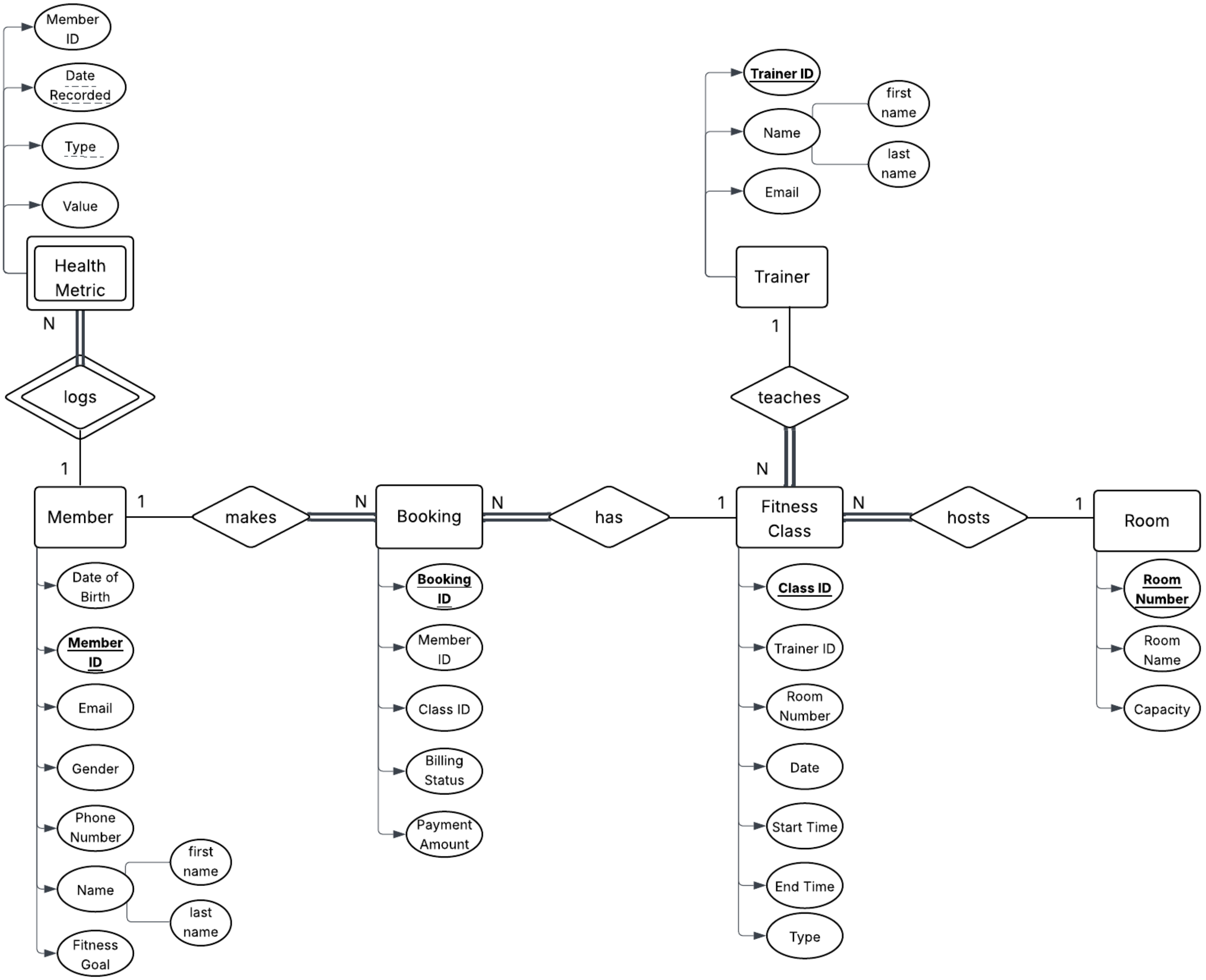
ER MODEL DIAGRAM



The database schema has already been designed to satisfy 3NF requirements :

- **1NF:** All attributes contain atomic values. For example, the Name attribute has been decomposed into first and last name in both Members and Trainers Table making sure no composite values exist
- **2NF – Full Functional Dependency:** all non-key attributes are fully dependent on the primary key. In HealthMetrics table, which uses a composite key (member_id + date + type), the value attribute depends on the *entire* key combination, not just a part of it (e.g., a value cannot be determined by member_id alone).
- **3NF – Transitive Dependencies:** There are no transitive dependencieswhere a non key attribute depends on another non-key attribute. For example, In the FitnessClasses table, we store room_number (a foreign key) but not room_capacity. Storing capacity there would create a transitive dependency (class_id -> room_number -> capacity). Instead, capacity is correctly stored only in the Rooms table, eliminating redundancy.

ERD MAPPING TABLE:

Requirements	Assumptions	Representation in ER model
• A member should be register by providing information such as name, date of birth, gender, and contact details • Members will have the ability to establish/track personalized fitness goal	• Name is composite (First/Last) for normalization. • Email is unique. • Only one contact detail per person (1 phone number) • Fitness Goal is a single current target.	Member - Entity Attributes: PK Member ID, First Name, Last Name, Email, Phone, Gender, DOB, Fitness Goal.
Trainers are uniquely identified by a system ID. Record name and email.	Name is composite (First/Last). Trainers do not have health metrics in this system.	Trainer - Entity Attributes: PK Trainer ID, First Name, Last Name, Email.
Health Metrics track a member's history (weight, heart rate, etc) over time. Entries must not be overwritten.	This is a Weak Entity. It cannot exist without a Member. A specific log is identified by the MemberID + Date + Type.	Health Metric - Weak Entity Owner: Member Relationship: logs (1:N, Total Participation)

		Attributes: member_id (FK), date_recorded, type, value. Primary Key: Composite (member id + date recorded + type).
Fitness Classes represent the schedule. Each class has a specific time, room, and trainer.	Personal Training is treated as a Class with capacity = 1.	Fitness Class – Entity Attributes: Class id (PK), Class type, Class date, Start Time, End Time. Foreign Keys: room number and trainer id Relationships: Trainer teaches **Fitness Class,** Room hosts **Fitness Class.**
Bookings link a member to a class and track payment status.	Payment amount is recorded per booking.	Booking – Entity Attributes: Booking id (PK), Payment Amount, Billing Status. Foreign Keys: Member id, Class id
Rooms are physical spaces with a unique number/name and capacity	Room Number is a string and unique.	Room - Entity Attributes: PK Room Number, Room Name, Capacity.

Application Implementation:

- The application logic was implemented in Java using JDBC
  - Database Connection: The application establishes a connection to the PostgreSQL/Pg Admin database using the DriverManager class
  - Data Retrieval: To retrieve data such as verifying a member's login credentials or viewing their dashboard, the application uses PreparedStatement to execute SELECT queries

    - Data Modification: For operations that modify the database state, such as registering a new user or booking a class, the executeUpdate() method is used.

- The system follows a command line interface structure where a main loop handles user navigation (such as user login + signup) and helper methods perform specific database operations using SQL

- The application uses try-catch blocks to handle the SQL exceptions and is especially important for catching the errors raised by the SQL trigger (check_room_availability) when a double- booking is attempted

- The view, trigger and index were implemented in this way:
    - **View (PublicSchedule):** A virtual table was created to simplify complex joins between FitnessClasses, Rooms, and Trainers. This provides a readable, view of the schedule for public display without exposing underlying IDs
    - **Trigger (check_room_availability):** A BEFORE INSERT trigger was implemented on the FitnessClasses table. This function automatically checks for time overlaps in a specific room before a new class is scheduled. If a conflict is detected, the database raises an exception, preventing double-booking.
    - **Index (idx_member_last_name):** An index was created on the last_name column of the Members table. This optimizes the search performance for the Trainer's "member lookup" function as the member base grows.