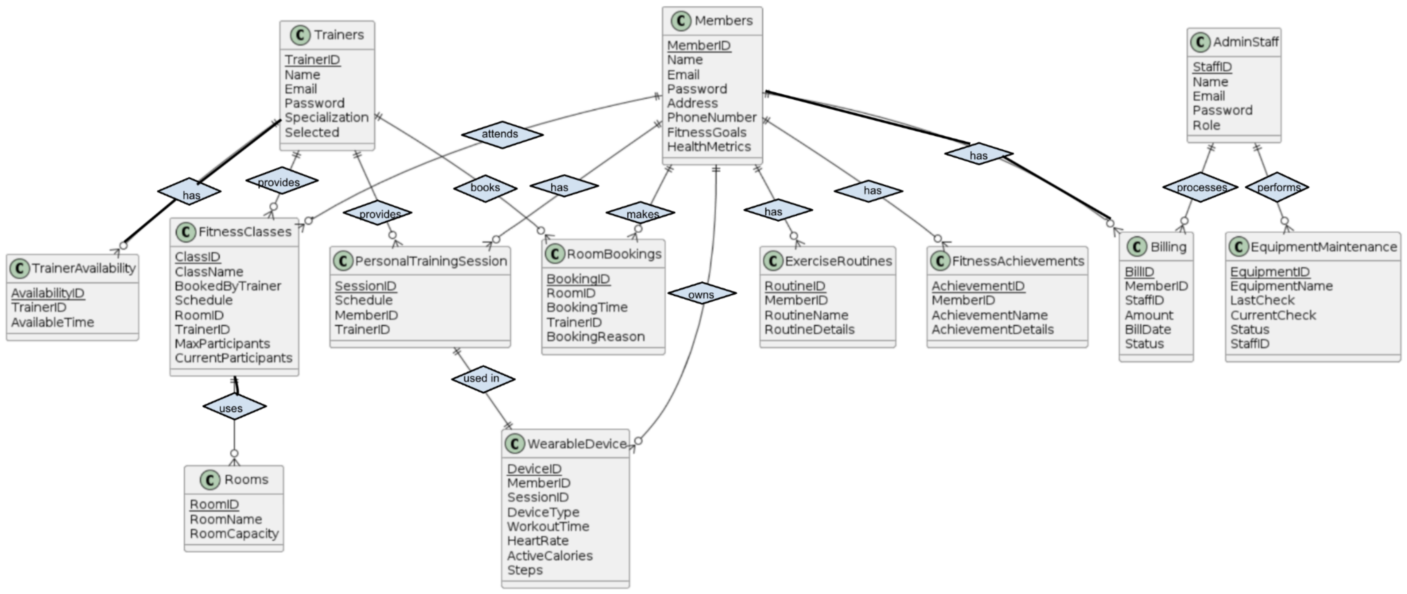
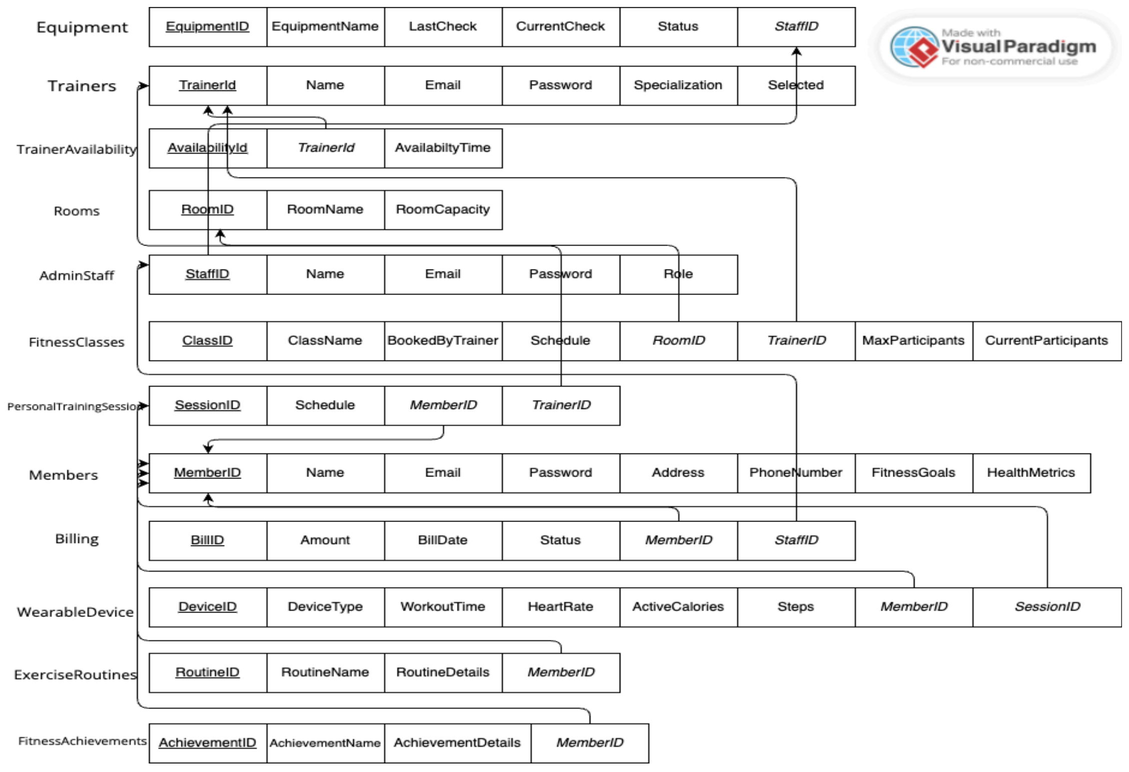
COMP 3005 - Final Project Report

Conceptual Design

This is our Entity Relation Diagram (ERD):



Reduction to Relation Schemas



Equipment	EquipmentID	EquipmentName	LastCheck	CurrentCheck	Status	*StaffID*		
Trainers	TrainerId	Name	Email	Password	Specialization	Selected		
TrainerAvailability	AvailabilityId	*TrainerId*	AvailabiltyTime					
Rooms	RoomID	RoomName	RoomCapacity					
AdminStaff	StaffID	Name	Email	Password	Role			
FitnessClasses	ClassID	ClassName	BookedByTrainer	Schedule	*RoomID*	*TrainerID*	MaxParticipants	CurrentParticipants
PersonalTrainingSession	SessionID	Schedule	*MemberID*	*TrainerID*				
Members	MemberID	Name	Email	Password	Address	PhoneNumber	FitnessGoals	HealthMetrics
Billing	BillID	Amount	BillDate	Status	*MemberID*	*StaffID*		
WearableDevice	DeviceID	DeviceType	WorkoutTime	HeartRate	ActiveCalories	Steps	*MemberID*	*SessionID*
ExerciseRoutines	RoutineID	RoutineName	RoutineDetails	*MemberID*				
FitnessAchievements	AchievementID	AchievementName	AchievementDetails	*MemberID*				

Mapping of ERD to Schema:

Requirement	Assumption	Representation in ER Model	Representation in Database Schema
Members must have a unique ID and personal details.	Each member is uniquely identified by a serially assigned MemberID.	Members Entity: Attributes include MemberID (PK), Name, Email, Password, Address, PhoneNumber, FitnessGoals, HealthMetrics.	Members Table: Attributes include MemberID (PRIMARY KEY), Name, Email (UNIQUE), Password, Address, PhoneNumber, FitnessGoals, HealthMetrics.
Trainers are uniquely identified and have specializations.	Each trainer has a unique TrainerID and fields for credentials and availability.	Trainers Entity: Attributes include TrainerID (PK), Name, Email, Password, Specialization, Selected	Trainers Table: Attributes include TrainerID (PRIMARY KEY), Name, Email (UNIQUE), Password, Specialization, Selected (BOOLEAN).
TrainerAvailability indicates available times for trainers.	Each availability entry is unique and linked to a trainer.	TrainerAvailability Entity: Attributes include AvailabilityID (PK), TrainerID, AvailableTime.	TrainerAvailability Table: Attributes include AvailabilityId (PRIMARY KEY), TrainerId (FOREIGN KEY), AvailableTime.
Fitness classes occur in specific rooms at scheduled times.	Each class has a unique ClassID and is associated with a room and a trainer.	FitnessClasses Entity: Attributes include ClassID (PK), ClassName, BookedByTrainer, Schedule, RoomID, TrainerID, MaxParticipants, CurrentParticipants	FitnessClasses Table: Attributes include ClassID (PRIMARY KEY), ClassName, BookedByTrainer, Schedule, RoomID (FOREIGN KEY), TrainerID (FOREIGN KEY), MaxParticipants, CurrentParticipants.
PersonalTrainingSession connects members with trainers.	Sessions are identified by SessionID and scheduled for	PersonalTrainingSession Entity: Attributes include SessionID (PK), Schedule,	PersonalTrainingSession Table: Attributes include SessionID (PRIMARY KEY),

	members with specific trainers.	MemberID, TrainerID.	Schedule, MemberID (FOREIGN KEY), TrainerID (FOREIGN KEY).
RoomBookings are made by trainers for sessions.	Bookings have a BookingID and are associated with a room and a trainer for a reason at a time.	RoomBookings Entity: Attributes include BookingID (PK), RoomID, BookingTime, TrainerID, BookingReason.	RoomBookings Table: Attributes include BookingID (PRIMARY KEY), RoomID (FOREIGN KEY), BookingTime, TrainerID (FOREIGN KEY), BookingReason.
AdminStaff processes billing and maintains equipment.	Admin staff identified by StaffID handle billing and equipment maintenance tasks.	AdminStaff Entity: Attributes include StaffID (PK), Name, Email, Password, Role.	AdminStaff Table: Attributes include StaffID (PRIMARY KEY), Name, Email (UNIQUE), Password, Role.
Billing records are associated with members and admin staff.	Bills are uniquely identified and linked to both members and admin staff.	Billing Entity: Attributes include BillID (PK), Amount, BillDate, Status, MemberID, StaffID.	Billing Table: Attributes include BillID (PRIMARY KEY), MemberID (FOREIGN KEY), StaffID (FOREIGN KEY), Amount, BillDate, Status.
WearableDevice records member workout data.	Devices have a unique DeviceID and log workout details for members.	WearableDevice Entity: Attributes include DeviceID (PK), MemberID, DeviceType, WorkoutTime, HeartRate, ActiveCalories, Steps.	WearableDevice Table: Attributes include DeviceID (PRIMARY KEY), MemberID (FOREIGN KEY), SessionID (FOREIGN KEY), DeviceType, WorkoutTime, HeartRate, ActiveCalories, Steps.
Members perform various ExerciseRoutines.	Routines have a unique RoutineID and are linked to the members who perform them.	ExerciseRoutines Entity: Attributes include RoutineID (PK), MemberID, RoutineName,	ExerciseRoutines Table: Attributes include RoutineID (PRIMARY KEY), MemberID

		RoutineDetails.	(FOREIGN KEY), RoutineName, RoutineDetails.
FitnessAchievements mark member milestones.	Achievements are uniquely identified by AchievementID and tied to members.	FitnessAchievements Entity: Attributes include AchievementID (PK), MemberID, AchievementName, AchievementDetails.	FitnessAchievements Table: Attributes include AchievementID (PRIMARY KEY), MemberID (FOREIGN KEY), AchievementName, AchievementDetails.

DDL (Data Definition Language

ddl.sql:

```
CREATE TABLE Members (
    MemberID SERIAL PRIMARY KEY,
    Name VARCHAR(255),
    Email VARCHAR(255) UNIQUE,
    Password VARCHAR(255),
    Address TEXT,
    PhoneNumber VARCHAR(20),
    FitnessGoals TEXT,
    HealthMetrics TEXT
);

CREATE TABLE Trainers (
    TrainerId SERIAL PRIMARY KEY,
    Name VARCHAR(255),
    Email VARCHAR(255) UNIQUE,
    Password VARCHAR(255),
    Specialization VARCHAR(255),
    Selected BOOLEAN
);

CREATE TABLE TrainerAvailability (
    AvailabilityId SERIAL PRIMARY KEY,
    TrainerId INT,
    AvailableTime TIMESTAMP,
    FOREIGN KEY (TrainerId) REFERENCES Trainers(TrainerId)
);
```

```
CREATE TABLE Rooms (
   RoomID SERIAL PRIMARY KEY,
   RoomName VARCHAR(255),
   RoomCapacity INT
);

CREATE TABLE AdminStaff (
   StaffID SERIAL PRIMARY KEY,
   Name VARCHAR(255),
   Email VARCHAR(255) UNIQUE,
   Password VARCHAR(255),
   Role VARCHAR(255)
);

CREATE TABLE FitnessClasses (
   ClassID SERIAL PRIMARY KEY,
   ClassName VARCHAR(255),
         BookedByTrainer BOOLEAN DEFAULT FALSE,
   Schedule TIMESTAMP,
   RoomID INT,
   TrainerID INT,
   MaxParticipants INT,
   CurrentParticipants INT,
         FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID),
   FOREIGN KEY (TrainerID) REFERENCES Trainers(TrainerId)
);

CREATE TABLE PersonalTrainingSession (
   SessionID SERIAL PRIMARY KEY,
   Schedule TIMESTAMP,
   MemberID INT,
   TrainerID INT,
   FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
   FOREIGN KEY (TrainerID) REFERENCES Trainers(TrainerId)
);

CREATE TABLE RoomBookings (
   BookingID SERIAL PRIMARY KEY,
   RoomID INT,
   BookingTime TIMESTAMP,
   TrainerID INT,
   BookingReason VARCHAR(255),
         FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID),
   FOREIGN KEY (TrainerID) REFERENCES Trainers(TrainerId)
```

```
);


CREATE TABLE EquipmentMaintenance (
   EquipmentID SERIAL PRIMARY KEY,
   EquipmentName VARCHAR(255),
   LastCheck DATE,
   CurrentCheck DATE,
   Status VARCHAR(100),
   StaffID INT,
   FOREIGN KEY (StaffID) REFERENCES AdminStaff(StaffID)
);

CREATE TABLE Billing (
   BillID SERIAL PRIMARY KEY,
   MemberID INT,
   StaffID INT,
   Amount DECIMAL(10,2),
   BillDate DATE,
   Status VARCHAR(100),
   FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
   FOREIGN KEY (StaffID) REFERENCES AdminStaff(StaffID)
);


CREATE TABLE WearableDevice (
   DeviceID SERIAL PRIMARY KEY,
   MemberID INT,
   SessionID INT,
   DeviceType VARCHAR(100),
   WorkoutTime TIMESTAMP,
   HeartRate INT,
   ActiveCalories INT,
   Steps INT,
   FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
   FOREIGN KEY (SessionID) REFERENCES PersonalTrainingSession(SessionID)
);

CREATE TABLE ExerciseRoutines (
   RoutineID SERIAL PRIMARY KEY,
   MemberID INT,
   RoutineName VARCHAR(255),
   RoutineDetails TEXT,
   FOREIGN KEY (MemberID) REFERENCES Members(MemberID)
```

```
);

CREATE TABLE FitnessAchievements (
   AchievementID SERIAL PRIMARY KEY,
   MemberID INT,
   AchievementName VARCHAR(255),
   AchievementDetails TEXT,
   FOREIGN KEY (MemberID) REFERENCES Members(MemberID)
);

CREATE OR REPLACE FUNCTION check_room_booking() RETURNS TRIGGER AS $$
BEGIN
   IF EXISTS (SELECT 1 FROM RoomBookings WHERE RoomID = NEW.RoomID AND
BookingTime = NEW.BookingTime) THEN
      RAISE EXCEPTION 'Room is already booked at the specified time.';
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER room_booking_trigger
BEFORE INSERT ON RoomBookings
FOR EACH ROW EXECUTE FUNCTION check_room_booking();

CREATE OR REPLACE FUNCTION update_trainer_id()
RETURNS TRIGGER AS $$
BEGIN
   IF NEW.BookedByTrainer THEN
      NEW.TrainerID = (SELECT TrainerID FROM TrainerAvailability WHERE StartTime <=
NEW.Schedule AND EndTime > NEW.Schedule LIMIT 1);
   ELSE
      NEW.TrainerID = NULL;
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_trainer_id_trigger
BEFORE INSERT OR UPDATE ON FitnessClasses
FOR EACH ROW EXECUTE FUNCTION update_trainer_id();

CREATE INDEX idx_email ON Members(Email);
CREATE INDEX idx_trainer_specialization ON Trainers(Specialization);
CREATE INDEX idx_scheduling ON FitnessClasses(Schedule);
```

DML (Data Manipulation Language)

dml.sql:

```
INSERT INTO Rooms (RoomID, RoomName, RoomCapacity) VALUES
(1, 'Room A', 10),
(2, 'Room B', 15),
(3, 'Room C', 20),
(4, 'Room D', 12);

INSERT INTO Members (Name, Email, Password, Address, PhoneNumber, FitnessGoals,
HealthMetrics) VALUES
('John Doe', 'john.doe@example.com', 'password123', '123 Main St, Anytown, USA',
'1234567890', 'Lose weight', 'Normal'),
('Jane Smith', 'jane.smith@example.com', 'password456', '456 Elm St, Anytown, USA',
'0987654321', 'Gain muscle', 'Normal'),
('Robert Brown', 'robert.brown@example.com', 'password789', '789 Pine St, Anytown, USA',
'1122334455', 'Improve stamina', 'Normal'),
('Emily Davis', 'emily.davis@example.com', 'password012', '012 Oak St, Anytown, USA',
'5566778899', 'Increase flexibility', 'Normal');

INSERT INTO Trainers (Name, Email, Password, Specialization, Selected) VALUES
('Tom Johnson', 'tom.johnson@example.com', 'password123', 'Weightlifting', FALSE),
('Sally Peterson', 'sally.peterson@example.com', 'password456', 'Cardio', FALSE),
('Bill Thompson', 'bill.thompson@example.com', 'password789', 'Yoga', FALSE),
('Linda Anderson', 'linda.anderson@example.com', 'password012', 'Pilates', FALSE);

INSERT INTO TrainerAvailability (TrainerId, AvailableTime) VALUES
(1, '2024-04-11 08:00:00'),
(1, '2024-04-12 14:00:00'),
(2, '2024-04-11 09:00:00'),
(3, '2024-04-12 10:00:00'),
(4, '2024-04-11 15:00:00');

INSERT INTO AdminStaff (Name, Email, Password, Role) VALUES
('George Wilson', 'george.wilson@example.com', 'password123', 'Manager'),
('Susan Miller', 'susan.miller@example.com', 'password456', 'Receptionist'),
('Fred Taylor', 'fred.taylor@example.com', 'password789', 'Maintenance'),
('Karen Davis', 'karen.davis@example.com', 'password012', 'Billing');

INSERT INTO FitnessClasses (ClassName, Schedule, RoomID, TrainerID, MaxParticipants,
CurrentParticipants) VALUES
('Yoga', '2024-04-11 10:00:00', 1, NULL, 20, 10),
('Zumba', '2024-04-11 15:00:00', 2, NULL, 20, 15),
('Weightlifting', '2024-04-10 09:00:00', 3, 1, 20, 12),
('Pilates', '2024-04-11 16:00:00', 4, 4, 20, 14);

INSERT INTO PersonalTrainingSession (Schedule, MemberID, TrainerID) VALUES
('2024-04-12 11:00:00', 1, 1),
('2024-04-12 16:00:00', 2, 2),
('2024-04-12 10:00:00', 3, 3),
('2024-04-12 15:00:00', 4, 4);

INSERT INTO RoomBookings (RoomID, BookingTime, TrainerID, BookingReason) VALUES
(1, '2024-04-11 12:00:00', 1, 'Weightlifting Session'),
(2, '2024-04-11 17:00:00', 2, 'Cardio Training'),
(3, '2024-04-12 11:00:00', 3, 'Yoga Class'),
(4, '2024-04-12 16:00:00', 4, 'Pilates Class');


INSERT INTO EquipmentMaintenance (EquipmentName, LastCheck, CurrentCheck, Status,
StaffID) VALUES
('Treadmill', '2022-11-01', '2022-12-01', 'Good', 3),
('Dumbbells', '2022-11-01', '2022-12-01', 'Good', 3),
('Yoga Mats', '2022-11-01', '2022-12-01', 'Good', 3),
('Exercise Balls', '2022-11-01', '2022-12-01', 'Good', 3);

INSERT INTO Billing (MemberID, StaffID, Amount, BillDate, Status) VALUES
(1, 4, 100.00, '2022-12-01', 'Paid'),
(2, 4, 150.00, '2022-12-01', 'Unpaid'),
(3, 4, 120.00, '2022-12-01', 'Paid'),
(4, 4, 130.00, '2022-12-01', 'Unpaid');

INSERT INTO WearableDevice (MemberID, SessionID, DeviceType, WorkoutTime, HeartRate,
ActiveCalories, Steps) VALUES
(1, 1, 'Smart Watch', '2022-12-05 13:00:00', 80, 200, 5000),
(2, 2, 'Fitness Tracker', '2022-12-05 18:00:00', 90, 300, 7000),
(3, 3, 'Smart Watch', '2022-12-05 12:00:00', 85, 250, 5500),
(4, 4, 'Fitness Tracker', '2022-12-05 17:00:00', 95, 350, 7500);

INSERT INTO ExerciseRoutines (MemberID, RoutineName, RoutineDetails) VALUES
(1, 'Morning Cardio', '30 minutes of running'),
(2, 'Evening Yoga', '20 minutes of yoga'),
(3, 'Afternoon Weights', '45 minutes of weightlifting'),
(4, 'Night Pilates', '30 minutes of pilates');
```

```
INSERT INTO FitnessAchievements (MemberID, AchievementName, AchievementDetails)
VALUES
(1, '5k Run', 'Completed a 5k run'),
(2, '10k Steps', 'Walked 10k steps in a day'),
(3, '1 Hour Yoga', 'Did 1 hour of yoga'),
(4, 'Lifted 50kg', 'Lifted 50kg weights');
```

Implementation

We built a Command-Line Interface (CLI) application for the Health and Fitness Club Management System in python using a psycopg, a PostgreSQL adapter . The application facilitates various required and additional functionalities such as member registration, login processes for members, trainers, and admin staff, and scheduling activities. Upon initiating the application, users are greeted with a main menu where they can choose to register as a new member or log in with existing credentials depending on their role (member, trainer, or admin).

For trainers, the application provides a dashboard where they can manage their availability for training sessions, view member profiles, and log out. Functions such as `set_available_time` enable trainers to add, update, or delete their available times, thus dynamically updating their schedules. Similarly, members have a dashboard where they can book personal training sessions, enroll in fitness classes, sync wearable device data, view their profiles, manage exercise routines and achievements, and update personal information. These actions involve database transactions like inserting, updating, and deleting records, which are carefully handled with try-except blocks to manage database errors effectively.

The admin section of the CLI offers administrative tools for room booking management, equipment maintenance monitoring, updating class schedules, and handling billing and payment processing. These functionalities demonstrate the use of more complex SQL queries and transaction handling, ensuring that the administrative staff can efficiently manage the backend operations of the fitness club.

As previously mentioned, the application uses the `psycopg` library to establish a secure connection to the PostgreSQL database using credentials such as database name, user, host, and password. Error handling is a crucial part of the application, ensuring that any operational or connection issues are caught and appropriately reported to the user without causing the application to crash.

Bonus Features (Optional)

We have implemented triggers and indexes in the database to help optimize operations in the Health and Fitness Club Management System. The triggers, such as check_room_booking and update_trainer_id, are designed to prevent booking conflicts and automatically manage trainer assignments based on availability. The check_room_booking trigger checks for existing bookings at the same time in the same room, preventing double bookings by raising an exception if a conflict is detected. The update_trainer_id trigger assigns trainers to fitness classes based on their availability, enhancing the automation of the scheduling process. Additionally, indexes like idx_email on the Members table, idx_trainer_specialization on the Trainers table, and idx_scheduling on the FitnessClasses table are used to speed up query processing.

Additionally, we have worked on integrating wearable devices. By allowing members to insert their workout data directly from their devices into the database, we can facilitate regular health tracking. This includes the tracking of heart rate, calories burned, and steps can be used to update fitness routines and track progress more accurately thus measuring user fitness metrics.