**AI-диетолог 3.0**

**Use Case**: составить план питания

На основе Use Case выделим основные сущности и их атрибуты:

1. Спортсмен (Athlete) - основной актор
2. План питания (MealPlan) - результат работы системы
3. Прием пищи (Meal) - завтрак/обед/ужин в плане
4. Тренировка (Workout) - режим тренировок спортсмена
5. Физическая активность (Activity) - активность вне тренировок

Теперь построим логическую модель данных с PK/FK без связей многие-ко-многим.

**Логическая модель данных**

1. **Athlete** (Спортсмен)
    - athlete_id (PK)
    - name
    - gender
    - height
    - current_weight
    - target_weight_category
    - competition_date
2. **MealPlan** (План питания)
    - plan_id (PK)
    - athlete_id (FK → Athlete)
    - total_calories
    - total_proteins
    - total_fats
    - total_carbs
    - start_date
    - end_date

3. **Meal** (Прием пищи)
    - meal_id (PK)
    - plan_id (FK → MealPlan)
    - meal_type (breakfast/lunch/dinner)
    - calories
    - proteins
    - fats
    - carbs
    - description
4. **Workout** (Тренировка)
    - workout_id (PK)
    - athlete_id (FK → Athlete)
    - sessions_per_week
    - exercises
    - equipment_weight
    - reps
    - sets
5. **Activity** (Физическая активность)
    - activity_id (PK)
    - athlete_id (FK → Athlete)
    - activity_type
    - duration_minutes
    - frequency_per_week

## Физическая модель данных (PostgreSQL)

```sql
-- SQL скрипт создания физической модели и заполнения тестовыми данными
-- Для СУБД PostgreSQL
-- Версия 1.2 (исправлена ошибка с подзапросами, добавлены все таблицы)
-- Дата создания: 2025-08-12

-- Удаление существующих таблиц (для пересоздания)
DROP TABLE IF EXISTS activities CASCADE;
DROP TABLE IF EXISTS workouts CASCADE;
DROP TABLE IF EXISTS meals CASCADE;
DROP TABLE IF EXISTS meal_plans CASCADE;
DROP TABLE IF EXISTS athletes CASCADE;

-- Создание таблицы спортсменов
CREATE TABLE athletes (
    athlete_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    gender CHAR(1) CHECK (gender IN ('M', 'F')) NOT NULL,
    height DECIMAL(5,2) NOT NULL CHECK (height > 0),
    current_weight DECIMAL(5,2) NOT NULL CHECK (current_weight > 0),
    target_weight_category DECIMAL(5,2) NOT NULL CHECK (target_weight_category >
0),
    competition_date DATE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT weight_check CHECK (ABS(current_weight - target_weight_category) <=
current_weight * 0.05)
);

-- Создание таблицы планов питания
CREATE TABLE meal_plans (
    plan_id SERIAL PRIMARY KEY,
    athlete_id INTEGER NOT NULL REFERENCES athletes(athlete_id) ON DELETE CASCADE,
    total_calories INTEGER NOT NULL CHECK (total_calories > 0),
    total_proteins DECIMAL(5,2) NOT NULL CHECK (total_proteins >= 0),
    total_fats DECIMAL(5,2) NOT NULL CHECK (total_fats >= 0),
    total_carbs DECIMAL(5,2) NOT NULL CHECK (total_carbs >= 0),
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT date_check CHECK (end_date > start_date)
);

-- Создание триггерной функции для проверки даты соревнований
CREATE OR REPLACE FUNCTION check_competition_date()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.start_date > (SELECT competition_date FROM athletes WHERE athlete_id =
NEW.athlete_id) - INTERVAL '7 days' THEN
        RAISE EXCEPTION 'План должен начинаться не позднее чем за неделю до
соревнований';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```sql
-- Создание триггера
CREATE TRIGGER meal_plan_date_check
BEFORE INSERT OR UPDATE ON meal_plans
FOR EACH ROW EXECUTE FUNCTION check_competition_date();

-- Создание таблицы приемов пищи
CREATE TABLE meals (
    meal_id SERIAL PRIMARY KEY,
    plan_id INTEGER NOT NULL REFERENCES meal_plans(plan_id) ON DELETE CASCADE,
    meal_type VARCHAR(10) NOT NULL CHECK (meal_type IN ('breakfast', 'lunch',
'dinner')),
    calories INTEGER NOT NULL CHECK (calories > 0),
    proteins DECIMAL(5,2) NOT NULL CHECK (proteins >= 0),
    fats DECIMAL(5,2) NOT NULL CHECK (fats >= 0),
    carbs DECIMAL(5,2) NOT NULL CHECK (carbs >= 0),
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Создание таблицы тренировок
CREATE TABLE workouts (
    workout_id SERIAL PRIMARY KEY,
    athlete_id INTEGER NOT NULL REFERENCES athletes(athlete_id) ON DELETE CASCADE,
    sessions_per_week INTEGER NOT NULL CHECK (sessions_per_week BETWEEN 1 AND 14),
    exercises TEXT NOT NULL,
    equipment_weight DECIMAL(5,2) CHECK (equipment_weight >= 0),
    reps INTEGER CHECK (reps > 0),
    sets INTEGER CHECK (sets > 0),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Создание таблицы физической активности
CREATE TABLE activities (
    activity_id SERIAL PRIMARY KEY,
    athlete_id INTEGER NOT NULL REFERENCES athletes(athlete_id) ON DELETE CASCADE,
    activity_type VARCHAR(50) NOT NULL,
    duration_minutes INTEGER NOT NULL CHECK (duration_minutes > 0),
    frequency_per_week INTEGER NOT NULL CHECK (frequency_per_week BETWEEN 1 AND
7),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

## SQL скрипт с тестовыми данными

```sql
-- Заполнение таблицы athletes тестовыми данными
INSERT INTO athletes (name, gender, height, current_weight,
target_weight_category, competition_date) VALUES
('Иван Петров', 'M', 180.5, 85.0, 83.0, '2025-09-01'),
('Мария Сидорова', 'F', 165.0, 60.0, 58.0, '2025-09-15'),
('Алексей Иванов', 'M', 175.0, 90.0, 87.0, '2025-10-01'),
('Елена Кузнецова', 'F', 170.0, 65.0, 63.0, '2025-10-15'),
('Дмитрий Смирнов', 'M', 185.0, 95.0, 92.0, '2025-11-01');

-- Заполнение таблицы meal_plans тестовыми данными
INSERT INTO meal_plans (athlete_id, total_calories, total_proteins, total_fats,
total_carbs, start_date, end_date) VALUES
(1, 2500, 180.0, 70.0, 250.0, '2025-08-25', '2025-08-31'),
(2, 2000, 150.0, 60.0, 200.0, '2025-09-08', '2025-09-14'),
(3, 2800, 200.0, 80.0, 280.0, '2025-09-24', '2025-09-30'),
(4, 2200, 160.0, 65.0, 220.0, '2025-10-08', '2025-10-14'),
(5, 3000, 220.0, 90.0, 300.0, '2025-10-25', '2025-10-31');

-- Заполнение таблицы meals тестовыми данными
INSERT INTO meals (plan_id, meal_type, calories, proteins, fats, carbs,
description) VALUES
(1, 'breakfast', 600, 40.0, 20.0, 70.0, 'Овсянка с ягодами и орехами, яйца'),
(1, 'lunch', 900, 70.0, 30.0, 90.0, 'Гречка с куриной грудкой, овощной салат'),
(1, 'dinner', 1000, 70.0, 20.0, 90.0, 'Рис с рыбой, овощи на пару'),
(2, 'breakfast', 500, 35.0, 15.0, 60.0, 'Творог с фруктами, тост с авокадо'),
(2, 'lunch', 800, 60.0, 25.0, 80.0, 'Киноа с индейкой, салат из свежих овощей'),
(2, 'dinner', 700, 55.0, 20.0, 60.0, 'Омлет с овощами, цельнозерновой хлеб'),
(3, 'breakfast', 700, 50.0, 25.0, 80.0, 'Яичница с беконом, тосты, фрукты'),
(3, 'lunch', 1000, 80.0, 35.0, 100.0, 'Макароны из твердых сортов с говядиной'),
(3, 'dinner', 1100, 70.0, 20.0, 100.0, 'Картофель с рыбой, овощной салат'),
(4, 'breakfast', 550, 40.0, 20.0, 65.0, 'Смузи из протеина, банан, орехи'),
(4, 'lunch', 850, 65.0, 25.0, 85.0, 'Рис с курицей и овощами'),
(4, 'dinner', 800, 55.0, 20.0, 70.0, 'Творожная запеканка, ягоды'),
(5, 'breakfast', 800, 60.0, 30.0, 90.0, 'Овсянка с протеином, орехи, фрукты'),
(5, 'lunch', 1200, 90.0, 40.0, 120.0, 'Гречка с говядиной, овощной салат'),
(5, 'dinner', 1000, 70.0, 20.0, 90.0, 'Куриная грудка с овощами и картофелем');

-- Заполнение таблицы workouts тестовыми данными
INSERT INTO workouts (athlete_id, sessions_per_week, exercises, equipment_weight,
reps, sets) VALUES
(1, 5, 'Жим лежа, приседания, становая тяга', 80.0, 8, 4),
(2, 4, 'Приседания, выпады, жим гантелей', 15.0, 12, 3),
(3, 6, 'Жим штанги, подтягивания, отжимания', 100.0, 6, 5),
(4, 3, 'Планка, скручивания, бег', NULL, NULL, NULL),
(5, 5, 'Становая тяга, жим ногами, подтягивания', 120.0, 5, 5);

-- Заполнение таблицы activities тестовыми данными
INSERT INTO activities (athlete_id, activity_type, duration_minutes,
frequency_per_week) VALUES
(1, 'Плавание', 60, 2),
(2, 'Йога', 45, 3),
(3, 'Велоспорт', 90, 2),
(4, 'Пешие прогулки', 120, 4),
(5, 'Кроссфит', 60, 3);
```

Полный скрипт доступен по ссылке: https://drive.google.com/file/d/1h-h8Cpi3F0gAlBWrJzKCl_-3EzCsafhj/view?usp=sharing

**Нормализация до 3NF**

Модель соответствует 3NF, так как:

1. Все атрибуты атомарны
2. Нет частичных зависимостей от составного ключа (все PK простые)
3. Нет транзитивных зависимостей - все неключевые атрибуты зависят только от PK

Дополнительные проверки реализованы через CHECK-ограничения:

- Проверка разницы веса (не более 5%)
- Проверка даты начала плана (не позднее чем за неделю до соревнований)
- Проверка типов данных и допустимых значений