Carleton University

School of Computer Science

Health and Fitness Club Management System Project Report

Due: Dec 10th, 2023

COMP 3005A

Professor: Abdelghny Orogat

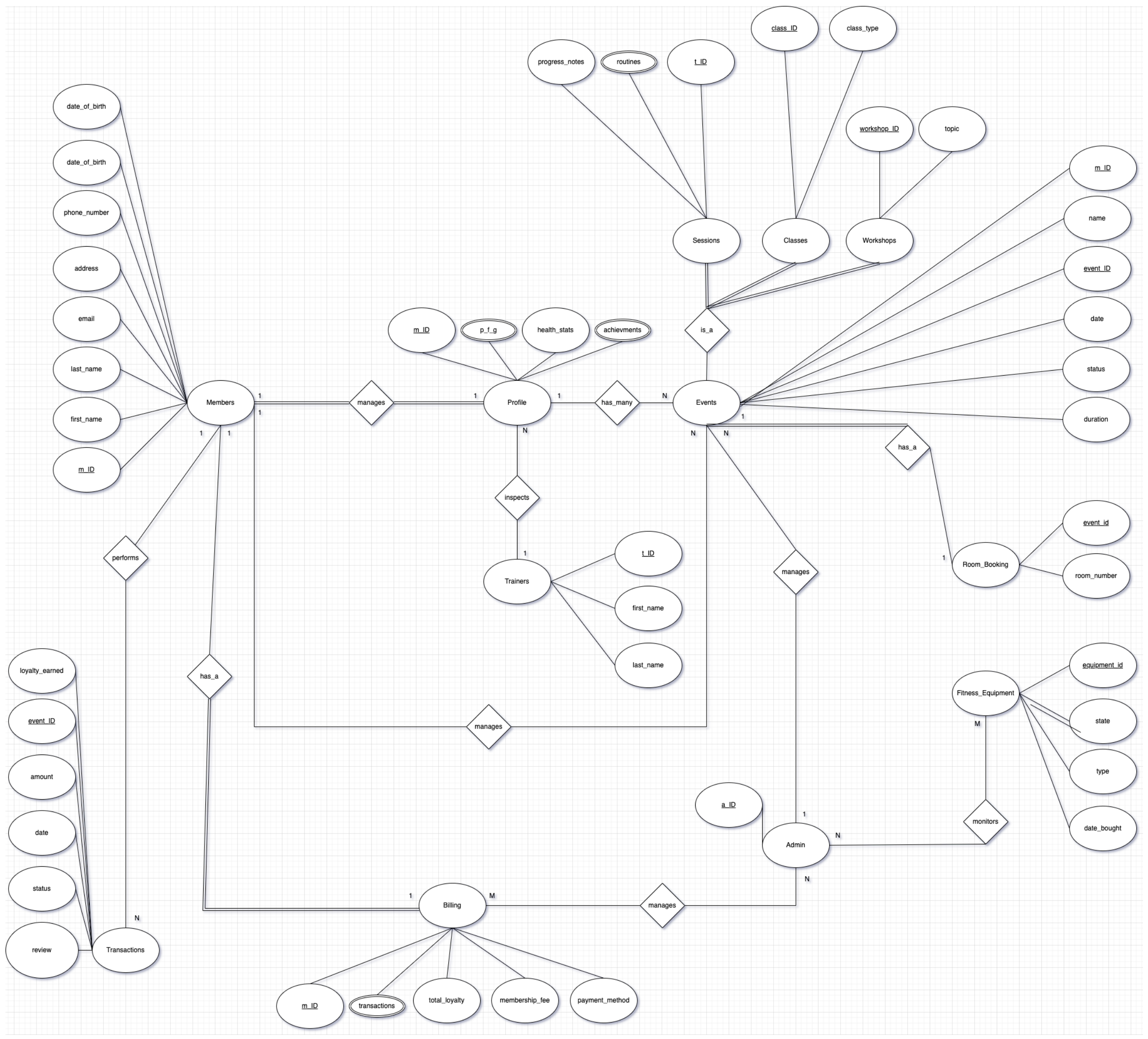
Ariba Rajput (101228808)
Alberto Chavez (100996906)

1 Problem Statement

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, setting personal fitness goals and inputting health metrics. Once registered, they will gain access to a personalized dashboard that tracks their exercise routines, fitness achievements, and health statistics. The platform will also enable members to schedule, reschedule, or cancel personal training sessions with certified trainers. Furthermore, members can register for group fitness classes, workshops, and other events, ensuring they always stay updated with their schedules and receive timely reminders for their sessions. On the other side, the system should empower trainers with tools to manage their schedules, view member profiles, and input progress notes after each training session. Administrative staff, the backbone of the club’s operations, should have features that allow them to oversee club resources effectively. This includes managing room bookings, monitoring fitness equipment maintenance, and updating class schedules. Additionally, they should have a robust system to oversee billing, process payments for membership fees, personal training sessions, and other services, and monitor club activities for quality assurance. The club’s unique selling point is its loyalty program; every transaction earns members loyalty points, which can be redeemed for future services.

2 Project Report

2.1 Conceptual Design



The following is the ER diagram for the implementation of the Health and Fitness Club Management System.

Profiles have a foreign key m_ID (for Member) so information on a member (first_name, last_name, etc) will also be on a profile. A profile will also have fitness goals and health metrics that are managed by a Member in a 1:1 relationship. This relationship is also total participation since a profile cannot exist without a Member.

1 Profile can have many events. Profiles do not need events to exist, and events can exist without being tied to a profile so the participation on both ends is partial. Events are either a Session, Workshop, or Class, and the event_ID is a foreign key to the primary key of a Session, Class, Workshop, or a unique event key for "other".

For a Session, Class, or Workshop to exist it must be an event. An event can be something "other" than those categories so its total-participation between Session, Class, and Workshop and Event, and partial between Event-Session/Workshop/Class.

1 member can also schedule multiple events, giving them the capability to schedule events (Sessions, Classes, Workshops, Others) so long as they have room.

1 trainer can inspect many profiles. They can update health_stats, edit events (add progress_notes to sessions, for example, and update any information on profiles that they have access to (Members they have sessions, workshops, with etc)

1 event will have 1 room_booking (assumption made that the club has rooms big enough that events can be held, rather than having a multivalued attribute for room bookings.)

Since equipment will need to be inspected on a rolling basis, many admins can monitor many equipment. This relationship is partial since the relationship is just monitoring does not rely on the existence of either party to exist.

1 admin will manage many events. We have assumed that 1 admin will oversee 1 event. This participation is partial, so that is not necessary, but if there were an admin overseeing the event there would only be 1 (assumed constraint)

Many admins oversee many Billing accounts. This where the transactions are stored, as well as billing information for each member which the admin will manage. There is a multivalued attribute called transactions on each billing account, and each transaction allows for loyalty to be earned.

1 member has one billing account that oversees their entire financial profile with the club. For a billing account to exist, the member needs to exist, to Billing→Member is total participation.

1 member will perform many transactions (which then are stored in each billing account), and this is after they complete an event (our club trusts our members [also store card information on file]). Each transaction has the option for the member to leave a review, which admins can access through the billing account of each member for quality assurance purposes. Transactions do not have to be with members (We are assuming that 1-off events at the club can happen where non-members can attend)

2.2 Reduction to Relational Schemas

Members
member_ID (Primary Key)

Profiles
m_ID (Primary Key)
References Members via member_ID (Foreign Key)

Events
event_ID (Primary Key)
References Profiles via m_ID (Foreign Key)

Sessions
session_ID (Primary Key)
References Events via event_ID (Foreign Key)

Workshops
workshop_ID (Primary Key)
References Events via event_ID (Foreign Key)

Classes
class_ID (Primary Key)
References Events via event_ID (Foreign Key)

Room_Booking
booking_ID (Primary Key)
References Events via event_ID (Foreign Key)

Fitness_Equipment
equipment_ID (Primary Key)

Billing
billing_ID (Primary Key)
References Members via m_ID (Foreign Key)

Transactions
transaction_ID (Primary Key)
References: Billing via billing_ID (Foreign Key)

Admin
admin_ID (Primary Key)

Admin_Billing_Log
Admin_billinig_entry (Primary Key)
References: Admin and Members via a_ID and m_ID (Foreign Keys)

Admin_FitnessEquipment_Log
Admin_fitness_equipment_entry (Primary Key)
References: admin amd equipment via a_ID and equipment_ID (Foreign Keys)

Routines
routine_ID (Primary Key)
References: session via session_ID (Foreign Key)

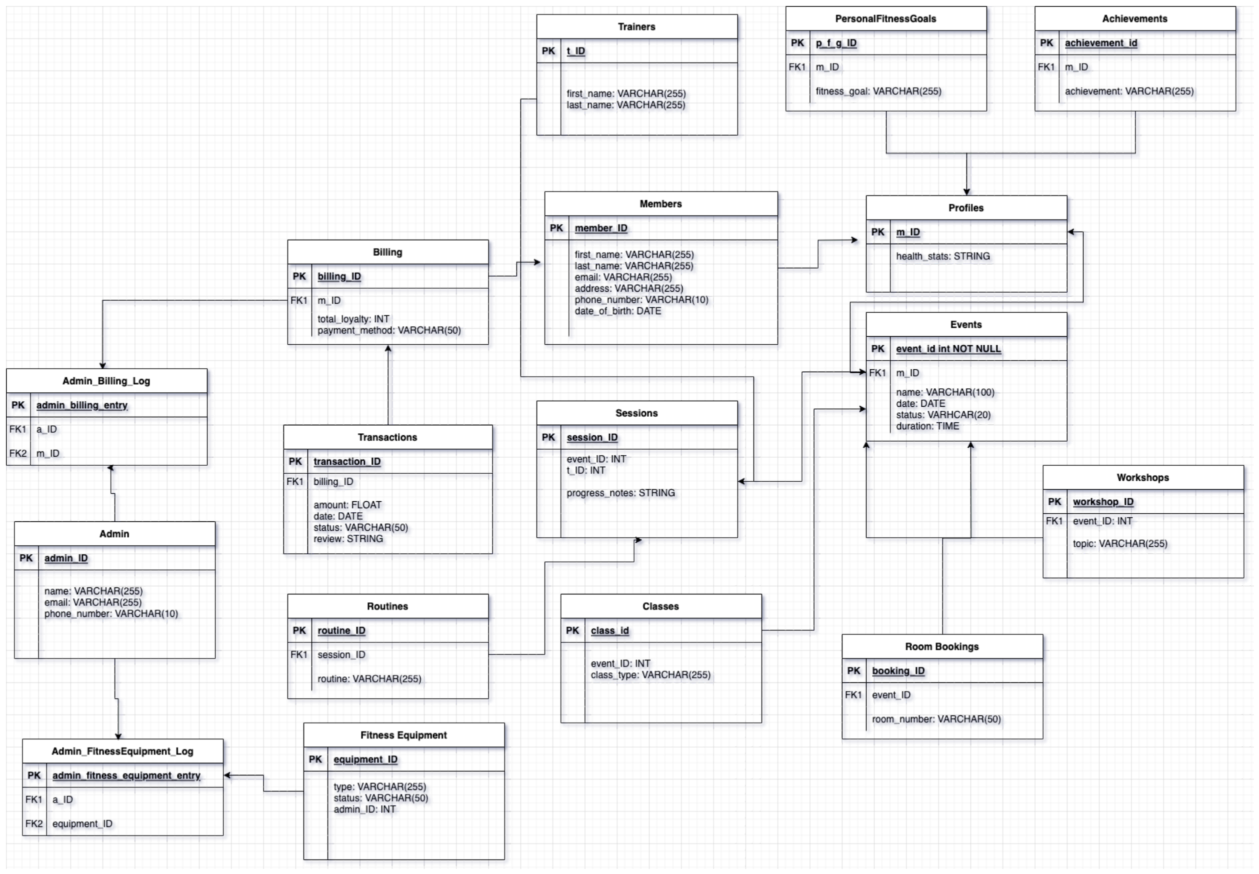
2.3 Normalization of Relational Schemas

We designed our database intending to be fully compliant with 2NF and 3NF from the ground up, by carefully manipulating the dependencies and creating new relationships as necessary. As a result, our database has been successfully normalized into 2NF AND 3NF.

2NF: In our database, every attribute is fully functionally dependent on the primary key.
For example, in the Members table, attributes like first_name, last_name, and email are all fully dependent on member_ID.

3NF: Our database has no transitive dependencies for non-prime attributes. For example, in the Billing table, attributes like total_loyalty and payment_method are dependent on billing_ID which is the primary key, and not on any other non-key attribute. Each non-key attribute is directly dependent on the primary key, which ensures that the tables are in 3NF.

2.4 Database Schema Diagram



2.5 Implementation

Table Creation - Members

```
CREATE TABLE Members (
    member_ID INT PRIMARY KEY,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    email VARCHAR(255),
    address VARCHAR(255),
    phone_number VARCHAR(15),
    date_of_birth DATE,
);
```

Inserting Data into Members Table

```
INSERT INTO Members (member_ID, first_name, last_name, email, address, phone_number,
date_of_birth, membership_date, membership_type, emergency_contact_name,
emergency_contact_phone)

VALUES (1, Alberto, 'C', 'alberto.c@email.com', '123 Rideau St', 613000000, '1810-02-19',
'2023-01-01', 'Annual', Ariba', 6138904534);
```

Table Creation - Events

```
CREATE TABLE Events (
    event_ID INT PRIMARY KEY,
    m_ID INT,
    type VARCHAR(255),
    FOREIGN KEY (m_ID) REFERENCES Profiles(m_ID)
);
```

Update Query

```
UPDATE Members
SET email = 'new.email@email.com'
WHERE member_ID = 1;
```

Deletion Query

```
DELETE FROM Members
```

```
WHERE member_ID = 1;
```

Select Query

```
SELECT * FROM Members
WHERE membership_type = 'Annual';
```

Table Creation - Billing

```
CREATE TABLE Billing (
   billing_ID INT PRIMARY KEY,
   m_ID INT,
   total_loyalty INT,
   payment_method VARCHAR(50),
   FOREIGN KEY (m_ID) REFERENCES Members(member_ID)
);
```

2.7 Github Repository

https://github.com/AribaRajput/Health-And-Fitness-Club-Management-Ariba-Alberto