these goals for themselves. Keeps track of members goals and status of these goals (complete or not).

Bookings

Group Event

Keeps track of group events and their attributes. Can be managed by multiple trainers and can be attended by multiple members. Is located in a one Room Also it keeps track of a cost as it is assumed these group events are sort of unique and not super regular.

Training Session

Each training session only has one trainer and one member and can be located in one Room There can be multiple sessions with the same member and trainer so date is also used as a unique key. Keeps track of personal training sessions past, present and future.

Other/Inventory

Room

Keeps track of the rooms owned by the club.

Item

Keeps track of items owned by the club. Treated as if they are stored in rooms.

Points Rate

Keeps track of what amount of points give what discounts. Can be used with transactions. It is why Service holds a base cost and the Transaction holds the actual cost paid. In this system money can be refunded easily but some calculations would have to be done to refund loyalty points.

Assumptions

In this section I will discuss specific assumptions made about the database, why they were made and the influence on the design.

Date Conflicts

Date conflicts in scheduling are assumed to be handled either by triggers or the program itself. This was specifically chosen because the way conflicts could be defined could change over time for example a person can be in two group events at once but personal training sessions can’t overlap. This is why it makes sense for this problem to be handled with code instead of in the database structure. Also such a format would likely be neigh impossible from my experience. Date fields and duration’s have been added to help a programmatic solution get values for specific scheduled activities.

Transactions

Transactions are assumed to take place sort of separately to actual bookings. It would be up to the front-end to prompt for transactions before allowing bookings. This led to a transaction table.

Permissions

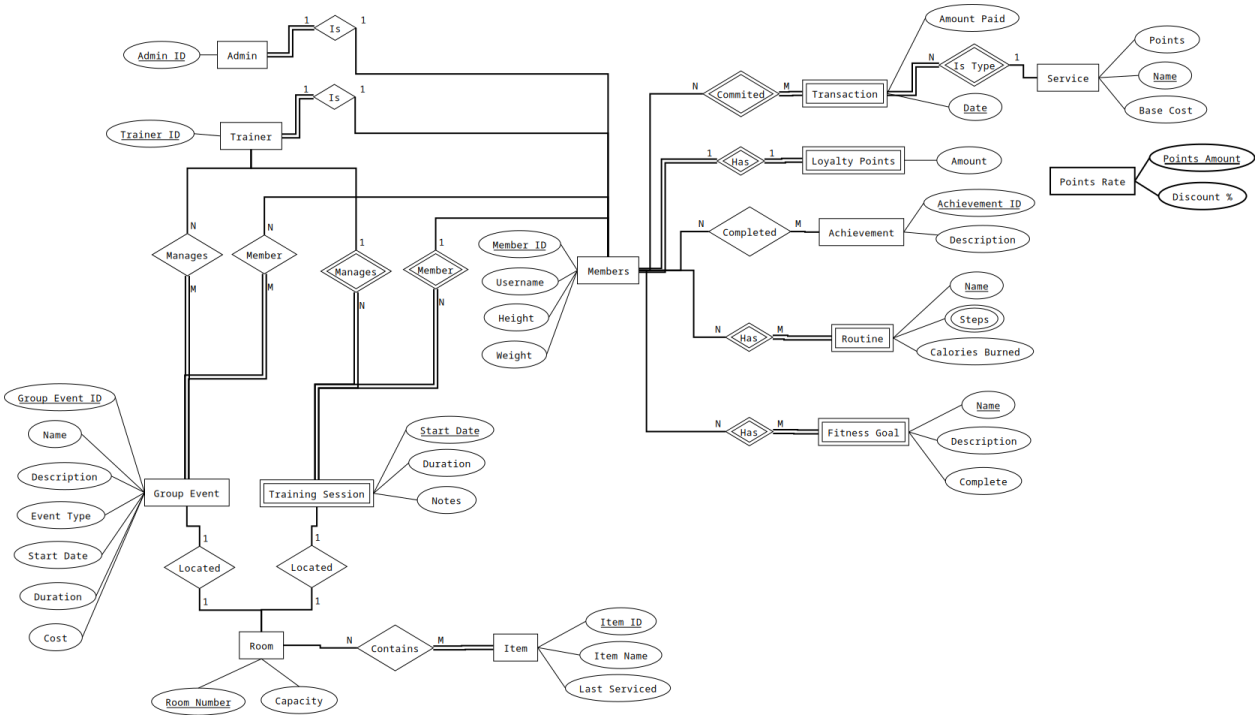
Handling who can do what will have to be handled by the program. This specifically refers to blocking normal members our of what trainers can do and what admins can do. I did add admin and trainer tables to separate out these members but permissions to what commands will have to be handled by the program.

Metrics And Statistics

I'm assuming for this project that the metrics will be stored more on members as attributes. Statistics will be reached from other tables and calculated by the front-end and displayed on the dashboard.

Note: A better way to allow for more expressive user defined metrics and statistics would probably be better handled by a NoSQL database.

ER Model



https://github.com/qwktrik/comp3005-final_project/blob/main/ERFitness.png

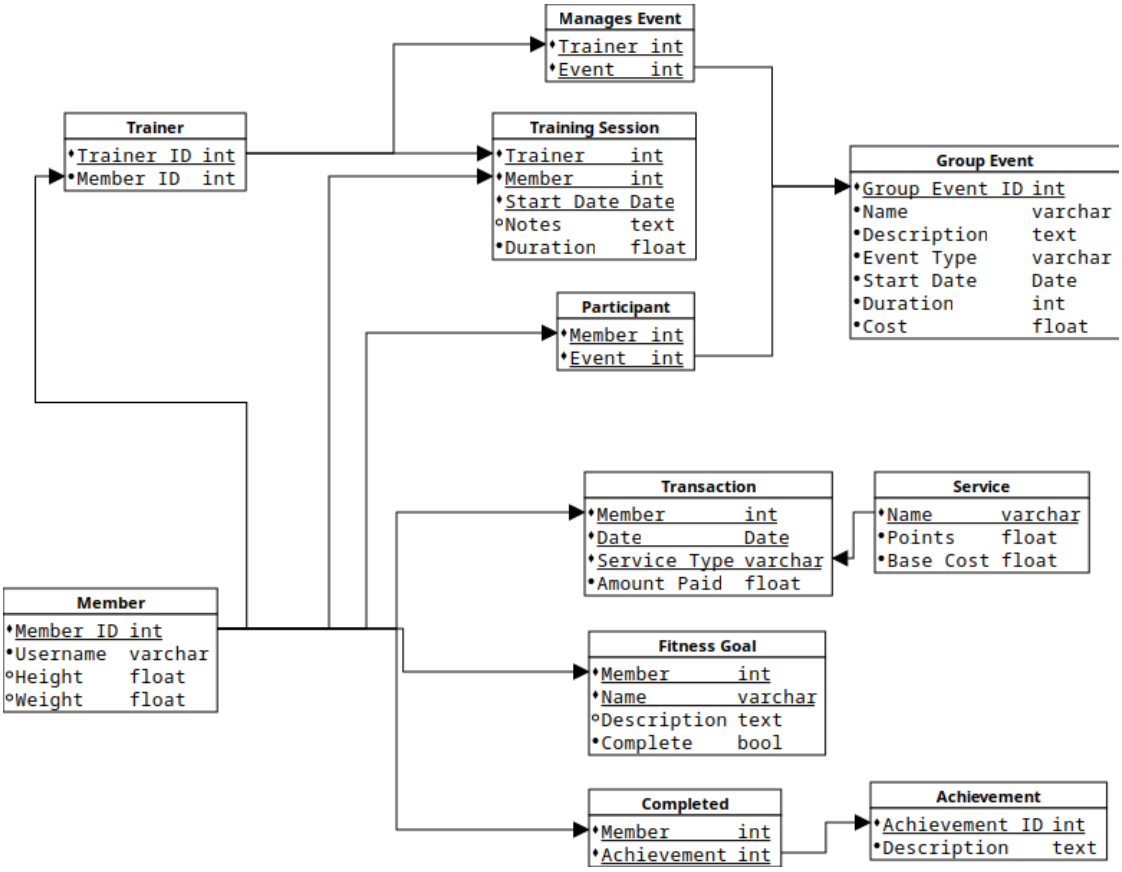
Database Structure (Version 1)

I decided to take 8 tables from the ER model and convert them to a relational database schema.

- Member

- Trainer
- Training Session
- Group Event
- Transaction
- Service
- Fitness Goal
- Achievement

This is because I will get some extra tables from many to many relationships and I think this represents a basic overview of what I was trying to accomplish with the database. It represents the three aspects of a member, Member Type, Member Attributes and Member Bookings.



https://github.com/qwktrik/comp3005-final_project/blob/main/dbversion1.png

Normalization

This section will explain the normalization process. I will be skipping many to many relationship tables in my analysis as they technically do not adhere to 3NF but are necessary to describe the relationships.

Member

Functional Dependencies

- Member ID → Username, Height, Weight

Second Normal Form

Already in second normal form. What differentiates each value from others is the Member ID.

Third Normal Form

Already in third normal form. Nothing is shared between keys (there is only one).

Trainer

Functional Dependencies

- Trainer ID → Member ID

Normalization

Already normalized this is just a very simple relation.

Group Event

Functional Dependencies

- Group Event ID → Name, Description, Event Type, Start Date, Duration, Cost

Normalization

Already in 2NF and 3NF as all non-prime elements directly dependent on the primary key (Group Event ID).

Training Session

Functional Dependencies

- Trainer, Member, Start Date → Notes, Duration

Normalization

Already in 2NF and 3NF as Notes and Duration directly depend on the composite primary key.

Transaction

Functional Dependencies

- Member, Date, Service Type → Amount Paid

Normalization

Already in 2NF and 3NF as Amount Paid directly depend on the composite primary key.

Service

Functional Dependencies

- Name → Points, Base Cost

Normalization

Already in 2NF and 3NF as Points and Base Cost directly depend on the service Name. This intuitively makes sense, each service will have its own base cost and amount of loyalty points given out. The name directly differentiates each entry from another.

Fitness Goal

Functional Dependencies

- Member, Name → Description, Complete

Normalization

Already in 2NF and 3NF as Description and Complete directly depend on the composite primary key.

Achievement

Functional Dependencies

- Achievement ID → Description

Normalization

Already in 2NF and 3NF as Description directly depend on Achievement ID.

It looks as if making a good ER model and carefully converting into a Relational Database Diagram gives you 2NF and 3NF for free without having to change anything. My conversion was very simple though as many things were taken out for simplicity from the ER model. But after going though this I would be fairly confident that no normalization issues would crop up even if I did the full conversion.

Database Structure (Version 2)

The diagram does not change. It is already in 2NF and 3NF.

PostgreSQL Database Creation

This section will go over the commands to create the database described by the relational database schema created.

```
CREATE TABLE Member (
    member_id SERIAL PRIMARY KEY,
    username VARCHAR(255),
    height DECIMAL,
    weight DECIMAL
);
CREATE TABLE Trainer (
    trainer_id SERIAL PRIMARY KEY,
    member_id INTEGER REFERENCES Member(member_id)
);
CREATE TABLE Training_Session (
    trainer_id INTEGER REFERENCES Trainer(trainer_id),
    member_id INTEGER REFERENCES Member(member_id),
    start_date DATE,
    notes TEXT,
    duration DECIMAL,
    PRIMARY KEY (trainer_id, member_id, start_date)
);
CREATE TABLE Group_Event (
    group_event_id SERIAL PRIMARY KEY,
    name VARCHAR(255),
    description TEXT,
    event_type VARCHAR(100),
    start_date DATE,
    duration DECIMAL,
    cost DECIMAL
);
CREATE TABLE Manages_Event (
    trainer_id INTEGER REFERENCES Trainer(trainer_id),
    group_event_id INTEGER REFERENCES Group_Event(group_event_id),
    PRIMARY KEY (trainer_id, group_event_id)
);
CREATE TABLE Participant (
    member_id INTEGER REFERENCES Member(member_id),
    group_event_id INTEGER REFERENCES Group_Event(group_event_id),
    PRIMARY KEY (member_id, group_event_id)
);
CREATE TABLE Service (
    name VARCHAR(255) PRIMARY KEY,
```

```
    points INTEGER,
    base_cost DECIMAL
);
CREATE TABLE Transaction (
    member_id INTEGER REFERENCES Member(member_id),
    date DATE,
    service_type VARCHAR(255) REFERENCES Service(name),
    amount_paid DECIMAL,
    PRIMARY KEY (member_id, date, service_type)
);
CREATE TABLE Fitness_Goal (
    member_id INTEGER REFERENCES Member(member_id),
    name VARCHAR(255),
    description TEXT,
    complete BOOLEAN,
    PRIMARY KEY (member_id, name)
);
CREATE TABLE Achievement (
    achievement_id SERIAL PRIMARY KEY,
    description TEXT
);
CREATE TABLE Completed (
    member_id INTEGER REFERENCES Member(member_id),
    achievement_id INTEGER REFERENCES Achievement(achievement_id),
    PRIMARY KEY (member_id, achievement_id)
);
```

https://github.com/qwktrik/comp3005-final_project/blob/main/DDL.sql

Test SQL Commands

In this section will go over populating the database and some other queries to test the database out. These selection statements will be inspired by ones we would possibly want if making a program.

Populating Database

```
INSERT INTO Member (username, height, weight) VALUES
    ('Quote', 180, 75),
    ('CurlyBrace', 165, 60),
    ('King', 175, 80),
    ('Sue', 160, 55),
    ('Balrog', 100, 200);
INSERT INTO Trainer (member_id) VALUES
    (1),
    (3);
INSERT INTO Training_Session (trainer_id, member_id, start_date, notes, duration) VALUES
    (1, 2, '2023-01-15', 'Targets 50m + 10km Run', 120),
    (2, 5, '2023-02-20', 'Spar', 45);
INSERT INTO Group_Event (name, description, event_type, start_date, duration, cost) VALUES
    ('Yoga Class', 'Calm down with some good tunes and good vibes', 'Yoga', '2023-03-10', 60, 15.99),
    ('Outer Wall Climb', 'Level 5 Climb (Bring a Harness!!)', 'Fitness', '2023-04-05', 45, 20.50);
```

```
INSERT INTO Manages_Event (trainer_id, group_event_id) VALUES
    (2, 1),
    (1, 2);
INSERT INTO Participant (member_id, group_event_id) VALUES
    (2, 1),
    (4, 2),
    (5, 1);
INSERT INTO Service (name, points, base_cost) VALUES
    ('Yoga Class', 10, 15.99),
    ('Fitness Climb', 15, 20.50),
    ('Membership', 0, 50);
INSERT INTO Transaction (member_id, date, service_type, amount_paid) VALUES
    (1, '2023-01-20', 'Membership', 50.00),
    (2, '2023-02-25', 'Yoga Class', 15.99),
    (4, '2023-03-12', 'Fitness Climb', 20.50);
INSERT INTO Fitness_Goal (member_id, name, description, complete) VALUES
    (1, 'Jetpack', 'Climb the Outerwall in less than a minute', FALSE),
    (2, 'Flexibility', 'Improve flexibility through yoga', TRUE),
    (5, 'Beat King in Spar', 'Kick his ass yooooo', FALSE);
INSERT INTO Achievement (description) VALUES
    ('Complete Sand Zone trail'),
    ('Win a Spar using the Nemesis'),
    ('All zones marathon in under 1 hour');
INSERT INTO Completed (member_id, achievement_id) VALUES
    (1, 2),
    (4, 1),
    (5, 3);
```

https://github.com/qwktrik/comp3005-final_project/blob/main/DML.sql

Select Statements

List all members

```
SELECT * FROM Member;
```

Members that have not participated in a group event

```
SELECT m.username FROM Member m
LEFT JOIN Participant p ON m.member_id = p.member_id
WHERE p.member_id IS NULL;
```

Get first members fitness goals

```
SELECT * FROM Fitness_Goal
WHERE member_id = 1;
```

Get member with ID 2 transactions after date

```
SELECT * FROM Transaction
WHERE member_id = 2
AND date > '2023-01-01'
ORDER BY date;
```

Show achievements and what members have completed them.

```
SELECT a.description AS achievement_description, m.username AS member_name
FROM Achievement a
JOIN Completed c ON a.achievement_id = c.achievement_id
JOIN Member m ON c.member_id = m.member_id;
```

List training sessions for Member ID 3

```
SELECT * FROM Training_Session WHERE member_id = 3;
```

List group events for Member ID 2

```
SELECT * FROM Group_Event
WHERE group_event_id IN (
    SELECT group_event_id FROM Participant WHERE member_id = 2
);
```

https://github.com/qwktrik/comp3005-final_project/blob/main/SELECT.sql

Update Statements

Update member weight

```
UPDATE Member SET weight = 78.5 WHERE member_id = 3;
```

Update training sessions notes

```
UPDATE Training_Session
SET notes = 'Great session see you next week'
WHERE trainer_id = 1 AND member_id = 2;
```

Update fitness goal as complete

```
UPDATE Fitness_Goal
SET complete = TRUE
WHERE member_id = 4 AND name = 'Flexibility';
```

https://github.com/qwktrik/comp3005-final_project/blob/main/UPDATE.sql

Conclusion

I did not have the time to add any application to go with this initially I was just going to make an ncuses client because I thought that would be fun. There are also no triggers or any extra stuff. I decided just to focus on the design as that was the main focus of the assignment.