COMP 3005: Database Management Systems

Health and Fitness Club Management System

Prepared by:

Jerick Liu

101225819

April 12, 2024

Contents

2 Problem Statement

This is verbatim with the problem statement given in the assignment specifications. Design and implement an application for a Health and Fitness Club Management System. This system will serve as a comprehensive platform catering to the diverse needs of club members, trainers, and administrative staff. Members should be able to register and manage their profiles, establish personal fitness goals (you can determine suitable fitness goals such as weight and time, and members will set the values), and input health metrics. They should have access to a personalized dashboard that tracks exercise routines, fitness achievements, and health statistics. Members can schedule, reschedule, or cancel personal training sessions with certified trainers. Additionally, they should be able to register for group fitness classes. Trainers should have the ability to manage their schedules and view member profiles. Administrative Staff should be equipped with features to manage room bookings, monitor fitness equipment maintenance, update class schedules, oversee billing, and process payments for membership fees, personal training sessions, and other services.

2.1 Conceptual Design

The conceptual design of the Health and Fitness Club Management System database is crafted with a focus on the various users: members, trainers, and administrative staff. Each user interacts with the system uniquely, necessitating distinct data organization to serve their needs efficiently.

The ER diagram presents a visual representation of the database's entities, their attributes, and the relationships among them:

- Users: The base entity representing all system users, including their username, password, email, and role. This acts as a parent entity for Members, Admins, and Trainers through an **"Is a"** relationship, illustrating inheritance where Admins and Trainers are specialized types of Users.

- Members: They are users with a focus on fitness activities. Members have personal attributes (`first_name, last_name`), fitness-related data (`fitness_goals, exercise, fitness_achievements`), and financial information (`balance`). The **"has"** relationship between Members and Health implies that member health details are an integral part of a member's profile.

- Health: An entity associated with Members that stores health metrics such as `blood_pressure`, `weight`, and `height`. The double line to Members signifies total participation, indicating that every Health record is associated with exactly one Member.

- Trainers: Specialized users who facilitate fitness activities. Each trainer is associated with dates of availability (`Dates`) and can host Sessions and Classes.

- Dates: Contains available dates and times for trainers, establishing when trainers can conduct Sessions and Classes. This signifies total participation, indicating that every Date record is associated with exactly one Trainer.

- Admins: Specialized users who manage the administrative aspects of the club.

- Rooms: Physical locations within the club, with attributes describing the room type and maintenance status. They are utilized for Sessions and Classes.

- Sessions: One-on-one training appointments between Members and Trainers that occur in Rooms. They are scheduled at specific times (`session_date`) and can include session details.

- Classes: Group fitness activities that Members can enroll in through the `ClassQueue`. Each class is defined by a name, date, description, and is linked to a trainer and room.

- ClassQueue: A junction table managing the many-to-many relationship between Members and Classes, indicating which members are enrolled in which classes.

Assumptions made:

- Each user can be a member, trainer, or admin, but roles do not overlap in the current design.
- Trainers have dedicated availability slots that are exclusively used for one session or class at a time.
- Members can have multiple health records over time, but each health record is uniquely tied to one member. Same with Trainers and Dates.
- Rooms can be scheduled for multiple sessions and classes, but not simultaneously.
- Class enrollment is managed through an honour attendance system.
- Financial transactions and member balances are managed within the system, without the use of an integrated payment processing functionality.

The design ensures that the provided details are sufficient to uniquely identify and facilitate the interactions required by the system's users. It balances normalization to avoid data redundancy with the practical need for efficient data retrieval for club operations.

2.2 Reduction to Relational Schemas

The process of converting the ER diagram to a relational schema involves creating tables that reflect the entities and their relationships, ensuring that data integrity and normalization standards are upheld. This reduction involves several systematic steps:

Representation of Entities:

Each entity in the ER diagram has been transformed into a corresponding table in the relational schema. The attributes of the entities have become columns within these tables. For instance, the Members entity has translated into the Members table with columns such as `first_name`, `last_name`, and `fitness_goals`.

Primary Keys:

The unique identifier for each entity, the primary key, is represented as a unique column in

each table, and denoted by the `SERIAL` datatype, which automatically increments with each new record. For example, the `user_id` in the Users table serves as the primary key, uniquely identifying each user.

Foreign Keys and Relationships:

Foreign keys are used to enforce referential integrity between tables. In the schema, they establish the relationships as defined in the ER diagram, such as the one-to-many relationship where a single Trainer can be linked to multiple Sessions. Foreign keys are indicated by columns that reference the primary key of another table. The `member_id` in the Health table, for instance, references the `member_id` in the Members table, creating a direct link between a member's profile and their health metrics.

Handling Many-to-Many Relationships:

Many-to-many relationships are managed through join tables. The `ClassQueue` table, for instance, implements the many-to-many relationship between Members and Classes. It contains two foreign keys that together form a composite primary key to uniquely identify each enrollment.

Integrity Constraints:

The relational schema reflects the integrity constraints implied by the ER diagram. The `NOT NULL` constraint ensures that certain fields, which are essential for each record, must be filled out. For example, a Session must have associated `member_id`, `trainer_id`, and `session_date` fields filled. The `UNIQUE` constraints ensure that values like username and email in the Users table are unique across the system.

Total Participation:

The double line in the ER diagram, which signifies total participation or mandatory relationships, is represented by `NOT NULL` foreign keys in the relational schema. This implies that a record in a child table cannot exist without a corresponding record in the parent table.

The reduction from the ER diagram to the relational schema has been executed with a clear understanding of the relationships, integrity constraints, and the need for normalization. The resulting schema provides a sturdy foundation for the Health and Fitness Club

Management System, enabling efficient data storage, retrieval, and maintenance, conducive to the system's robust and scalable performance.

2.3 DDL File

As discussed thoroughly before, the provided DDL file is comprehensive and sets the foundation with all the tables of entities for the Health and Fitness Club Management System's database. Upon execution in a PostgreSQL environment, the DDL script will lead to the creation of all necessary tables and establish the corresponding relationships through primary and foreign key constraints.

Constraints Overview:

In addition to primary and foreign keys, databases commonly enforce a variety of other constraints to ensure data integrity and adherence to business rules:

- `NOT NULL Constraints`: These are used to prevent null values in certain columns that must have a valid value.
- `UNIQUE Constraints`: Ensure all values in a column are unique across the dataset, which is particularly important for identifiers like usernames.
- `DEFAULT Values`: Provide a default value for a column when no value is specified. This is used to initialize the member balance.

2.4 DML File

The DML file provided is crucial for populating the Health and Fitness Club Management System with an initial set of data, which is essential for testing and demonstration purposes. By executing this file in PostgreSQL, the relevant tables will be filled with sample data, reflecting realistic scenarios that the system might encounter in a live environment.

Contents of the DML File:

The DML file contains `INSERT` statements for each table in the database, reflecting a diverse

range of data to cover various test cases:

- Users: Sample user accounts, including members, trainers, and an administrator, each with unique usernames and roles.
- Members: Details for gym members, including names, fitness goals, exercise routines, achievements, and account balances.
- Trainers: Information about trainers, their names, and associated user accounts.
- Health: Health metrics for each member, such as blood pressure, weight, and height measurements.
- Dates: Availability dates for trainers, indicating when they can hold personal training sessions or classes.
- Rooms: Descriptions of the different spaces in the club, like the yoga studio and cardio room, along with their maintenance status.
- Classes: Group fitness classes offered by the club, including the class name, schedule, description, and the assigned trainer and room.
- ClassQueue: Sign-ups for classes, indicating which members have enrolled in each class.
- Sessions: Scheduled personal training sessions, detailing the member, trainer, room, time, and session focus.

2.5 Implementation

Architecture Overview

The Health and Fitness Club Management System is implemented as a Python-based CLI application. It interacts with a PostgreSQL relational database, managing data persistence and complex queries integral to the application's functionality.

Application Architecture

- Database Layer: At the core is the PostgreSQL database, which holds the data within structured tables corresponding to the entities identified in the ER diagram. Relationships between these tables are enforced through foreign keys, maintaining the integrity of the data.

- Application Layer: The Python application serves as the middle layer, containing the business logic. It utilizes the `psycopg2` library to establish database connections and perform SQL operations.

- Presentation Layer: The CLI provides the user interface. Users interact with the system through text-based menus, prompts, and commands.

Feature Implementation

Each feature outlined in the project requirements is implemented as a function within the Python application. Here is a brief overview of some features:

- User Registration: Handled by `register_user()`, allowing members to create their profiles with unique usernames and passwords.

- User Login: Managed by the `login()` function, which authenticates users and directs them to the appropriate dashboard based on their role.

- Member Dashboard: Through `member_login()`, members can view and manage their profiles, fitness goals, and health metrics.

- Training Sessions Management: Members can schedule (`book_training()`), reschedule, and cancel (`cancel_training()`) their personal training sessions.

- Group Classes Registration: The function `event_registration()` allows members to sign up for classes.

- Trainer Dashboard: Accessed through `trainer_login()`, where trainers can set their availability (`set_trainer_availability()`) and view member profiles (`query_member()`).

- Administrative Functions: The `admin_login()` function leads to administrative tasks such as processing payments (`process_payments()`), managing room statuses (`manage_rooms()`), and updating class schedules.

Database Integration

The application makes extensive use of SQL operations such as `SELECT`, `INSERT`, `UPDATE`, and `DELETE` to interact with the database. These operations are performed within functions, wrapped in `try-except` blocks to handle any exceptions and rollback transactions if needed.

Error Handling and Transactions

The application includes error handling to manage database transaction integrity. If an error occurs during a database operation, the transaction is rolled back to ensure the database remains in a consistent state.

Startup and Initialization

Upon starting the application (`main()`), it initializes the database by running the DDL and DML SQL scripts to create the necessary schema and populate it with sample data.

The CLI-based Health and Fitness Club Management System is designed to be user-friendly, with a focus on robust database interaction and secure user operations. Each function correlates directly with the user stories and requirements, ensuring a comprehensive solution for managing health and fitness club operations.

2.6 Bonus Features

In addition to the core functionalities, several bonus features have been incorporated to enhance the user experience and system efficiency of the Health and Fitness Club Management System. These features add depth to the application and showcase advanced database concepts and integrations.

- Views for Reporting: Created database views to simplify complex queries, especially for reporting purposes. For example, a view combining member details with their latest health metrics streamlines generating health progress reports for trainers.

- Indexes for Query Optimization: Added indexes on frequently queried columns, such as the `username` in the Users table and the `session_date` in the Sessions table. This significantly speeds up the retrieval times for login processes and session management.

- Triggers for Data Validation: Utilized database triggers to ensure data integrity. For instance, a trigger to prevent insertion of a training session if it conflicts with another session already scheduled for the same room at the same time.

2.7 GitHub Repository

Please visit the following Github Repository to access the full project:

Repository: https://github.com/JerickLiu/COMP3005-Final-Project