Национальный исследовательский ядерный университет «МИФИ» (Московский Инженерно–Физический Институт)

Кафедра №42 «Криптология и кибербезопасность»

**Лабораторная работа №1-1:**

**«Построение модели данных»**

Антон Гатченко Б22-525

2025 г.

*Используемая рабочая среда:*

* Процессор - AMD Ryzen 5 5600H (laptop), 6c/12t
* Оперативная память – DDR4 16 ГБ
* ОС - Windows 10 Pro 22H2 19045.4780, 64 bit
* SQLite 3.49.1.0

*Неформальное описание предметной области:*

Выбранная область – фитнес-центр.

Клуб предлагает различные спортивные секции (фитнес, бассейн, единоборства). Есть тренеры с разной квалификацией. Клиенты могут приобретать абонементы на определенный период. Требуется учет посещаемости и расписания занятий. При расширении можно будет добавлять данные об акциях, новых секциях и т.п.

Необходимые данные:

* Информация о клиентах (ФИО, контакты, выбранные секции)
* Информация о тренерах (специализация, график работы)
* Абонементы (тип, срок действия, стоимость)
* Расписание занятий
* Посещаемость

*Спецификация таблиц:*

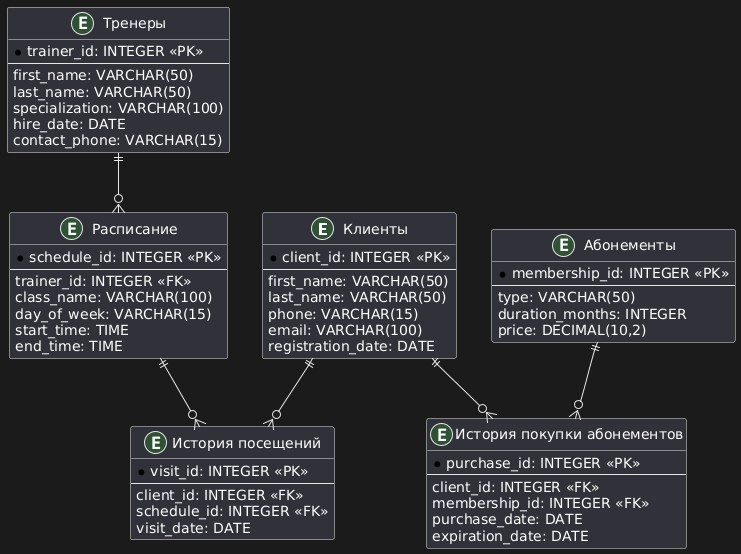


Рисунок 1 - Диаграмма отношения сущностей

*Связи между таблицами:*

* Каждый клиент может купить несколько абонементов (один-ко-многим)
* Каждый абонемент может быть куплен многими клиентами (многие-ко-многим через таблицу покупок)
* Каждый тренер ведет несколько занятий (один-ко-многим)
* Каждое занятие могут посещать многие клиенты (многие-ко-многим через таблицу посещений)
* membership\_id – это номер абонемента, он определяет длительность абонемента, например на 3/6/12 месяцев, и тип абонемента, например обычный, с доступом в бассейн и т.п.
* client\_id - определяет человека.
* purchase\_id - номер покупки, один и тот же человек может купить повторно такой же абонемент, как и в прошлый раз, и здесь пригодится purchase\_id для определения конкретной покупки.
* В случае изменений, таких как изменение цены абонемента или замена одного тренера другим можно добавить новую запись в memberships/schedule, а далее добавить запись в purchases/visits с указанием id нового абонемента/тренера.

*Соответствие третьей нормальной форме:*

* Все неключевые атрибуты полностью зависят от первичного ключа (id)
* Все неключевые атрибуты взаимно независимы
* Внешние ключи используются только для связи таблиц и не создают транзитивной зависимости

*Использованные команды, типы данных и понятия SQL:*

1. Команда **CREATE TABLE** используется для создания новой таблицы в базе данных.
2. **PRIMARY KEY** - уникальный идентификатор каждой строки в таблице, обеспечивает уникальность записей. **AUTOINCREMENT** автоматически увеличивает значение ключа при добавлении новой записи.
3. **FOREIGN KEY** - Ссылается на первичный ключ другой таблицы. **ON UPDATE CASCADE** -при изменении значения в родительской таблице оно автоматически меняется в связанной таблице.
4. **NOT NULL** – указывает на то, что столбец не может содержать пустые значения.
5. Типы данных:

* **INTEGER** – целое число,
* **VARCHAR(n)** - текстовый тип данных с фиксированной максимальной длиной **n**,
* **DECIMAL(n, m)** - число с фиксированным кол-вом цифр после запятой, **n** – общее количество цифр, **m** – кол-во цифр после запятой.
* **DATE** – тип данных для хранения даты
* **TIME** – тип данных для хранения времени

*Заключение:*

В ходе данной лабораторной работы были приобретены навыки построения модели данных на основе неформального описания предметной области. В качестве предметной области был выбран фитнес-центр. После её анализа была разработана система таблиц для хранения информации о клиентах, тренерах, абонементах, расписании занятий и посещаемости. Для визуализации зависимостей между таблицами была построена ER-диаграмма (рис. 1). В будущем модель данных может быть расширена за счет добавления новой функциональности, например учет акций и скидок.

Модель данных была проверена на соответствие требованиям третьей нормальной формы.

В процессе выполнения работы также были получены практические навыки работы с SQLite. Были рассмотрены некоторые из основных типов данных, команд и понятий SQL.

В приложении 1 находится SQL-сценарий для реализации этой модели на базе SQLite 3. Он реализует создание реляционной базы данных для фитнес-центра, описанной в лабораторной работе. Скрипт последовательно создает шесть взаимосвязанных таблиц. В каждой таблице определены первичные ключи, в некоторых также внешние ключи с каскадными обновлениями для обеспечения целостности данных между связанными таблицами. Все неключевые атрибуты зависят только от первичного ключа.

*Приложение:*

1. Листинг использованных инструкций SQL (SQL-сценарий)

CREATE TABLE clients (  
 client\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 first\_name VARCHAR(50) NOT NULL,  
 last\_name VARCHAR(50) NOT NULL,  
 phone VARCHAR(15),  
 email VARCHAR(100),  
 registration\_date DATE NOT NULL  
);  
  
CREATE TABLE trainers (  
 trainer\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 first\_name VARCHAR(50) NOT NULL,  
 last\_name VARCHAR(50) NOT NULL,  
 specialization VARCHAR(100),  
 hire\_date DATE NOT NULL,  
 contact\_phone VARCHAR(15)  
);  
  
CREATE TABLE memberships (  
 membership\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 type VARCHAR(50) NOT NULL,  
 duration\_months INTEGER NOT NULL,  
 price DECIMAL(10,2) NOT NULL  
);  
  
CREATE TABLE schedule (  
 schedule\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 trainer\_id INTEGER NOT NULL,  
 class\_name VARCHAR(100) NOT NULL,  
 day\_of\_week VARCHAR(15) NOT NULL,  
 start\_time TIME NOT NULL,  
 end\_time TIME NOT NULL,  
 FOREIGN KEY (trainer\_id) REFERENCES trainers(trainer\_id)  
 ON UPDATE CASCADE  
);  
  
CREATE TABLE visits (  
 visit\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 client\_id INTEGER NOT NULL,  
 schedule\_id INTEGER NOT NULL,  
 visit\_date DATE NOT NULL,  
 FOREIGN KEY (client\_id) REFERENCES clients(client\_id)  
 ON UPDATE CASCADE,  
 FOREIGN KEY (schedule\_id) REFERENCES schedule(schedule\_id)  
 ON UPDATE CASCADE  
);  
  
CREATE TABLE purchases (  
 purchase\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 client\_id INTEGER NOT NULL,  
 membership\_id INTEGER NOT NULL,  
 purchase\_date DATE NOT NULL,  
 expiration\_date DATE NOT NULL,  
 FOREIGN KEY (client\_id) REFERENCES clients(client\_id)  
 ON UPDATE CASCADE,  
 FOREIGN KEY (membership\_id) REFERENCES memberships(membership\_id)  
 ON UPDATE CASCADE  
);