DojoDB Database Management

Requirements

The goal of the project is to create a website for a gym. The gym needs to keep track of the members that use it, the certified trainers and their training sessions, events going on at the gym, and the gym's backend organization.

1. **Members**:
    - Members need to be able to view, create, and delete their goals, health metrics, workout routines, personal training sessions, classes they're in, and complaints if something goes wrong.
    - Members will need to pay bills/collect loyalty points.
    - Members should not be able to see other members' personal details.
2. **Trainers**:
    - Trainers must be able to create new sessions and track the sessions they have with members, as well as their general schedule.
    - Trainers should not be able to see members' personal details.
3. **Classes**:
    - Classes will be able to set aside space within the gym and keep track of all members who have signed up for the class
4. **Backend**:
    - There has to be a way for gym administrators to oversee all activity within the gym. They should have access to tools to manage classes, gym equipment, members, and trainers.

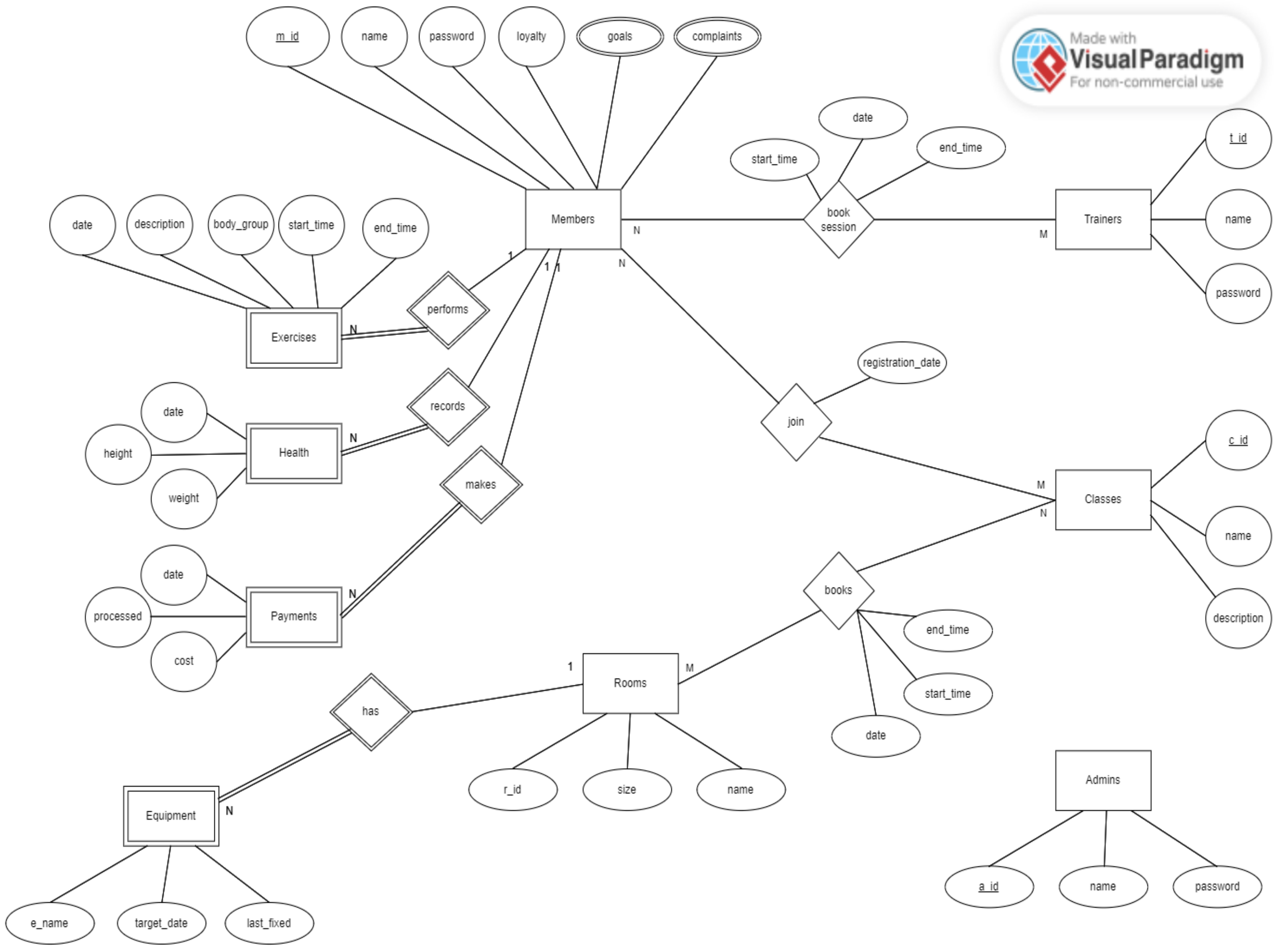
ER model Overview

The ER model for our gym website aligns closely with our requirements. Here's how they relate:

1. **Members**:
    - In the database, the **Members** entity is central. It supports functionalities like viewing, creating, and deleting goals, health metrics, workout routines, personal training sessions, classes, and complaints.
    - The weak entities like **Health**, **Payments**, and **Exercises** are dependent on **Members** and facilitate tracking of health metrics, financial transactions, and workout routines, respectively.
    - Multivalued attributes like **Goals** and **Complaints** are stored with the Member ID, aligning with the requirement for members to manage their goals and complaints.

2. **Trainers**:
    - The **Trainers** entity our database enables the creation and tracking of sessions with members.
    - The many-to-many relationship between **Members** and **Trainers** through the **Sessions** joining table(represented by book session diamond) allows trainers to manage their schedules and sessions with members.
3. **Classes**:
    - The **Classes** entity manages the scheduling and tracking of members enrolled in different classes.
    - The relationship between **Members** and **Classes** through the **MemberClasses** joining table(represented by join diamond) aligns with the requirement for classes to track member enrollments.
4. **Backend/Admins**:
    - The **Admins** standalone table in the ER model provides the backend framework.
    - Administrators have oversight over all activities within the gym, which includes managing classes, gym equipment, members, and trainers – aligning with the backend requirement of the website.
5. **Privacy and Security**:
    - The structure of the ER model ensures that members and trainers cannot access each other's personal details, aligning with the privacy requirements of the project.
6. **Gym Space and Equipment**:
    - The relationship between **Classes** and **Rooms** through the **Bookings** joining table (represented by books diamond) allows for the management of space within the gym for classes.
    - The **Equipment** entity, being a weak entity of **Rooms**, helps in managing gym equipment, which is part of the backend administrative functionality.

In summary, the ER model provides a robust framework for the gym website, addressing the core functionalities and privacy concerns as per the project's requirements. It efficiently maps out the necessary entities and their interactions to support the website's functionalities.



ER Model Explanation

1. **Members and Health(Weak):**
    - Why Weak: A health record (Health) is existentially dependent on a member (Members). Without being associated with a specific member, a health record has no context or purpose. It's identified not just by its own ID (h_id) but also by the member it belongs to (m_id)
    - Cardinality: One-to-Many (1:N)
    - Participation: Partial for Members, Total for Health

- Description: Each member can have multiple health records, but each health record is associated with exactly one member.

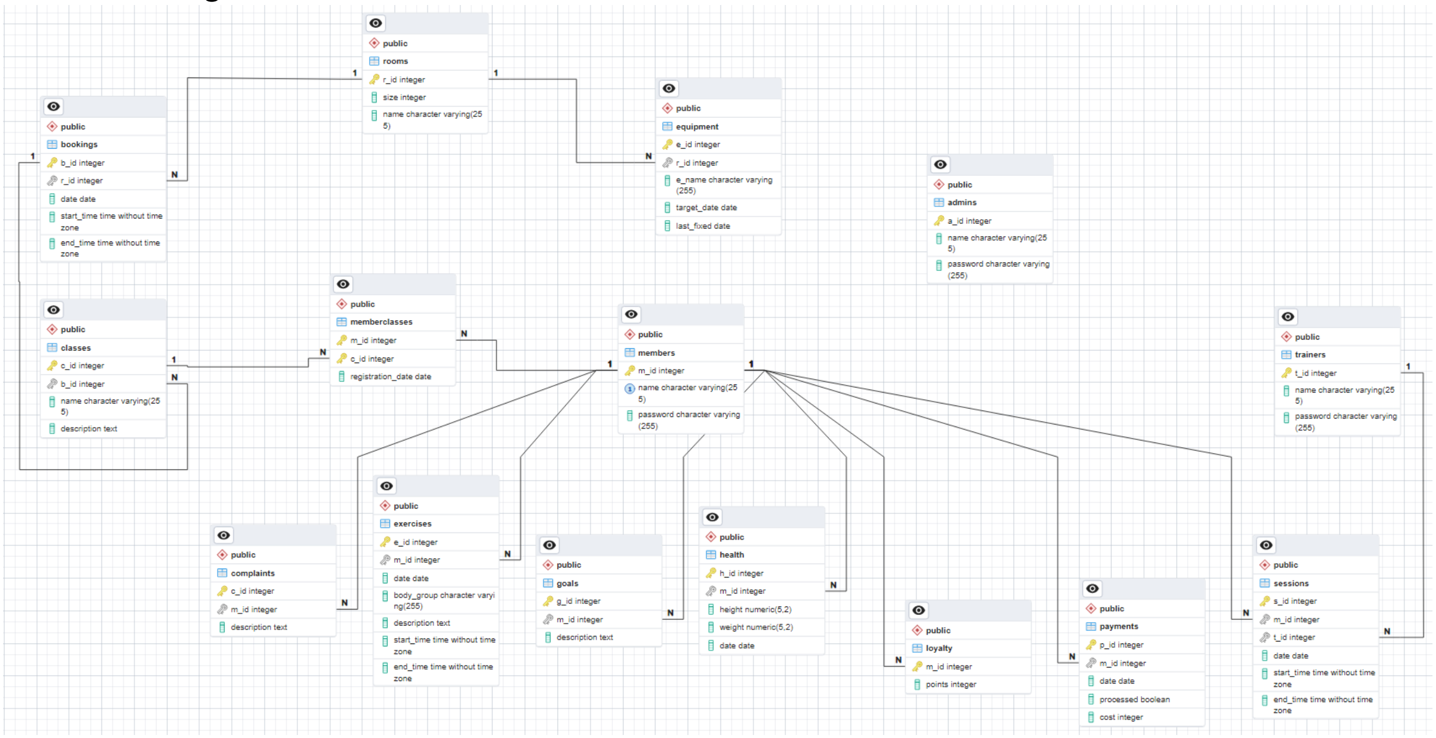
2. **Members and Exercises(Weak):**
    - Why Weak: Exercise records (Exercises) are tied directly to members. These records do not stand alone and are meaningful only when associated with a member. Their unique identification also depends on the member (m_id) they are associated with.
    - Cardinality: One-to-Many (1:N)
    - Participation: Partial for Members, Total for Exercises
    - Description: Each member can perform multiple exercises, but each exercise entry is associated with exactly one member.
3. **Members and Payments(Weak):**
    - Why Weak: Payment records (Payments) are inherently linked to members. A payment record is relevant only in the context of a member making the payment. The identity of a payment record is also dependent on the member to whom it is linked.
    - Cardinality: One-to-Many (1:N)
    - Participation: Partial for Members, Total for Payments
    - Description: Each member can make multiple payments, but each payment is associated with exactly one member.
4. **Members and Classes:**
    - Cardinality: One-to-Many (M:N)
    - Participation: Partial for both Members and Classes
    - Description: Members can register for multiple classes, and each class can have multiple members registered. The **MemberClasses**(represented by join) table links members to the classes they are enrolled in, including information like registration dates.
5. **Members and Trainers:**
    - Cardinality: Many-to-Many (M:N) through the Sessions table(represented by (book session), includes date, start_time and end_time
    - Participation: Partial for both Members and Trainers

- Description: Members can have sessions with various trainers, and trainers can have sessions with various members.

6. **Classes and Rooms:**
    - One-to-Many (1:M) through the Bookings table
    - Participation: Partial for both Classes and Rooms
    - Description: Each class is associated with a booking that specifies a room, indicating that each class takes place in a specific room. However, a room can host multiple classes. The **Bookings** (represented by books) table serves as the intermediary, linking each class to a room at a specific date and time. It includes date, start_time and end_time. This setup allows for the scheduling and organization of various classes in different rooms.
7. **Rooms and Equipment(Weak):**
    - Why Weak: Equipment (Equipment) is dependent on the room (Rooms) it is placed in. Equipment by itself does not have a standalone context; its identification and purpose are tied to the room it occupies.
    - Cardinality: One-to-Many (1:N)
    - Participation: Partial for Rooms, Total for Equipment
    - Description: Each room can contain multiple pieces of equipment, but each piece of equipment is placed in exactly one room.

These relationships form the foundation of how data is interconnected in your gym management database, providing a clear understanding of the interactions between different entities.

Schema Diagram



Tables in Normal Form

1. **Members**
   - Primary Key (PK): { m_id }
   - Functional Dependencies: m_id → name, password
   - Normalization: Meets 2NF and 3NF (No transitive dependencies)
2. **Payments**
   - PK: { p_id }
   - Functional Dependencies: p_id → m_id, date, processed, cost
   - Normalization: Meets 2NF and 3NF (Possible transitive dependency taken care of by p_id → m_id)
3. **Loyalty**
   - PK: { m_id }
   - Functional Dependencies: m_id → points
   - Normalization: Meets 2NF and 3NF (No transitive dependencies)
4. **Health**

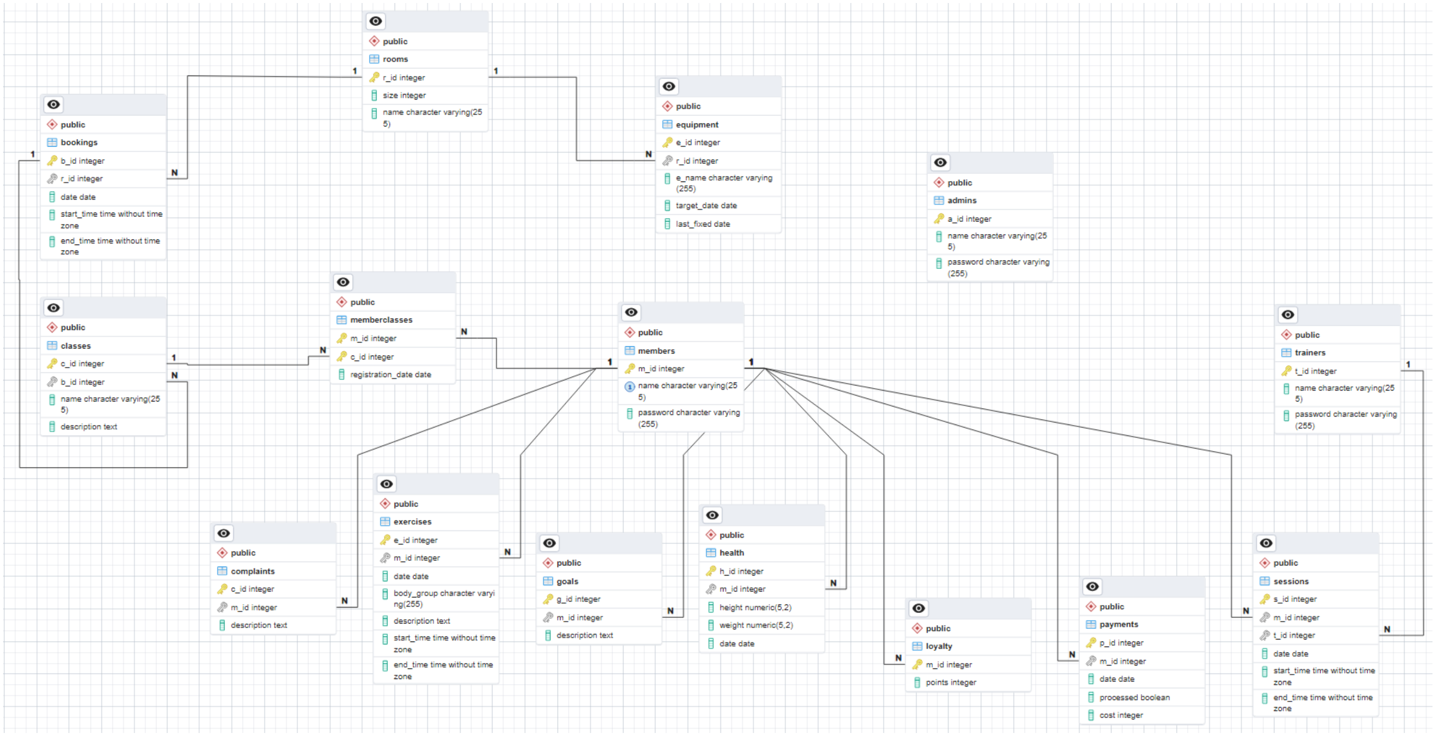
- PK: { h_id }
- Functional Dependencies: h_id → m_id, height, weight, date
- Normalization: Meets 2NF and 3NF (Possible transitive dependency taken care of by h_id → m_id)

5. **Goals**
   - PK: { g_id }
   - Functional Dependencies: g_id → m_id, description
   - Normalization: Meets 2NF and 3NF (Possible transitive dependency taken care of by g_id → m_id)
6. **Admins**
   - PK: { a_id }
   - Functional Dependencies: h_id → name, password
   - Normalization: Meets 2NF and 3NF (No transitive dependencies)
7. **Exercises**
   - PK: { e_id }
   - Functional Dependencies: e_id → m_id, date, description, start_time, end_time
   - Normalization: Meets 2NF and 3NF (Possible transitive dependency taken care of by e_id → m_id)
8. **Complaints**
   - PK: { c_id }
   - Functional Dependencies: c_id → m_id, description
   - Normalization: Meets 2NF and 3NF (No transitive dependencies)
9. **Trainers**
   - PK: { t_id }
   - Functional Dependencies: t_id → name, password
   - Normalization: Meets 2NF and 3NF (No transitive dependencies)
10. **Sessions**
    - PK: { s_id }

- Functional Dependencies: s_id → m_id, t_id, date, start_time, end_time
- Normalization: Meets 2NF and 3NF (Possible transitive dependencies taken care of by s_id → m_id and s_id → t_id)

11. **Rooms**
    - PK: { r_id }
    - Functional Dependencies: r_id → size, name
    - Normalization: Meets 2NF and 3NF (No transitive dependencies)
12. **Bookings**
    - PK: { b_id }
    - Functional Dependencies: b_id → r_id, date, start_time, end_time
    - Normalization: Meets 2NF and 3NF (Possible transitive dependency taken care of by b_id → r_id**)**
13. **Classes**
    - PK: { c_id }
    - Functional Dependencies: c_id → b_id, name, description
    - Normalization: Meets 2NF and 3NF (Possible transitive dependency taken care of by c_id → b_id)
14. **MemberClasses**
    - PK: { m_id, c_id }
    - Functional Dependencies: m_id, c_id → registration_date
    - Normalization: Meets 2NF and 3NF (No transitive dependencies)
15. **Equipment**
    - PK: { e_id }
    - Functional Dependencies: e_id → r_id, name, target_date, last_fixed
    - Normalization: Meets 2NF and 3NF (Possible transitive dependency taken care of by e_id → r_id**)**

Final Schema Diagram(Did not change after Normalization)


public
rooms
r_id integer
size integer
name character varying(255)
public
bookings
b_id integer
r_id integer
date date
start_time time without time zone
end_time time without time zone
public
equipment
e_id integer
r_id integer
e_name character varying (255)
target_date date
last_fixed date
public
admins
a_id integer
name character varying(255)
password character varying (255)
public
classes
c_id integer
b_id integer
name character varying(255)
description text
public
memberclasses
m_id integer
c_id integer
registration_date date
public
members
m_id integer
name character varying(255)
password character varying (255)
public
trainers
t_id integer
name character varying(255)
password character varying (255)
public
complaints
c_id integer
m_id integer
description text
public
exercises
e_id integer
m_id integer
date date
body_group character varying(255)
description text
start_time time without time zone
end_time time without time zone
public
goals
g_id integer
m_id integer
description text
public
health
h_id integer
m_id integer
height numeric(5,2)
weight numeric(5,2)
date date
public
loyalty
m_id integer
points integer
public
payments
p_id integer
m_id integer
date date
processed boolean
cost integer
public
sessions
s_id integer
m_id integer
t_id integer
date date
start_time time without time zone
end_time time without time zone
1
N