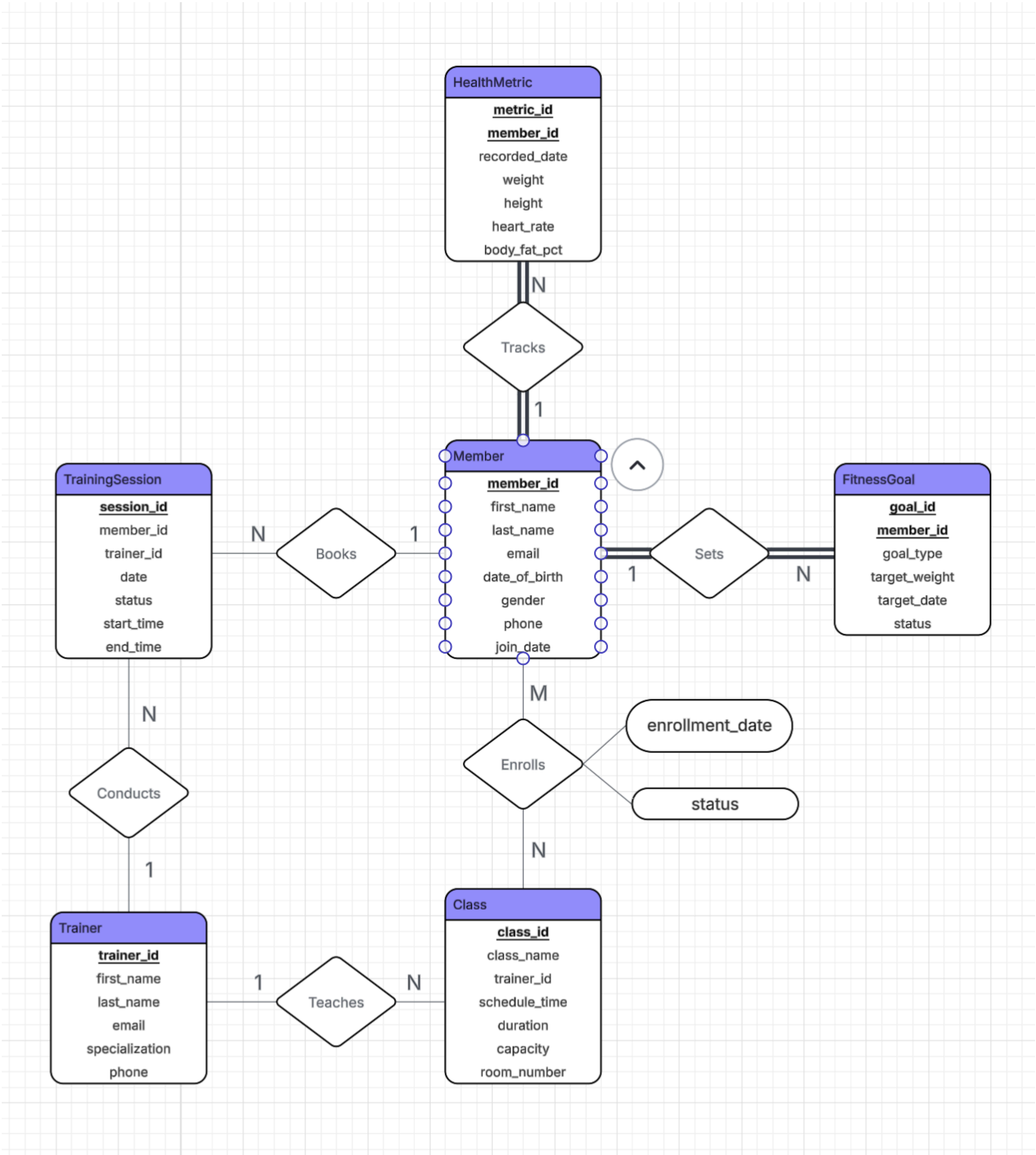
ER Model


HealthMetric
metric_id
member_id
recorded_date
weight
height
heart_rate
body_fat_pct
N
Tracks
1
Member
member_id
first_name
last_name
email
date_of_birth
gender
phone
join_date
TrainingSession
session_id
member_id
trainer_id
date
status
start_time
end_time
N
Books
1
1
Sets
N
FitnessGoal
goal_id
member_id
goal_type
target_weight
target_date
status
N
Conducts
1
M
Enrolls
enrollment_date
status
N
Trainer
trainer_id
first_name
last_name
email
specialization
phone
1
Teaches
N
Class
class_id
class_name
trainer_id
schedule_time
duration
capacity
room_number

ER to Relational Mapping

This secession documents the conversion of the Entity-Relationship model into a normalized relational database schema.

Step1: Regular Entities

Member(member_id PK, first_name, last_name, email UNIQUE, date_of_birth, gender, phone, join_date)

Trainer(trainer_id PK, first_name, last_name, email UNIQUE, specialization, phone)

Class(class_id PK, class_name, trainer_id FK, schedule_time, duration, capacity, room_number)

TrainingSession(session_id PK, member_id FK, trainer_id FK, date, start_time, end_time, status)

Step 2: Weak Entities

Weak entities cannot exist independently and require an owner entity for identification.

HealthMetric(member_id PK/FK, metric_id PK, recorded_date, weight,
height, heart_rate, body_fat_pct)
Explanation: A health metric cannot exist without a member. The metric_id is only unique within the context of a specific member.

FitnessGoal(member_id PK/FK, goal_id PK, goal_type, target_weight,
target_date, status)
Explanation: A fitness goal should be unique to a member, we can easily manage the many goals a member could have.

Step 3: One-To-One Relationships

There are no 1:1 relationships in this ER model.

Step 4: One-to-Many (1:N) Relationships

One-To-Many relationships are implemented by placing a foreign key in the table on the "many" side that references the primary key of the table on the "one" side.
Note: TrainingSession has two foreign keys because it participates in two 1:N relationships.

Step 5: Many-To-Many (M:N) Relationships

Unfortunately, many-to-many relationships cannot be represent in the relational model. Fortunately, we can implement it by creating another table that contains foreign keys referencing both participating entities.

Member ENROLLS Class (M:N):
ClassEnrollment(enrollment_id PK, member_id FK, class_id FK, enrollment_date, status)

Step 6: Multi-valued Attributes

There are no multivalued attributes in this ER model. All of them are single-valued.

Normalization

Second Normal Form (2NF): No partial dependencies exist. All non-key attributes are fully functionally dependent on the entire primary key. For tables with single-valued primary keys, 2NF is automatically satisfied because partial dependency can only occur with composite keys.

In the HealthMetric table, the composite primary key (member_id, metric_id) uniquely identifies each health measurement. Every non-key attribute depends on the entire primary key. For instance, the weight attribute cannot be determined by knowing only the member_id (which member?) or only the metric_id (which measurement for which member?). Both components of the composite key are required to identify a specific health metric record. If we had violated 2NF by including an attribute like member_name that depends solely on member_id, we would have created a partial dependency. The same was done with Fitness goal, where it uses a composite primary key (member_id, goal_id).

Third Normal Form (3NF): No transitive dependencies exist. No non-key attribute depends on another non-key attribute. Each non-key attribute depends only on the primary key. the Class table stores trainer_id as a foreign key but deliberately does not store trainer_name or trainer_email, which would create a transitive dependency where class_id → trainer_id → trainer_name. Instead, trainer details remain in the Trainer table. Each piece of information is stored in exactly one location, removing redundancy in all the tables.