Москва 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью ВКР является – Разработка базы данных, предоставляющей быстрый доступ к каталогу автосалона для оформления заказа.

Поводом для автоматизации организации заключается необходимость быстрого и подробного оформления заказа в автосалоне, минимизирующего очередь среди клиентов, путем его автоматизации при помощи базы данных.

эта работа слишком трудоемкая и не защищена от неточностей и ошибок

обусловленным человеческим фактором. Для облегчения и ускорения

работы, было принято решение создать базу данных, которая обеспечивает:

1)Надежность и безопасность;

2)Автоматизацию документооборота ;

3)Быстрое составление отчетов.

Автоматизация этих функций призвана, увеличить скорость и качество

обработки информации. Результатом окончания ВКР будет – прототип разработанной базы данных.

Какие задачи включает в себя работа над ВКР:

1) Анализ предметной области;

2) Анализ существующих базы данных;

3) Анализ схемы базы данных;

4) Проектирование базы данных;

5) Разработка базы данных;

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Исследование предметной области

1.1.1 Анализ основных понятий

Автомобиль – моторное дорожное и внедорожное транспортное средство, используемое для перевозки людей и грузов**.**

Клиенты – Люди, совершающие покупку в автосалоне.

1.1.2 Анализ основных объектов

База данных «Сервис такси» создается для ускорения и упрощения процесса покупки автомобиля клиентом через менеджера автосалона. Данные о заявке позволят более оперативно отслеживать исполнение этих заявок, а также проводить анализ данных по запросам покупателей.

Таблица №1 — Содержательное проектирование иерархического меню.

Пункт главного меню. Пункт подменю.	Экранная форма для ввода информации	Выходная форма (отчёт)
Заполнение справочников Справочник «Автомобили» Справочник «Водители» Справочник «Улицы» Выход	**Подменю** Справочник «Автомобили» Справочник «Водители» Справочник «Улицы» _____	_____ Справочник «Автомобили» Справочник «Водители» Справочник «Улицы» Главное меню
Заявки Открыть справочник «Водители» Выход	**Заявки** _____ _____	**Заявка** Справочник «Водители» Главное меню
Заявки по дате	**Дата**	**Отчет о заявках по дате**
Заявки по личному номеру	**Личный номер**	**Отчет о заявках по №**
Заявки	**Дата**	**Отчет о заявках по дате**
Заявки по номеру водителя	**Личный номер**	**Отчет о заявках по №**
Выход	_____	**Системное меню**

Проблемы, которые могут возникнуть при осуществлении данной деятельности такие:

1) Потеря данных менеджером, или неправильный ввод данных в смету;

2) Выбор недействительного автомобиля;

3) Не корректная форма представления отчетов и т.д.

Основными видами деятельности являются:

1) Обеспечение покупателя необходимым транспортом;

2) Обеспечение быстроты и точности составления заказа;

3) Своевременное выполнение заявок.

1.1.3 Анализ основных действий с объектами

Поводом для автоматизации организации Автосалона послужило необходимость в данной подсистеме, и облегчения ежедневной работы, т.к. эта работа слишком трудоемкая и не защищена от неточностей и ошибок обусловленным человеческим фактором. Для облегчения и ускорения работы, было принято решение создать базу данных, которая обеспечивает:

1)Надежность и безопасность;

2)Автоматизацию документооборота;

3)Быстрое составление отчетов.

Автоматизация этих функций призвана, увеличить скорость и качество обработки информации.

1.1.4 Анализ основных участников

Участниками предметной области будут:

1) **Клиент** — лицо, осуществляющее заказ.

2) **Менеджер** — Лицо, осуществляющее администрацию заказов в автосалоне.

1.2 Анализ существующих СУБД

Redis – это резидентная система класса NoSQL. Ориентирована на достижение максимальной производительности на атомарных операциях.

MySQL – это свободная реляционная система управления базами данных. Используется при разработке любых систем, которые должны хранить определенные данные, чаще всего это приложения или сайты.

MongoDB – это документно-ориентированная система управления базами данных, не требующая описания схемы таблиц. Используется как альтернатива MySQL для хранения информации с последующим расширением.

1.3 Обоснование выбора СУБД

В нашем случае наиболее выгодным вариантом будет использовать базу данных MySQL. В отличие от других баз данных у нее есть некоторые преимущества:

1) Возможность запуска на разных ОС;

2) Небольшие требования к системе;

3) Наличие различных движков, которые можно выбирать индивидуально для каждой таблицы;

4) Наличие языка SQL;

5) Использование многими крупными компаниями (Facebook, YouTube, Google);

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Формирование требований

2.1.1 Основные сущности базы данных

Описание сущностей можно упростить путем использования диаграмм, таким образом диаграммой сущностей для этого проекта будет диаграмма, которую можно просмотреть на Рисунке 2.1.

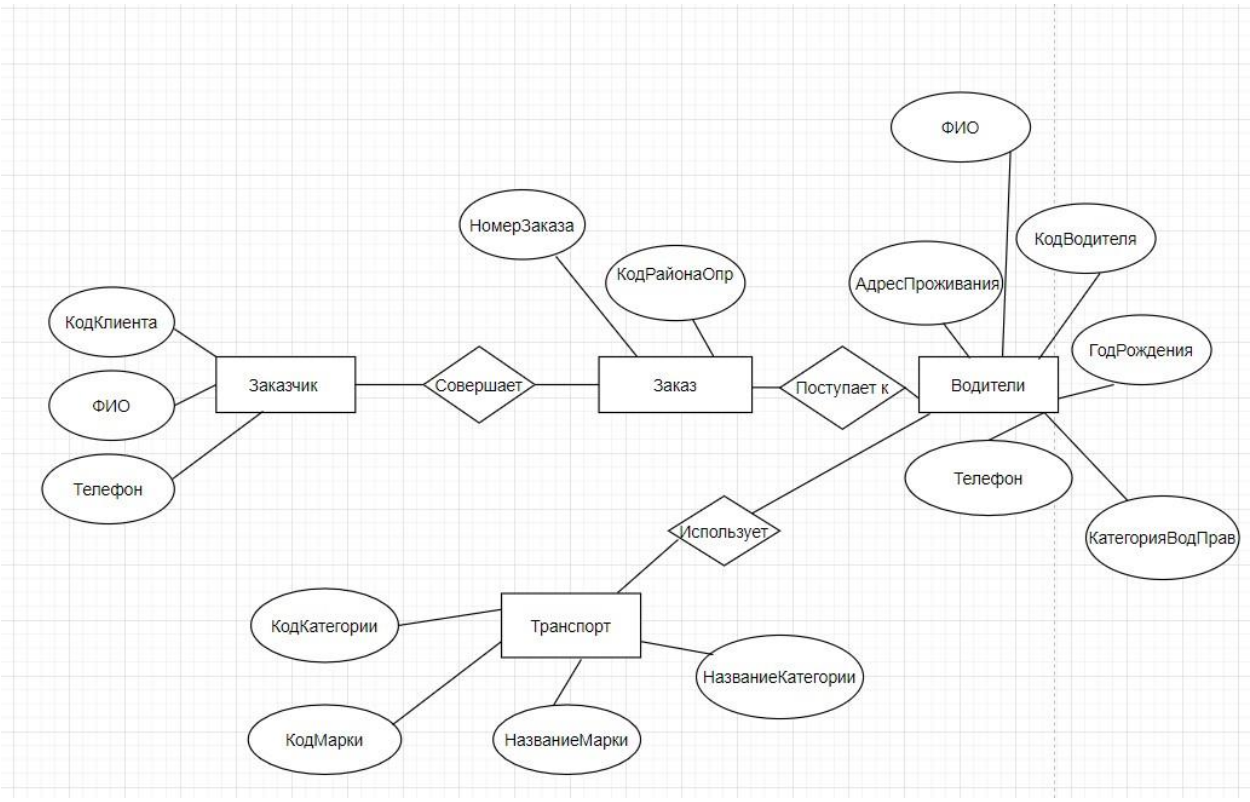


Рисунок 2.1 – Диаграмма сущностей

2.1.2 Варианты использования

Для разработки сценариев использования будет использоваться UML диаграмма использования, иначе use-cases. Она составляется для того, чтобы наглядно представить функциональные возможности разрабатываемой системы.

Для нашей базы данных сценарии использования будут такими, как представлено на диаграмме на рисунке 2.2.

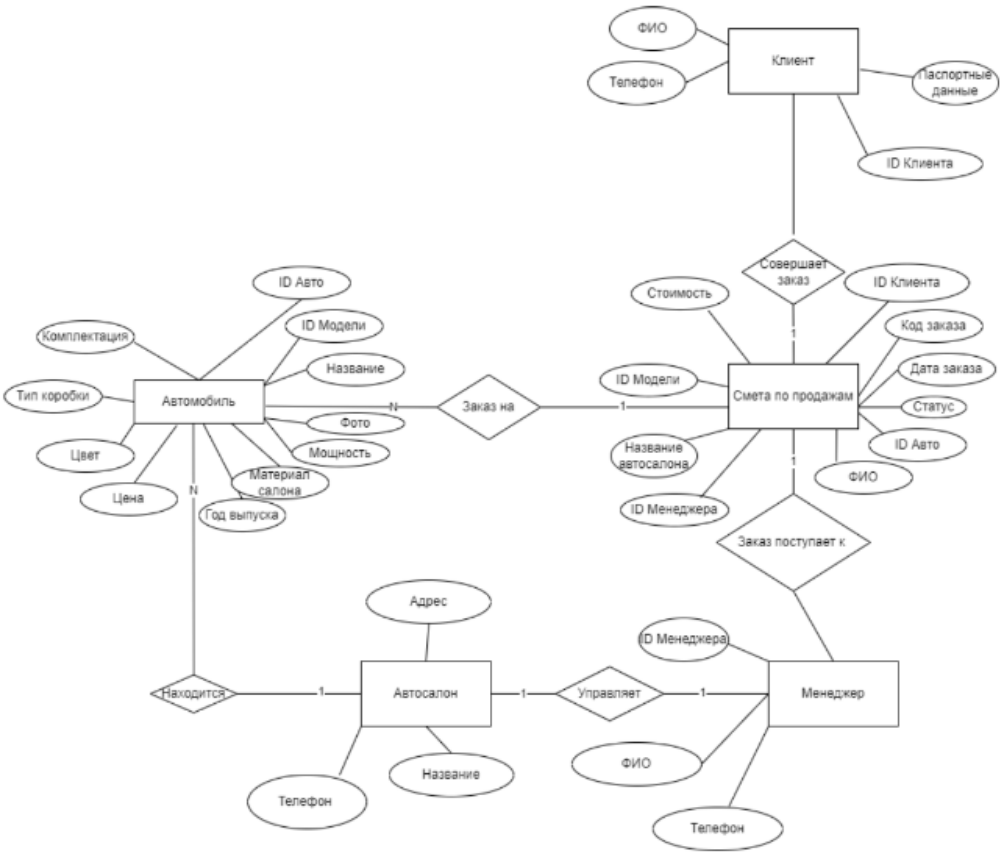


Рисунок 2.2 – Диаграмма сущностей

2.1.3 Определение API для взаимодействия с базой данных

Список методов разделен по таблицам 2.1, 2.2, 2.3, 2.4 для упрощения понимания.

Таблица 2.1 – Список методов

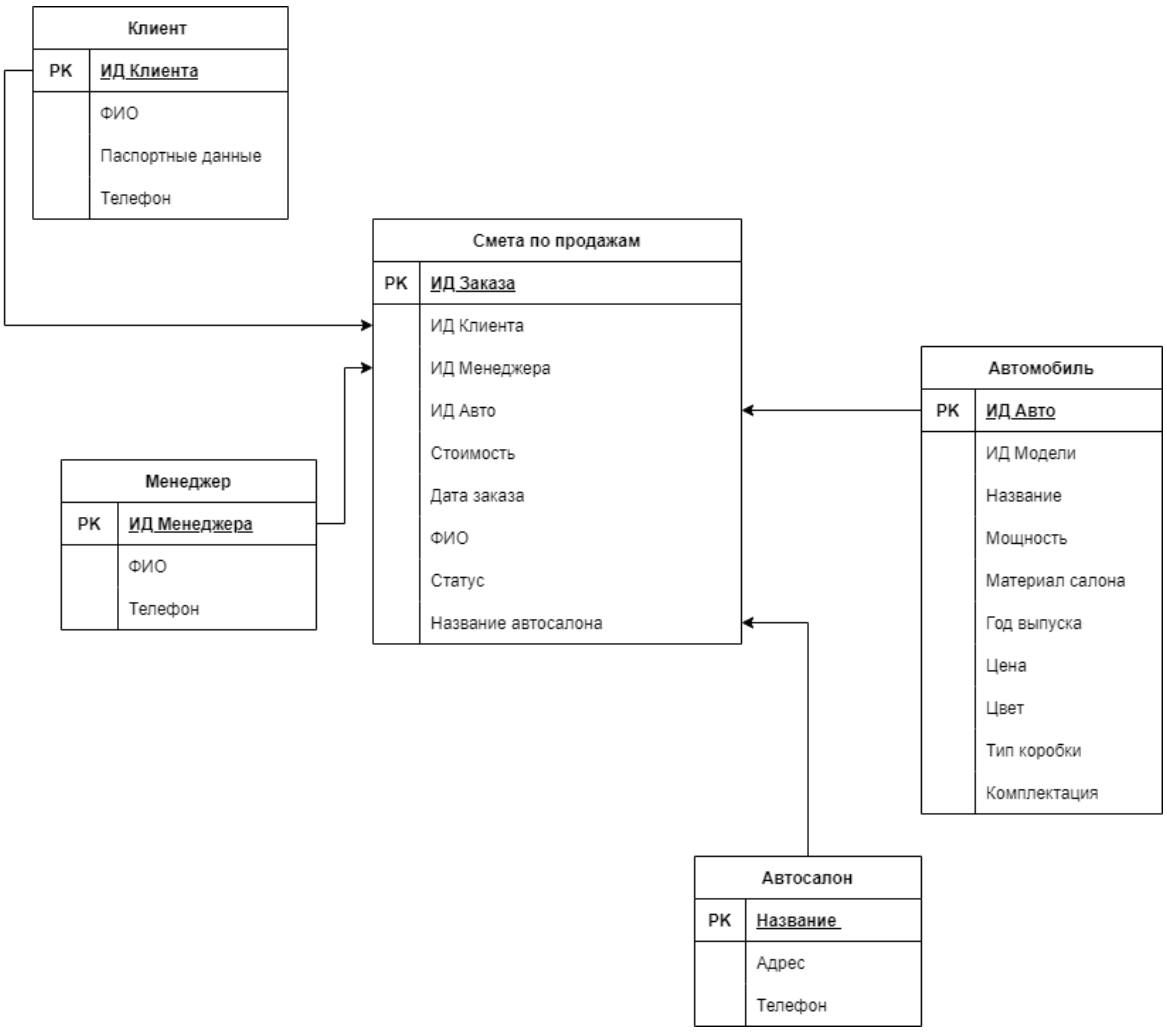
Метод	**Описание**	**Аргумент**
Поиск авто	Доступ: Менеджер Клиент	ID Модели , ID Авто.
Заключение договора	Доступ: Клиент Менеджер	Код заказа, ID Менеджера, ID Клиента.

Расторжение договора	Доступ: Менеджер, Клиент	Код заказа, ID Менеджера, ID Клиента, Статус.
Удалить транспорт	Доступ: Менеджер	ID Авто.
Продать транспорт	Доступ: Менеджер	ID Авто, ID Менеджера, Цена, ID Клиента.
Изменить данные(Клиент)	Доступ: Менеджер	ID Клиента, ФИО, Телефон, Паспортные данные.
Добавить данные(Клиент)	Доступ: Менеджер	ID Клиента, ФИО, Телефон, Паспортные данные.

2.2 Проектирование базы данных

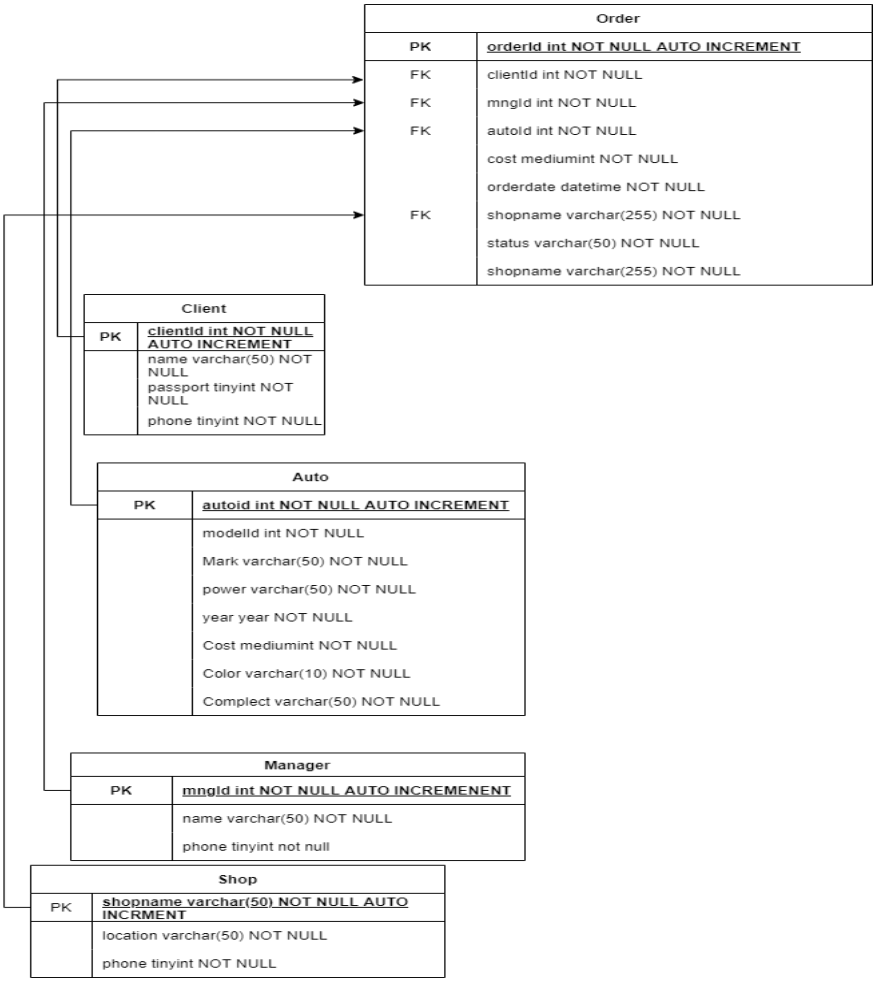
2.2.1 Логическая схема базы данных

Разработаем логическую схему базы данных. Диаграмма представлена на изображении 2.1



Изображение 2.1 – Логическая модель.

2.2.2 Физическая схема базы данных



Изображение 2.2 – Физическая модель.

2.2.3 Определение типовых запросов к объектам базы данных

Основные запросы к базе данных описаны в таблице 2.2.

Представление	**Таблицы**	**Содержание**	**Поля**
orders	– Заказы; – Районы; – Заказчики; – Водители; – Транспорт;	Заказы с полными подробностями о заказчике и водителе.	orderId (int)
			driverId (int)
			customerId (int)
			locationId (int)
			cardId (int)
			customerFio (varchar(255))
			customerPhone (varchar(255))

			customerLocation (varchar(255))
			driverFio (varchar(255))
			driverPhone (int)
			driverBirth (datetime)
			driverCategory (varchar(255))
			carName (varchar(255))
			carCategory (varchar(255))
drivers	– Водители; – Транспорт;	Список водителей в базе.	driverId (int)
			driverFio (varchar(255))
			driverPhone (int)
			driverBirth (datetime)
			driverCategory (varchar(255))
			carName (varchar(255))
			carCategory (varchar(255))

2.2.4 Определение процедур и функций API

Функции API базы данных представлены в таблице 2.3.

Таблица 2.3 – Определения функций API.

Таблица	Запрос	Пример
Клиенты	Список Клиентов	SELECT * FROM `Client`
	Добавить клиента	INSERT INTO `Client` (`name`, `passport`, `phone`) VALUES (‘Павлов Алексей Дмитриевич’, ’4512567890’, ’89775462854’)
	Изменить данные клиента	UPDATE `Client` SET name = IsNull(@name, name), passport = IsNull(@passport, passport) WHERE clientId=1
	Удалить клиента	DELETE FROM `Client` WHERE clientId=1
Автомобили	Список автомобилей	SELECT * FROM `Auto`
	Добавить автомобиль	INSERT INTO `Auto` (`modelId`, `mark`, `cost`, `color`, `year`, `type`,) VALUES (‘001’, ’Toyota’, ’4000000’, ’Black’, ’1977’, ’Mechanic’)
	Править тех. данные автомобиля	UPDATE `Auto` SET name = IsNull(@mark, mark), year = IsNull(@year, year) WHERE autoId=1
	Удалить автомобиль	DELETE FROM `Auto` WHERE autoId=1

2.3 Разработка базы данных

(Приводите пример выполнения запросов к таблицам, запросов к представлениям, вызовы процедур и функций)

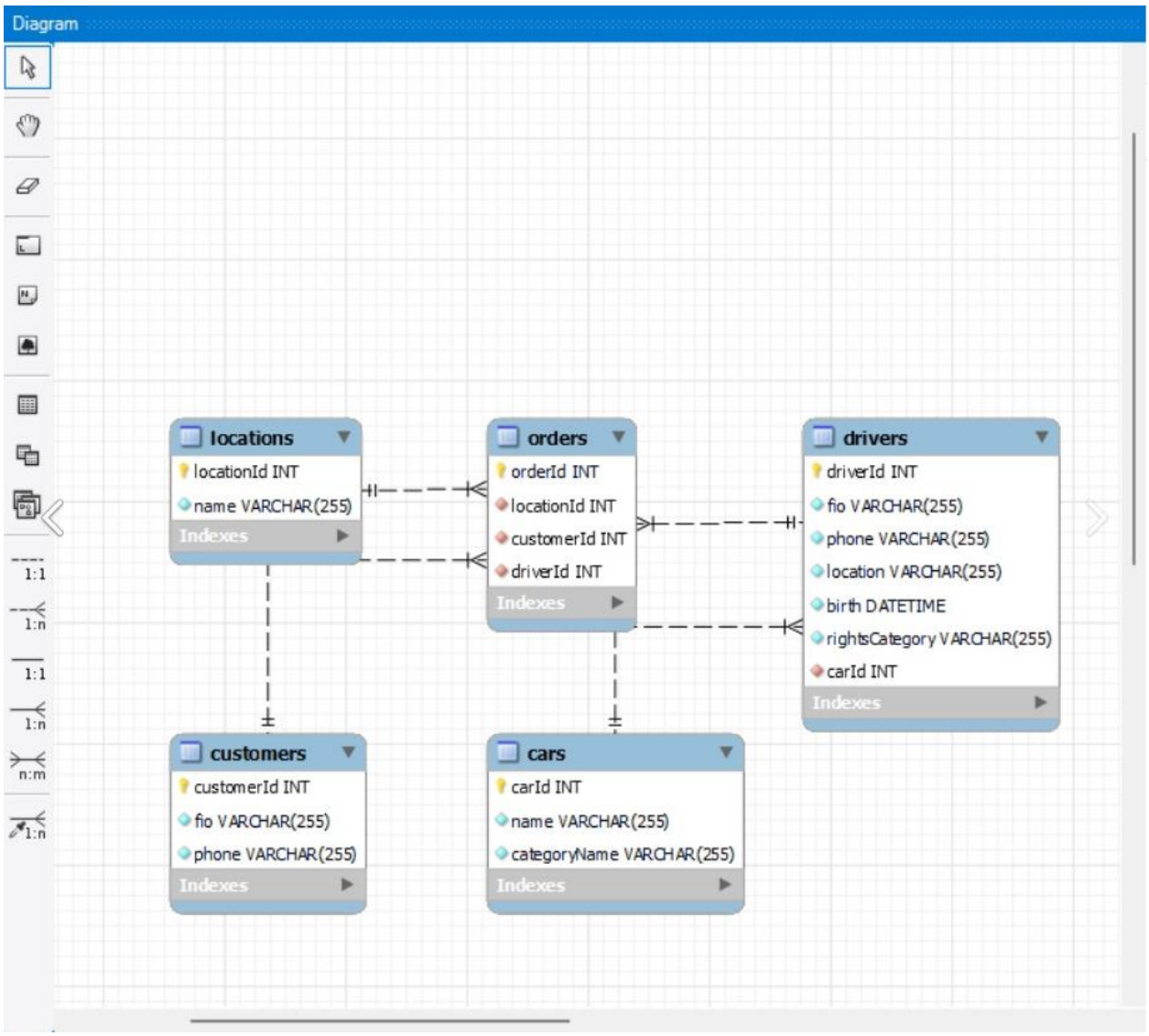
2.3.1 Разработанные таблицы

Таблицы созданы в соответствии с физической схемой базы данных, код для создания данных таблиц представлен в Приложении А (Таблица А.1)

```
use 'AutoBase'
-- create a table
CREATE TABLE Client (
  clientid INTEGER PRIMARY KEY,
  name varchar(50) NOT NULL,
  passport tinyint NOT NULL,
  phone tinyint NOT NULL
) ENGINE = Innodb;

CREATE TABLE Manager (
mngId INTEGER PRIMARY KEY,
name varchar(50) NOT NULL,
phone tinyint NOT NULL
) ENGINE = Innodb;
labelId )
-- insert some values
```

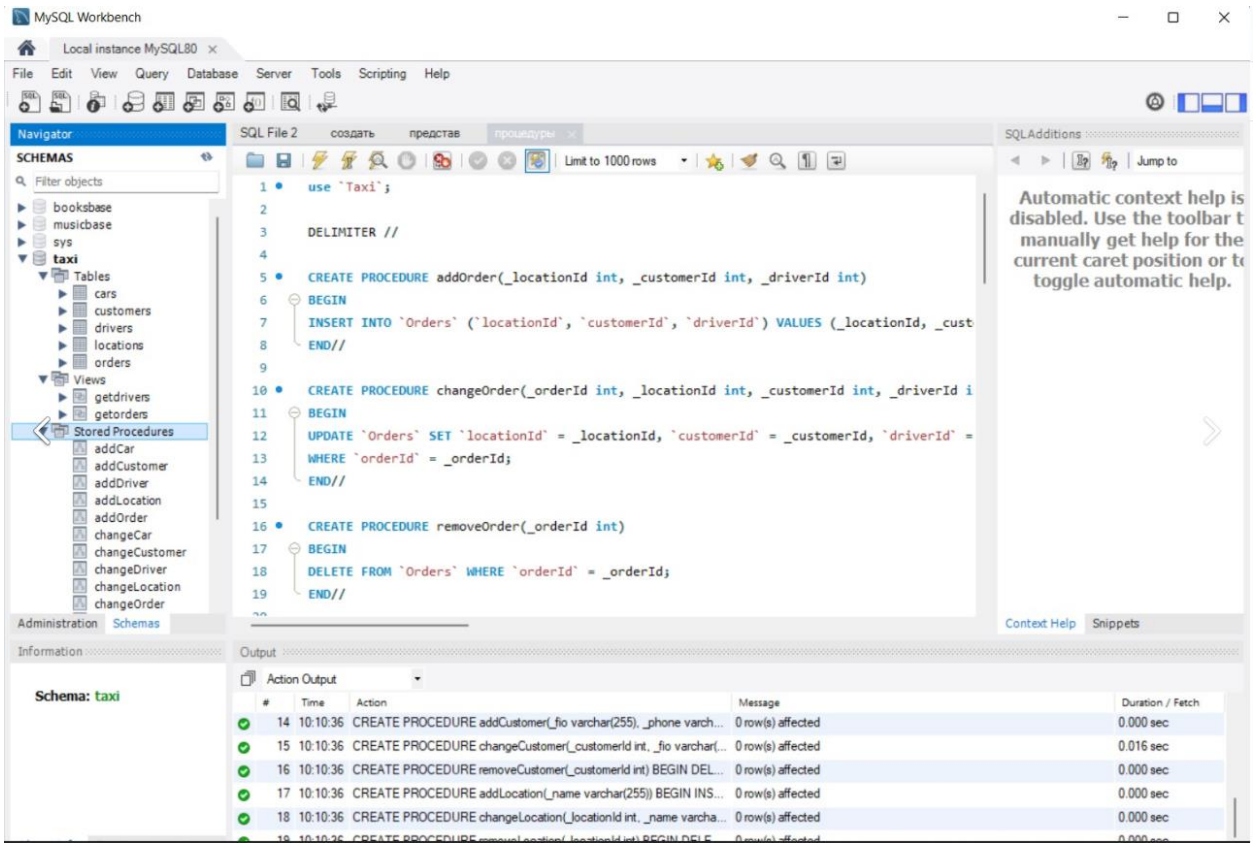
Изображение 3 – Результат выполнения запроса на создание базы данных.



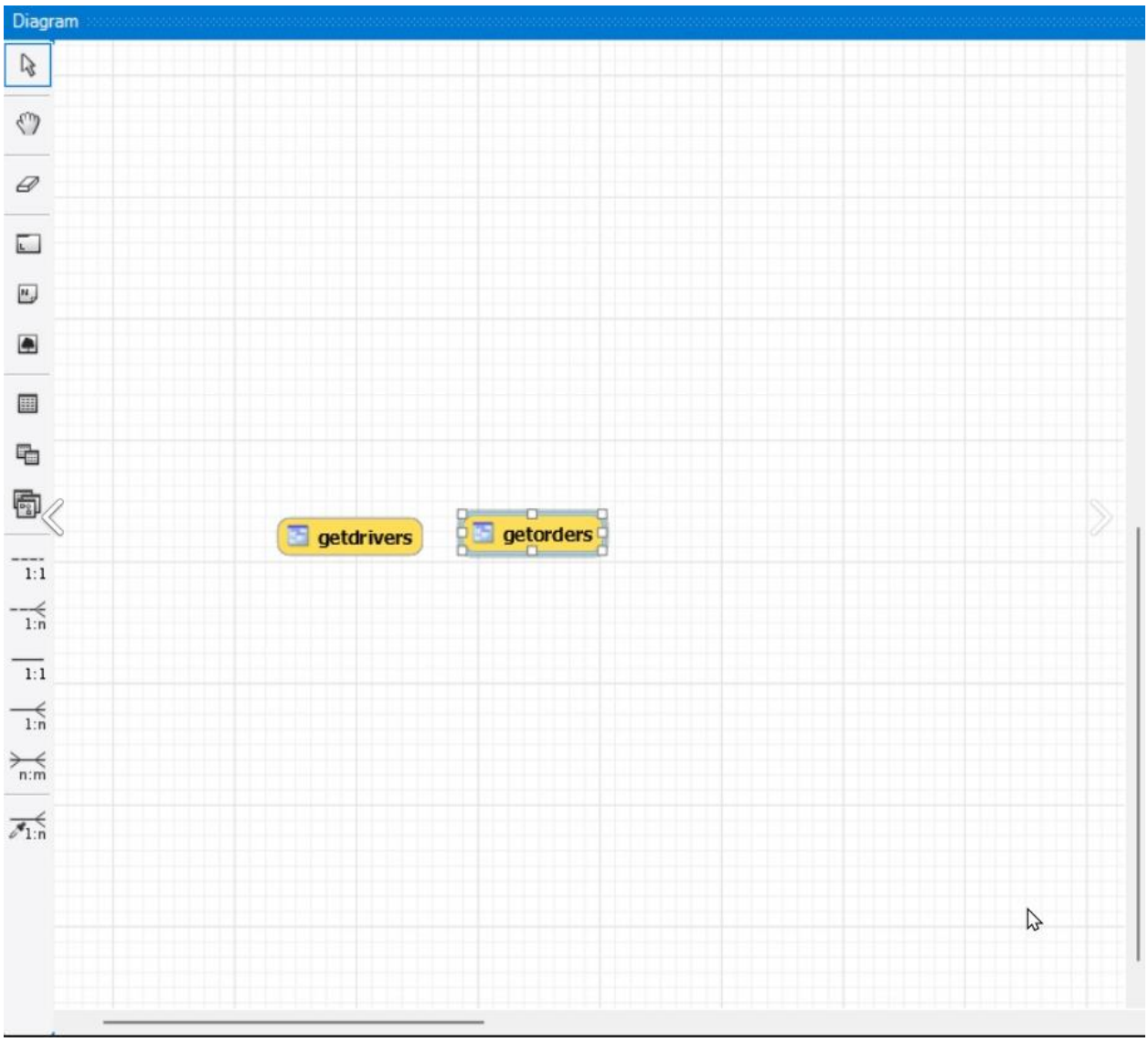
Изображение 4 – Связи в таблице после ее создания через запрос.

2.3.2 Разработанные представления

Описанные в требованиях представления разработаны и их исходный код описан в Приложении А (Таблица А.2)



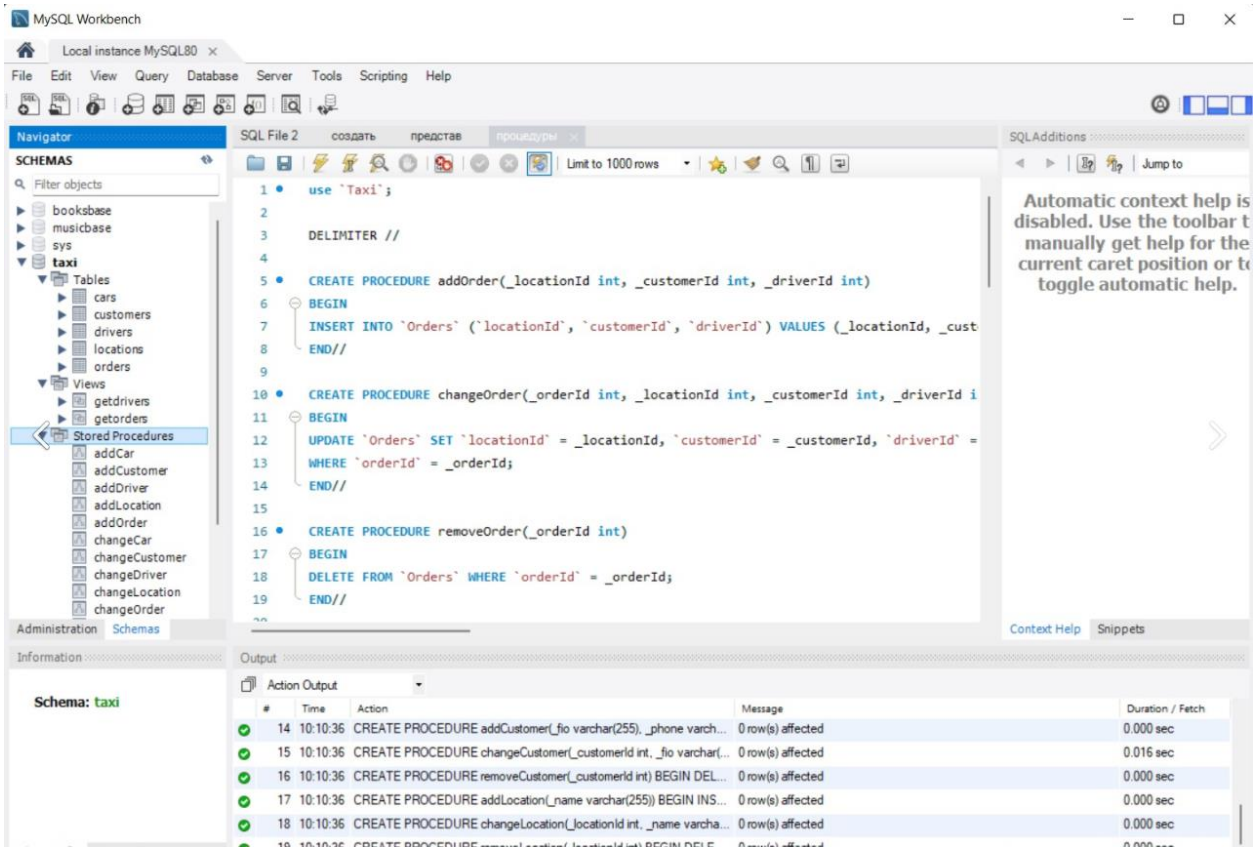
Изображение 5 – Результат выполнения запроса на создание представлений.



Изображение 6 – Представления на диаграмме.

2.3.3 Разработанные процедуры и функции

В соответствии с обозначенными требованиями были разработаны процедуры и функции, представленные в Приложении А (Таблица А.2)



Изображение 7 – Результат выполнения запроса на создание процедур.

3 ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ

ЗАКЛЮЧЕНИЕ

В результате выполнения ВКР была разработана база данных музыки.

В процессе выполнения ВКР были выполнены следующие задачи:

1) Анализ предметной области;

2) Анализ существующих базы данных;

3) Анализ схемы базы данных;

4) Проектирование базы данных;

5) Разработка базы данных;

Достоинства разработанной базы данных:

1) Возможность использования в небольших сервисах;

2) Расширяемость за счет возможностей выбранной СУБД;

3) Минимальные требования к знаниям для использования БД при использовании разработанного API;

4) Методы позволяют тратить меньше времени на добавление, изменение и удаление данных;

Недостатки разработанной базы данных:

1) Доступ пользователей на уровне БД не разграничен;

2) Отсутствие вспомогательных функций;

СПИСОК ЛИТЕРАТУРЫ

1) Официальный сайт MongoDB. [Электронный ресурс]. — URL: https://docs.mongodb.com/manual (Дата обращения: 17.11.2021).

2) Redis для начинающих. [Электронный ресурс]. — URL: https://webdevblog.ru/redis-dlya-nachinajushhij/ (Дата обращения: 17.11.2021)

3) MySQL Википедия. [Электронный ресурс]. — URL: https://ru.wikipedia.org/wiki/MySQL (Дата обращения: 17.11.2021).

4) MongoDB Datatypes. [Электронный ресурс]. — URL: https://www.tutorialspoint.com/mongodb/mongodb_datatype.htm (Дата обращения: 17.11.2021).

5) https://www.oracle.com/ru/database/what-is-database/

6) https://drawio-app.com/

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ОБЪЕКТОВ БАЗЫ ДАННЫХ

Таблица А.1 – Исходный код создания таблиц

createDB.sql

```
CREATE DATABASE Taxi CHARACTER SET utf8 COLLATE utf8 general ci;

use `Taxi`;

CREATE TABLE Customers (
          customerId int PRIMARY KEY AUTO INCREMENT,
    fio varchar(255) NOT NULL,
    phone varchar(255) NOT NULL
) ENGINE = InnoDB;

CREATE TABLE Locations (
          locationId int PRIMARY KEY AUTO INCREMENT,
    name varchar(255) NOT NULL
) ENGINE = InnoDB;

CREATE TABLE Orders (
          orderId int PRIMARY KEY AUTO_INCREMENT,
    locationId int NOT NULL,
    customerId int NOT NULL,
    driverId int NOT NULL
) ENGINE = InnoDB;

CREATE TABLE Drivers (
          driverId int PRIMARY KEY AUTO INCREMENT,
    fio varchar(255) NOT NULL,
    phone varchar(255) NOT NULL,
    location varchar(255) NOT NULL,
    birth datetime NOT NULL,
    rightsCategory varchar(255) NOT NULL,
    carId int NOT NULL
) ENGINE = InnoDB;

CREATE TABLE Cars (
          carId int PRIMARY KEY AUTO INCREMENT,
    name varchar(255) NOT NULL,
    categoryName varchar(255) NOT NULL
) ENGINE = InnoDB;

ALTER TABLE `Drivers`
  ADD CONSTRAINT `FK DriverCar`
  FOREIGN KEY (`carId`) REFERENCES `Cars` (`carId`)
  ON DELETE CASCADE ON UPDATE CASCADE;
```

Таблица А.2 – Исходный код создания хранимых процедур

procedureDB.sql

```
use `Taxi`;

DELIMITER //

CREATE PROCEDURE addOrder( locationId int,  customerId int,  driverId int)
BEGIN
INSERT INTO `Orders` (`locationId`, `customerId`, `driverId`) VALUES (_locationId, _customerId,
_driverId);
END//

CREATE PROCEDURE changeOrder( orderId int,  locationId int,  customerId int,  driverId int)
BEGIN
UPDATE `Orders` SET `locationId` = _locationId, `customerId` = _customerId, `driverId` =
_driverId
WHERE `orderId` =  orderId;
END//

CREATE PROCEDURE removeOrder(_orderId int)
BEGIN
DELETE FROM `Orders` WHERE `orderId` = _orderId;
END//INSERT INTO `Albums` (`name`, `creationDate`) VALUES (name, creationDate);
END//
```

```
CREATE PROCEDURE addOrder( locationId int,  customerId int,  driverId int)
BEGIN
INSERT INTO `Orders` (`locationId`, `customerId`, `driverId`) VALUES (_locationId, _customerId,
_driverId);
END//

CREATE PROCEDURE changeOrder( orderId int,  locationId int,  customerId int,  driverId int)
BEGIN
UPDATE `Orders` SET `locationId` = _locationId, `customerId` = _customerId, `driverId` =
_driverId
WHERE `orderId` =  orderId;
END//

CREATE PROCEDURE removeOrder(_orderId int)
BEGIN
DELETE FROM `Orders` WHERE `orderId` = _orderId;
END//

CREATE PROCEDURE addDriver(_fio varchar(255), _phone varchar(255), _location varchar(255),
_birth datetime, _rightsCategory varchar(255), _carId int)
BEGIN
INSERT INTO `Drivers` (`fio`, `phone`, `location`, `birth`, `rightsCategory`, `carId`) VALUES
( fio,  phone,  location,  birth,  rightsCategory,  carId);
END//

CREATE PROCEDURE changeDriver(_driverId int, _fio varchar(255), _phone varchar(255), _location
varchar(255), _birth datetime, _rightsCategory varchar(255), _carId int)
BEGIN
UPDATE `Drivers` SET `fio` =  fio, `phone` =  phone, `location` =  location, `birth` =  birth,
`rightsCategory` = _rightsCategory, `carId` = _carId
WHERE `driverId` = _driverId;
END//

CREATE PROCEDURE removeDriver( driverId int)
BEGIN
DELETE FROM `Drivers` WHERE `driverId` = _driverId;
END//

CREATE PROCEDURE addCar( name varchar(255),  categoryName varchar(255))
BEGIN
INSERT INTO `Cars` (`name`, `categoryName`) VALUES (_name, _categoryName);
END//

CREATE PROCEDURE changeCar( carId int,  name varchar(255),  categoryName varchar(255))
BEGIN
UPDATE `Cars` SET `name` = _name, `categoryName` = _categoryName
WHERE `carId` = _carId;
END//

CREATE PROCEDURE removeCar( carId int)
BEGIN
DELETE FROM `Cars` WHERE `carId` = _carId;
END//

CREATE PROCEDURE addCustomer( fio varchar(255),  phone varchar(255))
BEGIN
INSERT INTO `Customers` (`fio`, `phone`) VALUES ( fio,  phone);
END//

CREATE PROCEDURE changeCustomer(_customerId int, _fio varchar(255), _phone varchar(255))
BEGIN
UPDATE `Customers` SET `fio` =  fio, `phone` =  phone
WHERE `customerId` =  customerId;
END//

CREATE PROCEDURE removeCustomer( customerId int)
BEGIN
DELETE FROM `Customers` WHERE `customerId` =  customerId;
END//

CREATE PROCEDURE addLocation(_name varchar(255))
BEGIN
INSERT INTO `Locations` (`name`) VALUES ( name);
END//

CREATE PROCEDURE changeLocation(_locationId int, _name varchar(255))
BEGIN
UPDATE `Locations` SET `name` =  name
WHERE `locationId` =  locationId;
END//
```

```
CREATE PROCEDURE removeLocation(_locationId int)
BEGIN
DELETE FROM `Locations` WHERE `locationId` = _locationId;
END//

DELIMITER ;
```

Таблица А.3 – Исходный код создания представлений

viewDB.sql

```
use `Taxi`;

CREATE VIEW getOrders AS
SELECT
          orderId,
          Orders.driverId,
          Orders.customerId,
          Orders.locationId,
    Driver.carId,
          Customer.fio as customerFio,
          Customer.phone as customerPhone,
          Location.name as customerLocation,
          Driver.fio as driverFio,
          Driver.phone as driverPhone,
    Driver.birth as driverBirth,
    Driver.rightsCategory as driverCategory,
    Car.name as carName,
    Car.categoryName as carCategory
FROM `Orders`
          INNER JOIN `Drivers` as Driver ON Orders.driverId = Driver.driverId
          INNER JOIN `Customers` as Customer ON Orders.customerId = Customer.customerId
          INNER JOIN `Locations` as Location ON Orders.locationId = Location.locationId
    INNER JOIN `Cars` as Car ON Driver.carId = Car.carId
ORDER BY Orders.orderId;

CREATE VIEW getDrivers AS
SELECT
          driverId,
          fio as driverFio,
          phone as driverPhone,
    birth as driverBirth,
    rightsCategory as driverCategory,
    Car.name as carName,
    Car.categoryName as carCategory
FROM `Drivers` as Drivers INNER JOIN `Cars` as Car ON Drivers.carId = Car.carId
ORDER BY Drivers.driverId;
```