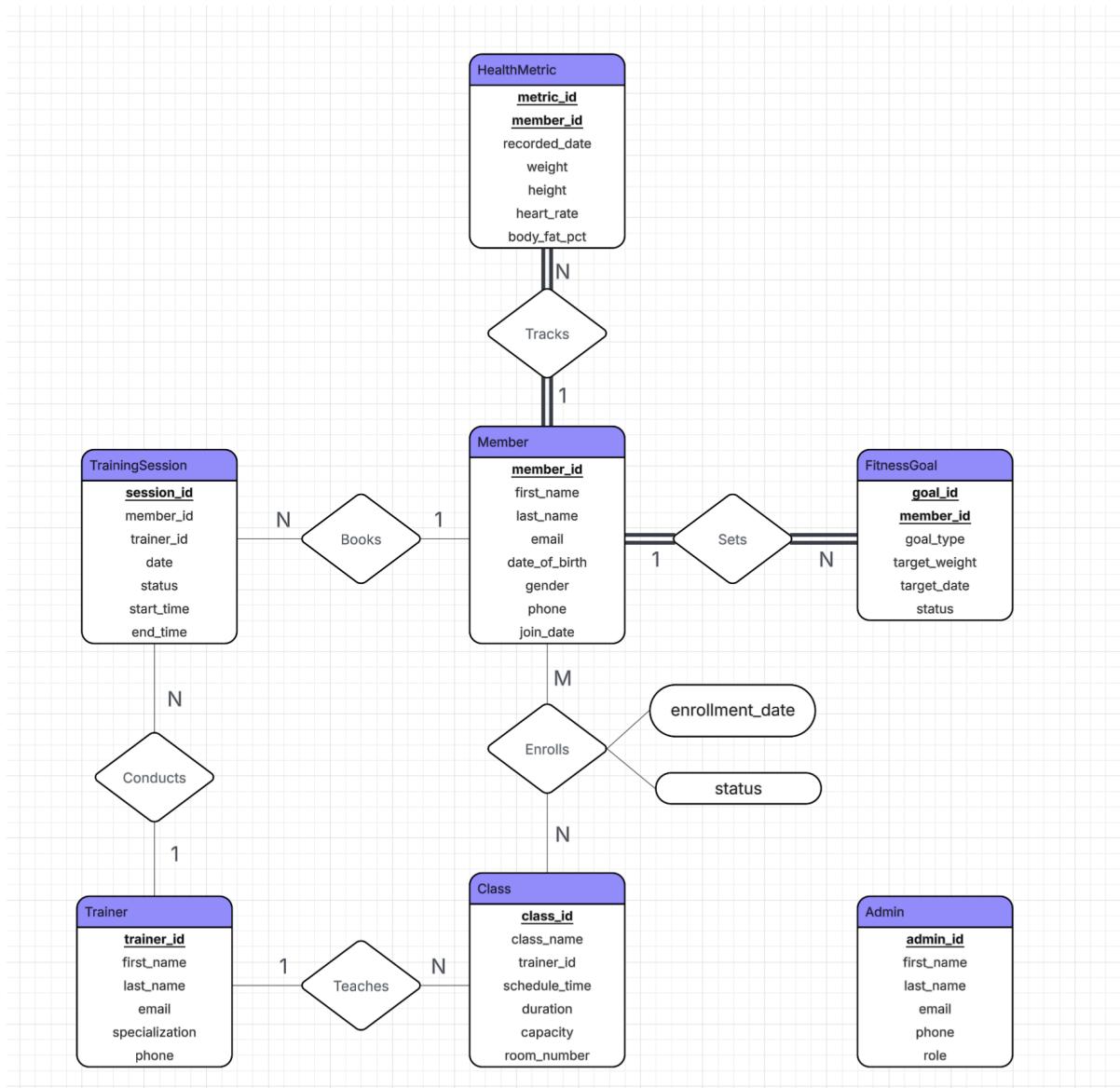


# ER Model



# ER to Relational Mapping

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

## Step 1: 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)

Admin(admin\_id PK, first\_name, last\_name, email UNIQUE, phone, role)

## 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 represented 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.