Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина «Модели данных и системы управления базами данных»

ОТЧЕТ
к лабораторной работе №3
на тему:
«РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ИСПОЛЬЗОВАНИЕМ БАЗЫ ДАННЫХ»
БГУИР 6-05-0612-02

Выполнил студент группы 353503
КОХАН Артём Игоревич
РЯЗАНЦЕВ Алексей Владимирович

_____
(дата, подпись студента)

Проверил ассистент каф. информатики
КОЖЕМЯКО Евгения Александровна

_____
(дата, подпись преподавателя)

Минск 2025

1 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Разработка физической структуры базы данных. Результат работы: полный набор SQL-запросов по созданию схемы данных в СУБД.

2 КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ

Схема базы данных представляет собой её структуру, включая таблицы, столбцы, типы данных, ограничения и связи между таблицами. Основным инструментом для создания структуры базы данных в SQL является DDL (Data Definition Language) CREATE TABLE. Оно позволяет определить таблицу, указав её имя, перечислив столбцы с их типами данных и наложив ограничения (CONSTRAINTS).

В PostgreSQL используются следующие основные типы данных: строковые (VARCHAR(n) для строк переменной длины с ограничением, TEXT для строк неограниченной длины), числовые (SMALLINT, INTEGER для целых чисел, BIGINT для больших целых чисел, REAL и DOUBLE PRECISION для чисел с плавающей точкой, NUMERIC(p, s) для точных десятичных чисел), дата и время (DATE для дат, TIME для времени, TIMESTAMP для даты и времени), логические (BOOLEAN), бинарные данные (BYTEA). Особенностью PostgreSQL является поддержка пользовательских типов данных через CREATE TYPE, включая перечисления (ENUM), составные типы и домены.

Связи между таблицами, обеспечивающие целостность данных, устанавливаются с помощью ограничений FOREIGN KEY, которые ссылаются на первичные или уникальные ключи в других таблицах. PostgreSQL поддерживает различные уровни ссылочной целостности через ON DELETE и ON UPDATE с опциями CASCADE, SET NULL, SET DEFAULT, RESTRICT. Также доступны сложные ограничения CHECK для проверки бизнес-логики на уровне базы данных.

3 ВЫПОЛНЕНИЕ РАБОТЫ

Представленный код (см. листинг 3.1) на языке SQL представляет собой законченный скрипт для создания реляционной базы данных, предназначенной для управления фитнес-клубом. Структура базы данных включает в себя десять взаимосвязанных таблиц, которые охватывают основные сущности предметной области: управление пользователями, ролями, абонементами, тренировками, шкафчиками и системой журналирования. Каждая таблица спроектирована с учетом целостности данных: определены первичные и внешние ключи для установления связей, а также добавлены различные ограничения (CHECK, NOT NULL, UNIQUE), которые гарантируют корректность и непротиворечивость хранимой информации. Для автоматической генерации уникальных идентификаторов записей в таблицах используется тип SERIAL. Особенностью данной реализации является использование пользовательских ENUM-типов PostgreSQL для стандартизации допустимых значений в таких атрибутах как роли пользователей, статусы шкафчиков, статусы тренировок и уровни важности логов. Политики обработки зависимых записей при удалении (ON DELETE CASCADE и ON DELETE RESTRICT) обеспечивают поддержание целостности связей в базе данных при выполнении операций удаления. Скрипт также включает предзаполнение таблицы ролей базовыми значениями: администратор, клиент и тренер, что обеспечивает готовность системы к использованию сразу после развертывания.

Листинг 3.1 – Создание физической структуры базы данных

```
DROP SCHEMA IF EXISTS fitness CASCADE;
CREATE SCHEMA fitness;
SET search_path TO fitness;

CREATE TYPE role_name_enum AS ENUM ('admin', 'member', 'coach');
CREATE TYPE gender_enum AS ENUM ('male', 'female', 'unisex');
CREATE TYPE locker_status_enum AS ENUM ('available', 'occupied', 'out of
service');
CREATE TYPE training_status_enum AS ENUM ('free', 'booked');
CREATE TYPE log_severity_enum AS ENUM ('info', 'warning', 'error');

CREATE TABLE role (
                      id SERIAL PRIMARY KEY,
                      name role_name_enum NOT NULL UNIQUE
);

CREATE TABLE person (
                        id SERIAL PRIMARY KEY,
                        role_id INTEGER NOT NULL REFERENCES role(id) ON DELETE
RESTRICT,
                        first_name VARCHAR(100) NOT NULL,
                        last_name VARCHAR(100) NOT NULL,
                        phone VARCHAR(30),
                        email VARCHAR(255) UNIQUE NOT NULL,
                        password VARCHAR(255) NOT NULL,
                        birth_date DATE
);
```

```
CREATE TABLE member_details (
                                person_id   INTEGER   PRIMARY   KEY   REFERENCES
person(id) ON DELETE CASCADE,
                                join_date DATE NOT NULL
);

CREATE TABLE coach_details (
                               person_id   INTEGER   PRIMARY   KEY   REFERENCES
person(id) ON DELETE CASCADE,
                               description VARCHAR(255),
                               experience_years          INTEGER          CHECK
(experience_years >= 0),
                               education VARCHAR(100)
);

CREATE TABLE admin_details (
                               person_id   INTEGER   PRIMARY   KEY   REFERENCES
person(id) ON DELETE CASCADE,
                               position VARCHAR(100),
                               permissions TEXT
);

CREATE TABLE membership (
                            id SERIAL PRIMARY KEY,
                            member_id    INTEGER    NOT    NULL    REFERENCES
member_details(person_id) ON DELETE CASCADE,
                            name VARCHAR(50) NOT NULL,
                            description VARCHAR(255),
                            price DECIMAL(10,2) CHECK (price >= 0),
                            start_date DATE NOT NULL,
                            end_date DATE
);

CREATE TABLE personal_training (
                                    id SERIAL PRIMARY KEY,
                                    coach_id   INTEGER   NOT   NULL   REFERENCES
coach_details(person_id) ON DELETE CASCADE,
                                    member_id   INTEGER   NOT   NULL   REFERENCES
member_details(person_id) ON DELETE CASCADE,
                                    date DATE NOT NULL,
                                    start_time TIME NOT NULL,
                                    end_time TIME NOT NULL,
                                    status   training_status_enum   NOT   NULL
DEFAULT 'free',
                                    price DECIMAL(10,2) CHECK (price >= 0),
                                    notes VARCHAR(255)
);

CREATE TABLE locker (
                        id SERIAL PRIMARY KEY,
                        number INTEGER NOT NULL UNIQUE,
                        gender gender_enum NOT NULL,
                        status locker_status_enum NOT NULL DEFAULT 'available'
);

CREATE TABLE locker_assignment (
                                    id SERIAL PRIMARY KEY,
                                    locker_id   INTEGER   NOT   NULL   REFERENCES
locker(id) ON DELETE CASCADE,
                                    person_id   INTEGER   NOT   NULL   REFERENCES
person(id) ON DELETE CASCADE,
                                    booked_from TIMESTAMP NOT NULL
);
```

```
CREATE TABLE user_log (
                          id SERIAL PRIMARY KEY,
                          person_id INTEGER NOT NULL REFERENCES person(id) ON
DELETE CASCADE,
                          action_type VARCHAR(100) NOT NULL,
                          action_description VARCHAR(255),
                          action_time    TIMESTAMP    NOT    NULL    DEFAULT
CURRENT_TIMESTAMP,
                          ip_address VARCHAR(45),
                          severity log_severity_enum NOT NULL DEFAULT 'info'
);

INSERT INTO role (name)
VALUES ('admin'), ('member'), ('coach');
```

ВЫВОД

В ходе выполнения индивидуального задания была успешно разработана физическая структура базы данных для системы управления фитнес-клубом. Создан полный набор SQL-запросов, реализующих схему данных в СУБД PostgreSQL. Разработанная структура включает десять взаимосвязанных таблиц, охватывающих все основные сущности предметной области: управление пользователями с различными ролями, систему абонементов, организацию персональных тренировок, управление шкафчиками и ведение журнала действий пользователей.

Особенностью реализации стало эффективное использование возможностей PostgreSQL, включая создание пользовательских ENUM-типов для стандартизации допустимых значений в ключевых атрибутах системы. При проектировании таблиц были применены различные ограничения целостности данных: первичные и внешние ключи, ограничения CHECK для проверки бизнес-логики, ограничения UNIQUE для обеспечения уникальности данных. Для автоматической генерации идентификаторов использовался тип SERIAL.

Правильно настроенные политики обработки зависимых записей при удалении (ON DELETE CASCADE и ON DELETE RESTRICT) обеспечивают поддержание целостности связей между таблицами. Начальное заполнение таблицы ролей базовыми значениями позволяет использовать систему после развертывания.

Разработанная физическая структура базы данных полностью соответствует требованиям задания и готова к использованию в реальной системе управления фитнес-клубом, обеспечивая надежное хранение данных и эффективное выполнение операций.