COMP 3005 Final Project Report

Zhenxuan Ding, Jiayu Hu, Andrew Verbovsky

Introduction

Implementation

A web-based application is built to implement the modification and visualization of the database. It is designed to be used by the Fitness Club's members, trainers and administrators.
The frameworks and libraries used for building the web app includes:

- Next.js - Web Framework
- NextAuth.js - Authentication Library
- Mantine 7.6.2 - Components Library
- node-postgres - PostgreSQL database driver for Node.js applications

Architecture:

1. Page directory: each file in the pages directory corresponds to a route in the application.
2. API routes: API routes are created under the pages/api directory, handling server-side logic (authorization, registration, etc.) and can be accessed from the client-side code.
    - **Each API route contains the corresponding functions (PostgreSQL queries) to handle CRUD operations.**

DDL/DML files

They are located in the /SQL directory. Furthermore, there is a DQL.sql file that displays all tables to verify the state of the database.

Bonus Features

Please consider giving us bonus marks for the following features:

1. A complete web app supporting multiple concurrent sessions, instead of a CLI program
2. Moreover, the web app is deployed on Vercel (https://comp-3005-w24.vercel.app/)
3. Authorization system based on NextAuth
4. Users with different roles have different frontend features, they are limited to viewing pages appropriate for their roles
5. Normalized database (in BCNF) to minimize redundancy and ensure consistency
6. Soft-deletion of accounts and rooms to retain data and referential integrity for future reference (as of now, only rooms can be deleted in the app, although deleted users are barred from logging in, deleting can only be done through the DBMS)
   Note: ON DELETE CASCADE clauses are on every table for the worst case scenario, hard-delete of accounts and rooms will lead to deletion of dependent data elsewhere

Conceptual Design

Assumptions

Assumptions for app features:

- **User Registration**: For the simplicity of testing, we assume that all types of accounts can be created on the registration page (not the case in actual commercial systems).
- **Profile Management**: We assume “personal information” only refers to age and gender.
- **Dashboard Display**: We assume that the “health statistics” on the dashboard will simply be the average of historical health metrics and the current metrics. The “achievements” include past attended classes, sessions, and completed fitness goals (member_goal). “Exercise routines” is essentially a member-defined todo list.
- **Member Profile Viewing**: We assume that the trainer is only permitted to view the age and gender of members.
- **Room Booking Management**: We assume room booking is for group classes only, so the booking system is a part of the schedule management system. We also assume that multiple classes can take place inside a room at a time.
- **Class Schedule Updating**: Since there is no information on who creates group classes, we assume that the administrator is responsible for this, and that creation and update would use the same system.
- **Billing and Payment Processing**: We assume that the system can generate bills automatically (upon personal training session and class signup), but an administrator can also create bills for members manually. Members can view and pay their own bills on their page, administrators can view all historical bills and manually set bills as paid (representing cash transaction).
- **Schedule Management**: We assume that the trainer can create time slots where they are available for booking, members can then book personal sessions (the fee is calculated based on session length and hourly fee set by the trainer) during these slots. Likewise, admins can book group classes during these slots, in this case, the admin sets the fee for joining the class. A member can then join available group classes. Bills for sessions and classes are generated automatically by the system. The scheduling is on a first-come-first-serve basis, a time slot can only have one class or session. Once scheduled, a personal session can be canceled before completion, and there will be a full refund to the member. For the sake of transparency, rescheduling is only done by canceling a session and booking a new one, as different trainers have different rates and booking fees. A group class cannot be canceled (but an admin can mark it as completed, or book a different time slot for it).

Assumptions for each database table (with explanation of the ER Model):

- **account**: All 3 types of users (members, trainers and administrators) will use the same platform, a user can only have one **user_type**. Each user needs a unique **username**, which is the primary key and login credential, but they can have identical **names** (that correspond to real names). All **accounts** should only be soft-deleted by marking the **is_deleted** attribute as **TRUE**.

- **administrator**: Attribute-defined specialization of **account**, does not have other attributes.

- **trainer**: Attribute-defined specialization of **account**, each trainer has a **rate_per_hour** attribute, which can be set by the trainer upon account creation, there is a constraint mandating it to be a positive number.

- **trainer_specialty**: We assume a **trainer** can have multiple specialties (a multi-valued attribute).

- **member**: Attribute-defined specialization of **account**, each member has an **age** attribute (an integer between 0 and 150) and a **gender** (enum: {"male", "female", "other"}) attribute that they can define when they create the account, they can also update these later in their profile.

- **exercise_routine**: Each member can have multiple routines which they can freely create or delete in their profile (a multi-valued attribute). We assume this can be as simple as a string (a to-do list).

- **member_goal**: A **member** can have any number of **member_goals** which they can freely create (or mark as complete) in their profile. Since each **member_goal** cannot be identified by its attributes alone, it is a weak entity.

- **has_goal**: Since **member_goal** is a weak entity**,** the relation between it and **member "has_goal"** is a weak relationship (one-to-many from strong entity to weak entity, the weak entity has total participation).

- **health_metric**: A **member** can have any number of **health_metrics**. Since each **health_metric** cannot be identified by its attributes alone, it is a weak entity. The **weight** is in lbs. The composite attribute **"blood_pressure"** is formed by "**systolic_pressure**" and "**diastolic_pressure**" (both are in mmHg, rounded to the closest integer).

- **uploaded:** Since **health_metric** is a weak entity**,** the relation between it and **member "uploaded"** is a weak relationship (one-to-many from strong entity to weak entity, the weak entity has total participation).

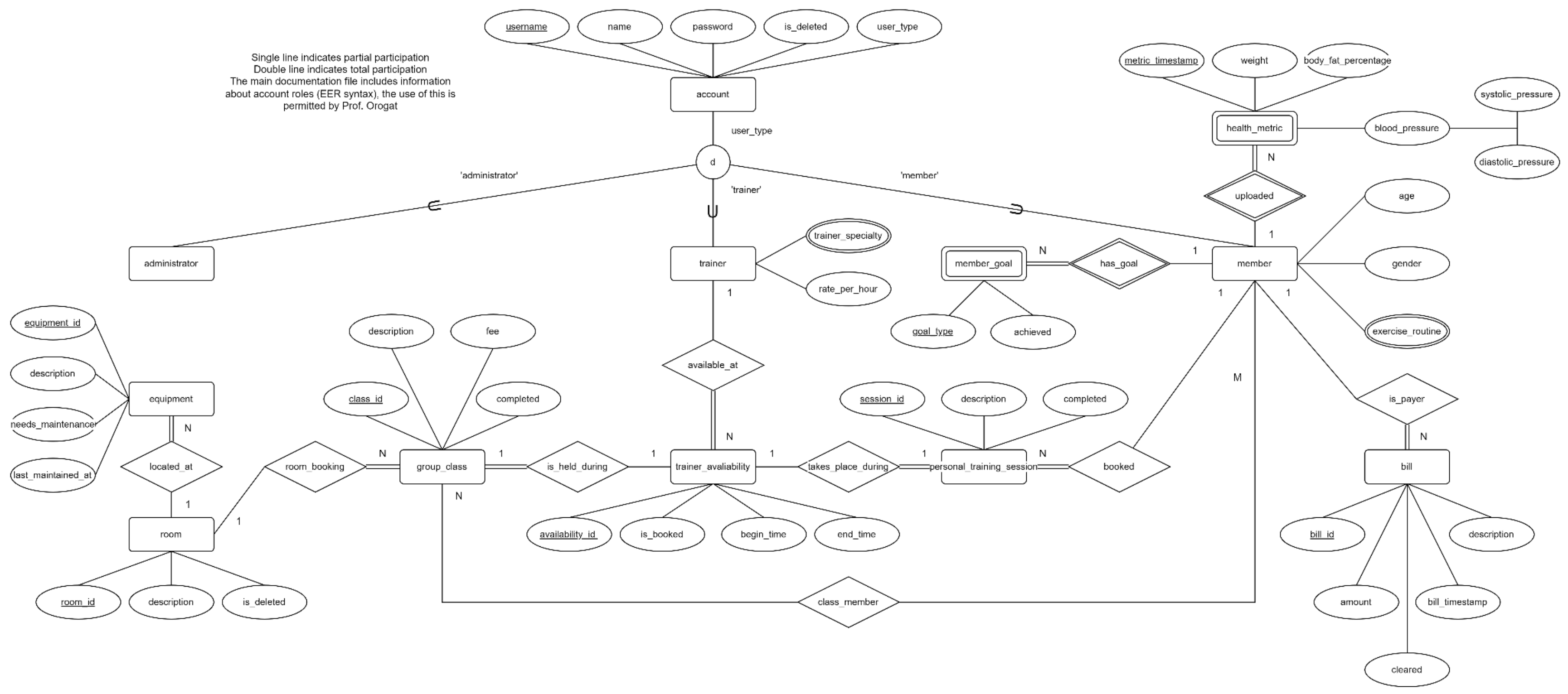
- **trainer_availability**: Represents a time slot where a **trainer** is available, each can be associated with a **group_class** or a **personal_training_session**, it is a strong entity and has its own primary key. Attributes **begin_time** and **end_time** describe the scheduling, **end_time** must be later than **begin_time**. We assume that there is no implicitly set duration (the duration can be any positive time interval).

- **available_at**: One-to-many relationship. Each **trainer_availability** must be associated with exactly one **trainer** (total participation) but a **trainer** may have listed no **trainer_availabilities** (partial participation) or many of them.

- **personal_training_session**: Represents a personal training session booked by a **member** during an available time slot (**trainer_availability**).
- **takes_place_during**: One-to-one relationship. Each **personal_training_session** must take place during a **trainer_availability** (total participation) but a **trainer_availability** can have no **personal_training_session** (partial participation).
- **booked**: One-to-many relationship. Each **personal_training_session** must be booked by exactly one **member** (total participation) but a **member** may have booked no **personal_training_sessions** (partial participation) or many of them.
- **group_class**: Represents a group class created by an **administrator** during an available time slot (**trainer_availability**).
- **is_held_during**: One-to-one relationship. Each **group_class** must take place during a **trainer_availability** (total participation) but a **trainer_availability** can have no **group_class** (partial participation).
- **class_member**: Many-to-many relationship. We assume a **group_class** can have any number of participating **members**, a **member** can take part in any number of **group_classes** (both partial participation).
- **room**: Represents a room available for group classes. All **rooms** should only be soft-deleted by marking the **is_deleted** attribute as **TRUE**. A deleted room may be recovered (the code for this is not required and thus there is no implementation).
- **room_booking**: One-to-many relationship. A **room** can be booked for any number of **group_classes** (partial participation), we assume more than one class can take place in a room at once. Every **group_class** must be booked at a certain room (total participation). **Administrators** are responsible for both room booking and class scheduling when they create a **group_class**.
- **equipment**: Represents a piece of equipment and its maintenance status, we assume that each equipment piece must be located in a **room**.
- **located_at**: One-to-many relationship. A **room** can contain any number of **equipment** (partial participation). A piece of **equipment** must be in exactly one **room** to exist (total participation).
- **bill**: Represents a transaction (created automatically upon session/class registration, or by an **administrator**). The **amount** can be negative in the case of a refund.
- **is_payer**: One-to-many relationship. A **member** may be associated with any number of bills (partial participation). A **bill** must be associated with exactly one **member** to exist (total participation).

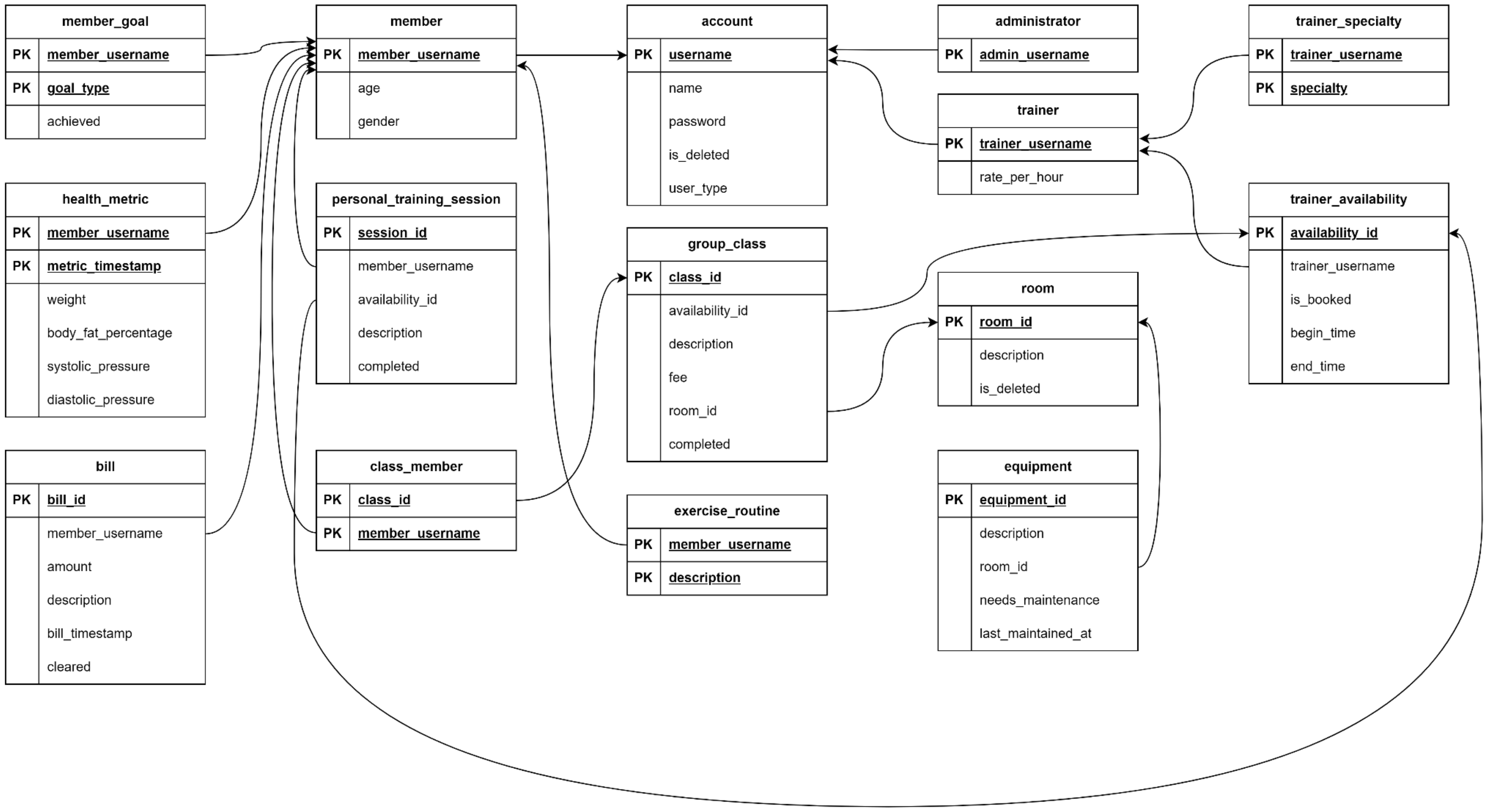
ER Diagram

Full size diagrams are included in the Documentation folder. This diagram uses EER (Enhanced Entity Relationship) syntax to properly represent the subclass relationship between accounts and different member roles. Professor Orogat told me about this, relevant info can be found on page 114 of *Fundamentals of Database Systems* by R. Elmasri and S. B. Navathe: https://www.auhd.edu.ye/upfiles/elibrary/Azal2020-01-22-12-28-11-76901.pdf



Reduction to Relational Schemas

Full size diagrams are included in the Documentation folder. The many-to-many relationship “class_member” is mapped to the “class_member” table here. All other tables represent either entities, or multi-valued attributes with the same name.



Appendix: Normalization

The underlying database is designed with minimal redundancy and maximal consistency in mind, it satisfies the requirements of the Boyce–Codd Normal Form (BCNF, or 3.5NF). The requirements of BCNF is:
For every functional dependency X → Y in a table, at least one of the following conditions is true:

- X → Y is a trivial functional dependency (Y ⊆ X)
- X is a superkey

We mainly focus on the second point (for all tables, all values are solely determined by the superkey, which is the primary key).

The database contains 15 tables, 5 of which are trivially in BCNF:

- **administrator**: only contains a PK.
- **trainer**: only contains PK **trainer_username** and another attribute **rate_per_hour**.
- **trainer_specialty**, **exercise_routine**, **class_member**: only contain composite PKs.

The rest of the tables are in BCNF because no attribute is determined by another non-primary key attribute, there are 2 important clarifications:

- **trainer_availability**: the constraint **CHECK (begin_time < end_time)** does not violate BCNF because the difference between **begin_time** and **end_time** can be any value as long as it satisfies the constraint, there is no implicit preset class length in this program.
- **health_metric**: similarly, the difference between **systolic_pressure** and **diastolic_pressure** can be any value satisfying their constraint, therefore, this constraint does not violate BCNF.