|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**Отчет по практическим работам**

по дисциплине «Разработка баз данных»

**Студент группы** ИКБО-20-19 Анваржонов Ж Т

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель практических работ** Володина А. М.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «20» декабря 2021 г.

Работа принята «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Москва 2021

**Введение**

**Описание предметной области:**

Основой предметной области по практическим работам послужила база данных «Автобусный парк». Данная тема была взята по причине того, что миллионы студентов по всей стране, в том числе и я, пользуются услугами государственных автобусных парков, когда добираются до своего учебного заведения. И вот в очередной день мотания кругов до университета мне взбрело в голову, что данная тема отлично подходит для практики по разработке базы данных.

В базу данных были включено несколько ключевых сущностей, таких как:

* Водитель автобуса
* Диспетчер, который связывается с водителем
* Автобусный рейс (bus\_flight)
* Автобус
* Валидатор
* Сущность «Страховая компания пассажиров»
* Реклама в автобусе (advertising on the bus)

Далее опишем Логическую и физическую модель данных.

**Логическая модель данных системы «Автобусный парк»**

Создадим cущности

* «Водитель автобуса» со следующими атрибутами:
* Id\_driver (Primary key, Number)
* Name&Surname (String)
* Work\_experience – опыт работы (Number)
* Driver\_license (String)
* «Dispatcher» со следующими атрибутами:
  + Id\_dispatcher (PK, Number)
  + Name (String)
  + Surname (String)
* «Bus\_flight»:
  + Id\_flight (PK, Number)
  + Name\_start\_flight (String)
  + Name\_end\_flight (String)
* «Bus»:
  + Id\_bus (PK, Number)
  + Bus\_number (String)
  + Bus\_name
* «Passenger\_insurance\_company»:
  + Id\_company (PK, Number)
  + Name\_insurance\_company(String)
  + Contract\_number (Number)
  + Date\_of\_conclusion (Date)
* «Validator»:
  + Id\_validator (PK, Number)
  + isActive (String)
* «Advertising\_on\_the\_bus»:
  + Id\_advertising (PK, number)
  + Name of company (String)
  + Start\_advertising\_period (Date)
  + End\_advertising\_period (Date)

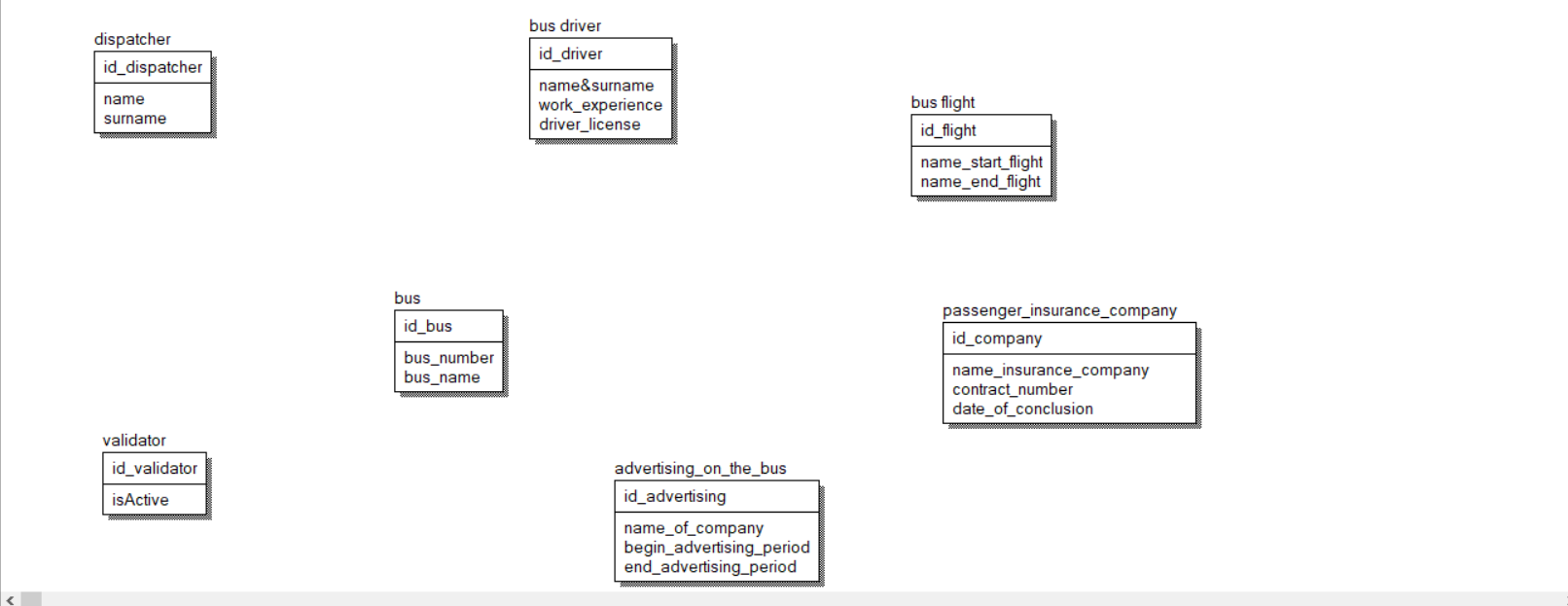


Рисунок 1. Отображение сущностей на логической модели.

Установим связи между сущностями:

* Dispatcher → Bus\_driver (Неидентифицирующая связь)

Здесь неидентифицирующая связь потому что экземпляр «водитель автобуса» может существовать без экземпляра «диспетчер». У многих автобусников может быть один диспетчер, значит если сделать id\_dispatcher как Primary Key, то диспетчеры у водителей автобусов не смогут повторяться, то есть одному автобуснику один диспетчер – здесь это не подходит.

* Bus\_flight→ Bus\_driver (Неидентифицирующая связь)

Здесь один рейс может быть у многих автобусников

* Bus\_driver→ Bus (Идентифицирующая связь)
* Validator→ Bus (Идентифицирующая связь)
* Passenger\_insurance\_company → Bus (Идентифицирующая связь)
* Advertising on the bus → Bus (Неидентифицирующая связь)

Данная связь с разрешением нулей, потому что рекламы в автобусе может и не быть

* Dispatcher → Bus\_driver (Идентифицирующая связь)

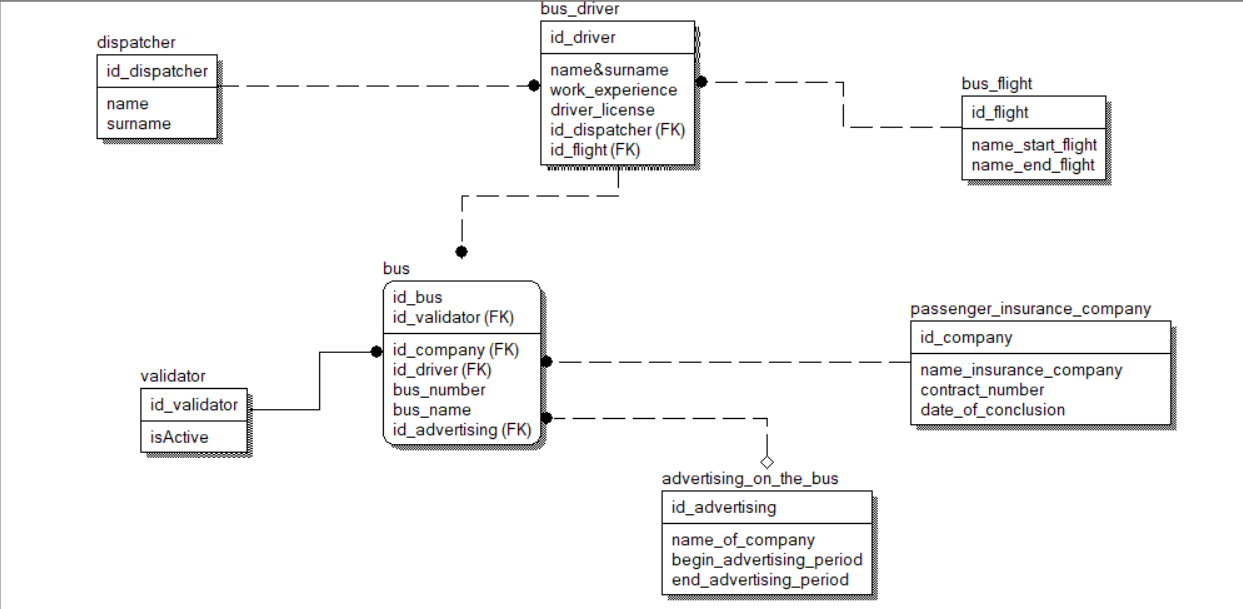


Рисунок 2. Отображение сущностей и их связей на логической модели

**Физическая модель данных системы «Автобусный парк»**

* «Водитель автобуса» со следующими атрибутами:
* Id\_driver - PK, int(10)
* Name&Surname - varchar(30)
* Work\_experience – опыт работы - int(20)
* Driver\_license - varchar(20)
* «Dispatcher» со следующими атрибутами:
  + Id\_dispatcher - PK, int(10)
  + Name – varchar(20)
  + Surname – varchar(20)
* «Bus\_flight»:
  + Id\_flight – PK, int(10)
  + Name\_start\_flight – varchar(30)
  + Name\_end\_flight – varchar(30)
* «Bus»:
  + Id\_bus – PK, int (10)
  + Bus\_number - varchar(20)
  + Bus\_name – varchar(20)
* «Passenger\_insurance\_company»:
  + Id\_company – PK, int(10)
  + Name\_insurance\_company – varchar(20)
  + Contract\_number – int(20)
  + Date\_of\_conclusion - DATE
* «Validator»:
  + Id\_validator – Pk, int(10)
  + isActive – varchar(10)
* «Advertising\_on\_the\_bus»:
  + Id\_advertising – Pk, int(10)
  + Name of company – varchar(20)
  + Start\_advertising\_period - Date
  + End\_advertising\_period – Date

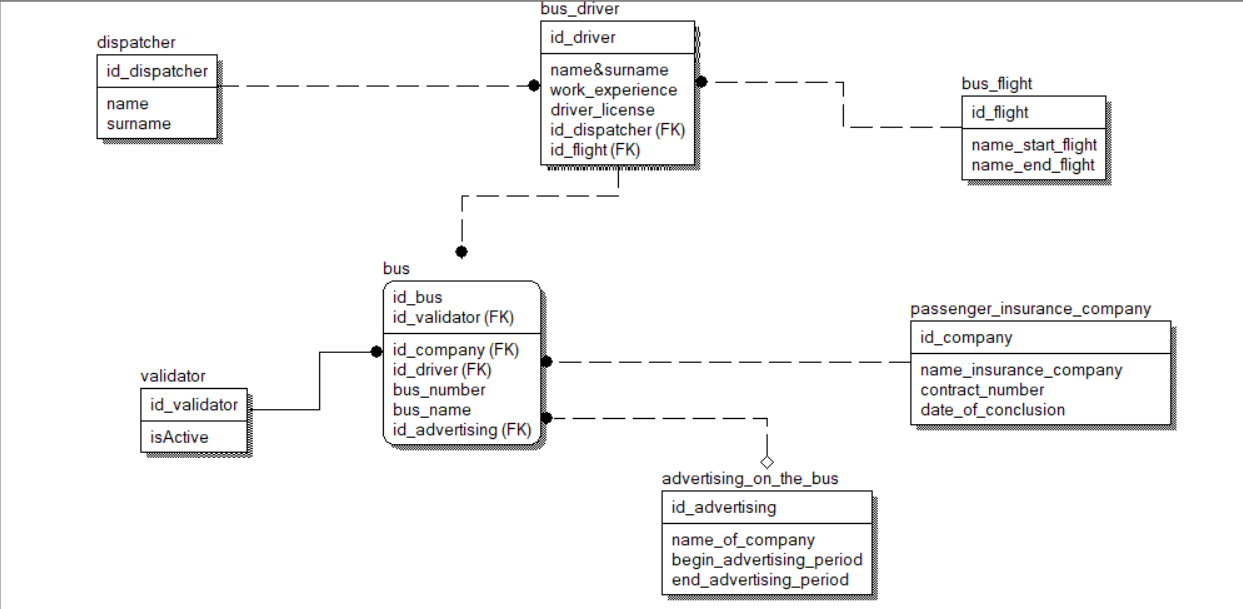


Рисунок 3. Физическая модель

**Создание таблиц в MySQL**

Создадим базу данных bus\_park:

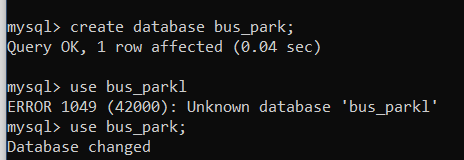
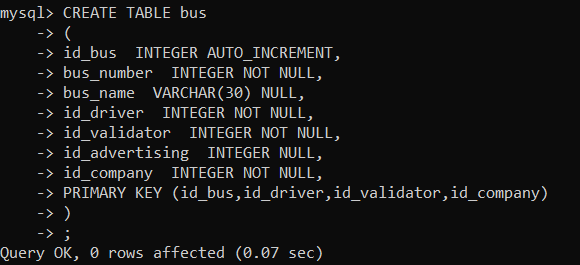
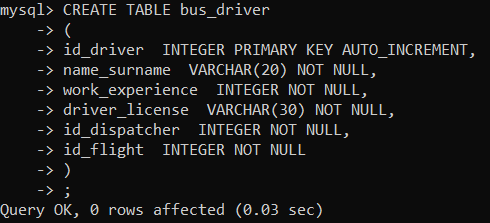


Рисунок 4. Создание таблицы

Создадим таблицы:





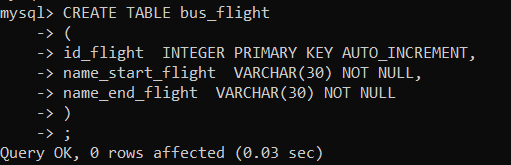


Рисунок 5. Таблицы bus,bus\_flight, bus\_driver

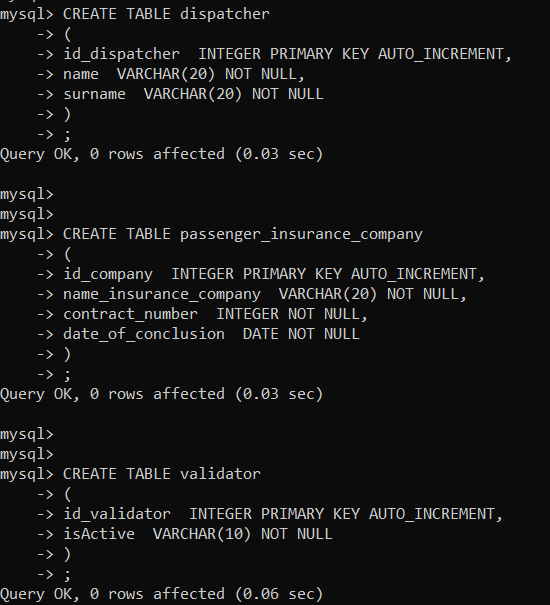


Рисунок 6. Таблицы dispatcher, passenger\_insurance, validator

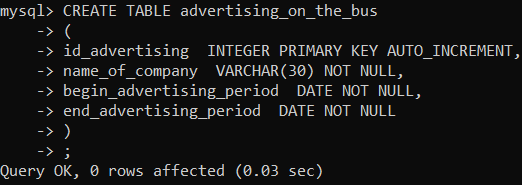


Рисунок 7 . Таблица advertising on the bus

Добавим в таблицы внешние ключи:

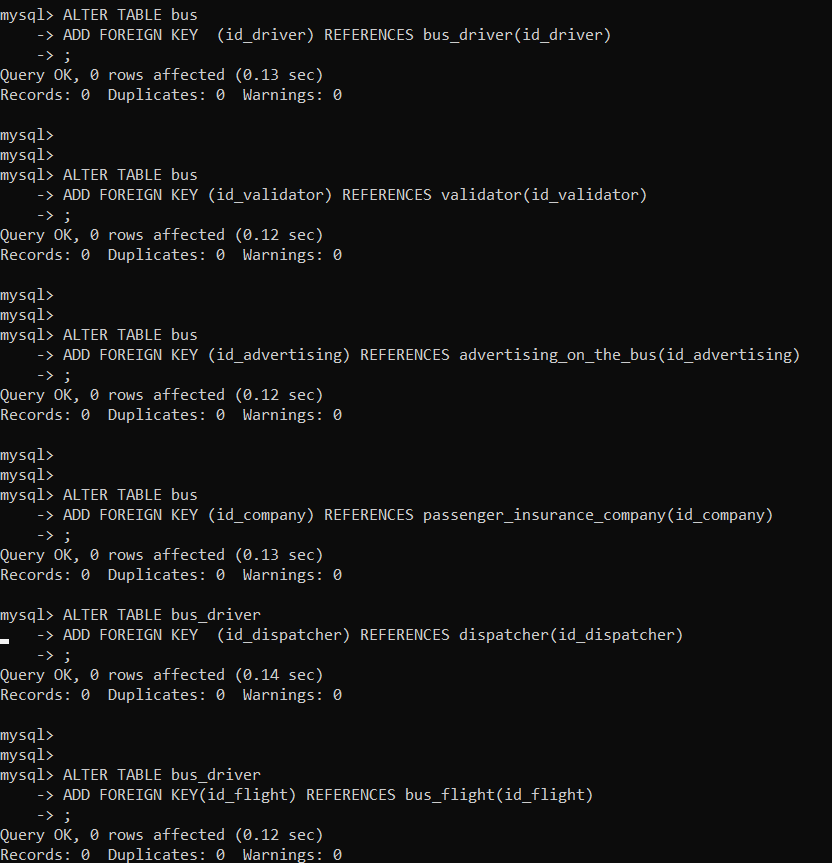


Рисунок 8. Добавление внешних ключей с помощью ALTER TABLE и foreign key.

Покажем созданные таблицы:

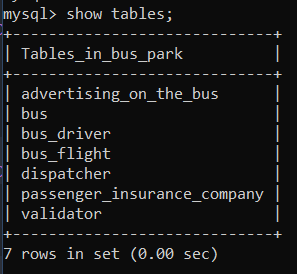


Рисунок 9 . Созданные таблицы

Заполним таблицы и покажем их:

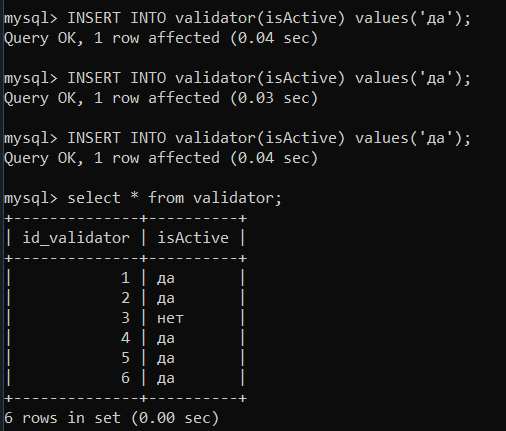


Рисунок 10. Заполнение таблицы validator и показ строк.

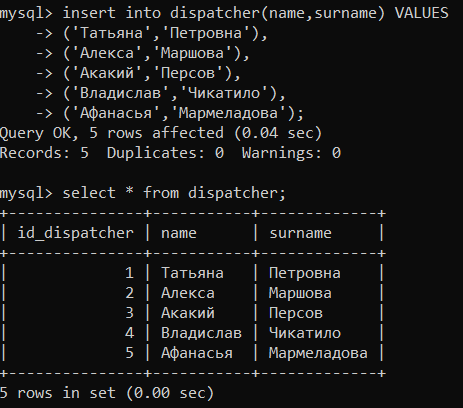


Рисунок 11. Заполнение таблицы dispatcher;

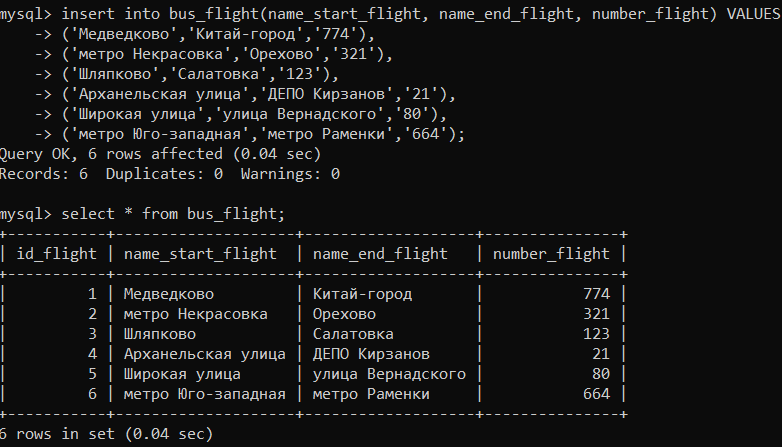


Рисунок 12. Заполнение таблицы bus\_flight;

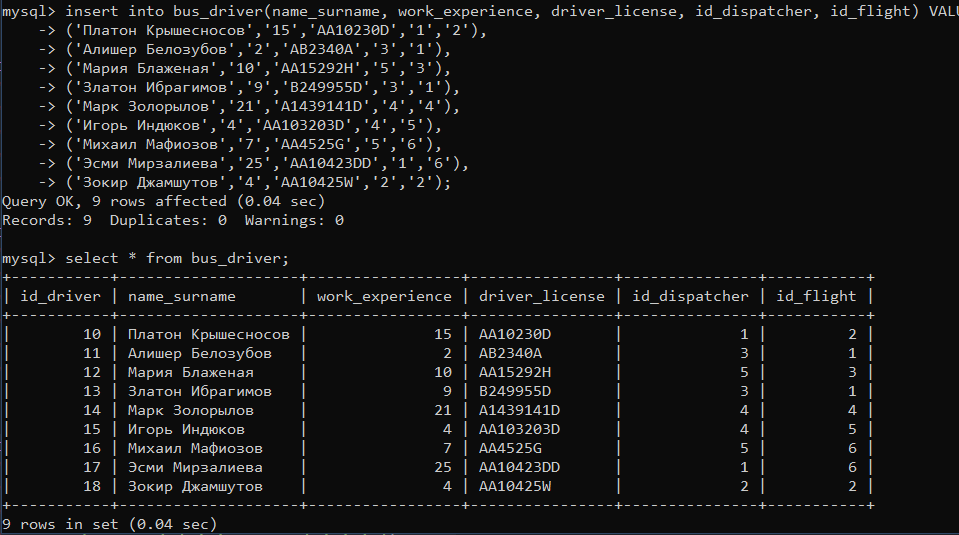


Рисунок 13. Заполнение таблицы bus\_driver

Попробуем удалить строки в таблице bus\_driver, где id\_driver>10:

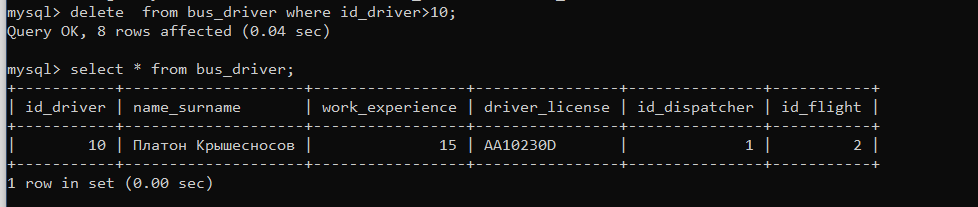


Рисунок 14. Удаление строк в таблице bus\_driver;

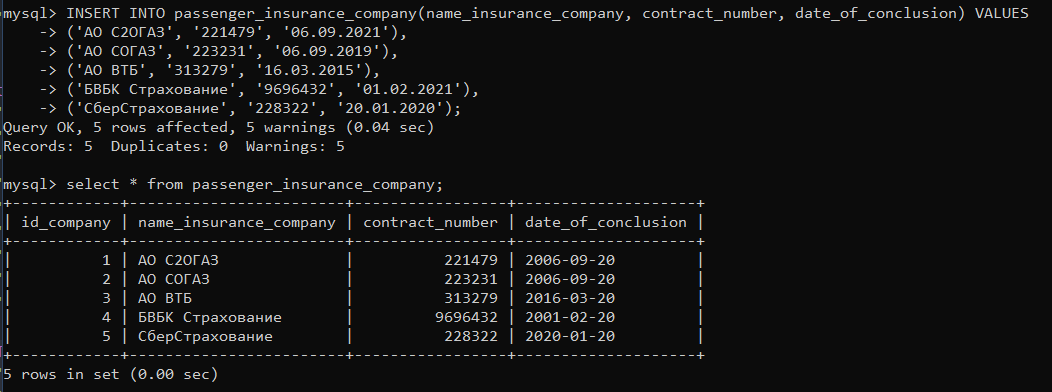


Рисунок 15. Заполнение таблицы «Страховая компания»

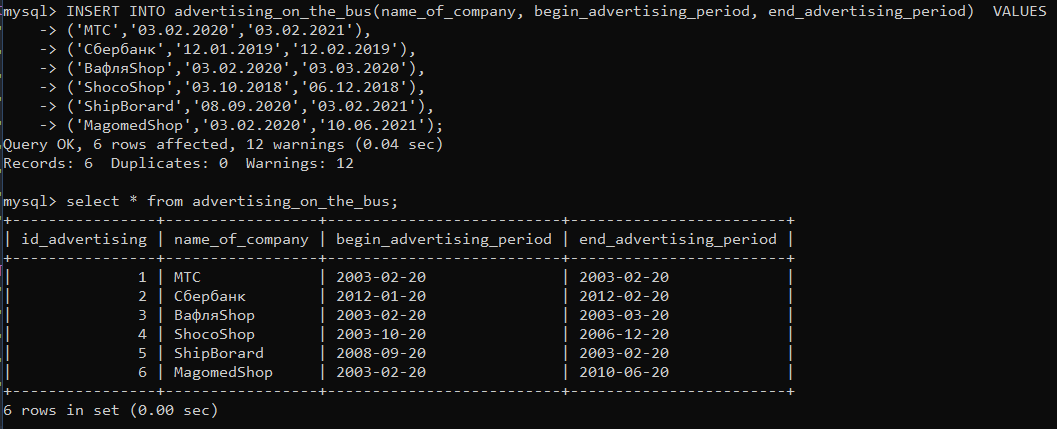


Рисунок 16. Заполнение таблицы «Реклама в автобусе»

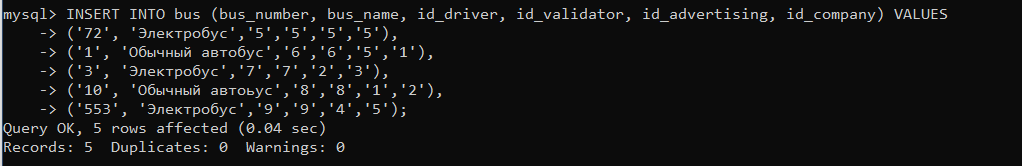


Рисунок 17. Заполнение таблицы «Автобус»

Выборка и сортировка данных

Просмотрим все столбцы из таблицы bus:

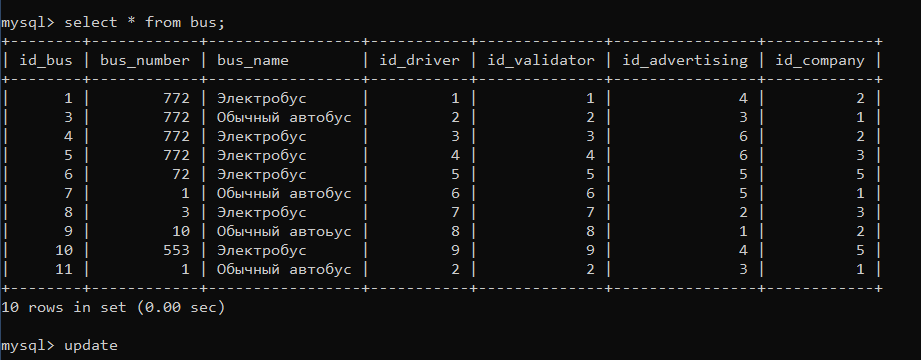


Рисунок 18. Все столбцы из таблицы bus

Просмотрим например только столбец bus\_number из таблицы bus:

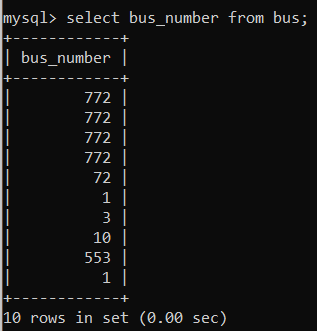


Рисунок 19. Cтолбец bus\_number.

Выведем имя и фамилию водителей по алфавиту с помощью ORDER BY:

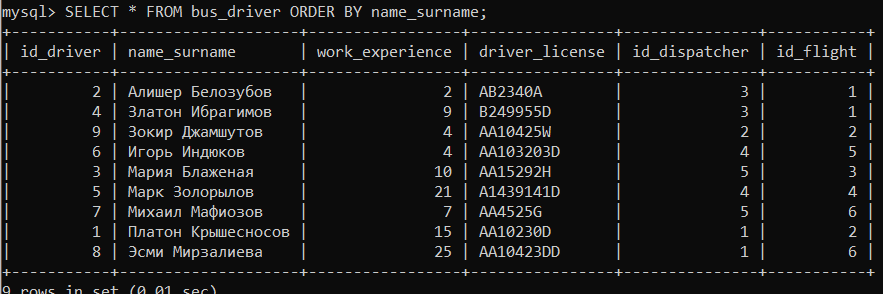


Рисунок 20. Вывод всех столбцов где сортировка идет по столбцу name\_surname;

Выведем имя и фамилию всех водителей, у которых id диспетчера равен 3 и отсортируем их по убыванию:

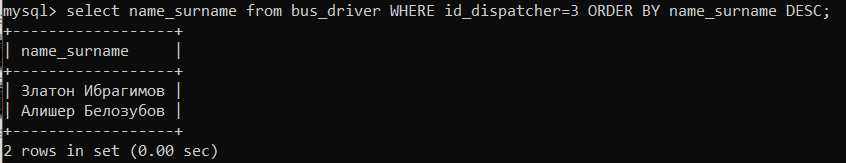


Рисунок 21. Вывод всех водителей, у которых id диспетчера равен 3

Покажем всех водителей, у которых стаж работы больше 5 лет:

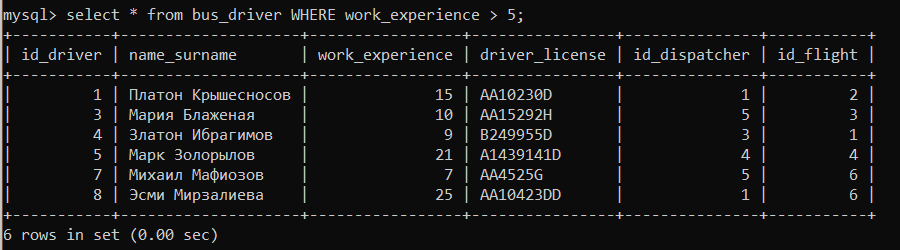


Рисунок 22. Вывод всех водителей у которых стаж работы больше 5 лет.

Выведем всех водителей у которых id рейса находится между 1 и 3

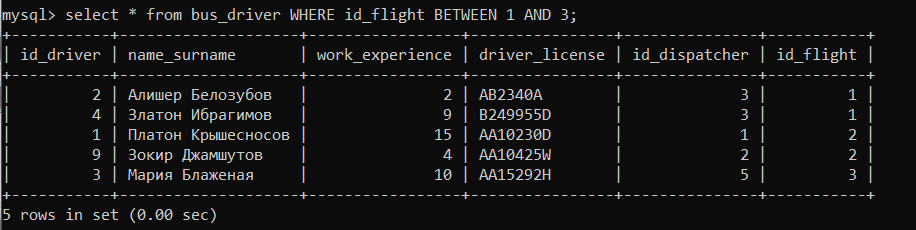


Рисунок 23. Вывод всех водителей у которых id рейса находится между 1 и 3

Выведем все страховые компании, названия которых начинаются с АО:

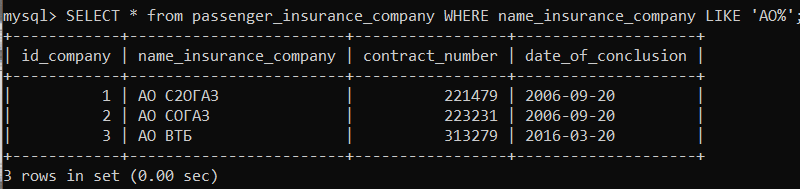


Рисунок 24. Вывод все страховые компании, названия которых начинаются с АО

Изменение данных в таблице

Обновим Стаж работы у водителя с id = 13 и у водителя с id=19:

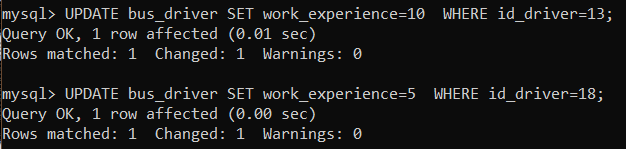


Рисунок 25. Обновление данных

Добавим один столбец в таблицу bus\_driver, которая показывает количество проделанных водителем рейсов

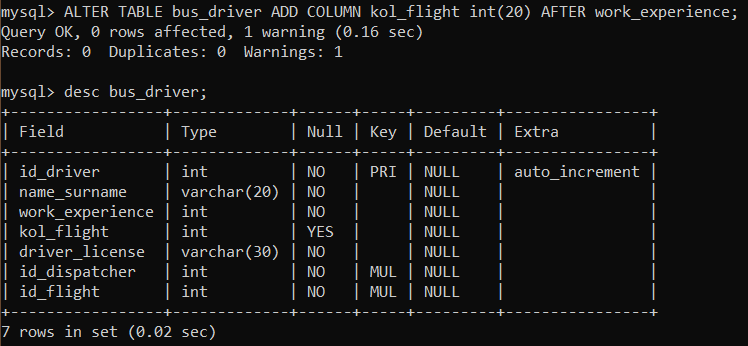


Рисунок 26. Добавление столбца kol\_flight после work\_experience;

Обновим информацию о проделанных рейсах у водителей у которых стаж работы больше 30

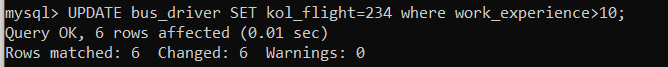


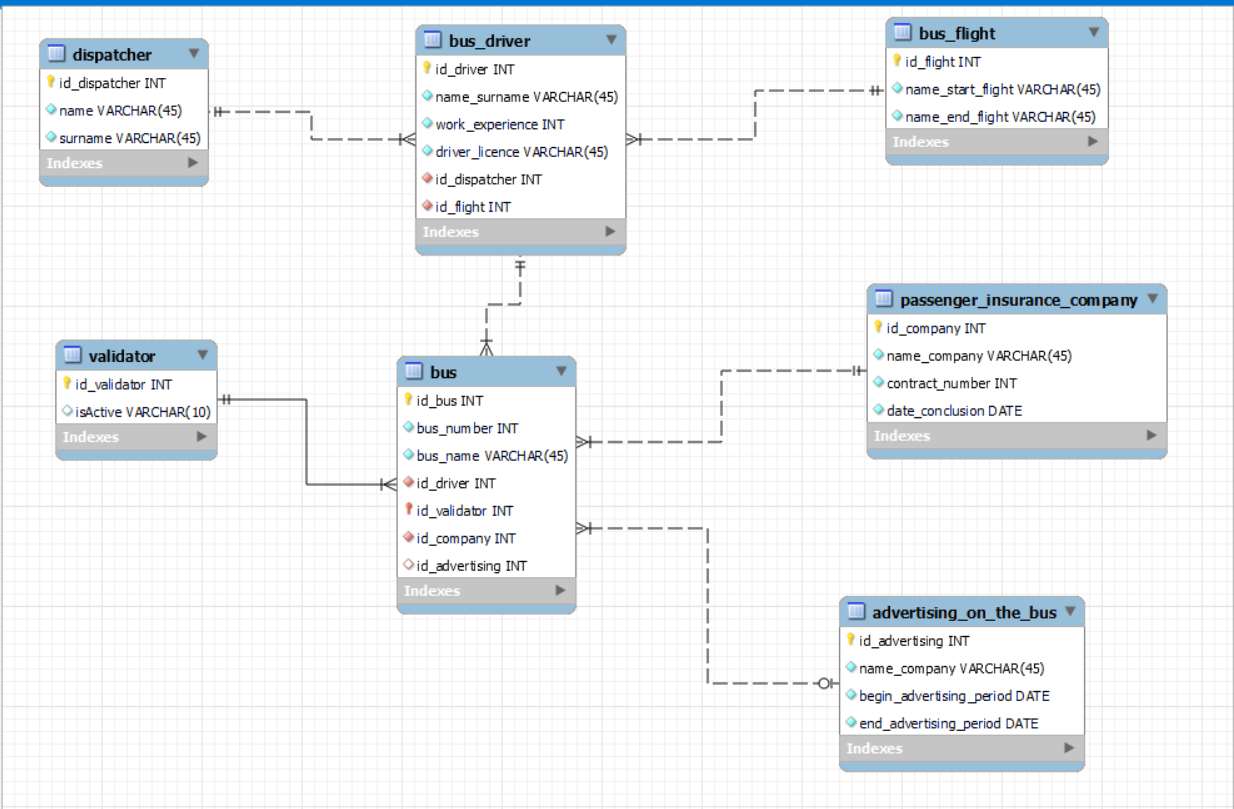
Рисунок 27. Обновление данных

**Вывод:** В ходе практических работ мы научились создавать логическую и физическую модель базы данных, делать простые и сложные sql запросы, обновлять данных в таблицах, делать выборку и сортировку данных в таблицах, изменять уже существующие данные используя различного рода запросы.

**Практическая №3**

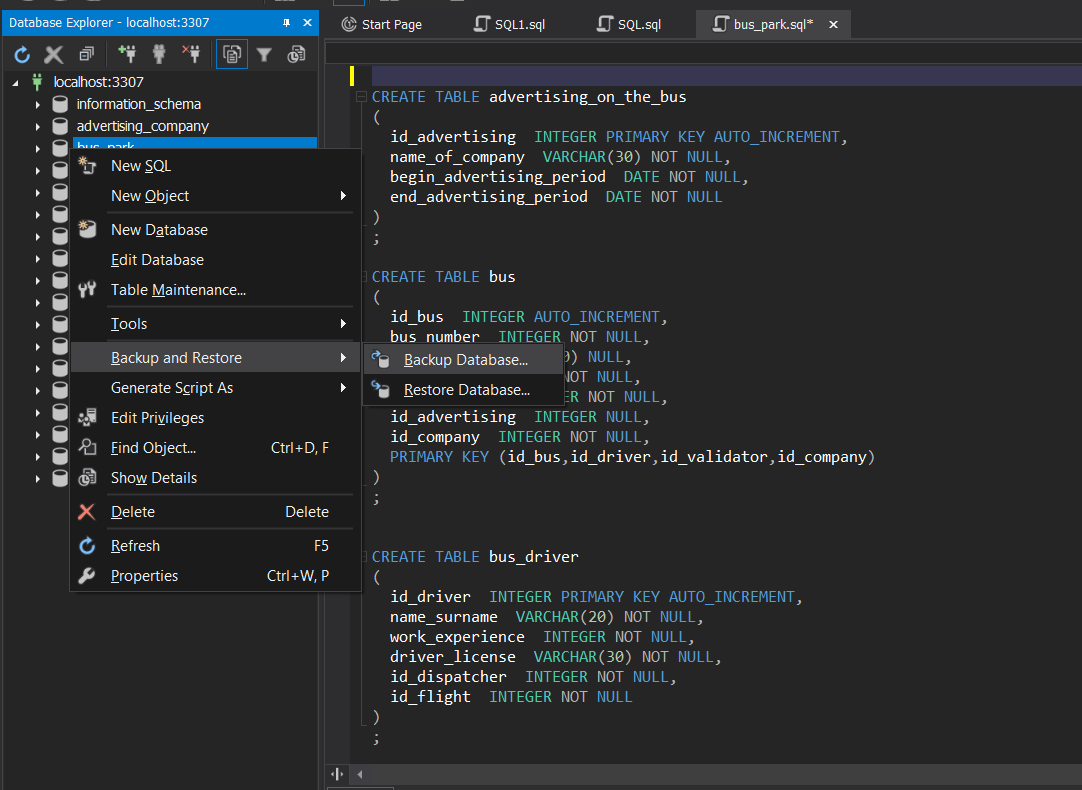
* 1. **Построение модели с помощью оболочки MYSQL WorkBench**

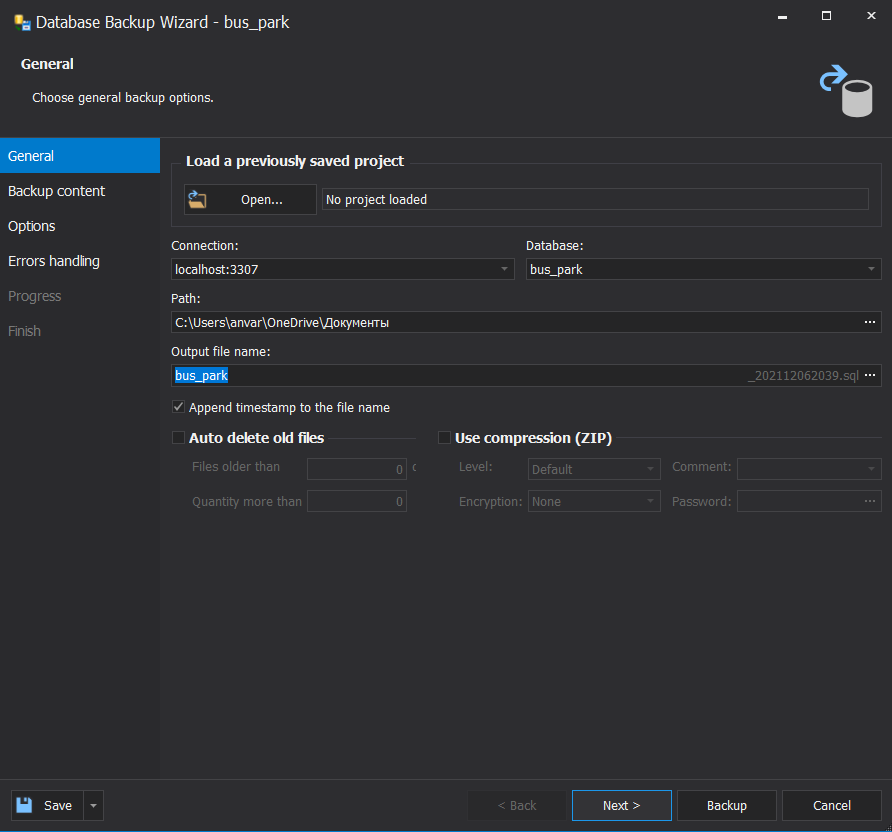
Создадим таблицы и их связи в виде диаграмм модели:



**1.2. Перенос базы данных на другой сервер**

Выберем в dbForce резервную копию базы данных





Покажем скрипт, который получился:

*--*

*-- Script was generated by Devart dbForge Studio 2020 for MySQL, Version 9.0.782.0*

*-- Product home page: http://www.devart.com/dbforge/mysql/studio*

*-- Script date 06.12.2021 20:40:15*

*-- Server version: 8.0.26*

*-- Client version: 4.1*

*--*

*--*

*-- Disable foreign keys*

*--*

*/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/*;

*--*

*-- Set SQL mode*

*--*

*/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/*;

*--*

*-- Set character set the client will use to send SQL statements to the server*

*--*

SET NAMES 'utf8';

DROP DATABASE IF EXISTS bus\_park;

CREATE DATABASE bus\_park

CHARACTER SET utf8mb4

COLLATE utf8mb4\_0900\_ai\_ci;

*--*

*-- Set default database*

*--*

USE bus\_park;

*--*

*-- Create table `dispatcher`*

*--*

CREATE TABLE dispatcher (

  id\_dispatcher int NOT NULL AUTO\_INCREMENT,

  name varchar(20) NOT NULL,

  surname varchar(20) NOT NULL,

  PRIMARY KEY (id\_dispatcher)

)

ENGINE = INNODB,

AUTO\_INCREMENT = 6,

AVG\_ROW\_LENGTH = 3276,

CHARACTER SET utf8mb4,

COLLATE utf8mb4\_0900\_ai\_ci;

*--*

*-- Create table `bus\_flight`*

*--*

CREATE TABLE bus\_flight (

  id\_flight int NOT NULL AUTO\_INCREMENT,

  name\_start\_flight varchar(30) NOT NULL,

  name\_end\_flight varchar(30) NOT NULL,

  number\_flight int NOT NULL,

  PRIMARY KEY (id\_flight)

)

ENGINE = INNODB,

AUTO\_INCREMENT = 7,

AVG\_ROW\_LENGTH = 2730,

CHARACTER SET utf8mb4,

COLLATE utf8mb4\_0900\_ai\_ci;

*--*

*-- Create table `bus\_driver`*

*--*

CREATE TABLE bus\_driver (

  id\_driver int NOT NULL AUTO\_INCREMENT,

  name\_surname varchar(20) NOT NULL,

  work\_experience int NOT NULL,

  kol\_flight int DEFAULT NULL,

  driver\_license varchar(30) NOT NULL,

  id\_dispatcher int NOT NULL,

  id\_flight int NOT NULL,

  PRIMARY KEY (id\_driver)

)

ENGINE = INNODB,

AUTO\_INCREMENT = 19,

AVG\_ROW\_LENGTH = 910,

CHARACTER SET utf8mb4,

COLLATE utf8mb4\_0900\_ai\_ci;

*--*

*-- Create foreign key*

*--*

ALTER TABLE bus\_driver

ADD CONSTRAINT bus\_driver\_ibfk\_1 FOREIGN KEY (id\_dispatcher)

REFERENCES dispatcher (id\_dispatcher);

*--*

*-- Create foreign key*

*--*

ALTER TABLE bus\_driver

ADD CONSTRAINT bus\_driver\_ibfk\_2 FOREIGN KEY (id\_flight)

REFERENCES bus\_flight (id\_flight);

*--*

*-- Create table `validator`*

*--*

CREATE TABLE validator (

  id\_validator int NOT NULL AUTO\_INCREMENT,

  isActive enum ('да', 'нет') NOT NULL DEFAULT (\_cp866 '¤ '),

  PRIMARY KEY (id\_validator)

)

ENGINE = INNODB,

AUTO\_INCREMENT = 28,

AVG\_ROW\_LENGTH = 606,

CHARACTER SET utf8mb4,

COLLATE utf8mb4\_0900\_ai\_ci;

*--*

*-- Create table `passenger\_insurance\_company`*

*--*

CREATE TABLE passenger\_insurance\_company (

  id\_company int NOT NULL AUTO\_INCREMENT,

  name\_insurance\_company varchar(20) NOT NULL,

  contract\_number int NOT NULL,

  date\_of\_conclusion date NOT NULL,

  PRIMARY KEY (id\_company)

)

ENGINE = INNODB,

AUTO\_INCREMENT = 6,

AVG\_ROW\_LENGTH = 3276,

CHARACTER SET utf8mb4,

COLLATE utf8mb4\_0900\_ai\_ci;

*--*

*-- Create table `advertising\_on\_the\_bus`*

*--*

CREATE TABLE advertising\_on\_the\_bus (

  id\_advertising int NOT NULL AUTO\_INCREMENT,

  name\_of\_company varchar(30) NOT NULL,

  begin\_advertising\_period date NOT NULL,

  end\_advertising\_period date NOT NULL,

  PRIMARY KEY (id\_advertising)

)

ENGINE = INNODB,

AUTO\_INCREMENT = 7,

AVG\_ROW\_LENGTH = 2730,

CHARACTER SET utf8mb4,

COLLATE utf8mb4\_0900\_ai\_ci;

*--*

*-- Create table `bus`*

*--*

CREATE TABLE bus (

  id\_bus int NOT NULL AUTO\_INCREMENT,

  bus\_number int NOT NULL,

  bus\_name varchar(30) DEFAULT NULL,

  id\_driver int NOT NULL,

  id\_validator int NOT NULL,

  id\_advertising int DEFAULT NULL,

  id\_company int NOT NULL,

  PRIMARY KEY (id\_bus, id\_driver, id\_validator, id\_company)

)

ENGINE = INNODB,

AUTO\_INCREMENT = 28,

AVG\_ROW\_LENGTH = 910,

CHARACTER SET utf8mb4,

COLLATE utf8mb4\_0900\_ai\_ci;

*--*

*-- Create foreign key*

*--*

ALTER TABLE bus

ADD CONSTRAINT bus\_ibfk\_1 FOREIGN KEY (id\_driver)

REFERENCES bus\_driver (id\_driver);

*--*

*-- Create foreign key*

*--*

ALTER TABLE bus

ADD CONSTRAINT bus\_ibfk\_2 FOREIGN KEY (id\_validator)

REFERENCES validator (id\_validator);

*--*

*-- Create foreign key*

*--*

ALTER TABLE bus

ADD CONSTRAINT bus\_ibfk\_3 FOREIGN KEY (id\_advertising)

REFERENCES advertising\_on\_the\_bus (id\_advertising);

*--*

*-- Create foreign key*

*--*

ALTER TABLE bus

ADD CONSTRAINT bus\_ibfk\_4 FOREIGN KEY (id\_company)

REFERENCES passenger\_insurance\_company (id\_company);

*--*

*-- Dumping data for table dispatcher*

*--*

INSERT INTO dispatcher VALUES

(1, 'Татьяна', 'Петровна'),

(2, 'Алекса', 'Маршова'),

(3, 'Акакий', 'Персов'),

(4, 'Владислав', 'Чикатило'),

(5, 'Афанасья', 'Мармеладова');

*--*

*-- Dumping data for table bus\_flight*

*--*

INSERT INTO bus\_flight VALUES

(1, 'Медведково', 'Китай-город', 774),

(2, 'метро Некрасовка', 'Орехово', 321),

(3, 'Шляпково', 'Салатовка', 123),

(4, 'Арханельская улица', 'ДЕПО Кирзанов', 21),

(5, 'Широкая улица', 'улица Вернадского', 80),

(6, 'метро Юго-западная', 'метро Раменки', 664);

*--*

*-- Dumping data for table validator*

*--*

INSERT INTO validator VALUES

(1, 'да'),

(2, 'да'),

(3, 'нет'),

(4, 'да'),

(5, 'да'),

(6, 'да'),

(7, 'да'),

(8, 'да'),

(9, 'да'),

(10, 'да'),

(11, 'да'),

(12, 'да'),

(13, 'да'),

(14, 'да'),

(15, 'да'),

(16, 'нет'),

(17, 'да'),

(18, 'да'),

(19, 'нет'),

(20, 'да'),

(21, 'нет'),

(22, 'да'),

(23, 'да'),

(24, 'нет'),

(25, 'да'),

(26, 'да'),

(27, 'да');

*--*

*-- Dumping data for table passenger\_insurance\_company*

*--*

INSERT INTO passenger\_insurance\_company VALUES

(1, 'АО С2ОГАЗ', 221479, '2006-09-20'),

(2, 'АО СОГАЗ', 223231, '2006-09-20'),

(3, 'АО ВТБ', 313279, '2016-03-20'),

(4, 'БВБК Страхование', 9696432, '2001-02-20'),

(5, 'CберСтрахование', 228322, '2020-01-20');

*--*

*-- Dumping data for table bus\_driver*

*--*

INSERT INTO bus\_driver VALUES

(1, 'Платон Крышесносов', 15, 234, 'AA10230D', 1, 2),

(2, 'Алишер Белозубов', 2, 234, 'AB2340A', 3, 1),

(3, 'Мария Блаженая', 10, NULL, 'AA15292H', 5, 3),

(4, 'Златон Ибрагимов', 9, 234, 'B249955D', 3, 1),

(5, 'Марк Золорылов', 21, 234, 'A1439141D', 4, 4),

(6, 'Игорь Индюков', 4, 234, 'AA103203D', 4, 5),

(7, 'Михаил Мафиозов', 7, 234, 'AA4525G', 5, 6),

(8, 'Эсми Мирзалиева', 25, 234, 'AA10423DD', 1, 6),

(9, 'Зокир Джамшутов', 4, 234, 'AA10425W', 2, 2),

(10, 'Мишутка Алфертов', 30, 234, 'A4141D', 1, 2),

(11, 'Али Белокуров', 6, 234, 'B7474', 3, 1),

(12, 'Мар Блаженая', 10, NULL, 'AA15192H', 5, 3),

(13, 'Злат ЗАбрагимов', 10, NULL, 'B219955D', 3, 1),

(14, 'Марк рылов', 21, 234, 'A1439741D', 4, 4),

(15, 'Игорь Индюк', 4, 234, 'AA109203D', 4, 5),

(16, 'Евгений Павлов', 7, 234, 'AA43525G', 5, 6),

(17, 'Эсвоьд Мирзалиев', 25, 234, 'A150423DD', 1, 6),

(18, 'Оскар Джамшутов', 5, 234, 'AA10485W', 2, 2);

*--*

*-- Dumping data for table advertising\_on\_the\_bus*

*--*

INSERT INTO advertising\_on\_the\_bus VALUES

(1, 'МТС', '2003-02-20', '2003-02-20'),

(2, 'Cбербанк', '2012-01-20', '2012-02-20'),

(3, 'ВафляShop', '2003-02-20', '2003-03-20'),

(4, 'ShocoShop', '2003-10-20', '2006-12-20'),

(5, 'ShipBorard', '2008-09-20', '2003-02-20'),

(6, 'MagomedShop', '2003-02-20', '2010-06-20');

*--*

*-- Dumping data for table bus*

*--*

INSERT INTO bus VALUES

(1, 772, 'Электробус', 1, 1, 4, 2),

(3, 772, 'Обычный автобус', 2, 2, 3, 1),

(4, 772, 'Электробус', 3, 3, 6, 2),

(5, 772, 'Электробус', 4, 4, 6, 3),

(6, 72, 'Электробус', 5, 5, 5, 5),

(7, 1, 'Обычный автобус', 6, 6, 5, 1),

(8, 3, 'Электробус', 7, 7, 2, 3),

(9, 10, 'Обычный автоьус', 8, 8, 1, 2),

(10, 553, 'Электробус', 9, 9, 4, 5),

(11, 1, 'Обычный автобус', 2, 2, 3, 1),

(20, 456, 'Обычный автобус', 6, 10, 3, 1),

(21, 966, 'Электробус', 9, 11, 5, 5),

(22, 453, 'Обычный автобус', 9, 6, 5, 1),

(23, 353, 'Электробус', 1, 13, 2, 3),

(24, 105, 'Обычный автоьус', 14, 8, 1, 2),

(25, 553, 'Электробус', 3, 16, 4, 5),

(26, 666, 'Электробус', 6, 18, 6, 2),

(27, 78, 'Электробус', 7, 4, 6, 3);

*--*

*-- Restore previous SQL mode*

*--*

*/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/*;

*--*

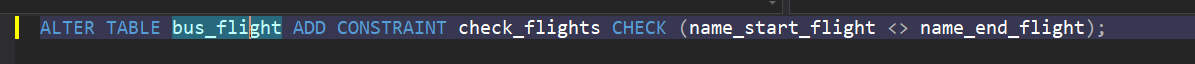
*-- Enable foreign keys*

*--*

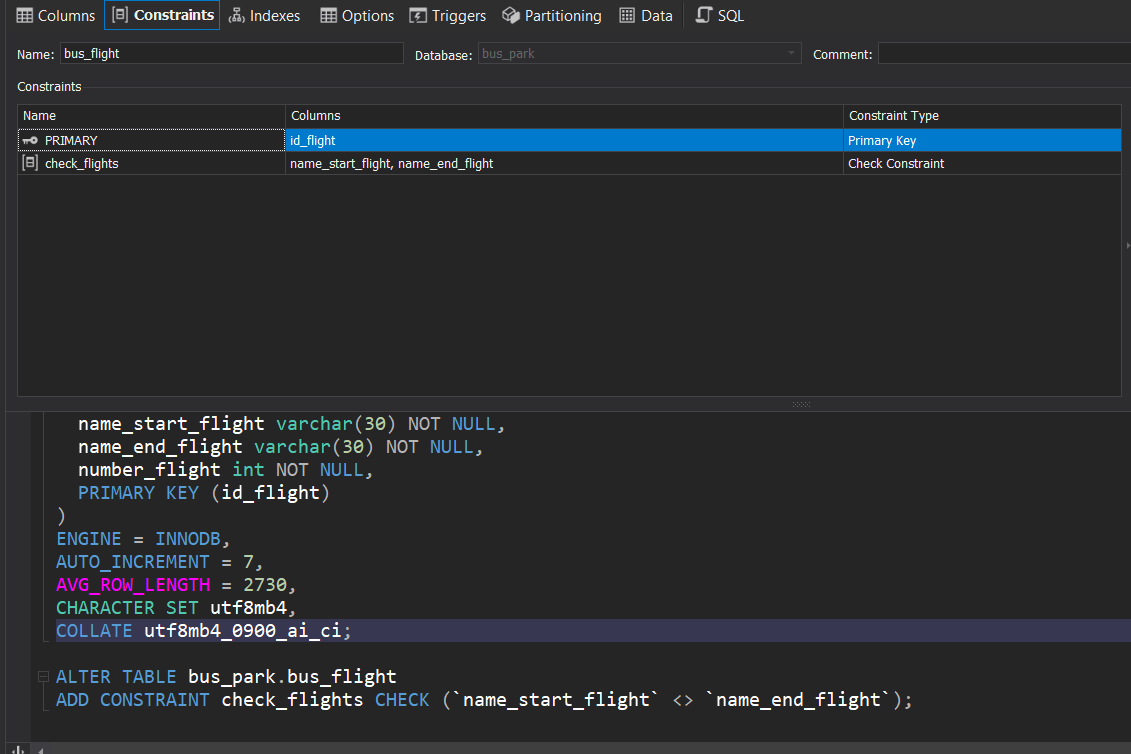
*/\*!40014 SET FOREIGN\_KEY\_CHECKS = @OLD\_FOREIGN\_KEY\_CHECKS \*/*;

Пример 1. Добавим ограничение уникальности на номер водительских прав водителя:

Пример 2. Добавим ограничение проверки условия для таблицы автобусных рейсов , означающее, что название начала рейса не должно быть равно названию конца рейса



Покажем успешность добавления:

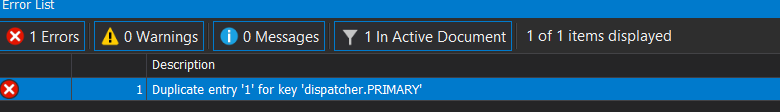


**1.3. Команды модификации данных (DML)**

Попытаемся вставить в таблицу диспетчеров с id, который уже существует в базе:



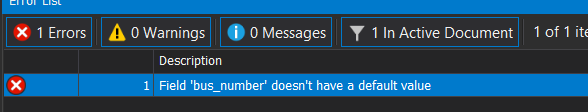
Ошибка, которая показывает, в чем мы ошиблись:



Попробуем сделать запись с нарушением ограничения внешнего ключа. Для примера возьмем таблицу bus и попытаемся вставить id водителя, которого не существует:



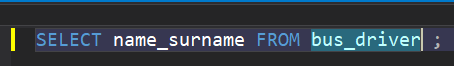
Ошибка:

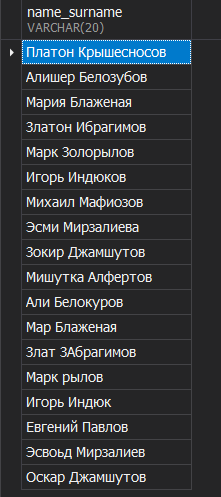


**1.4. Выборка данных. Оператор SELECT (DQL)**

**Запрос 1. Операции проекции.** Осуществляется выбор только части полей таблицы, т.е. производится вертикальная выборка данных.

Распечатаем имена всех водителей:



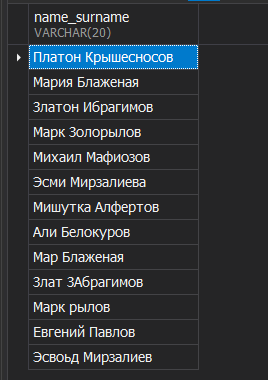


**Запрос 2. Операции селекции.** Осуществляется горизонтальная выборка

– в результат попадают только записи, удовлетворяющие условию.

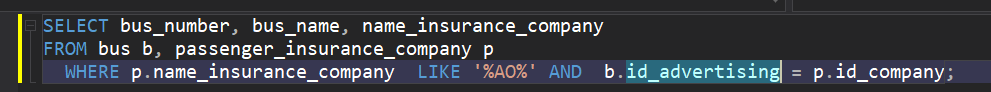
Для примера распечатаем имя и фамилию водителя у которого стаж работы

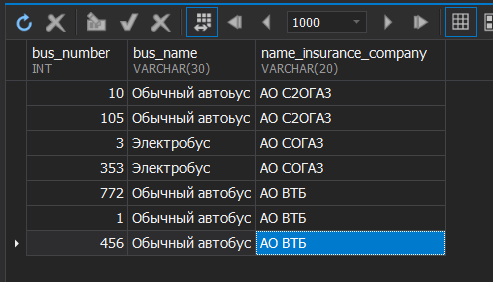




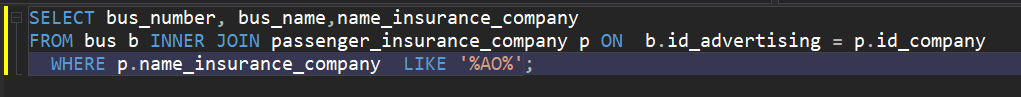
**Запрос 3. Операции соединения.** Здесь следует выделить декартово произведение и на его основе соединение по условию, а также естественное соединение (по одноименным полям или равенству полей с одинаковым смыслом).

Для примера выведем номер автобуса, его название и название страховой компании в этом автобусе, которая имеет аббревиатуру на «АО»





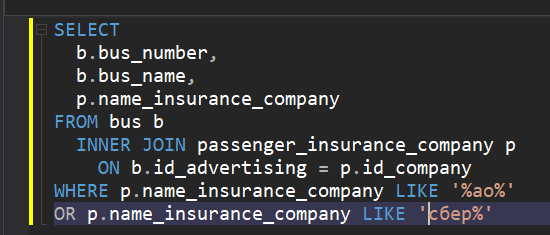
Запишем этот же запрос с помощью операции внутреннего соединения

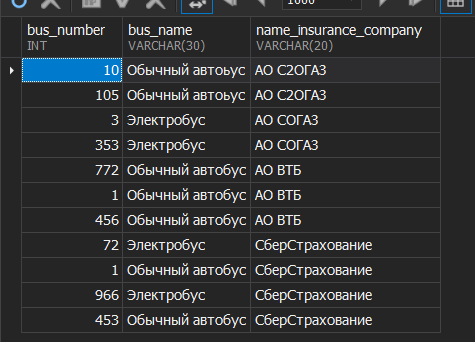


Результат будет таким же.

**Запрос 4. Операция объединения.** Теоретико-множественные операции

часто можно записать с помощью логических операций, примененных в конструкции WHERE запроса



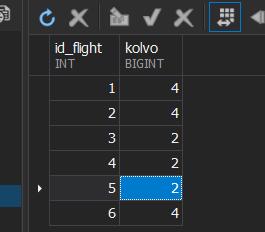


**Запрос 5. Операции группировки.** Эта операция связана со своеобразной

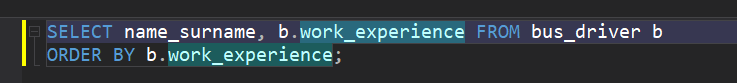
сверткой таблицы по полям группировки.

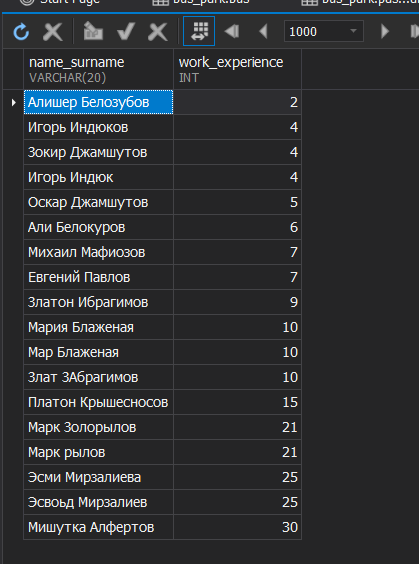
Выведем рейс и количество водителей, которые ездят на нем:





**Запрос 6. Операция сортировки.** Вывести имя и фамилию и стаж работы, отсортированной в возрастающем порядке по опыту работы.

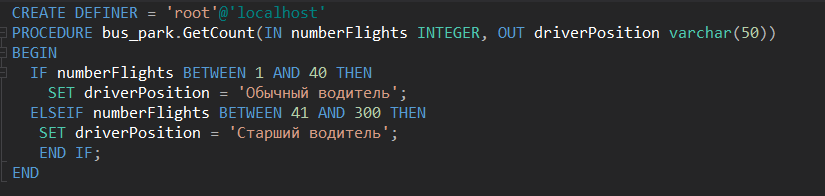




**1.5. ХРАНИМЫЕ ПРОЦЕДУРЫ, ФУНКЦИИ И ТРИГГЕРЫ**

**Пример 1.**  Напишем хранимую процедуру, которая получает в качестве входного параметра количество рейсов, которые проделал водитель автобуса и на основании шкалы проделанных рейсах вычисляет должность этого водителя.

Покажем скрипт для хранимой процедуры:

**

*Рис 10– скрипт для работы с хранимой процедурой*

Вывод:

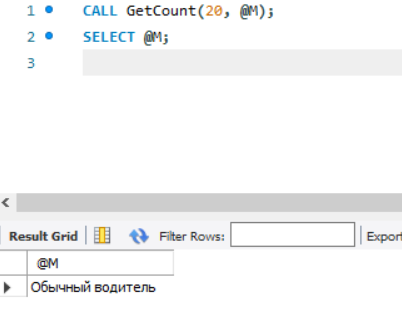


Рис – вызов процедуры и печать результата (1)

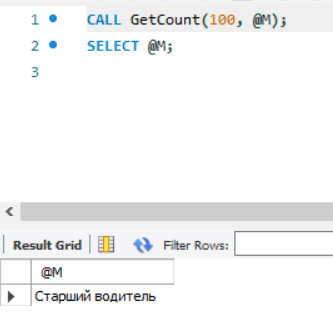


Рис – вызов процедуры и печать результата (2)

**Пример 2. (Функция)**

Cоздадим функцию для получения должности водители по количеству проделанных рейсов.

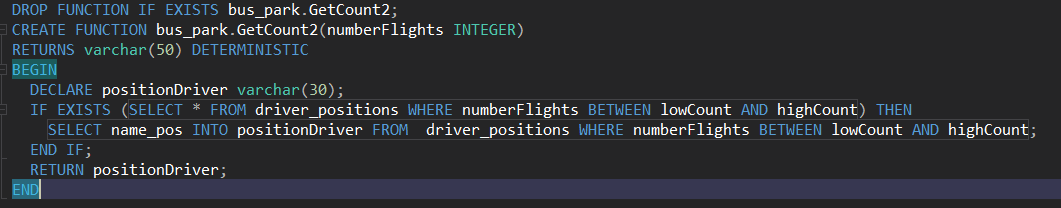


Рисунок – скрипт функции

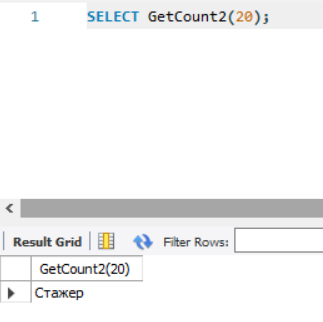


Рисунок – Вывод функции при количестве рейсов - 20

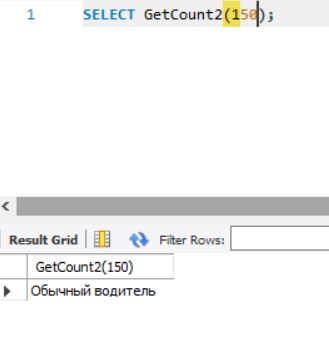


Рисунок – Вывод функции при количестве рейсов – 150

**Пример 3. (Триггер)**

В этом примере представлен скрипт триггера на вставку новой записи в таблицу количества рейсов водителя. Этот триггер должен срабатывать после вставки и быть связан с подсчетом актуального количества рейсов

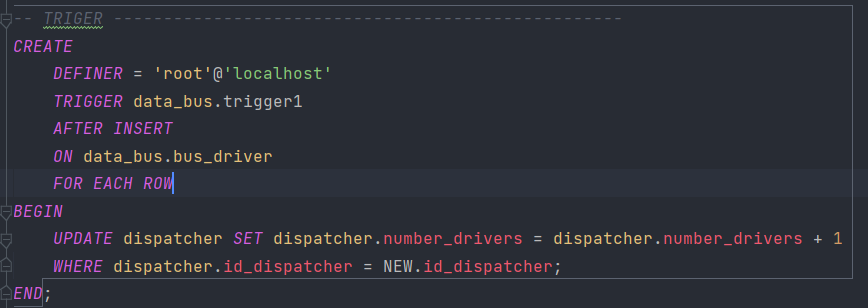
****

Рисунок – скрипт триггера

**Приложение к БД**

Сделаем пользовательский интерфейс к нашей составленной БД.

Функционал, который будет реализован в приложении:

* + Регистрация и авторизация пользовател в системе
  + добавление, изменение, удаление, обновление информации по теме;
  + фильтрация списков по соответствующим признакам
  + работоспособность триггеров, функций и процедур в приложении

Веб- приложение будет реализовано при помощи языка программирования Java, среды разработки Inlelij IDEA и фреймворка Spring Boot со всеми его составляющими.

1. ***Spring Boot*** - это полезный проект, целью которого является упрощение создания приложений на основе Spring. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода.

2. ***Spring Data JPA*** - это библиотека, которая добавляет дополнительный уровень абстракции поверх ORM реализации JPA. По умолчанию Spring Data JPA использует Hibernate, в качестве ORM провайдера

3. ***Thymeleaf*** - современный серверный механизм Java-шаблонов для веб- и автономных сред, способный обрабатывать HTML, XML, JavaScript, CSS и даже простой текст. В нашем приложении он нужен для динамического отображения шаблонов.

4. ***MySQL*** - подключает драйвера Java Database Connectivity для выполнения SQL-запросов к базе данных.

5. ***Maven*** - инструмент для автоматизации сборки проектов

Главная страница, показывающаяся пользователю при начальном входе:

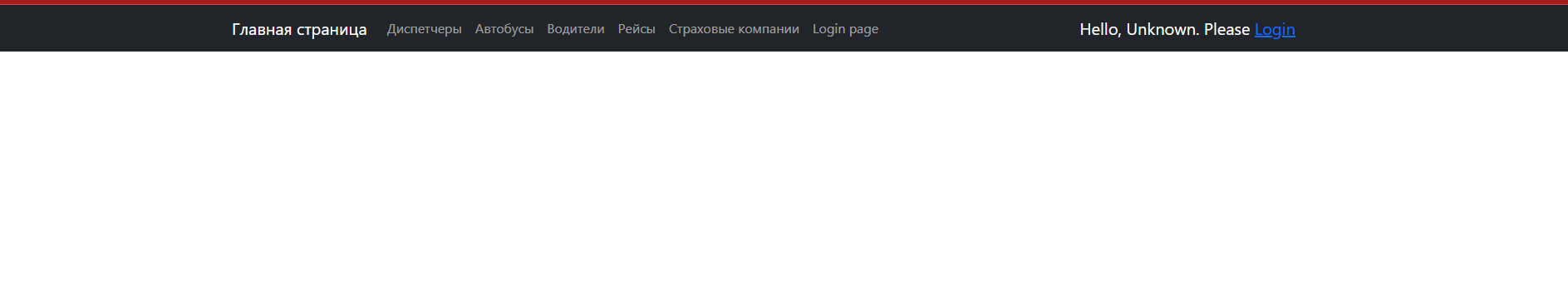


Рисунок – главная страница

Зайдем под пользователем admin, который имеет возможность редактировать, добавлять и обновлять информацию о сущностях.

Следующим шагом покажем интерфейс ранее добавленных в практических заданиях сущностях:

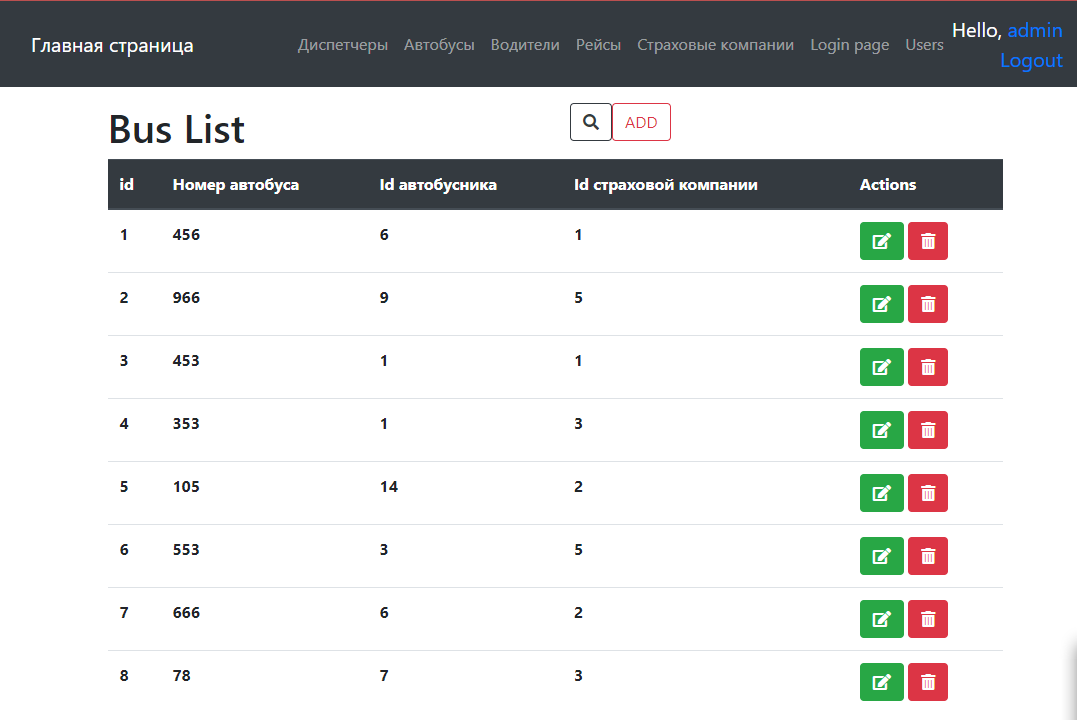


Рисунок – Страница со списком автобусов.

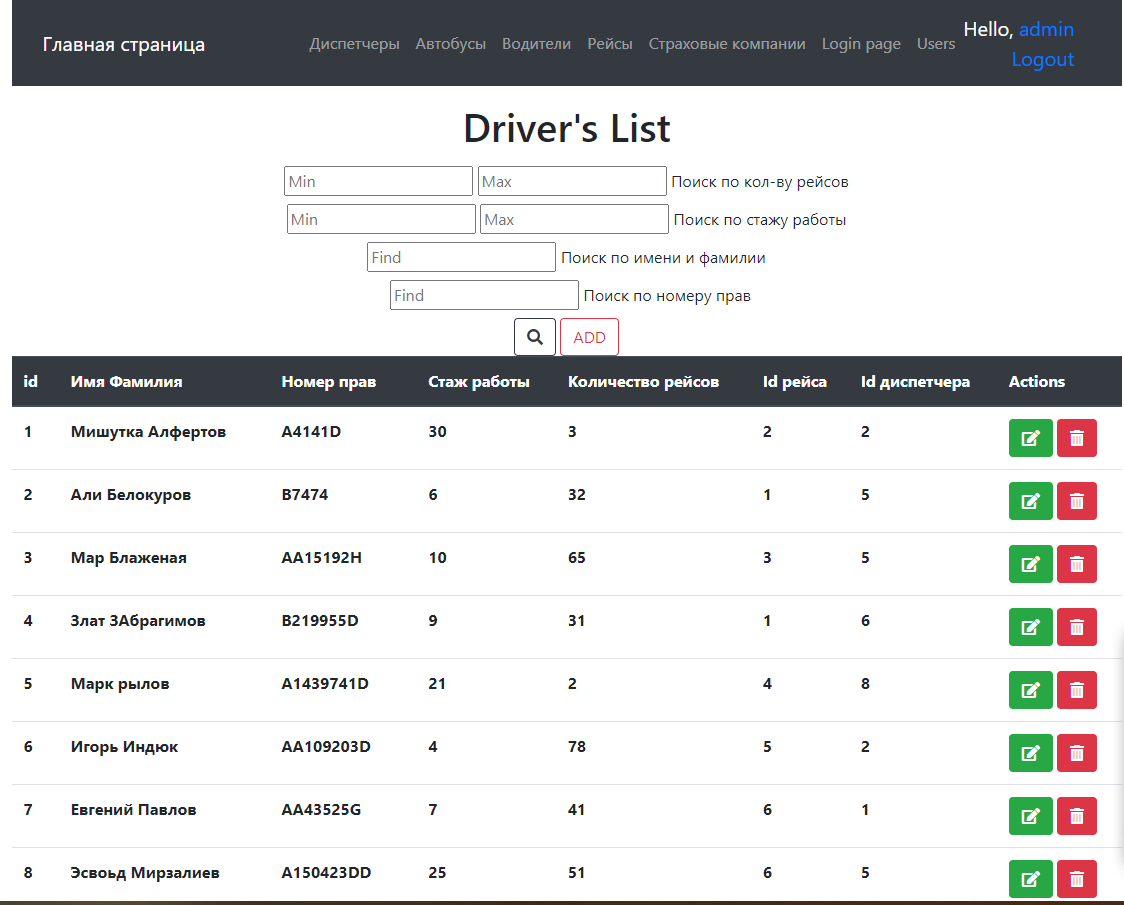


Рисунок – Список всех водителей, которые были добавлены

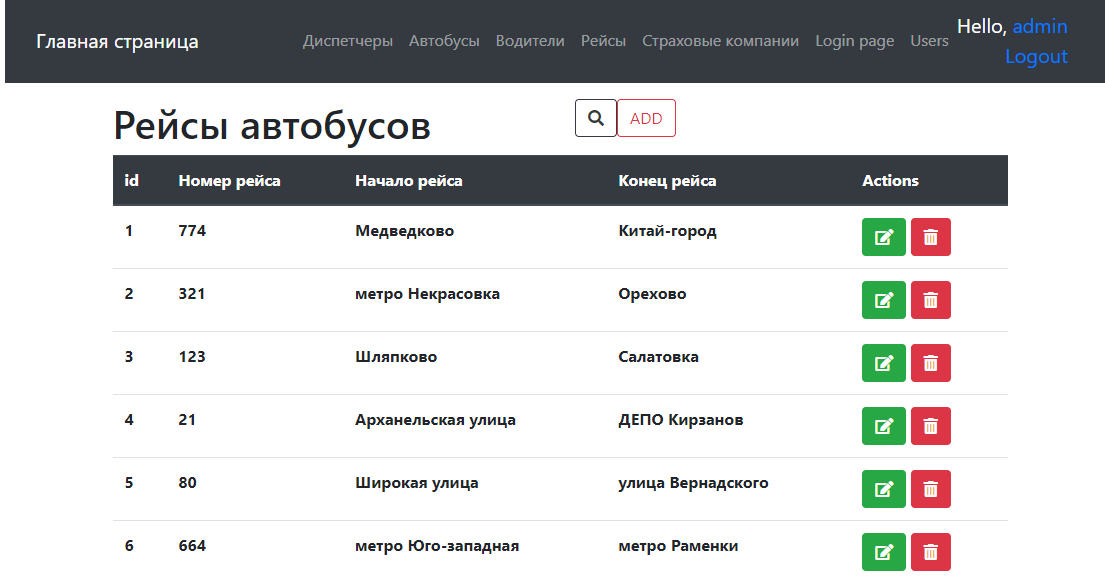


Рисунок – Список рейсов автобусов

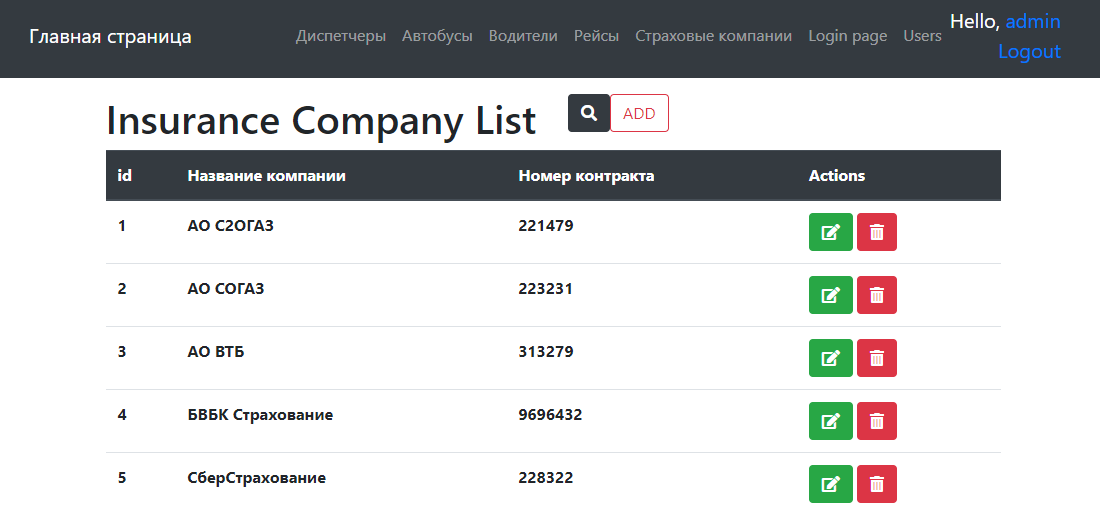


Рисунок – список страховых компаний

Попробуем добавить новую информацию о рейсах:

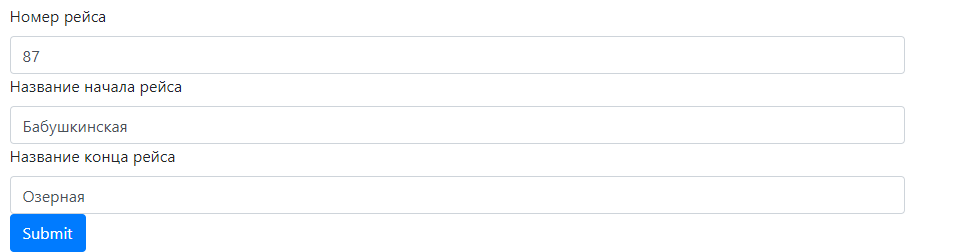


Рисунок - форма с добавлением нового рейса

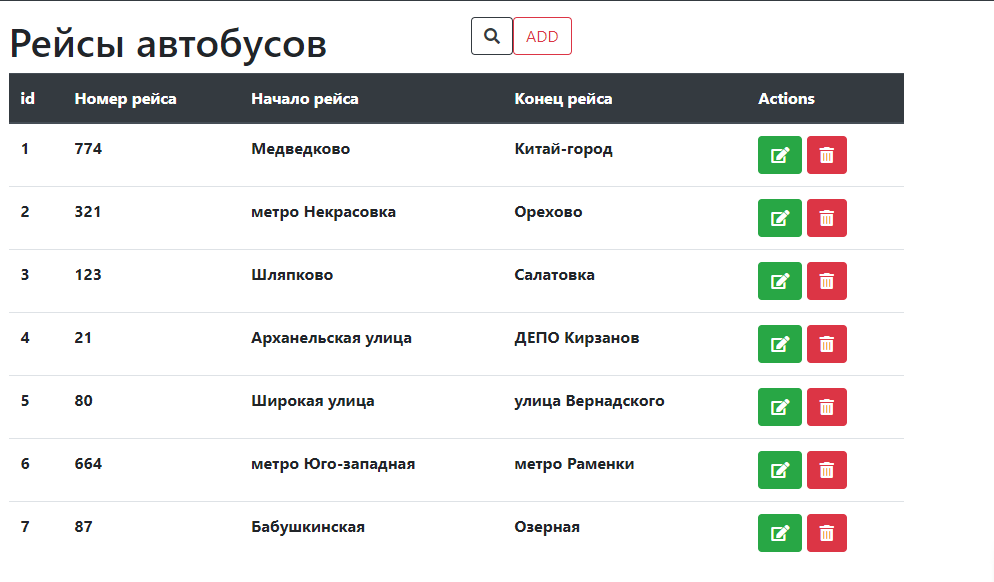


Рисунок – Успешно добавленная новая запись о рейсе

Покажем работоспобность обновления информации – попробуем обновить рейс, который мы только что создали и затем произведем удаление.

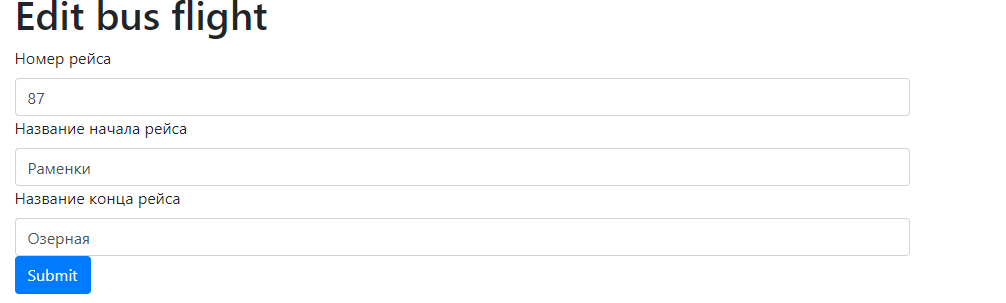


Рисунок – обновление информации о рейсе

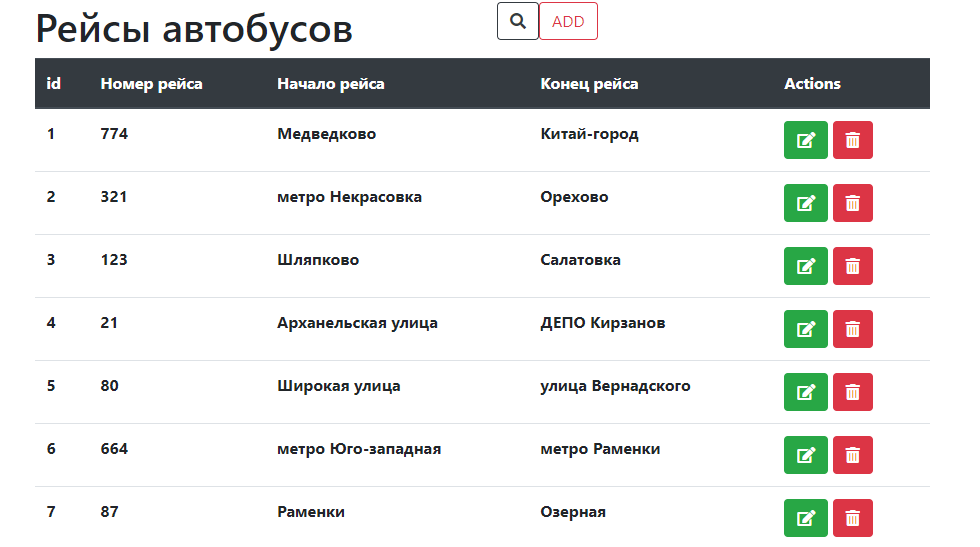


Рисунок – обновленные данные рейсов

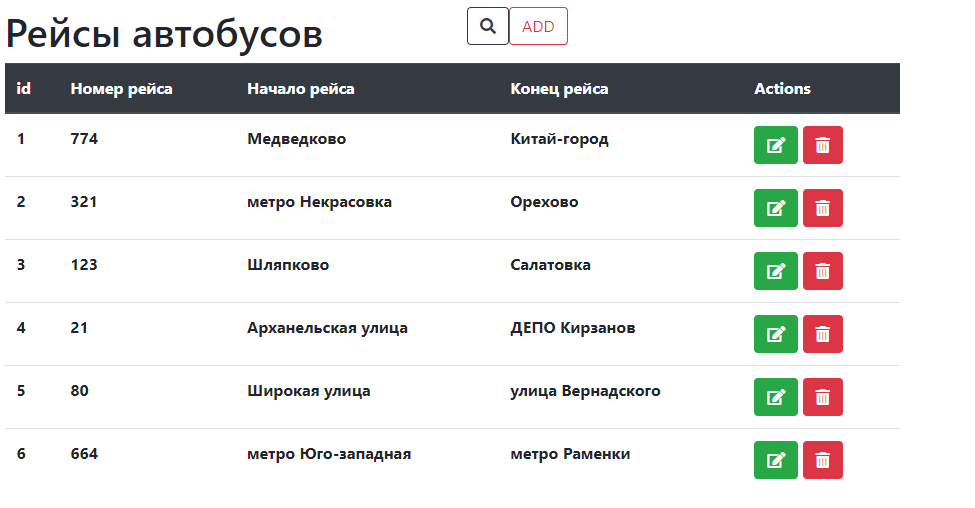


Рисунок – удаление новой записи

Покажем функционал поиска записей и фильтрации. Для примера возьмем таблицу с автобусниками.

Покажем всех водителей, у которых кол-во проделанных рейсов находится в промежутке от 2 до 50:

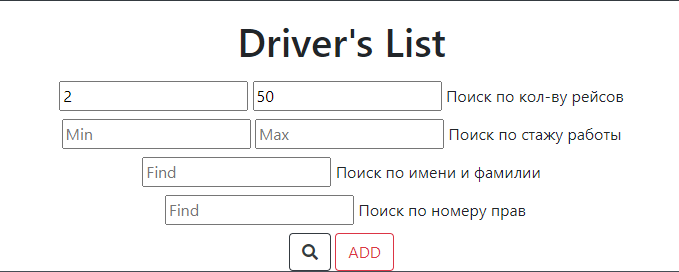


Рисунок – поиск по кол-ву рейсов

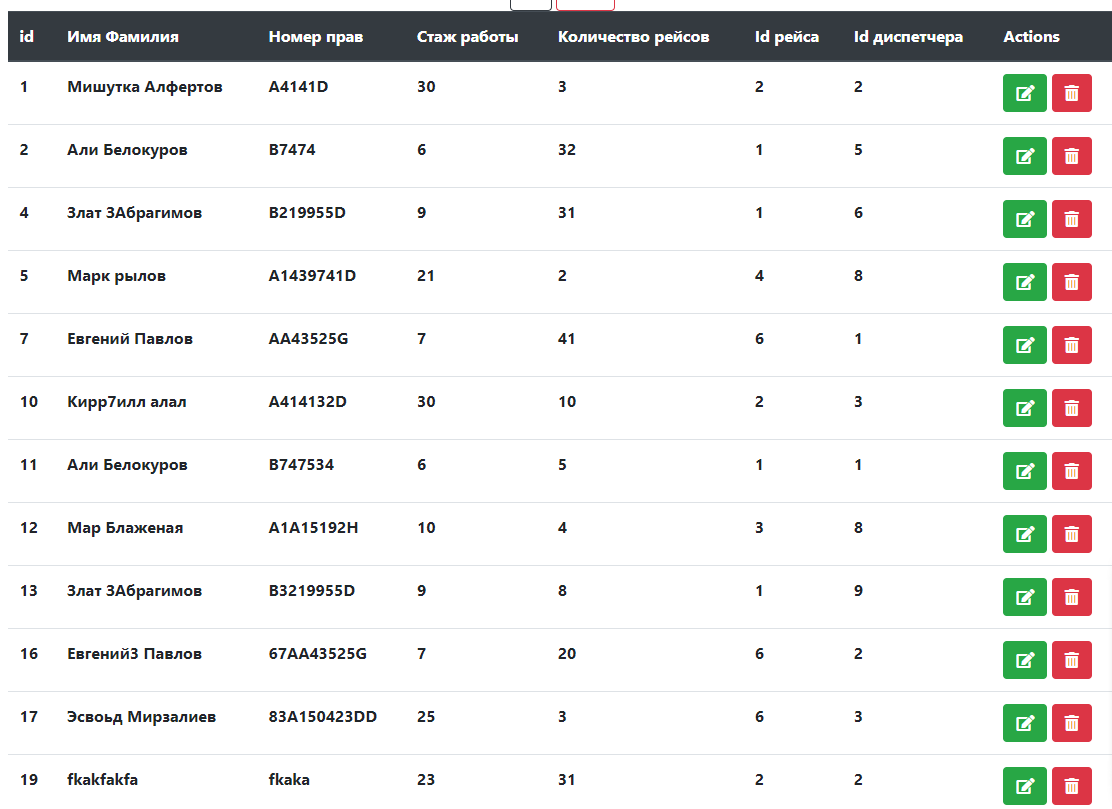
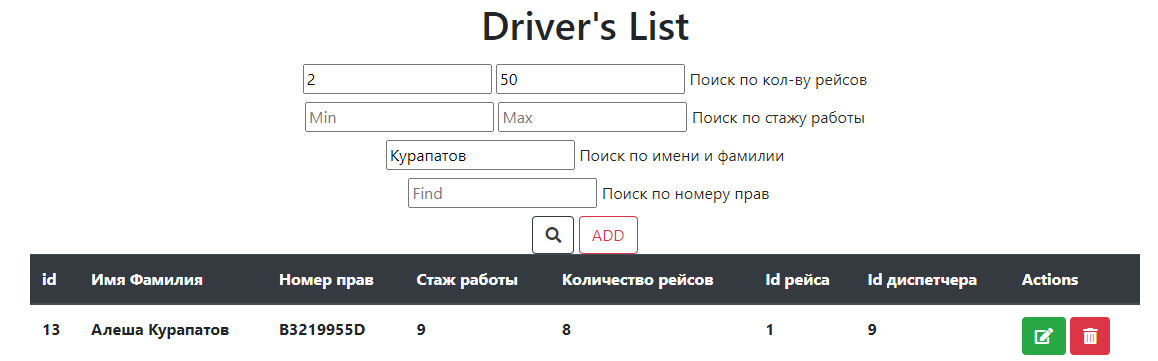


Рисунок – таблица с отфильтрованными по кол-ву стажа работы

Попробуем найти водителя с именем Курапатов:

 Рисунок – таблица с отфильтрованными по имени и фамилии

Покажем поиск по номеру прав:

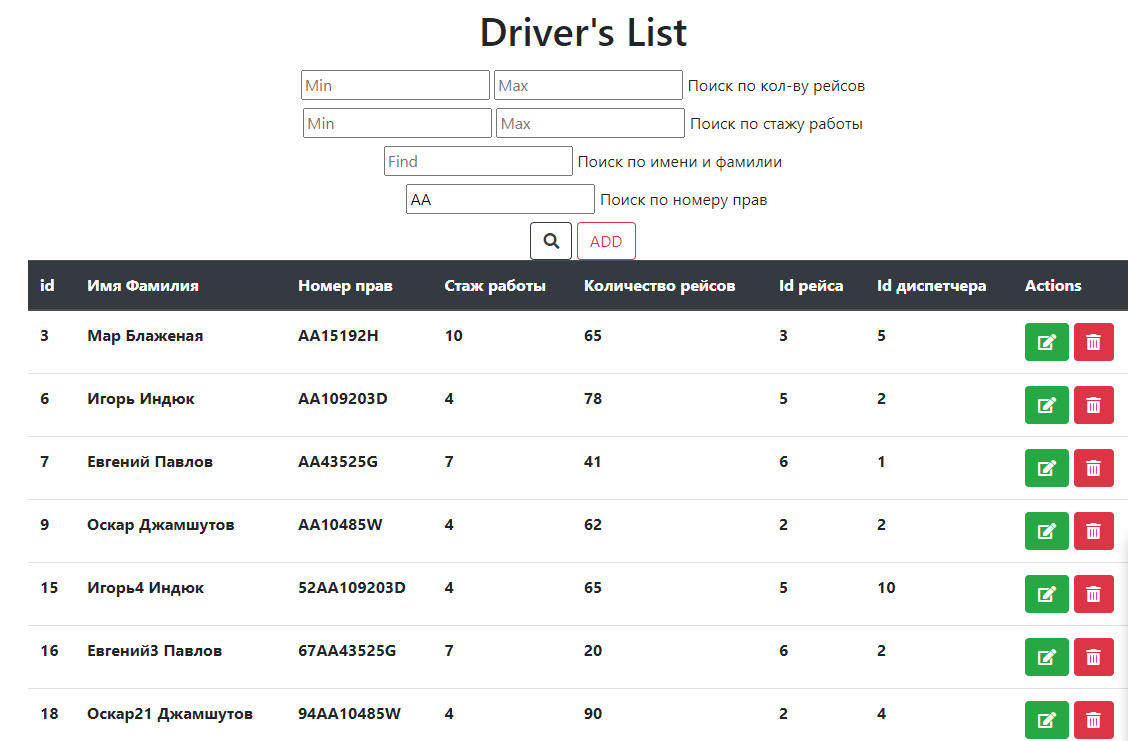


Рисунок – таблица с отфильтрованными данными по номеру прав

Попробуем сделать конбинацию фильтров: найдем всех водителей у которых стаж работы от от 10 до 20, у которого в имени и фамилии есть словосочетание «И» и у которого кол-во рейсов от 50:

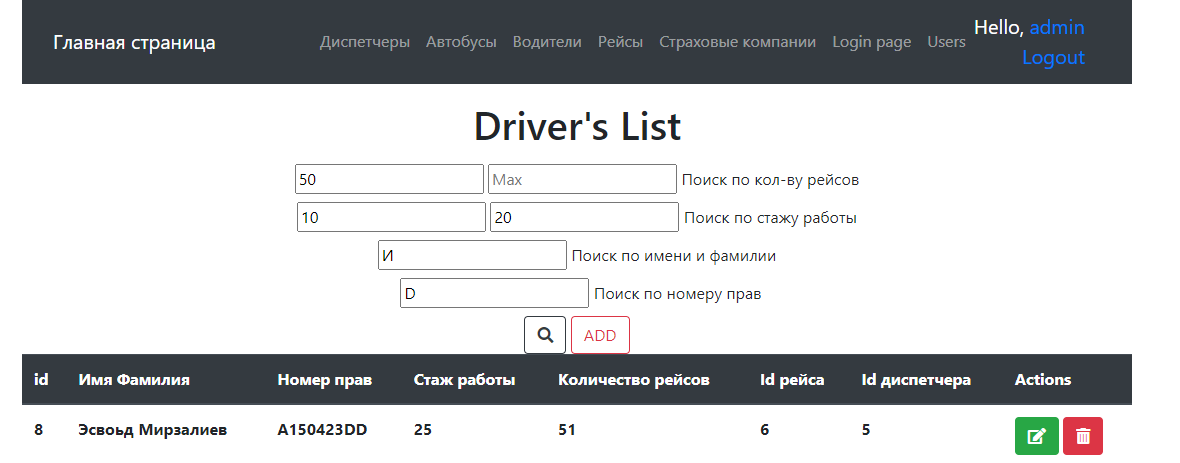


Рисунок – таблица с множественным фильтром

Покажем работу триггера, который автоматически увеличивает кол-во привязанных водителей у конкретного диспетчера.

Изначальная таблица с диспетчерами:

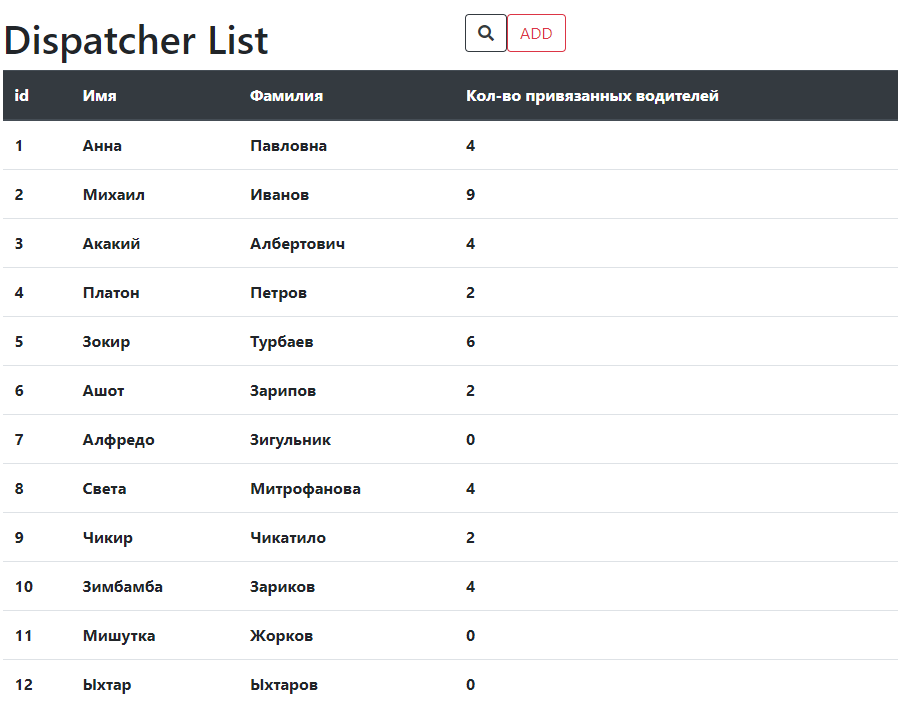


Рисунок – таблица с диспетчерами



Рисунок – добавление нового водителя, у которого диспетчер с id = 9

Покажем измененную таблицу с диспетчерами:

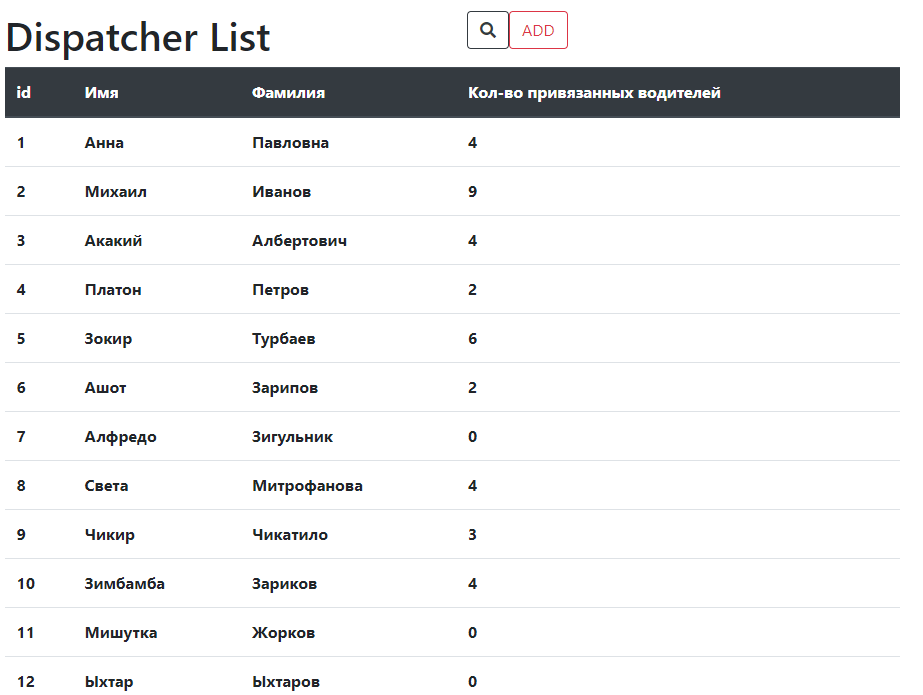


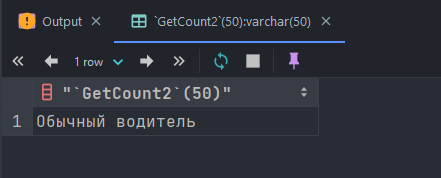
Рисунок - таблица с обновленной информацией

Как видим у диспетчера Чикира Чикатило увеличилось кол-во привязанных водителей

Код, который был добавлен в консоли самого приложения для показа работоспобности триггеров, функций и процедур:

*-- TRIGER ---------------------------------------------------  
CREATE  
 DEFINER* = 'root'@'localhost'  
 *TRIGGER* data\_bus.trigger1  
 *AFTER INSERT  
 ON* data\_bus.bus\_driver  
 *FOR EACH ROW  
BEGIN  
 UPDATE* dispatcher *SET* dispatcher.number\_drivers = dispatcher.number\_drivers + 1  
 *WHERE* dispatcher.id\_dispatcher = NEW.id\_dispatcher;  
*END*;  
*-- PROCEDURE -----------------------------------------------  
CREATE DEFINER*=`root`@`localhost` *PROCEDURE* `select\_user`(*IN* busId *int*)  
*BEGIN  
SELECT* `bus\_number` *FROM* bus b *WHERE* id\_bus = busId;  
*END*;  
*CAll* `select\_user`(2);  
*-- --------------------------------  
-- FUNCTION -------------------------------------------------------------------------------------  
CREATE DEFINER*=`root`@`localhost` *FUNCTION* `GetCount2`(numberFlights *INTEGER*) *RETURNS varchar*(50) *CHARSET* utf8mb4  
 *DETERMINISTIC  
BEGIN  
 DECLARE* positionDriver *varchar*(30);  
 *IF EXISTS* (*SELECT* \* *FROM* driver\_positions *WHERE* numberFlights *BETWEEN* lowCount *AND* highCount) *THEN  
 SELECT* name\_pos *INTO* positionDriver *FROM* driver\_positions *WHERE* numberFlights *BETWEEN* lowCount *AND* highCount;  
 *END IF*;  
 *RETURN* positionDriver;  
*END*;  
*SELECT* `GetCount2`();  
*-- ------------------*

Работоспособность функции, которая была реализована в практической работе №3:



Процедура:

