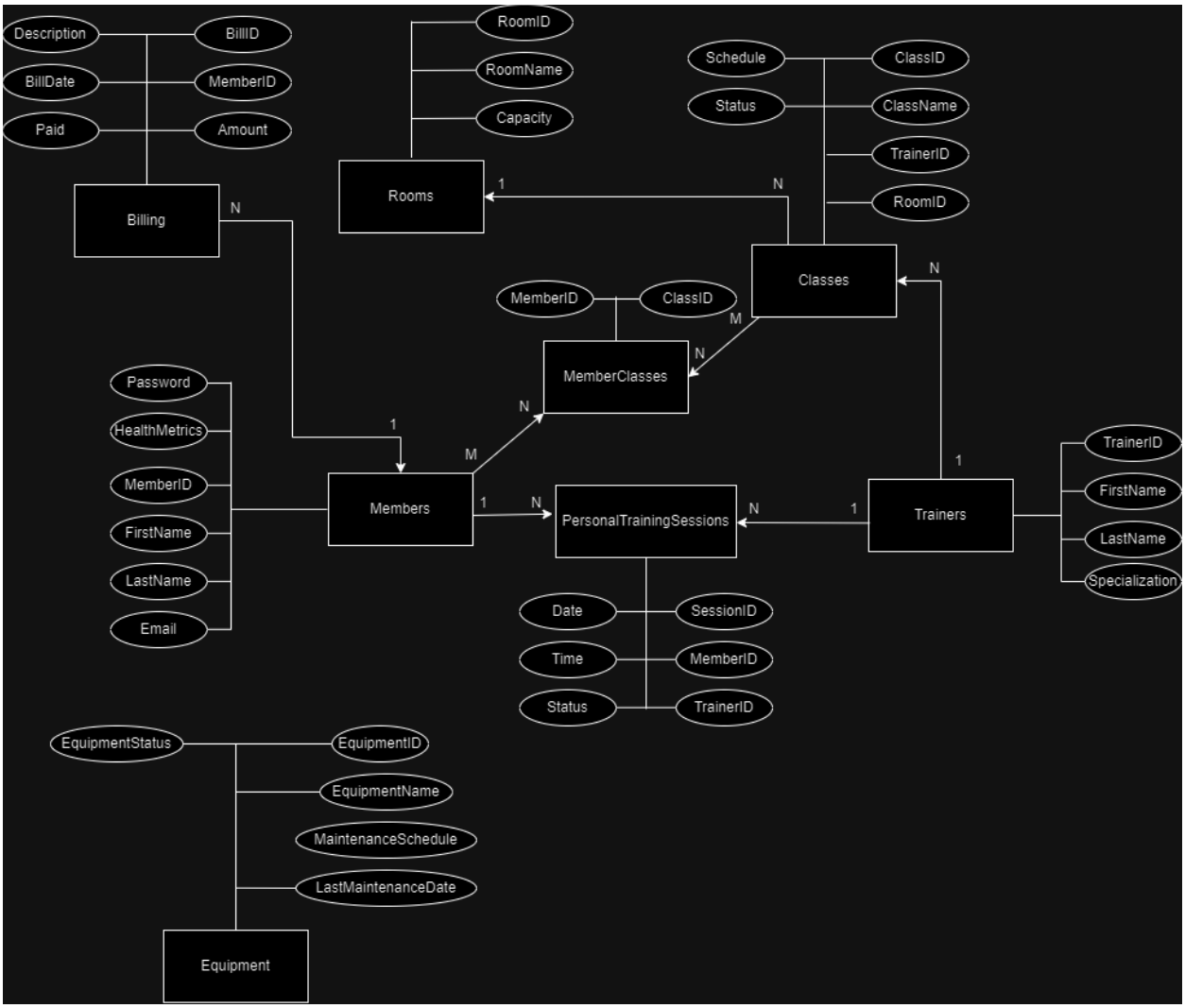
Project Report

COMP 3005 Project 2

Jonathan Dorfman

101241425

2.1 Conceptual Design



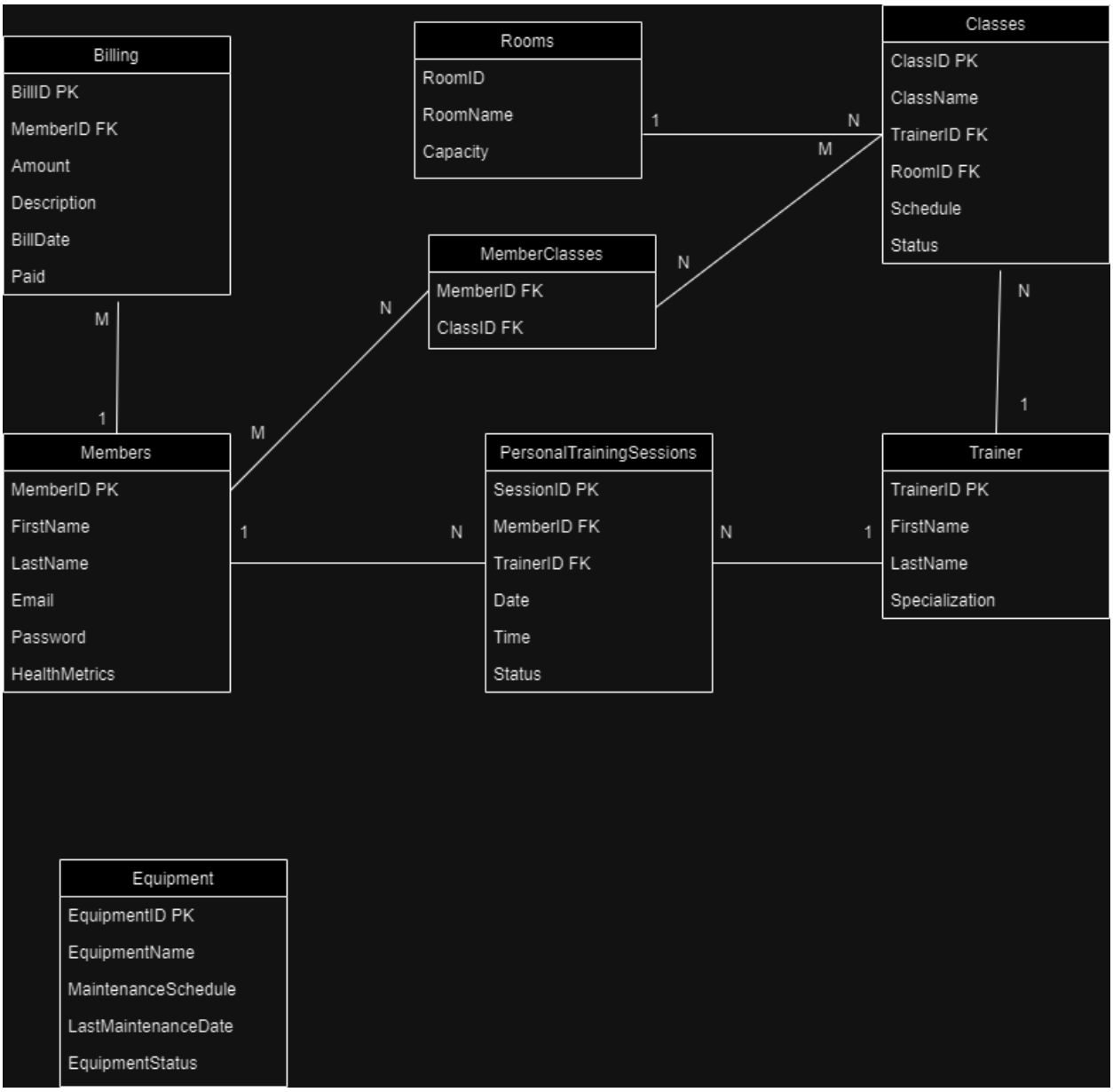
Assumptions for Database Design:

- Members: Every member is unique and can be identified by a MemberID. Members can participate in multiple personal training sessions and can enroll in multiple classes, reflecting diverse health and fitness goals.

- Trainers: Trainers are unique individuals identified by TrainerID and can conduct multiple training sessions and classes. They have specialties but are flexible in the types of classes they can lead, allowing for a dynamic schedule.
- Classes: Each class is unique, requires a room, and is led by a trainer. Classes can accommodate multiple members, and each class has a status indicating whether it is open for more members to join.
- Rooms: Rooms are unique spaces identified by RoomID and are assumed to have a capacity limit. Each room can host multiple classes at different times.
- Personal Training Sessions: These are specific interactions between a member and a trainer and are scheduled at specific times, requiring availability checks for trainers.
- Members to Personal Training Sessions: One-to-Many. Each member can book none or multiple personal training sessions.
- Trainers to Personal Training Sessions: One-to-Many. Each trainer can be scheduled for none or multiple sessions.
- Members to Classes: Many-to-Many. Members can enroll in multiple classes and each class can include multiple members, necessitating an associative entity (MemberClasses).
- Trainers to Classes: One-to-Many. A trainer can lead multiple classes but each class is led by only one trainer.
- Classes to Rooms: Many-to-One. Multiple classes can be held in a single room, but each class occurs in one specific room at a time.
- Mandatory Participation: Every class must have a trainer and a room; therefore, the participation of trainers in Classes and rooms in Classes is mandatory (total participation).
- Optional Participation: Members and trainers have optional participation in personal training sessions as not all members or trainers may choose to participate in personal training.
- Billing: Each billing entry is directly linked to a member, ensuring that all financial transactions are trackable to individual members. This setup supports the administrative function of managing finances efficiently.
- Equipment Maintenance: Equipment has scheduled maintenance dates, and the system allows for tracking the status (Available or Under Maintenance), aligning with administrative duties to ensure equipment availability and safety.
- Data Integrity and Security: Assume robust security measures for sensitive data like email and passwords. Passwords are stored securely, and emails are unique to prevent duplicate registrations.

- The system is designed to accommodate changes in class schedules, trainer assignments, and room bookings without significant restructuring, supporting administrative functions efficiently.
- Health metrics are stored in a flexible JSON format, allowing for the addition of new types of health data without altering the database schema.

2.2 Reduction to Relation Schemas



2.3 DDL File

-- Members Table

```
CREATE TABLE Members (
    MemberID SERIAL PRIMARY KEY,
    FirstName VARCHAR(255) NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    Email VARCHAR(255) UNIQUE NOT NULL,
    Password VARCHAR(255) NOT NULL,
    HealthMetrics JSONB -- For storing health metrics like weight, goals, etc.
);

-- Trainers Table
CREATE TABLE Trainers (
    TrainerID SERIAL PRIMARY KEY,
    FirstName VARCHAR(255) NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    Specialization VARCHAR(255)
);

-- Rooms Table
CREATE TABLE Rooms (
    RoomID SERIAL PRIMARY KEY,
    RoomName VARCHAR(255) NOT NULL,
    Capacity INT CHECK (Capacity > 0) NOT NULL
);

-- Classes Table
CREATE TABLE Classes (
    ClassID SERIAL PRIMARY KEY,
    ClassName VARCHAR(255) NOT NULL,
    TrainerID INT NOT NULL,
```

```
    RoomID INT NOT NULL,
    Schedule TIMESTAMP NOT NULL,
    Status VARCHAR(255) DEFAULT 'Open', -- Added status field
    FOREIGN KEY (TrainerID) REFERENCES Trainers(TrainerID),
    FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID)
);

-- Personal Training Sessions Table
CREATE TABLE PersonalTrainingSessions (
    SessionID SERIAL PRIMARY KEY,
    MemberID INT NOT NULL,
    TrainerID INT NOT NULL,
    Date DATE NOT NULL,
    Time TIME NOT NULL,
    Status VARCHAR(255) DEFAULT 'Scheduled', -- Added status field
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
    FOREIGN KEY (TrainerID) REFERENCES Trainers(TrainerID)
);

-- Equipment Table
CREATE TABLE Equipment (
    EquipmentID SERIAL PRIMARY KEY,
    EquipmentName VARCHAR(255) NOT NULL,
    MaintenanceSchedule DATE NOT NULL,
    LastMaintenanceDate DATE, -- Added last maintenance date field
    EquipmentStatus VARCHAR(255) DEFAULT 'Available' -- Added equipment status field
);

-- Member Classes Table (Association Table for Members and Classes)
```

```
CREATE TABLE MemberClasses (
  MemberID INT NOT NULL,
  ClassID INT NOT NULL,
  PRIMARY KEY (MemberID, ClassID),
  FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
  FOREIGN KEY (ClassID) REFERENCES Classes(ClassID)
);

-- Billing Table
CREATE TABLE Billing (
  BillID SERIAL PRIMARY KEY,
  MemberID INT NOT NULL,
  Amount DECIMAL(10, 2) NOT NULL,
  Description TEXT,
  BillDate DATE NOT NULL,
  Paid BOOLEAN NOT NULL DEFAULT FALSE,
  FOREIGN KEY (MemberID) REFERENCES Members(MemberID)
);
```

2.4 DML File

```
BEGIN;

-- Resetting the sequence for clear starting points in case of repeated tests
ALTER SEQUENCE trainers_trainerid_seq RESTART WITH 1;
ALTER SEQUENCE rooms_roomid_seq RESTART WITH 1;
ALTER SEQUENCE classes_classid_seq RESTART WITH 1;
ALTER SEQUENCE members_memberid_seq RESTART WITH 1;

-- Insert Members
```

```
INSERT INTO Members (FirstName, LastName, Email, Password, HealthMetrics) VALUES

('John', 'Doe', 'john.doe@example.com', 'password123', '{"weight": 80, "height": 180, "body_fat": 15, "fitness_goals": {"goal_weight": 75, "goal_body_fat": 12}}'),

('Jane', 'Smith', 'jane.smith@example.com', 'password123', '{"weight": 65, "height": 165, "body_fat": 25, "fitness_goals": {"goal_weight": 60, "goal_body_fat": 20}}'),

('Alice', 'Johnson', 'alice.johnson@example.com', 'password123', '{"weight": 70, "height": 170, "body_fat": 30, "fitness_goals": {"goal_weight": 65, "goal_body_fat": 25}}'),

('Bob', 'Brown', 'bob.brown@example.com', 'password123', '{"weight": 90, "height": 175, "body_fat": 20, "fitness_goals": {"goal_weight": 85, "goal_body_fat": 15}}'),

('Emma', 'Williams', 'emma.williams@example.com', 'password123', '{"weight": 55, "height": 160, "body_fat": 22, "fitness_goals": {"goal_weight": 50, "goal_body_fat": 18}}');


-- Display all MemberIDs for verification

SELECT MemberID FROM Members;


-- Insert Trainers

INSERT INTO Trainers (FirstName, LastName, Specialization) VALUES

('Dave', 'Roberts', 'Yoga'),

('Lucy', 'Taylor', 'Weightlifting'),

('Mark', 'Wilson', 'Cardio'),

('Chloe', 'Davis', 'Pilates'),

('Nina', 'Garcia', 'Crossfit');


-- Insert Rooms

INSERT INTO Rooms (RoomName, Capacity) VALUES

('Aerobics Room', 20),

('Yoga Studio', 15),

('Weight Room', 30),

('Cardio Zone', 25),

('Pilates Studio', 10);
```

```
-- Insert Classes ensuring TrainerID and RoomID are available
INSERT INTO Classes (ClassName, TrainerID, RoomID, Schedule, Status) VALUES
('Morning Yoga', 1, 1, '2024-05-01 08:00:00', 'Open'),
('Evening Weightlifting', 2, 2, '2024-05-01 19:00:00', 'Open'),
('Cardio Blast', 3, 3, '2024-05-02 09:00:00', 'Open'),
('Pilates Beginner', 4, 4, '2024-05-02 10:30:00', 'Open'),
('Crossfit Challenge', 5, 5, '2024-05-02 18:00:00', 'Open');

-- Insert Personal Training Sessions ensuring MemberID and TrainerID are available
INSERT INTO PersonalTrainingSessions (MemberID, TrainerID, Date, Time, Status) VALUES
(1, 1, '2024-05-01', '10:00', 'Scheduled'),
(2, 2, '2024-05-02', '11:00', 'Scheduled'),
(3, 3, '2024-05-03', '12:00', 'Scheduled'),
(4, 4, '2024-05-04', '13:00', 'Scheduled'),
(5, 5, '2024-05-05', '14:00', 'Scheduled');

-- Insert Equipment
INSERT INTO Equipment (EquipmentName, MaintenanceSchedule, LastMaintenanceDate,
EquipmentStatus) VALUES
('Treadmill 1', '2024-12-01', '2024-04-01', 'Available'),
('Elliptical 2', '2024-12-02', '2024-04-02', 'Maintenance'),
('Rowing Machine 3', '2024-12-03', '2024-04-03', 'Available'),
('Exercise Bike 4', '2024-12-04', '2024-04-04', 'Available'),
('Stair Climber 5', '2024-12-05', '2024-04-05', 'Maintenance');

-- Insert Member Classes ensuring MemberID and ClassID are available
INSERT INTO MemberClasses (MemberID, ClassID) VALUES
(1, 1),
```

(2, 2),

(3, 3),

(4, 4),

(5, 5);

-- Insert Billing ensuring MemberID is available

INSERT INTO Billing (MemberID, Amount, Description, BillDate, Paid) VALUES

(1, 50.00, 'Monthly Membership Fee', '2024-05-01', FALSE),

(2, 150.00, 'Personal Training Package', '2024-05-01', TRUE),

(3, 100.00, 'Annual Gym Access', '2024-05-01', FALSE),

(4, 75.00, 'Six-month Yoga Pass', '2024-05-01', TRUE),

(5, 80.00, 'Pilates Class Fees', '2024-05-01', FALSE);

COMMIT;

2.5 Implementation

My application is designed as a command-line interface (CLI) management system for a Health and Fitness Club, using Python as the programming language and PostgreSQL for database management. It employs the psycopg2 library to establish connections to the PostgreSQL database, which stores and manages data related to club members, trainers, and administrative staff. This setup allows for executing SQL commands directly from Python, facilitating operations such as user registration, profile updates, scheduling, and the management of room bookings and equipment maintenance.

2.7 Github Repo

https://github.com/Pohi11/COMP3005-Project-2