МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Направление подготовки
09.03.01 Информатика и вычислительная техника

Направленность (профиль)
«Технологии разработки программного обеспечения и обработки больших данных»

ОТЧЁТ

по реализации проекта для дисциплины «Базы данных»

Разработка системы электронного документооборота в МЧС по Санкт-Петербургу

Преподаватель: к.ф-м.н., доцент кафедры ИТиЭО

_____
(Жуков Н. Н.)

Студенты 2 курса:

Иванова О. Д. _____
Вяткина А. П. _____

Санкт-Петербург
2025

Оглавление

Ответственные

Иванова О. Д. – разработчик проекта. В обязанности Ивановой О. Д. входили анализ предметной области, нормализация данных, создание ER-диаграммы и моделей SQLModel.

Вяткина А. П. – разработчик проекта. В обязанности Вяткиной А. П. входили настройка подключения к базе данных, реализация авторизации и разграничения ролей.

Оба разработали основные API-эндпоинты на FastAPI, писали сервисы для работы с пользователями и сотрудниками, а также вместе тестировали систему и оформляли отчёт.

Предметная область

Предметной областью проекта является автоматизация электронного документооборота в рамках работы территориального управления МЧС по Санкт-Петербургу.

Система предназначена для хранения, просмотра и частичного редактирования информации, связанной с профессиональной деятельностью сотрудников. Основная цель проекта — обеспечить централизованный и безопасный доступ к данным о сотрудниках, их обучении, аттестации и задействованной технике.

В системе предусмотрено два типа пользователей:

- Администратор — имеет полный доступ ко всем функциям: просмотр, редактирование, добавление и удаление записей в системе.
- Пользователь — получает доступ только к просмотру информации, без возможности внесения изменений.

Хранимая информация:

- данные о сотрудниках (ФИО, дата рождения, звание, должность),
- данные об аттестации и её статусе,
- данные о занятиях,
- данные об автомобильной технике,
- данные о выездах.

Ход выполнения нормализации

После выделения основных сущностей предметной области была выполнена поэтапная нормализация данных от 1НФ до 3НФ. Ниже представлен список сущностей и их атрибутов:

Сотрудник (employee):

Каждый сотрудник имеет фамилию, имя, отчество, дату рождения, а также связан с определённой должностью и званием. В качестве ключевого атрибута используется surrogate key — id. Атрибуты position и rank являются внешними ключами на соответствующие справочники должностей и званий. Все атрибуты обязательны к заполнению.

Аттестация (attestation):

Каждая аттестация относится к сотруднику (employee_id), имеет тип (type — внешний ключ на справочник типов аттестаций), статус, дату проведения аттестации, а также поле причины, если аттестация не была пройдена. Ключевой атрибут — id.

Занятия (exercise):

Каждое занятие связано с сотрудником (employee_id), содержит дату, тип (exercise_type — внешний ключ) и адрес проведения. Используется surrogate key id.

Отчёт по занятиям (exercises_report):

Отдельная сущность, отражающая плановые и фактические показатели: даты начала и окончания, запланированное и фактическое количество проведенных занятий, а также комментарии. Имеет собственный id.

Должность (position):

Справочник должностей, включает наименование и группу. Первичный ключ — id.

Звание (rank):

Справочник званий, включает наименование и срок подготовки. Первичный ключ — id.

Типы аттестаций и занятий (attestation_type, exercise_type):

Отдельные справочники, каждый содержит уникальный идентификатор и наименование.

Автомобильная техника (fire_vehicle)

Хранит данные о пожарной и специальной технике. Имеет уникальный номер, тип (ПА, АГ и т.д.), марку, год изготовления и год ввода в эксплуатацию. Один автомобиль может участвовать во множестве выездов.

Выезды (callouts)

Представляет собой отдельный выезд. Включает дату, длительность, руководителя (head_of_depature_id), тип работы, адрес, ранг пожара и прочее. Связан с автомобилем (fire_vehicle_id) и сотрудниками через таблицу-участник.

Employee_to_attestation, employee_to_exercises, employee_to_callout

Реализуют связь «многие ко многим» между сотрудниками и аттестациями/занятиями/выездами. Например, у каждого выезда может быть множество участников, и сотрудник может участвовать в нескольких выездах.

В ходе нормализации:

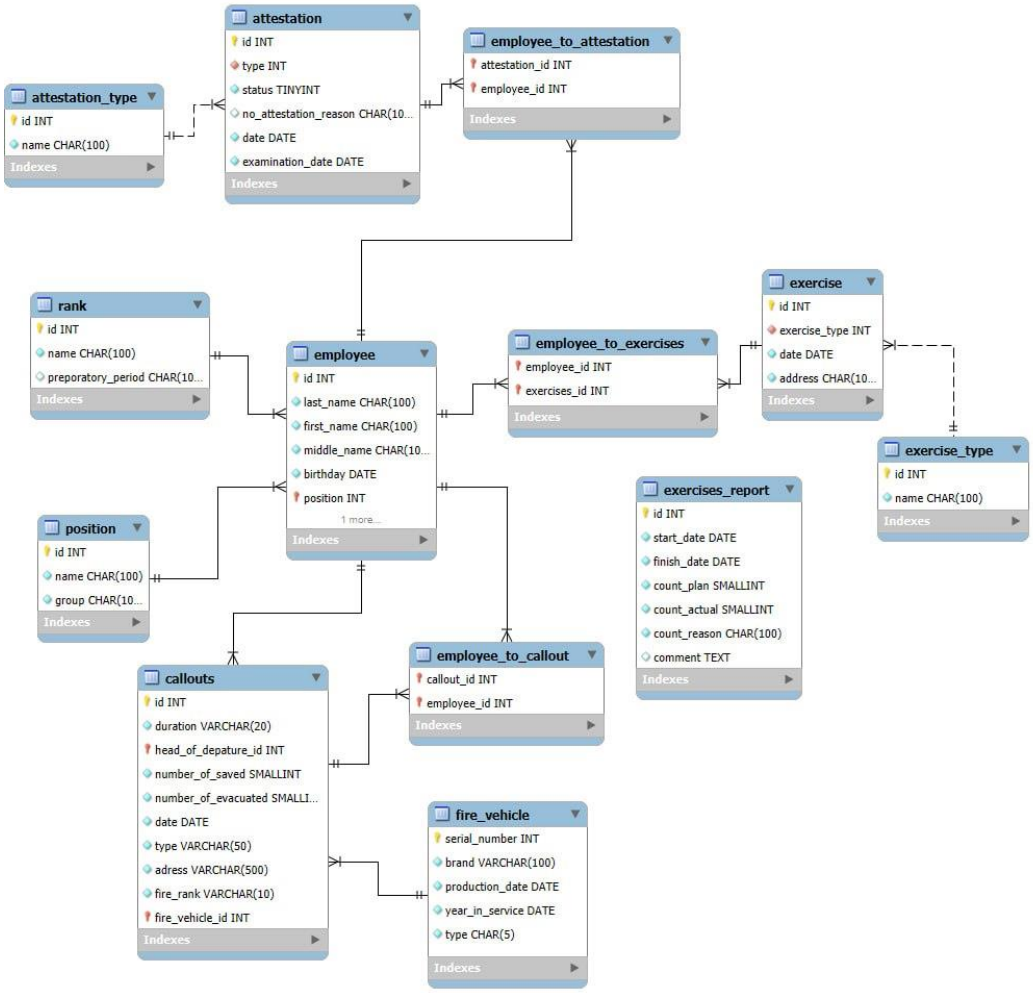
- Устранены дублирующиеся данные за счёт вынесения повторяющихся атрибутов в отдельные справочники;
- Введены surrogate-ключи для обеспечения уникальности;
- Реализованы связи между сущностями по внешним ключам;
- Таблицы приведены к 3НФ, транзитивные зависимости устранены.

На основе нормализованных сущностей построена ER-диаграмма, отражающая структуру базы данных и взаимосвязи между объектами системы.

Объяснение выбранной СУБД

Для реализации проекта была выбрана СУБД MySQL в связке с MySQL Workbench. Данный выбор был обусловлен тем, что Workbench используется для удобного проектирования ER-диаграмм и генерации SQL-кода, а также простотой, надёжностью и широким распространением MySQL в учебных и рабочих проектах.

ER–диаграмма



Исходный текст запросов

По созданию таблиц

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZE
RO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_E
NGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema mchs_project
-- -----------------------------------------------------
```

```
-- -----------------------------------------------------
-- Schema mchs_project
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS mchs_project DEFAULT CHARACTER SET utf8mb3 ;
USE mchs_project ;

-- -----------------------------------------------------
-- Table mchs_project.attestation_type
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS mchs_project.attestation_type (
  id INT NOT NULL AUTO_INCREMENT,
  name CHAR(100) NOT NULL,
  PRIMARY KEY (id))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------
-- Table mchs_project.attestation
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS mchs_project.attestation (
  id INT NOT NULL AUTO_INCREMENT,
  type INT NOT NULL,
  status TINYINT NOT NULL,
  no_attestation_reason CHAR(100) NULL DEFAULT NULL,
  date DATE NOT NULL,
  examination_date DATE NOT NULL,
  PRIMARY KEY (id),
  INDEX attestation_to_attestation_type_idx (type ASC) VISIBLE,
  CONSTRAINT attestation_to_attestation_type
    FOREIGN KEY (type)
    REFERENCES mchs_project.attestation_type (id))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------
-- Table mchs_project.position
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS mchs_project.position (
  id INT NOT NULL AUTO_INCREMENT,
  name CHAR(100) NOT NULL,
  group CHAR(100) NOT NULL,
  PRIMARY KEY (id))
```

```
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------
-- Table mchs_project.rank
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS mchs_project.rank (
  id INT NOT NULL AUTO_INCREMENT,
  name CHAR(100) NOT NULL,
  preporatory_period CHAR(100) NULL DEFAULT NULL,
  PRIMARY KEY (id))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------
-- Table mchs_project.employee
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS mchs_project.employee (
  id INT NOT NULL AUTO_INCREMENT,
  last_name CHAR(100) NOT NULL,
  first_name CHAR(100) NOT NULL,
  middle_name CHAR(100) NOT NULL,
  birthday DATE NOT NULL,
  position INT NOT NULL,
  rank INT NOT NULL,
  PRIMARY KEY (id, position, rank),
  INDEX employee_to_position_idx (position ASC) VISIBLE,
  INDEX employee_to_rank_idx (rank ASC) VISIBLE,
  CONSTRAINT employee_to_position
    FOREIGN KEY (position)
    REFERENCES mchs_project.position (id),
  CONSTRAINT employee_to_rank
    FOREIGN KEY (rank)
    REFERENCES mchs_project.rank (id))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------
-- Table mchs_project.exercise_type
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS mchs_project.exercise_type (
  id INT NOT NULL AUTO_INCREMENT,
```

```
  name CHAR(100) NOT NULL,
  PRIMARY KEY (id))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------
-- Table mchs_project.exercise
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS mchs_project.exercise (
  id INT NOT NULL AUTO_INCREMENT,
  exercise_type INT NOT NULL,
  date DATE NOT NULL,
  address CHAR(100) NOT NULL,
  PRIMARY KEY (id),
  INDEX exercise_to_exercise_type_idx (exercise_type ASC) VISIBLE,
  CONSTRAINT exercise_to_exercise_type
    FOREIGN KEY (exercise_type)
    REFERENCES mchs_project.exercise_type (id))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------
-- Table mchs_project.exercises_report
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS mchs_project.exercises_report (
  id INT NOT NULL AUTO_INCREMENT,
  start_date DATE NOT NULL,
  finish_date DATE NOT NULL,
  count_plan SMALLINT NOT NULL,
  count_actual SMALLINT NOT NULL,
  count_reason CHAR(100) NOT NULL,
  comment TEXT NULL DEFAULT NULL,
  PRIMARY KEY (id))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------
-- Table mchs_project.employee_to_exercises
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS mchs_project.employee_to_exercises (
  employee_id INT NOT NULL,
  exercises_id INT NOT NULL,
```

```
  PRIMARY KEY (employee_id, exercises_id),
  INDEX middle_table_to_exercises_idx (exercises_id ASC) VISIBLE,
  CONSTRAINT middle_table_to_exercises
    FOREIGN KEY (exercises_id)
    REFERENCES mchs_project.exercise (id)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT middle_table_to_employee
    FOREIGN KEY (employee_id)
    REFERENCES mchs_project.employee (id)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table mchs_project.employee_to_attestation
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS mchs_project.employee_to_attestation (
  attestation_id INT NOT NULL,
  employee_id INT NOT NULL,
  PRIMARY KEY (attestation_id, employee_id),
  INDEX middle_table_to_employee_idx (employee_id ASC) VISIBLE,
  CONSTRAINT middle_table_to_attestation
    FOREIGN KEY (attestation_id)
    REFERENCES mchs_project.attestation (id)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT middle_table_to_employee
    FOREIGN KEY (employee_id)
    REFERENCES mchs_project.employee (id)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table mchs_project.fire_vehicle
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS mchs_project.fire_vehicle (
  serial_number INT NOT NULL,
  brand VARCHAR(100) NOT NULL,
  production_date DATE NOT NULL,
  year_in_service DATE NOT NULL,
  type CHAR(5) NOT NULL,
```

```
  PRIMARY KEY (serial_number))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table mchs_project.callouts
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS mchs_project.callouts (
  id INT NOT NULL AUTO_INCREMENT,
  duration VARCHAR(20) NOT NULL,
  head_of_depature_id INT NOT NULL,
  number_of_saved SMALLINT NOT NULL,
  number_of_evacuated SMALLINT NOT NULL,
  date DATE NOT NULL,
  type VARCHAR(50) NOT NULL,
  adress VARCHAR(500) NOT NULL,
  fire_rank VARCHAR(10) NOT NULL,
  fire_vehicle_id INT NOT NULL,
  PRIMARY KEY (id, fire_vehicle_id, head_of_depature_id),
  INDEX callout_to_vehicle_idx (fire_vehicle_id ASC) VISIBLE,
  INDEX gh_idx (head_of_depature_id ASC) VISIBLE,
  CONSTRAINT callout_to_vehicle
    FOREIGN KEY (fire_vehicle_id)
    REFERENCES mchs_project.fire_vehicle (serial_number)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT gh
    FOREIGN KEY (head_of_depature_id)
    REFERENCES mchs_project.employee (id)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table mchs_project.employee_to_callout
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS mchs_project.employee_to_callout (
  callout_id INT NOT NULL,
  employee_id INT NOT NULL,
  PRIMARY KEY (callout_id, employee_id),
  INDEX middle_table_to_employee_idx (employee_id ASC) VISIBLE,
  CONSTRAINT middle_table_to_employee
    FOREIGN KEY (employee_id)
    REFERENCES mchs_project.employee (id)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT middle_table_to_callouts
    FOREIGN KEY (callout_id)
    REFERENCES mchs_project.callouts (id)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Триггеры и транзакции

В рамках проекта предполагается использование транзакций и триггеров для обеспечения целостности и корректности данных.

В нашем проекте транзакции могут использоваться в следующих ситуациях:

- **Добавление выезда.** При создании выезда (**callouts**) сначала записывается основной выезд, затем добавляются его участники в таблицу **employee_to_callout**. В случае ошибки на любом этапе — например, если один из сотрудников больше не работает — вся операция должна быть отменена.
- **Удаление пользователя**. При удалении пользователя необходимо также удалить связанного с ним сотрудника, а возможно — и все связанные с ним выезды и участие в них. Все эти действия должны выполняться как единое целое, иначе останутся «висячие» записи.
- **Изменение ранга пожара.** Допустим, система позволяет вручную изменять ранг пожара (**fire_rank**) по результатам выезда. Чтобы избежать противоречий, изменение должно сопровождаться проверкой и сохранением причин или комментариев.

В проекте можно реализовать следующие триггеры:

- **Установка даты по умолчанию.** В таблице **attestation** или **callouts** может быть триггер BEFORE INSERT, который автоматически подставляет текущую дату, если она не указана при добавлении записи.
- **Очистка зависимых данных.** При удалении сотрудника (**employee**) можно настроить AFTER DELETE-триггер, который удалит все связанные с ним записи в **employee_to_callout**.
- **Проверка готовности техники.** Перед вставкой новой записи в таблицу **callouts** может использоваться триггер BEFORE INSERT, который проверяет, существует ли указанный автомобиль (**fire_vehicle_id**) и находится ли он в исправном состоянии или готов к выезду. Это позволяет исключить случаи, когда в выезд назначается техника, находящаяся на обслуживании или списании.