VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Database Systems (CO2014)

CC01

Assignment 1

Gym Membership Management System

Instructor: Do Thanh Thai, M.Sc.
Student: Truong Tuan Kiet - 2252411
Vu Minh Quan - 2212828
Le Trung Kien - 2252394

HO CHI MINH CITY, November 28, 2024

Contents

1 Introduction

This report presents the design and implementation of a Gym Membership Management System that efficiently manages gym members, their memberships, trainers, gym branches, and payments. The system is designed to provide a streamlined interface for gym administrators to handle memberships and payments. It is built using a relational database model and implemented with SQL constraints to ensure data integrity and security.

2 Data Requirements

2.1 Requirement description

The gym management system must maintain detailed records of all individuals involved in its operations. Each person is uniquely identified by their Social Security Number (SSN) and includes additional personal information such as first and last name, date of birth, and age, which is derived from their date of birth. A person can be a gym member, a trainer, or both.

Each gym member is assigned a unique Member ID along with their join date. Members have the option to register with a personal trainer, and a trainer may simultaneously train multiple members.

Trainers are identified by a unique Trainer ID, and their profile includes details like specialization and employment type, indicating whether they work full-time or part-time. Trainers may lead multiple classes and can also be assigned to individual members as personal trainers.

Members can enroll in various memberships, each represented as a program with a unique Program ID. Each program includes a name, price, and current status. There are two types of memberships: Basic Membership, which grants access to all gym equipment and is associated with a specific duration, and Class Memberships, which allow members to enroll in specific classes. Classes are defined by the number of periods, a weekly schedule specifying the day and time, and the duration of each session. Each class has an assigned trainer, and one trainer may lead multiple classes.

Members can make multiple payments for memberships. The system should track each membership's start and end date, payment details including method, bill number, purchase date, and payment amount. Whenever a member registers for a new membership, this information is updated accordingly.

The gym operates multiple branches, each uniquely identified by a Gym Branch ID and address. Each branch is divided into areas, distinguished by area name and floor location. Classes are held in these areas, and each area can host multiple classes.

2.2 Entities and Relationships

This section covers the key entities, relationships, and constraints used in the Gym Membership Management System.

- **Person**: Represents general information about a person, serve as a superclass for individuals associated with the gym

- **Member**: Represents customers with gym memberships.
- **Trainer**: Represents fitness instructors.
- **Membership**: Tracks membership type and details for each member.
- **GymStore**: Represents gym stores where members can purchase products.
- **GymBranch**: Represents different gym branches.
- **Payment**: Tracks payments made by members.
- **Basic**: A subclass of Membership, representing a basic membership type.
- **Class**: A subclass of Membership, representing member that can join classes, includes additional details specific to classes.
- **Area**: A weak entity which are specific sections within a branch. Depends on entity GymBranch.

The relationships include:

- **HAS**: Connects `PERSON` to `MEMBER` and `TRAINER`, indicating that a person can be either a member or a trainer.
- **REGISTER**: Connects `MEMBER` to `MEMBERSHIP`, allowing members to register for memberships. Also keep track of the payment between members and memberships
- **TEACH**: Connects `TRAINER` to `CLASS`, indicating that trainers teach specific classes.
- **WORK_AT**: Connects `TRAINER` to `GYMBRANCH`, showing that trainers work at a specific gym branch.
- **INCLUDE**: Connects `GYMBRANCH` to `AREA`, indicating that a gym branch includes different areas within it.
- **LOCATE_AT**: Connects `CLASS` to `AREA`, specifying where classes are located within a gym branch.
- **PURCHASE_FROM**: Connects `MEMBER` to `GYMSTORE` through the `BASIC` membership, indicating that members with a BASIC membership can make purchases from the gym store.

2.3 Key Attributes and Descriptions

Entity	Key Attribute	Description
PERSON	SSN	Social Security Number, unique identifier for a person
MEMBER	MemberID	Unique identifier for each gym member
TRAINER	TrainerID	Unique identifier for each trainer
MEMBERSHIP	ProgramID	Unique identifier for each membership program
	Number	Unique identifier for each payment transaction
GYMBRANCH	GymBranchID	Unique identifier for each gym branch
GYMSTORE	GymStoreID	Unique identifier for each gym store
BASIC	ProgramID	Unique identifier inherited from MEMBERSHIP
CLASS	ProgramID	Unique identifier inherited from MEMBERSHIP
AREA	(GymBranchID, Floor, Name)	Composite key identifying each unique area in a gym branch

Table 1: Key Attributes and Descriptions in the ER Diagram

3 ER Diagram and Mapping to Relational Database Schema

3.1 ER Diagram

Below is an illustration of the ER Diagram of the Gym Management System.

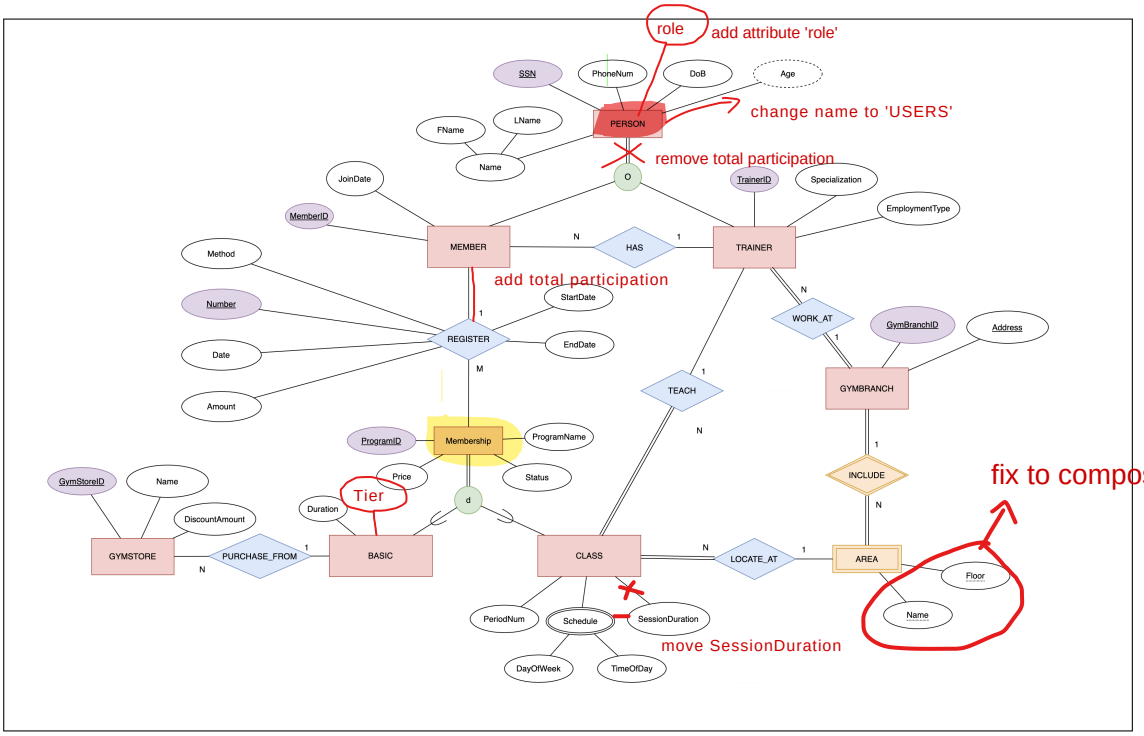

Figure 1: ER Diagram for Gym Management System

3.2 Mapping ER Diagram to Relational Database Schema

Below is the relational schema derived from the ER diagram, including primary keys, secondary keys, foreign keys, and constraints.

- **PERSON** (SSN, FName, LName, PhoneNum, DoB)
  - Primary key: SSN

+ Password
PERSON -> USER
secondary key: PhoneNum

- **MEMBER** (SSN, MemberID, JoinDate, TrainerID)
  - Primary key: MemberID
  - Secondary key (unique, not null): SSN
  - Foreign key: SSN references PERSON.SSN, TrainerID references TRAINER.TrainerID
- **TRAINER** (SSN, TrainerID, Specialization, EmploymentType, Workplace)
  - Primary key: TrainerID
  - Secondary key (unique, not null): SSN
  - Foreign key: SSN references PERSON.SSN, Workplace references GYMBRANCH.GymBranchID
  - Not null: Workplace
  - Range constraint: EmploymentType in 'parttime', 'fulltime'
  - Check: Total participation of GYMBRANCH (every trainer is associated with a gym branch)
- **REGISTER** (MemberID, ProgramID, Number, StartDate, EndDate, Method, Date, Amount)
  - Primary key: (MemberID, ProgramID)
  - Secondary key (unique, not null): Number
  - Foreign key: MemberID references MEMBER.MemberID, ProgramID references MEMBERSHIP.ProgramID

Method IN ('cash', 'credit')

- **MEMBERSHIP** (ProgramID, ProgramName, Price, Status)
  - Primary key: ProgramID

MEMBERSHIP -> PROGRAM

- **BASIC** (ProgramID, Duration)
  - Primary key: ProgramID
  - Foreign key: ProgramID references MEMBERSHIP.ProgramID
- **GYMSTORE** (GymstoreID, Name, DiscountAmount, ProgramID)
  - Primary key: GymstoreID
  - Foreign key: ProgramID references BASIC.ProgramID

CHECK DiscountAmount < 100

- **CLASS** (ProgramID, PeriodNum, SessionDuration, GymBranchID, Floor, Area, TrainerID)
  - Primary key: ProgramID
  - Foreign keys: ProgramID references MEMBERSHIP.ProgramID, GymBranchID references GYMBRANCH.GymBranchID, Floor references AREA.Floor, Area references AREA.Name, TrainerID references TRAINER.TrainerID

remove SessionDuration

not null TrainerID

CLASS_SCHED (ProgramID, DayOfWeek, TimeOfDay, SessionDuration)
- Primary key: ProgramID, DayOfWeek, TimeOfDay)
- Foreign keys: ProgramID references CLASS(ProgramID)
CHECK (DayOfWeek IN ('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'))

- **GYMBRANCH** (GymBranchID, Address)
  - Primary key: GymBranchID
  - Secondary key (unique, not null): Address
- **AREA** (GymBranchID, Floor, Name)
  - Primary key: (GymBranchID, Floor, Name)
  - Foreign key: GymBranchID references GYMBRANCH.GymBranchID
  - Check: Total participation of GYMBRANCH (every area is associated with a gym branch)

4 Database Implementation and Technology Preparation

4.1 Chosen DBMS and Installation

PostgreSQL was chosen for this project for its robustness and ease of setup. The installation was completed using the PostgreSQL installer, allowing efficient database management.

4.2 Defining user groups

```sql
CREATE ROLE admin_group;
CREATE ROLE read_only_group;

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO admin_group;
GRANT SELECT ON ALL TABLEs TO read_only_group;

-- Create some sample users
CREATE USER alice WITH PASSWORD 'password123';
CREATE USER bob WITH PASSWORD 'password456';
CREATE USER carol WITH PASSWORD 'password789';

-- Assign users to groups
GRANT admin_group TO alice;
GRANT read_only_group TO bob;
GRANT write_only_group TO carol;
```

Listing 1: User groups definition with PostgreSQL

We have two user groups: admin and read_only. Admins is given all privileges to view, edit, and manage the database. While the read-only group can make queries for data.

4.3 SQL Statements

```sql
-- Table for GYMBRANCH entity (create this first for dependency)
CREATE TABLE GYMBRANCH (
    GymBranchID SERIAL PRIMARY KEY,
    Address VARCHAR(255) UNIQUE NOT NULL
);

-- Table for PERSON entity
CREATE TABLE PERSON (
    SSN CHAR(9) PRIMARY KEY,
    FName VARCHAR(50) NOT NULL,
    LName VARCHAR(50) NOT NULL,
    PhoneNum VARCHAR(15),
```

```
    DoB DATE
);

-- Table for TRAINER entity
CREATE TABLE TRAINER (
    TrainerID SERIAL PRIMARY KEY,
    SSN CHAR(9) UNIQUE NOT NULL,
    Specialization VARCHAR(50),
    EmploymentType VARCHAR(20) CHECK(EmploymentType IN ('parttime','fulltime')),
    Workplace INT NOT NULL,
    FOREIGN KEY (SSN) REFERENCES PERSON(SSN),
    FOREIGN KEY (Workplace) REFERENCES GYMBRANCH(GymBranchID)
);

-- Table for MEMBER entity
CREATE TABLE MEMBER (
    MemberID SERIAL PRIMARY KEY,
    SSN CHAR(9) UNIQUE NOT NULL,
    JoinDate DATE NOT NULL,
    TrainerID INT,
    FOREIGN KEY (SSN) REFERENCES PERSON(SSN),
    FOREIGN KEY (TrainerID) REFERENCES TRAINER(TrainerID)
);

-- Table for AREA entity
CREATE TABLE AREA (
    GymBranchID INT,
    Floor INT,
    Name VARCHAR(50),
    PRIMARY KEY (GymBranchID, Floor, Name),
    FOREIGN KEY (GymBranchID) REFERENCES GYMBRANCH(GymBranchID) ON DELETE CASCADE
);

-- Table for MEMBERSHIP entity
CREATE TABLE MEMBERSHIP (
    ProgramID SERIAL PRIMARY KEY,
    ProgramName VARCHAR(100) NOT NULL,
    Price NUMERIC(10, 2) NOT NULL,
    Status VARCHAR(20) NOT NULL
);

-- Table for BASIC entity, extending MEMBERSHIP with duration info
CREATE TABLE BASIC (
    ProgramID INT PRIMARY KEY,
    Duration INTERVAL NOT NULL,
    FOREIGN KEY (ProgramID) REFERENCES MEMBERSHIP(ProgramID)
);

-- Table for GYMSTORE entity
CREATE TABLE GYMSTORE (
    GymstoreID SERIAL PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    DiscountAmount NUMERIC(5, 2),
    ProgramID INT,
    FOREIGN KEY (ProgramID) REFERENCES BASIC(ProgramID)
);

-- Table for CLASS entity
CREATE TABLE CLASS (
    ProgramID INT,
    PeriodNum INT NOT NULL,
    SessionDuration INTERVAL NOT NULL,
```

```
    GymBranchID INT,
    Floor INT,
    Area VARCHAR(50),
    TrainerID INT,
    PRIMARY KEY (ProgramID),
    FOREIGN KEY (ProgramID) REFERENCES MEMBERSHIP(ProgramID),
    FOREIGN KEY (GymBranchID,Floor, Area) REFERENCES AREA(GymBranchID,Floor, Name)
    ,
    FOREIGN KEY (TrainerID) REFERENCES TRAINER(TrainerID)
);

-- Table for REGISTER entity
CREATE TABLE REGISTER (
    MemberID INT,
    ProgramID INT,
    Number INT UNIQUE NOT NULL,
    StartDate DATE NOT NULL,
    EndDate DATE NOT NULL,
    PRIMARY KEY (MemberID, ProgramID),
    FOREIGN KEY (MemberID) REFERENCES MEMBER(MemberID),
    FOREIGN KEY (ProgramID) REFERENCES MEMBERSHIP(ProgramID),
    Method VARCHAR(50) NOT NULL,
    Date DATE NOT NULL,
    Amount NUMERIC(10, 2) NOT NULL
);
```

Listing 2: PostgreSQL's DDL

4.4 Technology Preparation

The front-end of the Gym Membership Management System will be developed using ***React.js***, chosen for its efficiency in creating dynamic, interactive user interfaces. The back-end will be implemented with ***Node.js***, allowing scalable and high-performance server-side processing. This setup ensures seamless interaction with the PostgreSQL database through RESTful API endpoints.

5 Conclusion and Reflections

This project successfully developed a Gym Membership Management System, achieving a well-structured relational database that meets the specified requirements for efficient management of gym members, trainers, memberships, and payments. The implemented PostgreSQL database schema, along with defined relationships and constraints, has ensured data integrity and consistency across all entities. Additionally, by employing user role definitions, the system provides secure, role-based access to sensitive data, with administrators, trainers, and members having permissions tailored to their roles.

Future improvements could include enhanced security measures, such as multi-factor authentication, and extended features, such as biometric access, to further increase system reliability and ease of use. Overall, the Gym Membership Management System demonstrates a scalable, practical solution for gym operations, fulfilling its goals of efficient and secure membership management.

References

[1] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 7th ed., Pearson, 2016.