Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Gym Management System Report
(Information Management Project Report)

Submitted by:
Prachi Singhroha
(21355131)

Submitted To:
Prof. Vincent Patrick Wade

Module ID:
CSU44D01

Declaration:
I understand that this is an individual assessment and that collaboration is not permitted. I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at http://www.tcd.ie/calendar. I understand that by returning this declaration with my work, I am agreeing with the above statement.

Contents of Report

1. Application description……..3
2. Entity Relationship Diagram……..4
3. Mapping to Relational Schema……..5
4. Functional Dependency Diagrams……. 5-7

Explanations of data and SQL Code:

5. Explanation and SQL Code for Creating the database Tables…7-9
6. Explanation and SQL Code for Altering tables……..9
7. Explanation and SQL Code for Creating Views……..10
8. Explanation and SQL Code for Populating the Tables……10
9. Explanation and SQL Code for retrieving information from the database……..……..10
10. Explanation and SQL Code for Triggers……..10
11. Explanation and SQL Code for Security……..11
12. Explanation of Additional SQL Features……..11
13. References…….11

Application description

As part of the CSU44D01 course, this report is for the project, Gym Management System. The system is designed for the Gyms belonging to a particular company, here named Fitness101. Members enrolled in a specific workout plan can access them in any gym, part of Fitness101, teaching that plan, given their membership is still valid. The following entities are used to create the database.

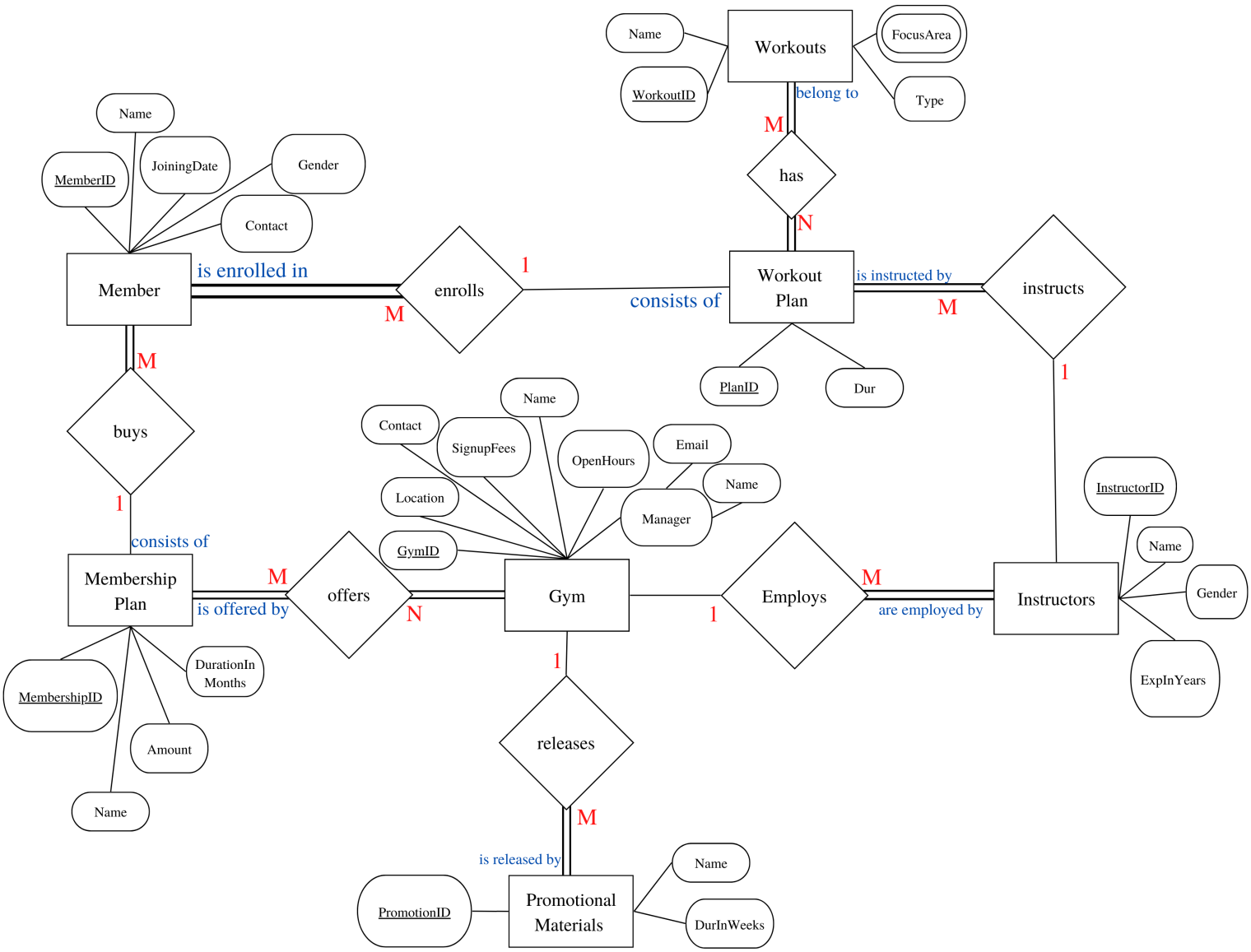
- Gym
- Instructors
- Membership Plans
- Members
- Workout
- Workout Plans
- Promotional Materials

The database can be used for the following purposes.

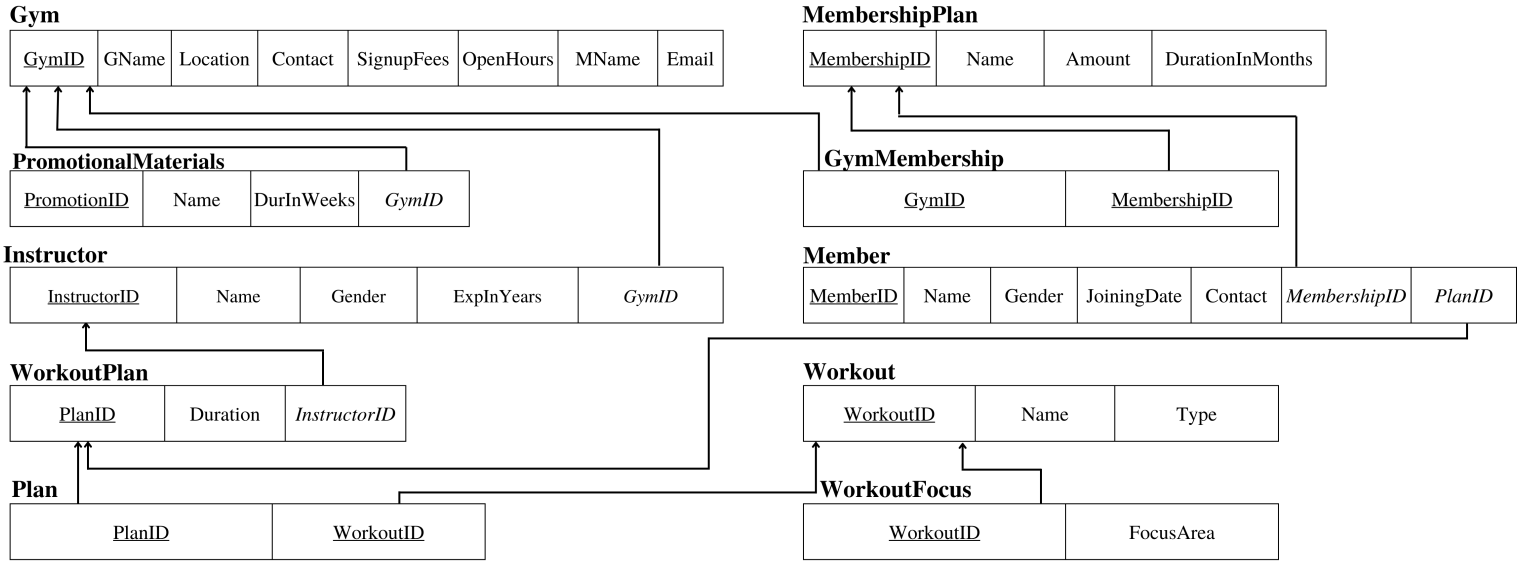
- Retrieving information about any of the above entities
- Managing database for Fitness101
- Analysing data to increase Fitness101's presence
- Reduce redundancy
- Find the cheapest gyms
- Look up for female instructors
- Look up for experiences instructors

All the data represented is fictional and does not include any internal information about gyms and their workers. The sole purpose of the project is to understand ER models, functional dependencies, and SQL.

Entity Relationship Diagram



Mapping to Relational Schema



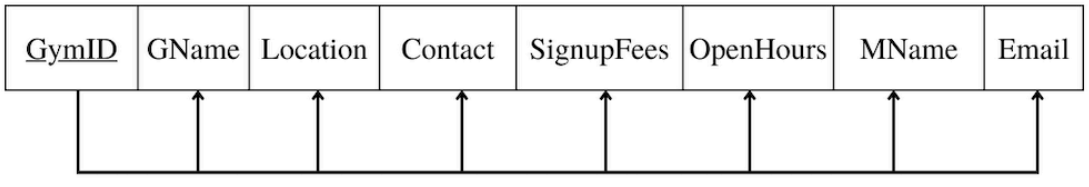
Relation between entities are as follows.

- Every gym employs multiple instructors (1:M relation)
- Every gym releases multiple promotional materials (1:M relation)
- Multiple gyms offers multiple workout plans (M:N relation)
- Every member buys a membership plan(M:1 relation)
- Every member enrols in a workout plan(M:1 relation)
- Instructors instruct multiple workout plans (1:M relation)
- Multiple workouts are part of multiple workout plans (M:N relation)

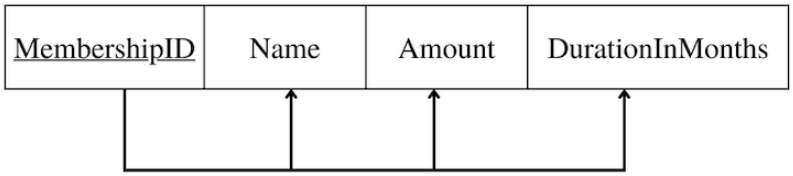
Functional Dependencies

1. Table for '**Gym**'

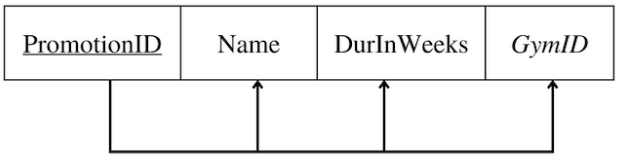
Primary Key: GymID



2. Table for '**MembershipPlan**'
Primary Key: MembershipID

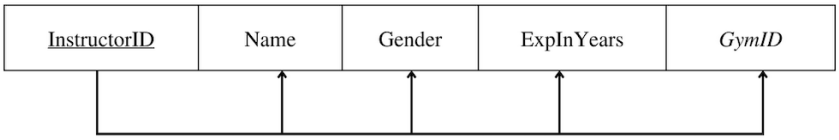


3. Table for '**PromotionalMaterial**'
Primary Key: PromotionID
Foreign Key: GymID

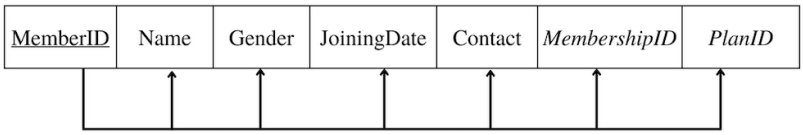


4. Table for '**GymMembership**'
Primary Key: MembershipID, GymID

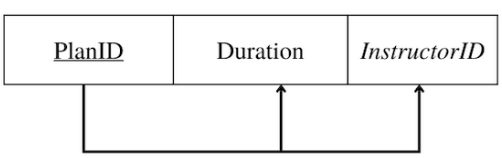
5. Table for '**Instructor**'
Primary Key: InstructorID
Foreign Key: GymID



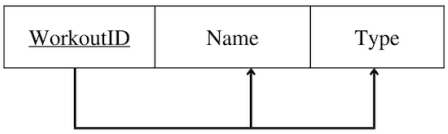
6. Table for '**Members**'
Primary Key: MemberID
Foreign Key: MembershipID, PlanID



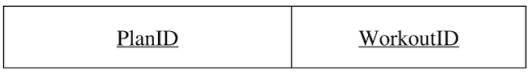
7. Table for '**WorkoutPlan**'
Primary Key: PlanID
Foreign Key: InstructorID



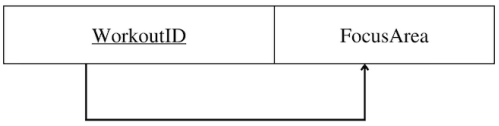
8. Table for '**Workout**'
Primary Key: WorkoutID



9. Table for '**Plan**'
Primary Key: PlanID, WorkoutID



10. Table for '**WorkoutFocus**'
Primary Key: WorkoutID



Explanations of data and SQL Code:

Explanation and SQL Code for Creating the database Tables

To create the tables following code was used.

CREATE TABLE *table_name* (*attributes* *constraints*);

Following are semantics for the tables created.

1. Gym

Attribute	Contraints
GymID	int, Auto_increment
GName	varchar(10), NOT NULL
Location	varchar(20), DEFAULT 'Dublin'
Contact	int, NOT NULL
SignupFees	varchar(5), NOT NULL
OpenHours	time, NOT NULL
Mname	varchar(10), NOT NULL
Email	varchar(50), NOT NULL

2. MembershipPlan

Attribute	Contraints
MembershipID	int, Auto_increment
MPname	varchar(10), NOT NULL
Amount	varchar(20), DEFAULT 'Dublin'
DurationInMonths	int, NOT NULL

3. GymMembership

Attribute	Contraints
MembershipID	int(3), NOT NULL
gymID	int(3), NOT NULL

4. Instructor

Attribute	Contraints
InstructorID	int, Auto_increment
IName	varchar(10), NOT NULL
ExpInYears	int, NOT NULL
GymID	int, NOT NULL
Gender	varchar(1), NOT NULL

5. PromotionalMaterial

Attribute	Contraints
PromotionID	int, Auto_increment
PName	varchar(20), NOT NULL
DurInWeeks	int, NOT NULL
GymID	int, NOT NULL

6. Workout

Attribute	Contraints
WorkoutID	int, Auto_increment
WName	varchar(10), NOT NULL
Type	varchar(15), NOT NULL

7. WorkoutPlan

Attribute	Contraints
PlanID	int, Auto_increment
DurInWeeks	int, NOT NULL
InstructorID	int, NOT NULL

8. Plan

Attribute	Contraints
PlanID	int(3), NOT NULL
GymID	int(3), NOT NULL

9. WorkoutFocus

Attribute	Contraints
WorkoutID	int, NOT NULL
Focus	varchar(15)

10. Members

Attribute	Contraints
MemberID	int, Auto_increment
MemberName	varchar(10), NOT NULL
Gender	varchar(1), NOT NULL
Contact	int, NOT NULL
Joining Date	DATE, NOT NULL
MembershipID	int(3), NOT NULL
PlanID	int(3), NOT NULL

Explanation and SQL Code for Altering tables

I have added foreign keys and constraints for altering tables. To add constraint I've used the following code. I added constraint to make sure that the values added in Gender (Male, Female) and Workout types (beginner, intermediate, advanced) are correct.

ALTER TABLE *table_name*
ADD CONSTRAINT *constraint_name* CHECK (*condition*);

To add foreign keys I've used the following code.
ALTER TABLE *table_name*
ADD FOREIGN KEY (*Attribute*) REFERENCES RefTable(*RefAttribute*);

Explanation and SQL Code for Creating Views

I created 3 views, one to find the cheapest gyms, female instructors' name and experience and instructors with more than 5 years of experience . Following code was used to create the views.

CREATE VIEW *viewName* AS
SELECT *attributes* FROM *tableName*
WHERE *condition*;

Explanation and SQL Code for Populating the table

I populated the tables by adding at least 5 tuples to each using the following code.

INSERT *tableName* (*attributes*) VALUES (*values*);

Explanation and SQL Code for retrieving information from the database

I displayed the tables in the terminal using the following command.

SELECT * FROM *tableName;*

I also displayed a table of gyms members can use for a particular workout plan. Following example is for MembershipID = 2 but it can be used for any required ID.

SELECT Members.MemberName, Gym.GName
FROM Members
INNER JOIN Gym ON Members.MembershipID = 2;

Explanation and SQL Code for Triggers

I created 4 triggers for the database.

1. Set limit on membership duration
2. Set limit on joining date if members
3. Set limit on duration of workout plans
4. Set limit on amount of membership plans

I've defined a variable as well to use in one of the triggers. Syntax is as follows.

SET *@variableName* = *@value*;

Following code was used since all the triggers were delimiters.

delimiter //
CREATE TRIGGER *triggerName* BEFORE INSERT
ON *tableName*
FOR EACH ROW
IF NEW.*attribute* > *quantity* THEN

```
SIGNAL SQLSTATE '50001' SET MESSAGE_TEXT = 'message';
END IF;//
delimiter ;
```

Explanation and SQL Code for Security

I've added security constraints while creating the tables to reduce redundancy. Also the manager has access to view and edit the data while others can just login to check their own status. I did that using GRANT command in the terminal. The database can be made more secure by adding more layers of security. I also locked the tables using the following syntax before populating to reduce the chances of redundancy and invasion.

```
LOCK TABLES `tableName` WRITE;
UNLOCK TABLES;
```

Explanation of Additional SQL Features

I used a few functions of MySQL while creating this database.

- CURDATE()
- IF loop

References

- w3schools
- stackoverflow
- mysqlDocumentation