FitTrack Pro: ER to Relational Mapping Approach

HW2: Logical Database Schema

Course: Databases Project 2025
Assignment: 2
Deadline: September 25, 2025, 23:59
Team Members: Aleksandr Zinovev, Siwoo Lee, Arslan Ahmet Berk

---

Mapping Strategy Overview

This document describes our approach to converting the FitTrack Pro ER diagram into a relational database schema. We followed standard ER-to-relational mapping rules while making specific design decisions for ISA hierarchies and relationship sets.

ISA Hierarchy Mapping Decisions

Approach Used: Separate Relation per Entity Set (Alt 1)

We chose the **separate relation per entity set** approach for all three ISA hierarchies rather than other alternatives like relations only for subclasses or one big relation.

Rationale: - **Data Integrity**: Each subclass has distinct attributes that are always populated - **Query Performance**: Avoids NULL values and reduces table width - **Maintainability**: Clear separation of concerns for different user types - **Extensibility**: Easy to add new subclasses without affecting existing tables

ISA Hierarchy 1: User Specialization

Mapping Decision:

```
user (superclass) → user table
individual_user (subclass) → individual_user table
gym_member (subclass) → gym_member table
staff (subclass) → staff table
```

Alternative Considered: One big relation (Alt 3) with discriminator column **Why Rejected:** Would create many NULL values since user types have very different attributes

ISA Hierarchy 2: Staff Specialization

Mapping Decision:

```
staff (superclass) → staff table (already created from User hierarchy)
trainer (subclass) → trainer table
manager (subclass) → manager table
receptionist (subclass) → receptionist table
```

Alternative Considered: Merging all staff types into staff table **Why Rejected:** Trainer certifications, manager access levels, and receptionist schedules are too specialized

ISA Hierarchy 3: Exercise Categories

Mapping Decision:

```
exercise (superclass) → exercise table
cardio (subclass) → cardio table
strength (subclass) → strength table
flexibility (subclass) → flexibility table
```

Alternative Considered: Single exercise table with category-specific JSON fields **Why Rejected:** Loses type safety and makes queries more complex

Entity Set Mapping

Strong Entities

All strong entities mapped directly to tables with primary keys: - `user` → `user_id` (AUTO_INCREMENT) - `gym` → `gym_id` (AUTO_INCREMENT) - `workout` → `workout_id` (AUTO_INCREMENT) - `exercise` → `exercise_id` (AUTO_INCREMENT) - `class` → `class_id` (AUTO_INCREMENT) - `equipment` → `equipment_id` (AUTO_INCREMENT)

Weak Entities

- `progress_tracking` → Uses composite key (user_id, date) but added surrogate key `tracking_id` for simplicity

Relationship Set Mapping

Many-to-Many Relationships

1. Workout ↔ Exercise (M:N)

Mapping: Junction table `workout_exercise`

```
workout_exercise (
    workout_id (FK),
    exercise_id (FK),
    sets, reps, weight, duration, rest_time
)
```

Rationale: Stores workout-specific exercise data (sets, reps, weight)

2. Gym Member ↔ Class (M:N)

Mapping: Junction table `class_booking`

```
class_booking (
    booking_id (PK),
    class_id (FK),
    member_id (FK),
    booking_date, status
)
```

Rationale: Tracks booking history and status changes

One-to-Many Relationships

Implemented using foreign keys in the "many" side: - `user` → `workout` (user_id FK in workout) - `gym` → `gym_member` (gym_id FK in gym_member) - `gym` → `staff` (gym_id FK in staff) - `gym` → `class` (gym_id FK in class) - `gym` → `equipment` (gym_id FK in equipment) - `trainer` → `class` (trainer_id FK in class)

Constraint Implementation

Domain Constraints

- **ENUM types** for categorical data (gender, membership_type, status)
- **CHECK constraints** for data validation (positive values, date ranges)
- **VARCHAR lengths** appropriate for each field

Key Constraints

- **Primary keys** for all entities
- **Unique constraints** for business keys (username, email, membership_id)
- **Composite primary keys** for junction tables

Referential Integrity

- **FOREIGN KEY constraints** with appropriate CASCADE/RESTRICT actions
- **ON DELETE CASCADE** for dependent entities
- **ON DELETE RESTRICT** for referenced entities that shouldn't be deleted

Business Rules

- **Unique booking constraint** prevents double-booking same class
- **Date validation** ensures end_date > start_date
- **Realistic value ranges** for weight, body fat percentage
- **Email format validation** using CHECK constraints

Schema Statistics

Requirement Compliance: - **Entity Sets:** 12 tables (Required: 6+ for 3-person team) ✅ - **Relationship Sets:** 4 explicit relationships (Required: 3+ for 3-person team) ✅
- **ISA Hierarchies:** 3 hierarchies (Required: 3 for 3-person team) ✅

Table Count: - **Superclass tables:** 3 (user, exercise, staff) - **Subclass tables:** 7 (individual_user, gym_member, trainer, manager, receptionist, cardio, strength, flexibility) - **Regular entity tables:** 6 (gym, workout, class, equipment, progress_tracking) - **Junction tables:** 2 (workout_exercise, class_booking) - **Total:** 18 tables

Performance Considerations

Indexing Strategy

- **Primary key indexes** (automatic)
- **Foreign key indexes** for join performance
- **Composite indexes** for common query patterns (user_id, date)
- **Unique indexes** for business constraints

Query Optimization

- **Normalized design** reduces data redundancy
- **Appropriate data types** minimize storage overhead
- **Selective indexes** on frequently queried columns

Testing Strategy

Schema Validation

1. **CREATE TABLE statements** execute without errors
2. **Constraint validation** prevents invalid data insertion
3. **Foreign key relationships** maintain referential integrity
4. **ISA hierarchy queries** work correctly across inheritance levels

Sample Data Testing

- Insert test data for each entity type
- Verify constraint enforcement
- Test cascade delete behavior
- Validate complex queries across multiple tables
