Relational Schemas

Gym Management System

---

1. DATABASE SCHEMA OVERVIEW

Database Name: gym_management
DBMS: PostgreSQL 12+
ORM: Prisma 5.7.1
Total Tables: 6
Total Relationships: 7

---

2. RELATIONAL SCHEMA NOTATION

```
Table_Name (PK, attribute1, attribute2, FK, ...)

PK = Primary Key (underlined)
FK = Foreign Key (italicized)
UNIQUE = Unique constraint
NOT NULL = Cannot be null
DEFAULT = Default value
```

---

3. NORMALIZED RELATIONAL SCHEMAS

3.1 MEMBERS Table

```
MEMBERS (
    member_id,
    name,
    email,
    password,
    age,
    gender,
    phone,
    join_date,
    status
)
```

Formal Notation:

```
MEMBERS = (member_id: STRING [PK],
           name: STRING [NOT NULL],
           email: STRING [UNIQUE, NOT NULL],
           password: STRING [NOT NULL],
           age: INTEGER [NOT NULL],
           gender: STRING [NOT NULL],
           phone: STRING [NOT NULL],
           join_date: TIMESTAMP [NOT NULL, DEFAULT NOW()],
           status: STRING [NOT NULL, DEFAULT 'active'])
```

Constraints:

- PRIMARY KEY: member_id
- UNIQUE: email
- NOT NULL: all attributes
- DEFAULT: join_date = CURRENT_TIMESTAMP, status = 'active'

3.2 TRAINERS Table

```
TRAINERS (
    trainer_id,
    name,
    email,
    password,
    specialization
)
```

Formal Notation:

```
TRAINERS = (trainer_id: STRING [PK],
            name: STRING [NOT NULL],
            email: STRING [UNIQUE, NOT NULL],
            password: STRING [NOT NULL],
            specialization: STRING [NOT NULL])
```

Constraints:

- PRIMARY KEY: trainer_id
- UNIQUE: email
- NOT NULL: all attributes

3.3 WORKOUT_PLANS Table

```
WORKOUT_PLANS (
    plan_id,
    member_id,
    trainer_id,
    plan_details,
    created_at
)
```

Formal Notation:

```
WORKOUT_PLANS = (plan_id: STRING [PK],
                 member_id: STRING [FK → MEMBERS.member_id, NOT NULL],
                 trainer_id: STRING [FK → TRAINERS.trainer_id, NOT NULL],
                 plan_details: TEXT [NOT NULL],
                 created_at: TIMESTAMP [NOT NULL, DEFAULT NOW()])
```

Constraints:

- PRIMARY KEY: plan_id
- FOREIGN KEY: member_id REFERENCES MEMBERS(member_id) ON DELETE CASCADE
- FOREIGN KEY: trainer_id REFERENCES TRAINERS(trainer_id) ON DELETE CASCADE
- NOT NULL: all attributes
- DEFAULT: created_at = CURRENT_TIMESTAMP

3.4 DIET_PLANS Table

```
DIET_PLANS (
    diet_id,
    member_id,
    trainer_id,
    diet_details,
    created_at
)
```

Formal Notation:

```
DIET_PLANS = (diet_id: STRING [PK],
              member_id: STRING [FK → MEMBERS.member_id, NOT NULL],
              trainer_id: STRING [FK → TRAINERS.trainer_id, NOT NULL],
              diet_details: TEXT [NOT NULL],
              created_at: TIMESTAMP [NOT NULL, DEFAULT NOW()])
```

Constraints:

- PRIMARY KEY: diet_id
- FOREIGN KEY: member_id REFERENCES MEMBERS(member_id) ON DELETE CASCADE
- FOREIGN KEY: trainer_id REFERENCES TRAINERS(trainer_id) ON DELETE CASCADE
- NOT NULL: all attributes
- DEFAULT: created_at = CURRENT_TIMESTAMP

3.5 ATTENDANCES Table

```
ATTENDANCES (
    attendance_id,
    member_id,
    date,
    status
)
```

Formal Notation:

```
ATTENDANCES = (attendance_id: STRING [PK],
               member_id: STRING [FK → MEMBERS.member_id, NOT NULL],
               date: TIMESTAMP [NOT NULL, DEFAULT NOW()],
               status: STRING [NOT NULL])
```

Constraints:

- PRIMARY KEY: attendance_id
- FOREIGN KEY: member_id REFERENCES MEMBERS(member_id) ON DELETE CASCADE
- NOT NULL: all attributes
- DEFAULT: date = CURRENT_TIMESTAMP

3.6 PROGRESS Table

```
PROGRESS (
    progress_id,
    member_id,
    trainer_id,
    weight,
    body_fat,
    muscle_mass,
    notes,
```

```
    updated_at
)
```

Formal Notation:

```
PROGRESS = (progress_id: STRING [PK],
            member_id: STRING [FK → MEMBERS.member_id, NOT NULL],
            trainer_id: STRING [FK → TRAINERS.trainer_id, NOT NULL],
            weight: FLOAT [NOT NULL],
            body_fat: FLOAT [NOT NULL],
            muscle_mass: FLOAT [NOT NULL],
            notes: TEXT [NULLABLE],
            updated_at: TIMESTAMP [NOT NULL, DEFAULT NOW()])
```

Constraints:

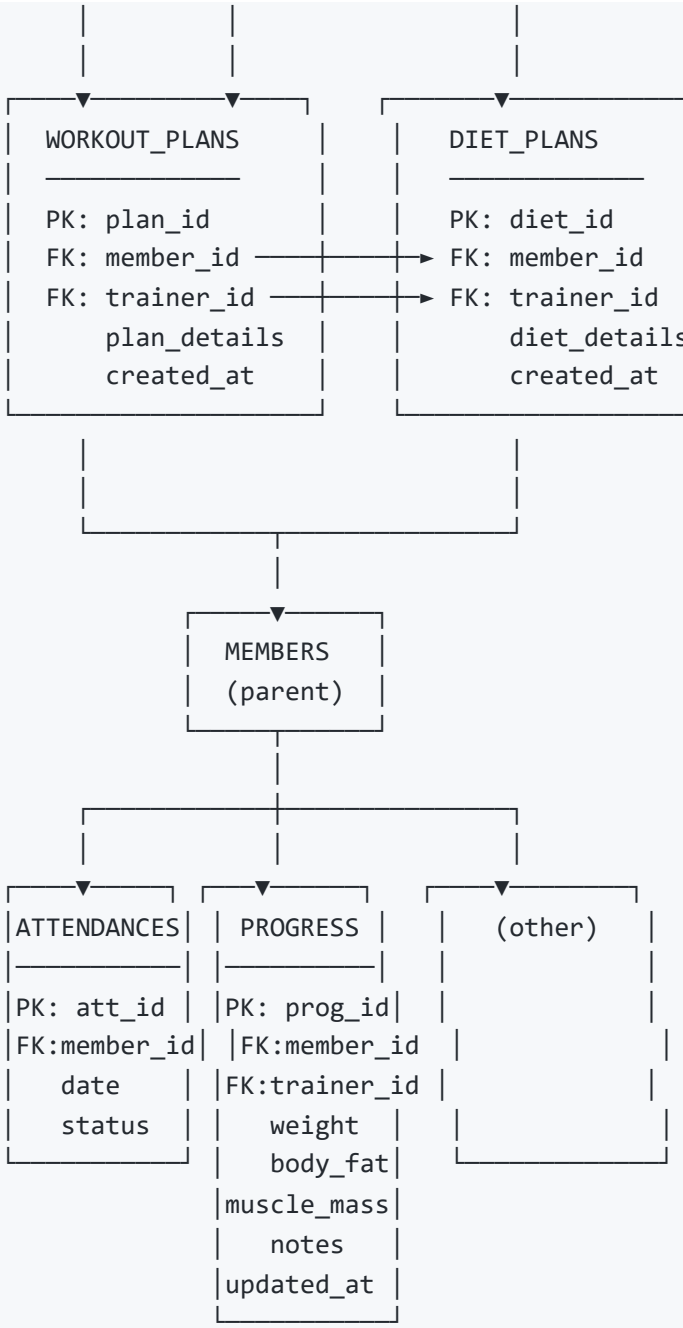
- PRIMARY KEY: progress_id
- FOREIGN KEY: member_id REFERENCES MEMBERS(member_id) ON DELETE CASCADE
- FOREIGN KEY: trainer_id REFERENCES TRAINERS(trainer_id) ON DELETE CASCADE
- NOT NULL: all attributes except notes
- DEFAULT: updated_at = CURRENT_TIMESTAMP

---

4. COMPLETE SCHEMA WITH RELATIONSHIPS

```
┌──────────────────────────────────────────────────────────────────────┐
│                    RELATIONAL SCHEMA DIAGRAM                         │
└──────────────────────────────────────────────────────────────────────┘

MEMBERS                          TRAINERS
_____________                    _____________
PK: member_id                    PK: trainer_id
    name                             name
    email (UNIQUE)                   email (UNIQUE)
    password                         password
    age                              specialization
    gender
    phone
    join_date
    status
    │                                 │
    │                                 │
    └──────┬──────────────────────────┘
           │                          │
           │                          │
           │                          │
           │          ┌───────────────┤
```



---

5. FUNCTIONAL DEPENDENCIES

5.1 MEMBERS

```
FD1: member_id → name, email, password, age, gender, phone, join_date, status
FD2: email → member_id, name, password, age, gender, phone, join_date, status
```

Candidate Keys: {member_id}, {email}
Primary Key: member_id
Alternate Key: email

5.2 TRAINERS

```
FD1: trainer_id → name, email, password, specialization
FD2: email → trainer_id, name, password, specialization
```

Candidate Keys: {trainer_id}, {email}
Primary Key: trainer_id
Alternate Key: email

5.3 WORKOUT_PLANS

```
FD1: plan_id → member_id, trainer_id, plan_details, created_at
```

Candidate Keys: {plan_id}
Primary Key: plan_id

5.4 DIET_PLANS

```
FD1: diet_id → member_id, trainer_id, diet_details, created_at
```

Candidate Keys: {diet_id}
Primary Key: diet_id

5.5 ATTENDANCES

```
FD1: attendance_id → member_id, date, status
```

Candidate Keys: {attendance_id}
Primary Key: attendance_id

5.6 PROGRESS

```
FD1: progress_id → member_id, trainer_id, weight, body_fat, muscle_mass, notes, updated_at
```

Candidate Keys: {progress_id}
Primary Key: progress_id

---

6. NORMALIZATION ANALYSIS

6.1 First Normal Form (1NF)

✅ **All tables are in 1NF**

- All attributes contain atomic values
- No repeating groups
- Each column contains values of a single type
- Each column has a unique name
- Order of rows/columns doesn't matter

6.2 Second Normal Form (2NF)

✅ **All tables are in 2NF**

- All tables are in 1NF
- No partial dependencies (all non-key attributes fully depend on the entire primary key)
- All tables have single-attribute primary keys (UUID), so no partial dependencies exist

6.3 Third Normal Form (3NF)

✅ **All tables are in 3NF**

- All tables are in 2NF
- No transitive dependencies (non-key attributes don't depend on other non-key attributes)

Analysis:

- MEMBERS: No transitive dependencies (age doesn't determine gender, etc.)
- TRAINERS: Specialization is directly dependent on trainer_id
- WORKOUT_PLANS: plan_details depends only on plan_id
- DIET_PLANS: diet_details depends only on diet_id
- ATTENDANCES: date and status are independent attributes
- PROGRESS: weight, body_fat, muscle_mass are independent metrics

6.4 Boyce-Codd Normal Form (BCNF)

✅ **All tables are in BCNF**

- All tables are in 3NF
- For every functional dependency X → Y, X is a superkey

Verification:

- All functional dependencies have primary keys or candidate keys on the left side
- No anomalies detected

---

7. REFERENTIAL INTEGRITY CONSTRAINTS

7.1 Foreign Key Constraints

```
-- WORKOUT_PLANS → MEMBERS
FOREIGN KEY (member_id)
    REFERENCES MEMBERS(member_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE

-- WORKOUT_PLANS → TRAINERS
FOREIGN KEY (trainer_id)
    REFERENCES TRAINERS(trainer_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE

-- DIET_PLANS → MEMBERS
FOREIGN KEY (member_id)
    REFERENCES MEMBERS(member_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE

-- DIET_PLANS → TRAINERS
FOREIGN KEY (trainer_id)
    REFERENCES TRAINERS(trainer_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE

-- ATTENDANCES → MEMBERS
FOREIGN KEY (member_id)
    REFERENCES MEMBERS(member_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE

-- PROGRESS → MEMBERS
FOREIGN KEY (member_id)
    REFERENCES MEMBERS(member_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
```

```
-- PROGRESS → TRAINERS
FOREIGN KEY (trainer_id)
    REFERENCES TRAINERS(trainer_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
```

7.2 CASCADE Rules

ON DELETE CASCADE:

- When a member is deleted, all related workout plans, diet plans, attendances, and progress records are automatically deleted
- When a trainer is deleted, all related workout plans, diet plans, and progress records are automatically deleted

ON UPDATE CASCADE:

- When a member_id or trainer_id is updated, all foreign key references are automatically updated

---

8. DOMAIN CONSTRAINTS

8.1 Data Types

Table	Attribute	Data Type	Constraints
MEMBERS	member_id	UUID	PRIMARY KEY
	name	VARCHAR(255)	NOT NULL
	email	VARCHAR(255)	NOT NULL, UNIQUE
	password	VARCHAR(255)	NOT NULL (hashed)
	age	INTEGER	NOT NULL, > 0
	gender	VARCHAR(50)	NOT NULL
	phone	VARCHAR(20)	NOT NULL
	join_date	TIMESTAMP	NOT NULL, DEFAULT NOW()
	status	VARCHAR(50)	NOT NULL, DEFAULT 'active'
TRAINERS	trainer_id	UUID	PRIMARY KEY
	name	VARCHAR(255)	NOT NULL

Table	Attribute	Data Type	Constraints
	email	VARCHAR(255)	NOT NULL, UNIQUE
	password	VARCHAR(255)	NOT NULL (hashed)
	specialization	VARCHAR(255)	NOT NULL
WORKOUT_PLANS	plan_id	UUID	PRIMARY KEY
	member_id	UUID	FOREIGN KEY, NOT NULL
	trainer_id	UUID	FOREIGN KEY, NOT NULL
	plan_details	TEXT	NOT NULL
	created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()
DIET_PLANS	diet_id	UUID	PRIMARY KEY
	member_id	UUID	FOREIGN KEY, NOT NULL
	trainer_id	UUID	FOREIGN KEY, NOT NULL
	diet_details	TEXT	NOT NULL
	created_at	TIMESTAMP	NOT NULL, DEFAULT NOW()
ATTENDANCES	attendance_id	UUID	PRIMARY KEY
	member_id	UUID	FOREIGN KEY, NOT NULL
	date	TIMESTAMP	NOT NULL, DEFAULT NOW()
	status	VARCHAR(50)	NOT NULL
PROGRESS	progress_id	UUID	PRIMARY KEY
	member_id	UUID	FOREIGN KEY, NOT NULL
	trainer_id	UUID	FOREIGN KEY, NOT NULL
	weight	FLOAT	NOT NULL, > 0
	body_fat	FLOAT	NOT NULL, >= 0, <= 100
	muscle_mass	FLOAT	NOT NULL, > 0
	notes	TEXT	NULLABLE
	updated_at	TIMESTAMP	NOT NULL, DEFAULT NOW()

9. CARDINALITY AND PARTICIPATION

Parent Table	Child Table	Cardinality	Parent Participation	Child Participation
MEMBERS	WORKOUT_PLANS	1:N	Partial (0..*)	Total (1..1)
MEMBERS	DIET_PLANS	1:N	Partial (0..*)	Total (1..1)
MEMBERS	ATTENDANCES	1:N	Partial (0..*)	Total (1..1)
MEMBERS	PROGRESS	1:N	Partial (0..*)	Total (1..1)
TRAINERS	WORKOUT_PLANS	1:N	Partial (0..*)	Total (1..1)
TRAINERS	DIET_PLANS	1:N	Partial (0..*)	Total (1..1)
TRAINERS	PROGRESS	1:N	Partial (0..*)	Total (1..1)

10. SQL DDL STATEMENTS

10.1 Create Tables

```sql
-- Create MEMBERS table
CREATE TABLE members (
    member_id TEXT PRIMARY KEY,
    name TEXT NOT NULL,
    email TEXT UNIQUE NOT NULL,
    password TEXT NOT NULL,
    age INTEGER NOT NULL,
    gender TEXT NOT NULL,
    phone TEXT NOT NULL,
    join_date TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP,
    status TEXT NOT NULL DEFAULT 'active'
);

-- Create TRAINERS table
CREATE TABLE trainers (
    trainer_id TEXT PRIMARY KEY,
    name TEXT NOT NULL,
    email TEXT UNIQUE NOT NULL,
    password TEXT NOT NULL,
    specialization TEXT NOT NULL
);

-- Create WORKOUT_PLANS table
CREATE TABLE workout_plans (
    plan_id TEXT PRIMARY KEY,
```

```
    member_id TEXT NOT NULL,
    trainer_id TEXT NOT NULL,
    plan_details TEXT NOT NULL,
    created_at TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (member_id) REFERENCES members(member_id) ON DELETE CASCADE ON UPDATE CASC
    FOREIGN KEY (trainer_id) REFERENCES trainers(trainer_id) ON DELETE CASCADE ON UPDATE C
);

-- Create DIET_PLANS table
CREATE TABLE diet_plans (
    diet_id TEXT PRIMARY KEY,
    member_id TEXT NOT NULL,
    trainer_id TEXT NOT NULL,
    diet_details TEXT NOT NULL,
    created_at TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (member_id) REFERENCES members(member_id) ON DELETE CASCADE ON UPDATE CASC
    FOREIGN KEY (trainer_id) REFERENCES trainers(trainer_id) ON DELETE CASCADE ON UPDATE C
);

-- Create ATTENDANCES table
CREATE TABLE attendances (
    attendance_id TEXT PRIMARY KEY,
    member_id TEXT NOT NULL,
    date TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP,
    status TEXT NOT NULL,
    FOREIGN KEY (member_id) REFERENCES members(member_id) ON DELETE CASCADE ON UPDATE CASC
);

-- Create PROGRESS table
CREATE TABLE progress (
    progress_id TEXT PRIMARY KEY,
    member_id TEXT NOT NULL,
    trainer_id TEXT NOT NULL,
    weight DOUBLE PRECISION NOT NULL,
    body_fat DOUBLE PRECISION NOT NULL,
    muscle_mass DOUBLE PRECISION NOT NULL,
    notes TEXT,
    updated_at TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (member_id) REFERENCES members(member_id) ON DELETE CASCADE ON UPDATE CASC
    FOREIGN KEY (trainer_id) REFERENCES trainers(trainer_id) ON DELETE CASCADE ON UPDATE C
);
```

10.2 Create Indexes

```
-- Create indexes for foreign keys
CREATE INDEX idx_workout_plans_member_id ON workout_plans(member_id);
CREATE INDEX idx_workout_plans_trainer_id ON workout_plans(trainer_id);
CREATE INDEX idx_diet_plans_member_id ON diet_plans(member_id);
CREATE INDEX idx_diet_plans_trainer_id ON diet_plans(trainer_id);
CREATE INDEX idx_attendances_member_id ON attendances(member_id);
```

```sql
CREATE INDEX idx_progress_member_id ON progress(member_id);
CREATE INDEX idx_progress_trainer_id ON progress(trainer_id);

-- Create indexes for commonly queried columns
CREATE INDEX idx_members_email ON members(email);
CREATE INDEX idx_trainers_email ON trainers(email);
CREATE INDEX idx_members_status ON members(status);
CREATE INDEX idx_attendances_date ON attendances(date);
CREATE INDEX idx_progress_updated_at ON progress(updated_at);
```

---

11. RELATIONAL ALGEBRA QUERIES

11.1 Get all members

```
π member_id, name, email, age, gender, phone, join_date, status (MEMBERS)
```

11.2 Get all workout plans for a specific member

```
π plan_id, plan_details, created_at (
    σ member_id = 'xyz' (WORKOUT_PLANS)
)
```

11.3 Get all members with their workout plans (JOIN)

```
π MEMBERS.name, WORKOUT_PLANS.plan_details (
    MEMBERS ⋈ member_id WORKOUT_PLANS
)
```

11.4 Get trainer information for a member's workout plan

```
π TRAINERS.name, TRAINERS.specialization (
    (σ WORKOUT_PLANS.member_id = 'xyz' (WORKOUT_PLANS))
    ⋈ trainer_id TRAINERS
)
```

11.5 Count attendance records for each member

```
member_id ℱ COUNT(attendance_id) (ATTENDANCES)
```

11.6 Get latest progress for a member

```
π weight, body_fat, muscle_mass, notes, updated_at (
    σ member_id = 'xyz' (
        PROGRESS
    )
) ORDER BY updated_at DESC LIMIT 1
```