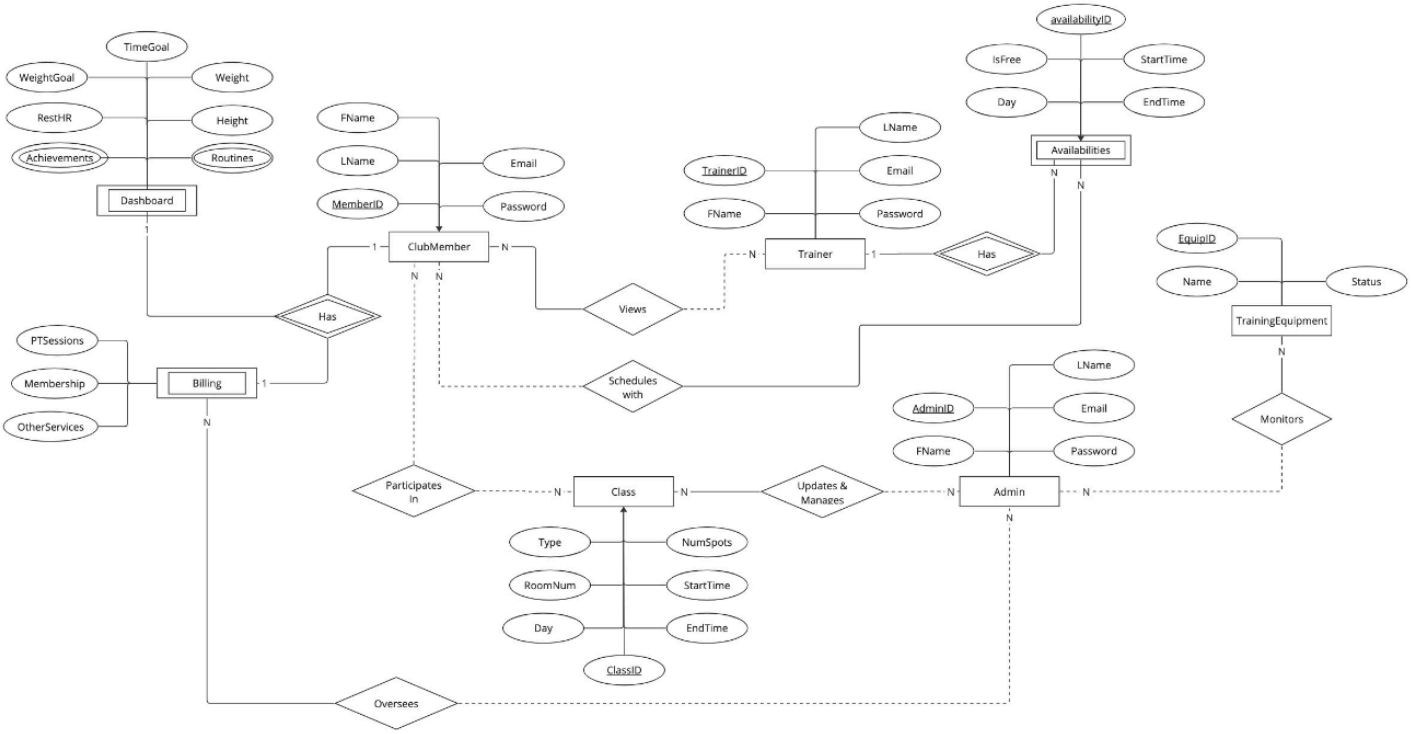
Conceptual Design



(for a clearer picture please refer to the github)

<u>Club member related conceptual design choices:</u>

- **Attributes**: name (first and last), email and password (to login), memberID
- Each member should **have** one dashboard. This dashboard keeps track of health statistics of their resting heart rate, height and weight, their goals (time and weight), as well as any achievements they have.
    - Dashboards are represented as weak entities, as they ARE tied to a club member and can exist on their own as the purpose of the dashboard is to be personalized to the club member
- Each club member may **participate in** a class, and a class can have multiple members
    - Partial participation is used to represent this relationship as not all club members have to attend classes, and class registration can exist without anyone registering for it
    - As for the classes' attributes they include: the type of class, the room number it is taking place in, the number of spots available for people to join, and the date and time for when the class occurs.
- Each Club member can **train with** a trainer for a training session

    - A club member has one trainer, and a trainer can take on multiple club members
    - Partial participation is also used here as not all club members have to sign up for training session with a trainer
        - But a trainer has train club members.
- Each Club member **has** a bill to cover
    - Billing details include the types of payments (Membership, Personal training sessions, OtherServices)
        - Membership bills are added to the system as the standard
        - Personal training session fees are applied whenever club members signs up for a training session
        - Other services fees include class fees (only applied when the club member participate in a class)
    - Billing is also represented as a weak entity as the billing is specific to the club member and can't exist on its own

<u>Trainer related conceptual design choices:</u>

- **Attributes:** name (first and last), email and password (to login), and trainerID
- A trainer **has** all their availabilities in the system
    - This relation is used to indicate all the times when a trainer is available for a training session
        - Thus this relation is a weak entity as its tied to the trainer
    - Availabilities have the day and time period to indicate when the trainer available
        - The isFree attribute is used to filter and then show only the training sessions that club member can choose
- Any trainer can **view** any club member's profile
    - Represented as partial participation relationship as trainers are able to view club member's profiles but the club member's aren't able to view the details of a trainer.
- See above for the **trains with** relation with club member

<u>Admin related conceptual design choices:</u>

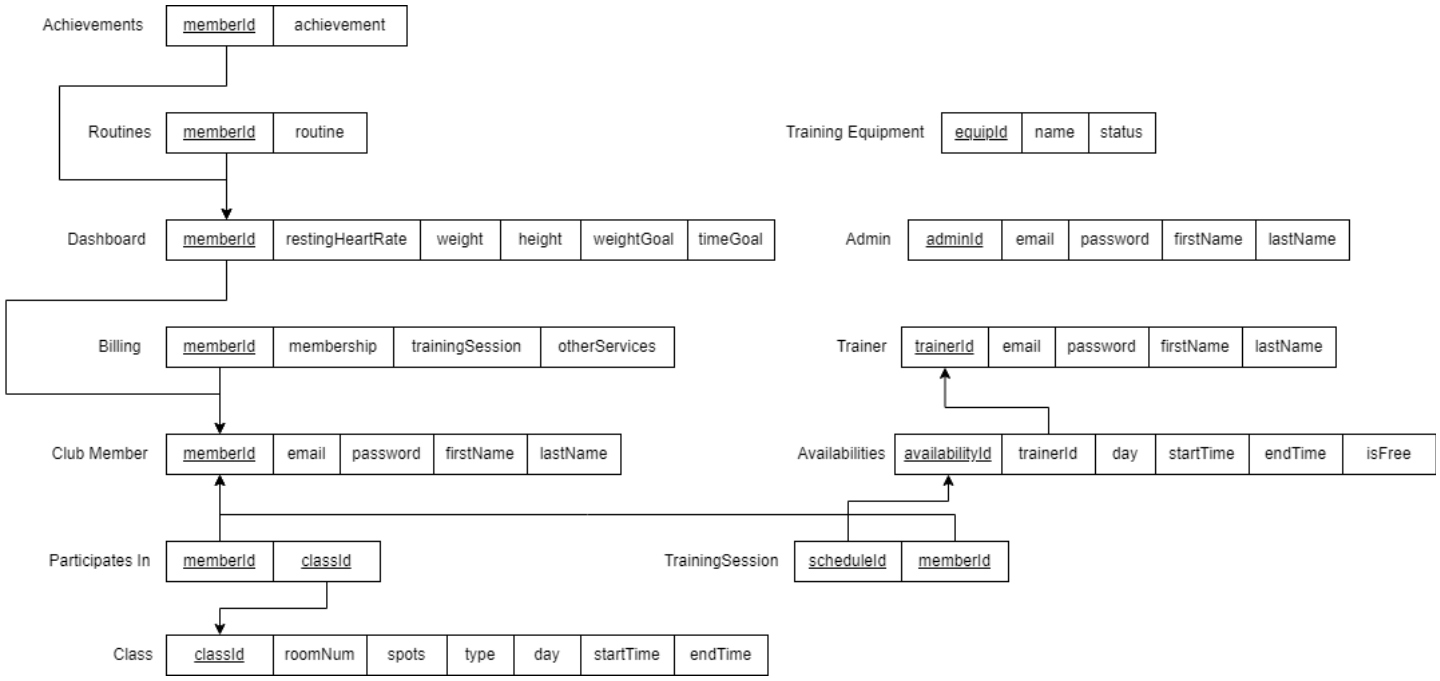
- **Attributes**: name (first and last), email and password (to login), and AdminID
- All relationships that the admin has are partial relationships as the admin should be able to interact with (and modify) any part of the system if they so wish
  (but not all admins are required to interact with all elements)
- Any admin can **monitor** any piece of training equipment (N:N)
    - Training equipment have the attributes of a name, status (the condition of the piece of equipment), and the equipment's ID

- Any admin can **manage** any of the bookings
- Any admin can **update and manage** any class
- Any admin can **oversee** the billings of any of the club members

Some additional assumptions that were made:

- Room bookings are exclusive to classes
    - (thus its implied that when an admin manages a class, they book a room for that class)

Reduction to Relation Schemas



Tables

Most of the translation from the Entity Relationship Diagram to the Relation Schema are simple as the entities become their own table with their respective attributes:

- ClubMember, Trainer, Admin, TrainingEquipment, Availabilities, Class, Dashboard, and Billing

The attributes Achievements and Routines that are a part of Dashboard must also be represented as their own table as these are multivalued attributes (club members can have multiple achievements and routines). Additionally two relations were represented as tables (due to N:N relationships), these are the Schedules With (TrainingSession), and Participates In. We wanted to avoid NULL values so we decided it would be best to represent these relations as a separate table.

Primary/Foreign Keys

- Club members are uniquely identified by their memberId, as this is guaranteed to be unique for every member. Then everything that is associated with the specific club member, such as their dashboard, achievements, routines, and billing all use the club members' id as a primary key.
- The Participates In table the memberId is used as a primary key as we intended for a club member to only be able to register for one class.
- Trainers are identified by their unique trainerId
- Schedule Id is used for both a trainer's Availability and for TrainingSession
    - This is because trainingSessions are dependent on the trainer's availabilities
- Admin's are identified by their unique trainerId
- Class's have their own classId to identify them
- Training equipment has their own equipId to identify them

Relations

- Note that admin does not display any relations between any of the other tables. While Admin had partial relations between monitoring training equipment, updating and managing classes, and overseeing billing, no record is kept in which admin specifically interacts with any of these entities.
    - Thus no adminId foreign keys are found on any of the other entities
- Training equipment is a similar case. It has no need to keep track of where it's located, or who uses it. The way training equipment is represented is just for the admin to keep track of its condition.

DDL File (see github for the file called DDL_Statements.sql)

DML File (see file initializeDB.sql)

```
--  ClubMember
INSERT INTO ClubMember (email, password, fname, lname)
VALUES
    ('bob@cm.com', 'bob', 'Bob', 'Z'),
    ('bobby@cm.com', 'bobby', 'Bobby', 'Z');

-- Trainer
INSERT INTO Trainer (email, password, fname, lname)
VALUES
    ('liamkelleher@trainer.com', 'liam', 'Liam', 'Kelleher'),
    ('lilyczarnecki@trainer.com', 'lily', 'Lily', 'Czarnecki'),
    ('hughhang@trainer.com', 'hugh', 'Hugh', 'Hang');

-- Availabilities
INSERT INTO Availabilities (trainerID, day, startTime, endTime, isFree)
VALUES
    (1, '2023-10-01', '08:00:00', '10:00:00', TRUE),
    (1, '2023-10-01', '10:00:00', '12:00:00', FALSE),
    (2, '2023-10-02', '09:00:00', '11:00:00', TRUE),
    (2, '2023-10-02', '13:00:00', '15:00:00', TRUE),
    (3, '2023-10-03', '10:00:00', '12:00:00', TRUE),
    (3, '2023-10-03', '14:00:00', '16:00:00', FALSE);

-- Admin
INSERT INTO Admin (email, password, fname, lname)
VALUES
    ('admin1@admin.com', 'admin1', 'Admin', '1'),
    ('admin2@admin.com', 'admin2', 'Admin', '2'),
    ('admin3@admin.com', 'admin3', 'Admin', '3');
```

```
-- Insert sample data into TrainingEquipment
INSERT INTO TrainingEquipment (name, status)
VALUES
    ('Bench', 'Good'),
    ('Squat rack', 'Decent'),
    ('Treadmill', 'Not functional');
```

Implementation

Our application is in the form of a Command-Line Interface and we have programmed in using Python. The program is divided into five different python files (connection.py, program.py, admin.py, clubMember.py, and trainer.py).

The connection.py file is responsible for connecting to the database in postgres. It takes care of making queries to the database that are required within the other classes (see function executeQuery(query, parameters=())). Additionally this class also takes care of logging in for the different types of users (clubMember/admin/trainer).

The program.py file is the main flow of the program, this is the file that runs on startup. The user will be presented with options, where they can login (or register as a club member) by choosing the appropriate method. Choosing one of the methods to login will direct the logic flow to the appropriate class, where the classes each contain methods that support the queries they need to make.

The clubMember class contains all of the queries relevant to what a member can do once they are a club member. Just like in the main flow, the user is now presented with a separate menu of options, where they can select the option by entering in the corresponding number. From there they enter in whatever inputs are necessary to complete the action.

This process is also the same for the trainer and admin for their respective actions they can take.