Alexandre Berube

101054165

April 13, 2024

COMP 3005 Final Project

2.1 Conceptual Design

Requirement	Assumption	Representation in ER Model
• Members register, identified by an id number and first/last names	N/A	Members Entity: Attributes include member_id (PK), first_name, and last_name.
• Members can establish fitness goals	• No limit to goal(s) specified • Fitness goals are not necessarily required to be used by members; therefore, total participation is not necessary.	Fitness Goals Entity: Attributes include member_id (FK and PK), goal_title, end_date, initial_value, and goal_value. Relationship: "Establish" between Members and Fitness Goals. One-to-many from Members to Fitness Goals, as each member can have multiple goals, but each goal has one member.
• Members can keep track of health metrics	• Each metric is optional to what the user wants to track • Total participation is not necessary since metrics are optional • Assumption that each member has only one set of metrics at a given time, and these are simply updated periodically. Members do not keep multiple sets of metrics at once.	Metrics Entity: Attributes include member_id (FK and PK), weight, age, bmi, daily_calories, bodyfat_percent. Relationship: "Tracks" between Members and Metrics. One-to-one relation from Members to Metrics, each member has one set of metrics at once and each set of metrics is maintained by a single member.
• Members can track health statistics	• Health stats will update periodically based on information contained in the 'metrics'	Health Stats Entity: member_id (PK and FK), date_of_entry, weight, age, bmi, bodyfat_percent.

	• Members themselves are not updating this information, it is instead periodically updated based on information within the "Metrics" and maintains information for each subsequent update. • Total participation for both, since if Metrics are used by the member then so will Health Stats. Likewise, Health Stats will only exist if Metrics are being used.	Relationship: "Updates" between Health Stats and Metrics. One-to-one relation, as there is only one of each for each assigned member. Relationship: "Maintain" between Health Stats and Members. One-to-Many relation from members to health stats, as each member could have many health stats if they have had frequent updates, but each health stats would have one member assigned to it.
• Members can schedule training sessions with a trainer, or in a class	• Members simply need to choose an open timeslot, whether it be a trainer or a class	Trainer Booking Entity: attributes include trainer_id (FK), member_id (FK), confirmation_number (PK), date, and time. Relationship: "Schedule" between Members and Trainer Booking. One-to-Many relation from members to trainer booking, as each member can book multiple sessions, but each individual session has one member. Relationship: "Sign up" between Members and Class Schedule. Many-to-Many relation from members to class schedule, as multiple members can sign up for classes, and members can sign up to multiple classes each.
• Trainers are identified by an id number, and first/last names	N/A	Trainer Entity: attributes include trainer_id (PK), first_name, last_name.
• Trainers can manage their own schedules	• Trainers would work 5 days a week, typically	Trainer Schedule Entity: attributes include trainer_id

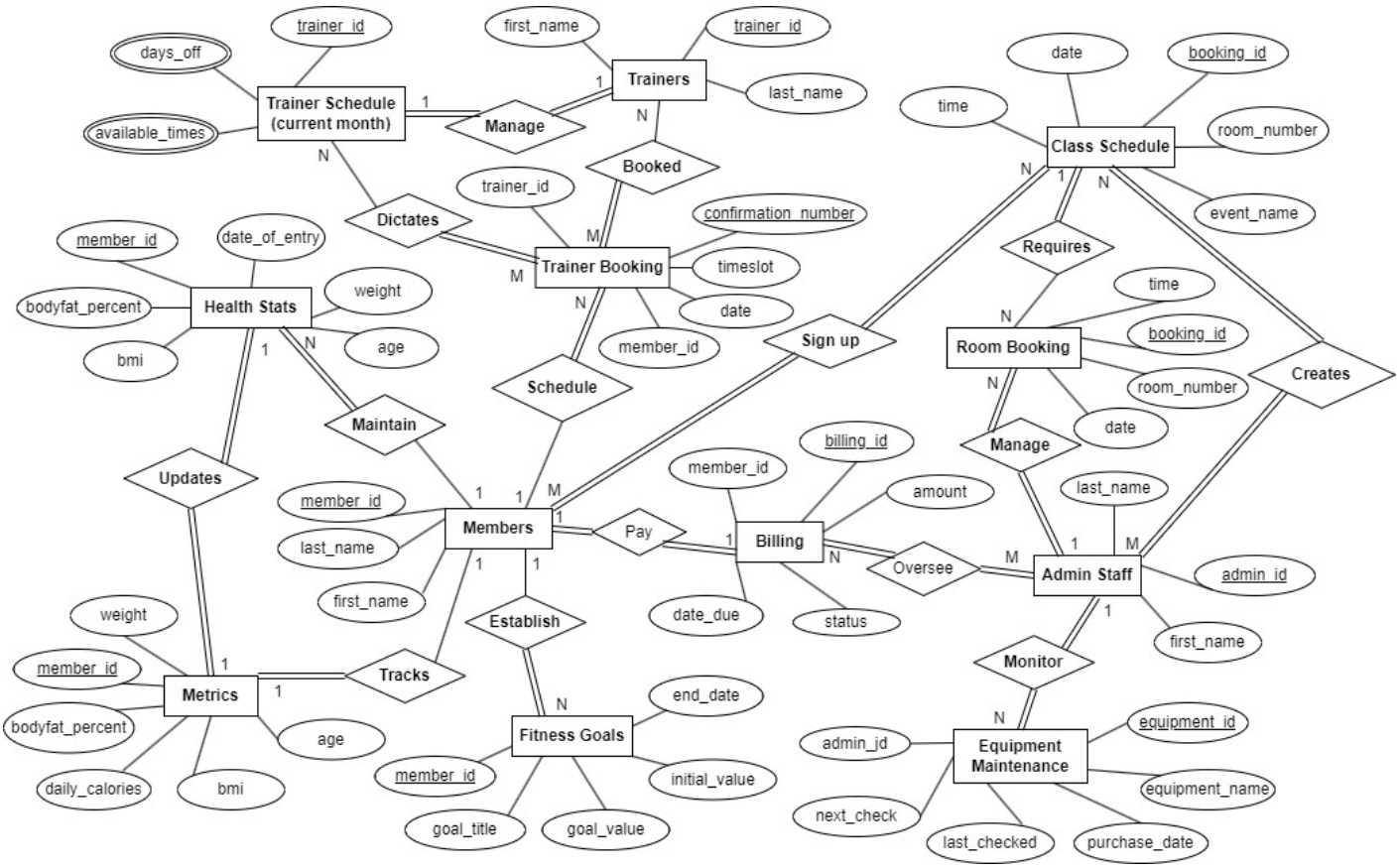
	having the same 2 days off a week for a given month (these 2 days can change month-to-month • Assumption made that while a trainer booking must of course have a trainer assigned, a trainer may not necessarily have any training booked.	(FK and PK), days_off, and available_times. Relationship: "Manage" between Trainers and Trainer Schedule. One-to-one relationship from Trainers to Trainer Schedule as each trainer has one schedule and each schedule has one trainer. Relationship: "Dictates" between Trainer Schedule and Trainer Booking. Many-to-Many relation from trainer schedule to trainer booking, as there are multiple trainer schedules (multiple trainers) and each of these trainers can have multiple bookings. Relationship: "Booked" between Trainers and Trainer Booking. Many-to-Many relation from Trainers to Trainer Booking, as each trainer can be booked multiple times, and each timeslot could have multiple trainers at a time.
• Admin staff are identified by an admin_id, and their first/last names	N/A	Admin Staff Entity: attributes include admin_id (PK), first_name, last_name.
• Admin staff can manage the room bookings	• Room booking will only be accessible to admins, and are created with an id, date, time, and event name	Room Booking Entity: attributes include booking_id (PK), room_number, date, and time. Relationship: "Manage" between Admin Staff and Room Booking. One-to-Many relation from admin staff to room booking, as each booking would be arranged by a single staff member, but a

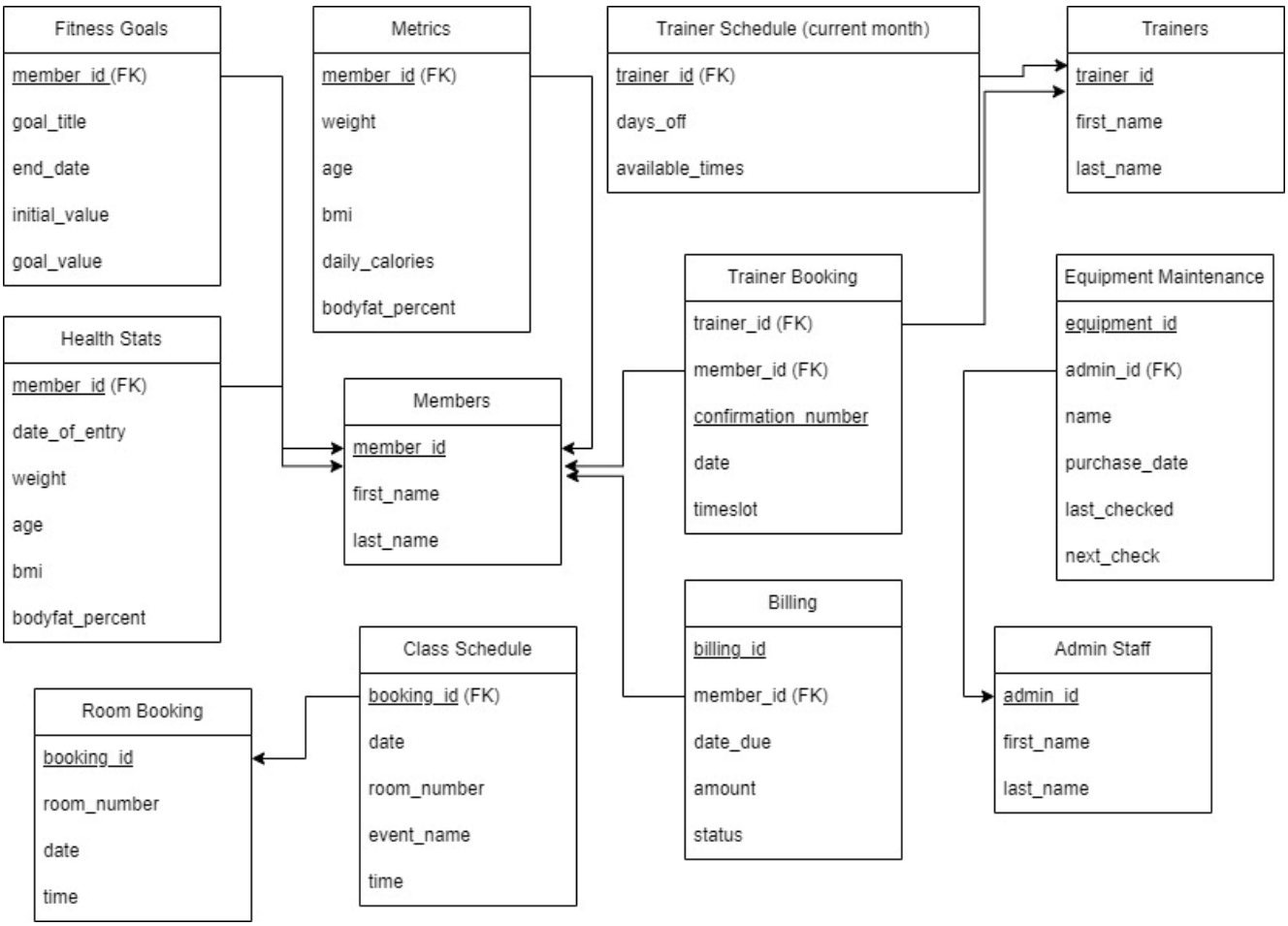
		single staff member may manage multiple room bookings.
• Admin staff monitor the equipment maintenance	• Monitor equipment based on id, name, date purchased, last checked, and note down the next check date	Equipment Maintenance Entity: attributes include equipment_id (PK), equipment_name, purchase_date, last_checked, next_check. Relationship: "Monitor" between Admin Staff and Equipment Maintenance. One-to-Many relation from admin to equipment maintenance, as there would be one admin staff per equipment piece monitored, but a single admin staff could monitor multiple pieces of equipment.
• Admin staff update/create class schedules	• Classes are scheduled to a room that is free with an id, given a name and the time it is at • A class schedule can have multiple room bookings. For example, a weekly class every Tuesday at 6 pm would still need a room booking for each of those classes. • A room booking is for an individual class session, not for every class in a schedule.	Class Schedule Entity: attributes include booking_id (PK and FK), date, room_number, event_name, and time. Relationship: "Creates" between Admin Staff and Class Schedule. Many-to-Many relation from admin to class schedules, as there could be multiple staff that create and update the schedules, and admin would update/create multiple class schedules. Relationship: "Requires" between Room Booking and Class Schedule. One-to-Many relation from Class Schedule to Room Booking, as class schedule might have multiple different rooms booked for different sessions, but each

		booking would only be associated with a single class.
• Admin staff oversee the billing to members	• Need to track date payment is due, member, amount, and whether it has been paid or not	Billing Entity: attributes include billing_id (PK), member_id (FK), date_due, amount, and status. Relationship: “Pay” between Billing and Members. One-to-one relation from members to billing, as each member maintains a single payment (not being charged multiple times), and each billing comes from a single member. Relationship: “Oversee” between Admin Staff and Billing. Many-to-Many relation from admin staff to billing, as there is no limit to how many billings can be overseen, and no limit to the amount of admin staff that can oversee billing.
• Admin staff process payments for membership fees and other services	• This would go in the same table as the above section, under ‘billing’	See Above entry.

2.2 Reduction to Relation Schemas

Note: In ER diagram, all relationship and entity names have been boldened for increased visibility.





2.3 DDL

```
CREATE TABLE "Admin Staff" (
        admin_id serial PRIMARY KEY,
        first_name TEXT NOT NULL,
        last_name TEXT NOT NULL
);

CREATE TABLE Members (
        member_id serial PRIMARY KEY,
        first_name TEXT NOT NULL,
        last_name TEXT NOT NULL
);
```

```
CREATE TABLE Trainers (
        trainer_id serial PRIMARY KEY,
        first_name TEXT NOT NULL,
        last_name TEXT NOT NULL
);

CREATE TABLE "Fitness Goals" (
        member_id serial PRIMARY KEY REFERENCES Members(member_id),
        goal_title TEXT NOT NULL,
        end_date DATE NOT NULL,
        initial_value TEXT NOT NULL,
        goal_value TEXT NOT NULL
);

CREATE TABLE Metrics (
        member_id serial PRIMARY KEY REFERENCES Members(member_id),
        weight TEXT,
        age INT,
        bmi INT,
        daily_calories INT,
        bodyfat_percent INT
);

CREATE TABLE "Health Stats" (
        member_id serial PRIMARY KEY REFERENCES Members(member_id),
        date_of_entry DATE NOT NULL DEFAULT CURRENT_DATE,
        weight INT,
        age INT,
        bmi INT,
```

```
    bodyfat_percent INT
);

CREATE TABLE "Room Booking"(
    booking_id serial PRIMARY KEY,
    room_number INT NOT NULL,
    date DATE NOT NULL,
    time TEXT NOT NULL
);

CREATE TABLE "Class Schedule"(
    booking_id serial PRIMARY KEY REFERENCES "Room Booking"(booking_id),
    date DATE NOT NULL,
    room_number INT NOT NULL,
    event_name TEXT NOT NULL,
    time TEXT NOT NULL
);

CREATE TABLE "Trainer Schedule"(
    trainer_id serial PRIMARY KEY REFERENCES Trainers(trainer_id),
    days_off TEXT[] NOT NULL,
    available_times TEXT[] NOT NULL
);

CREATE TABLE "Trainer Booking"(
    trainer_id serial REFERENCES Trainers(trainer_id),
    member_id serial REFERENCES Members(member_id),
    confirmation_number serial PRIMARY KEY,
    date DATE NOT NULL,
    timeslot TEXT NOT NULL
```

```
);

CREATE TABLE Billing(
        billing_id serial PRIMARY KEY,
        member_id serial REFERENCES Members(member_id),
        date_due DATE NOT NULL,
        amount NUMERIC(7,5) NOT NULL,
        status TEXT NOT NULL
);

CREATE TABLE "Equipment Maintenance" (
        equipment_id serial PRIMARY KEY,
        admin_id serial REFERENCES "Admin Staff"(admin_id),
        equipment_name TEXT NOT NULL,
        purchase_date DATE NOT NULL,
        last_checked DATE NOT NULL,
        next_check DATE NOT NULL
);
```

2.7 GitHub Repository

https://github.com/Rormund/COMP-3005-Final

3 Video Demonstration

https://youtu.be/3bX517WN-fQ