Государственное бюджетное профессиональное образовательное учреждение города Москвы

«Московский государственный колледж электромеханики и информационных технологий»

(ГБПОУ МГКЭИТ)

Отчет по практической работе № 4

Проектирование базы данных

«Автосалон»

Выполнил:

Студент группы 3ИП-11-19

Шибанов Д.

Проверил:

Преподаватель Басыров С. А.

Оглавление

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 2](#_Toc89174181)

[1.1 Исследование предметной области 2](#_Toc89174182)

[1.2 Анализ существующих СУБД 5](#_Toc89174183)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 16](#_Toc89174184)

[2.1 Формирование требований 16](#_Toc89174185)

[2.2 Проектирование базы данных 18](#_Toc89174186)

[2.3 Разработка базы данных 22](#_Toc89174187)

[3 ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ 26](#_Toc89174188)

[ЗАКЛЮЧЕНИЕ 27](#_Toc89174189)

[СПИСОК ЛИТЕРАТУРЫ 28](#_Toc89174190)

**ВВЕДЕНИЕ**

Целью ВКР является – Разработка базы данных в программе MySQL WorkBench

Объектом исследования является создание БД

Предметом исследования является БД «Автосалон»

# 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1 Исследование предметной области

**1** **Анализ основных понятий**

Опираясь на выбранную тему, основными понятиями будут являться:

1) Пользователь – лицо или группа лиц, которое использует действующую систему для выполнения конкретной функции.

2) БД автосалона – база данных хранящая информацию о автомобилях, находящихся в автосалоне

**2 Основные объекты**

Основным объектом является:

Автомобиль (таблица 1).

Работники (таблица 2)

Клиент (таблица 3)

Продажи (таблица 4)

Таблица 1 – Сведения об автомобиле

|  |  |
| --- | --- |
| **Характеристики объекта** | **Описание характеристик** |
| Марка автомобиля | Какая марка у автомобиля |
| Страна производства | В какой стране был произведен автомобиль |
| Год производства | Информация о том в каком году был произведен автомобиль |
| Цвет | Цвет автомобиля |
| Модель автомобиля | Информация о модели автомобиля |
| Цена | Информация о цене автомобиля |
| Код автомобиля | Код автомобиля в БД |

Таблица 2 – Сведения о работниках

|  |  |
| --- | --- |
| **Характеристики объекта** | **Описание характеристик** |
| ФИО | Информация о ФИО работника |
| Должность | Информация о должности работника |
| Паспортные данные | Информация о паспортных данных |
| Номер моб. Телефона | Информация о номере моб. Телефона |
| Код работника | Код работника в бд |

Таблица 3 – Сведения о клиенте

|  |  |
| --- | --- |
| **Характеристики объекта** | **Описание характеристик** |
| Фамилия | Информация о фамилии клиента |
| Имя | Информация о имени клиента |
| Отчества | Информация о отчестве клиента |
| Телефон | Информация о телефоне клиента |
| Код клиента | Код клиента в БД |

Таблица 4 – Сведения о продажах

|  |  |
| --- | --- |
| **Характеристики объекта** | **Описание характеристик** |
| Код продажи | Код продажи в БД |
| Код клиента | Код клиента в БД |
| Код работника | Код работника в БД |
| Кот автомобиля | Код автомобиля в БД |
| Дата продажи | Информация о дате продажи |

**3 Основные действия с объектами**

Каждый из объектов имеет несколько основных действий:

1) Объект «автомобиль»:

− Посмотреть информацию об автомобиле и о его наличии;

2) Объект «Работники»:

− изменения должности работника;

− изменение информации о моб. Телефоне ;

− удалить;

3) Объект «Клиент»

- изменения ФИО и телефона

4) Объект «Продажи»

- Изменения даты продажи, кода автомобиля, кода клиента, кода работника

**4 Основные участники предметной области**

Основным участником предметной области является сотрудники. Это человек, который взаимодействует с информацией в БД. Он может добавлять, удалять и управлять информацией в БД.

## 1.2 Анализ существующих СУБД

**1. Neo4j**

**1.1 Анализ**

Neo4j – графовая система управления базами данных с открытым исходным кодом, реализованная на Java. Данные хранит в собственном формате, специализированно приспособленном для представления графовой информации, такой подход в сравнении с моделированием графовой базы данных средствами реляционной СУБД позволяет применять дополнительную оптимизацию в случае данных с более сложной структурой. Также утверждается о наличии специальных оптимизаций для SSD-накопителей, при этом для обработки графа не требуется его помещение целиком в оперативную память вычислительного узла, таким образом, возможна обработка достаточно больших графов.

**1.2 Основные возможности**

Основные возможности — поддержка ACID и соответствие спецификациям JTA, JTS и XA. Интерфейс программирования приложений для СУБД реализован для многих языков программирования, включая Java, Python, Clojure, Ruby, PHP, также реализовано API в стиле REST. Расширить программный интерфейс можно как с помощью серверных плагинов, так и с помощью неуправляемых расширений (unmanaged extensions); плагины могут добавлять новые ресурсы к REST-интерфейсу для конечных пользователей, а расширения позволяют получить полный контроль над программным интерфейсом, и могут содержать произвольный код, поэтому их следует использовать с осторожностью.

**1.3 Типы данных**

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Тип данных CQL** | **использование** |
| 1 | логический | Используется для представления логических литералов: true, false. |
| 2 | байт | Он используется для представления 8-битных целых чисел. |
| 3 | короткая | Он используется для представления 16-битных целых чисел. |
| 4 | ИНТ | Он используется для представления 32-битных целых чисел. |
| 5 | долго | Он используется для представления 64-битных целых чисел. |
| 6 | поплавок | Он используется для представления 32-битных чисел с плавающей точкой. |
| 7 | двойной | Он используется для представления 64-битных чисел с плавающей точкой. |
| 8 | голец | Используется для представления 16-битных символов. |
| 9 | строка | Он используется для представления строк. |

**1.4 Язык запроса**

**Язык запросов Cypher** — Neo4j предоставляет декларативный язык запросов для визуального представления графика с использованием синтаксиса ascii-art. Команды этого языка в удобочитаемом формате и очень просты в освоении.

**2 Cassandra**

**2.1 Анализ**

**Apache Cassandra** — распределённая система управления базами данных, относящаяся к классу NoSQL-систем и рассчитанная на создание высокомасштабируемых и надёжных хранилищ огромных массивов данных, представленных в виде хэша.

**2.2 Основные возможности**

Данная СУБД имеет следующие возможности:

1. Эластичная масштабируемость – Cassandra отлично масштабируется; это позволяет добавить больше оборудования, чтобы разместить больше клиентов и больше данных согласно требованию;
2. Всегда на архитектуре – Cassandra не имеет единой точки отказа, и она постоянно доступна для критически важных для бизнеса приложений, которые не могут допустить сбоя;
3. Высокая производительность в линейном масштабе – Cassandra линейно масштабируется, т. Е. Увеличивает пропускную способность при увеличении количества узлов в кластере. Поэтому он поддерживает быстрое время отклика;
4. Простое распространение данных – Cassandra обеспечивает гибкость в распределении данных там, где вам нужно, путем репликации данных между несколькими центрами обработки данных;
5. Поддержка транзакций – Cassandra поддерживает такие свойства, как атомарность, согласованность, изоляция и долговечность (ACID).

**2.3 Типы данных**

|  |  |  |
| --- | --- | --- |
| **Тип данных** | **Константы** | **Описание** |
| ASCII | строки | Представляет строку символов ASCII |
| BIGINT | BIGINT | Представляет 64-битную подписанную длинную |
| **капля** | сгустки | Представляет произвольные байты |
| логический | булевы | Представляет истину или ложь |
| **счетчик** | целые | Представляет счетчик столбец |
| десятичный | целые числа, числа с плавающей запятой | Представляет десятичную с переменной точностью |
| двойной | целые | Представляет 64-битную IEEE-754 с плавающей точкой |
| поплавок | целые числа, числа с плавающей запятой | Представляет 32-битную IEEE-754 с плавающей точкой |
| инет | строки | Представляет IP-адрес, IPv4 или IPv6 |
| ИНТ | целые | Представляет 32-битный со знаком int |
| текст | строки | Представляет строку в кодировке UTF8 |
| **отметка времени** | целые числа, строки | Представляет метку времени |
| **timeuuid** | UUID, | Представляет UUID типа 1 |
| **UUID** | UUID, | Представляет тип 1 или тип 4 |
|  |  | UUID |
| VARCHAR | строки | Представляет строку в кодировке uTF8 |
| varint | целые | Представляет целое число произвольной точности |

**2.4 Языки запроса**

Пользователи могут получить доступ к Cassandra через его узлы, используя Cassandra Query Language (CQL). CQL рассматривает базу данных **(Keyspace)** как контейнер таблиц. Программисты используют **cqlsh:** приглашение работать с CQL или отдельными драйверами языка приложения.

Клиенты обращаются к любому из узлов за своими операциями чтения-записи. Этот узел (координатор) воспроизводит прокси между клиентом и узлами, содержащими данные.

**3 MySQL**

**3.1 Анализ**

**MySQL** – свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

**3.2 Основные возможности**

* Полностью многопоточное использование ядерных нитей. Это означает, что пакет может легко использовать много CPUs, если они есть.
* Интерфейсы для языков C, C++, Eiffel, Java, Perl, PHP, Python и Tcl.
* Работает на многих различных платформах.
* Много типов столбцов: целые со знаком или без него длиной 1, 2, 3, 4 и 8 байт, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET и ENUM.
* Очень быстрые объединения, использующие оптимизированное однопроходное объединение многих таблиц.
* Полная поддержка операторов и функций в частях запроса SELECT и WHERE. Например: mysql> SELECT CONCAT(first\_name, " ", last\_name) FROM tbl\_name WHERE income/dependents > 10000 AND age > 30;
* SQL-функции выполнены через хорошо оптимизированную библиотеку классов и должны выполняться с такой скоростью, с какой только возможно! Обычно не имеется никакого распределения памяти вообще после инициализации запроса.
* Полная поддержка предложений SQL GROUP BY и ORDER BY. Поддержка групповых функций (COUNT(), COUNT(DISTINCT ...), AVG(), STD(), SUM(), MAX() и MIN()).
* Поддержка LEFT OUTER JOIN и RIGHT OUTER JOIN с синтаксисами ANSI SQL и ODBC.
* Вы можете смешивать таблицы из разных баз данных в одном запросе.
* Привилегии и система паролей, которая является очень гибкой и безопасной, и позволяет проверку, основанную на имени хоста. Пароли безопасны потому, что вся передача пароля шифрована, когда Вы соединяетесь с сервером.
* ODBC (Open-DataBase-Connectivity) поддерживается для Win32 (с исходниками). Все функции ODBC 2.5 и многие другие реализованы. Например, Вы можете использовать MS Access для связи с сервером MySQL.
* Очень быстрые дисковые таблицы B-tree с индексным сжатием.
* Можно иметь до 32 индексов на таблицу. Каждый индекс может состоять от 1 до 16 столбцов или частей столбцов. Максимальная индексная длина 500 байт (это может быть изменено при компиляции MySQL). Индекс может использовать префикс поля CHAR или VARCHAR.
* Записи фиксированной и переменной длины.
* Таблицы в памяти, которые используются как временные таблицы.
* Поддержка поистине огромных объемов данных. Известен случай использования MySQL на 60000 таблиц, хранящих около 5000000000 строк.
* Все столбцы имеют значения по умолчанию. Вы можете использовать вызов INSERT, чтобы вставить подмножество столбцов таблицы. Те столбцы, которым явно не заданы значения, будут автоматически установлены к их значениям по умолчанию.
* Для переносимости использованы GNU Automake, Autoconf и Libtool.
* Пакет написан на C и C++. Оттестирован на всех распространенных компиляторах этих языков.
* Очень быстрая поточно-безопасная система управления памятью.
* Никаких утечек памяти. MySQL тестировался с помощью Purify, коммерческого детектора утечек памяти.
* Есть myisamchk, очень быстрая утилита для проверки таблицы, оптимизации и ремонта. Все функциональные возможности myisamchk также доступны через интерфейс SQL.
* Полная поддержка для нескольких различных наборов символов, включая ISO-8859-1 (Latin1), german, big5, ujis и много других. Например, скандинавские символы `@ringaccent{a}', `@"a' и `@"o' позволяются в именах столбцов и таблиц.
* Все данные сохранены в выбранном наборе символов. Все сравнения для нормальных столбцов нечувствительны к регистру.
* Сортировка выполнена согласно выбранному набору символов (шведский по умолчанию). Возможно изменить это, когда сервер MySQL работает. Чтобы увидеть пример очень продвинутой сортировки, рассмотрите сортировочный код для Czech. MySQL поддерживает много различных наборов символов, которые могут быть определены при компиляции или во время выполнения.
* Псевдонимы на таблицах и именах столбцов доступны как в стандарте SQL92.
* DELETE, INSERT, REPLACE и UPDATE возвращают число строк, которые были изменены (обработаны). Можно взамен вернуть число согласованных строк, устанавливая флажок при соединении с сервером.
* Имена функции не сталкиваются с именами столбцов или таблиц. Например, ABS представляет собой имеющее силу имя столбца. Единственное ограничение: для обращения к функции никакие пробелы не позволяются между именем функции и символом скобки (`('), который следует за ним.
* Все программы пакета MySQL понимают параметры командной строки --help или -? для выдачи справки о параметрах запуска конкретной программы.
* Сервер умеет выдавать сообщения об ошибках и диагностику на разных языках.
* Клиенты могут соединяться с сервером MySQL, используя все мыслимые способы, допустимые в сегодняшних сетях: сокеты TCP/IP, сокеты Unix (под Unix) или даже именованные каналы (под NT).
* MySQL-специфичная команда SHOW может использоваться, чтобы получить информацию относительно баз данных, таблиц и индексов. Команда EXPLAIN может использоваться, чтобы определить, как именно оптимизатор решает запрос.

**3.3 Типы данных**

1. CHAR: представляет строку фиксированной длины;
2. VARCHAR: представляет строку переменной длины;
3. TINYTEXT: представляет текст длиной до 255 байт;
4. INT: представляет целые числа от -2147483648 до 2147483647, занимает 4 байта;
5. DECIMAL: хранит числа с фиксированной точностью. Данный тип может принимать два параметра precision и scale: DECIMAL (precision, scale);
6. FLOAT: хранит дробные числа с плавающей точкой одинарной точности;
7. DOUBLE: хранит дробные числа с плавающей точкой двойной точности.

**3.4 Язык запроса**

SQL или Structured Query Language (язык структурированных запросов) — язык программирования, предназначенный для управления данными в СУБД. Все современные СУБД поддерживают SQL.

На языке SQL выражаются все действия, которые можно провести с данными: от записи и чтения данных, до администрирования самого сервера СУБД.  
Для повседневной работы совсем не обязательно знать весь этот язык; достаточно ознакомиться лишь с основными понятиями синтаксиса и ключевыми словами. Кроме того, SQL очень простой язык по своей структуре, поэтому его освоение не составит большого труда.

**1.3 Обоснование выбора СУБД**

Основные преимущества Microsoft SQL Server 2017:

* Полная Web ориентированность.
* Масштабируемость и надежность.
* Скорость создания решений.
* Возможность обработки вычислений в оперативной памяти.
* Возможность взаимодействия с публичным облаком Windows Azure.
* Улучшенная технология AlwaysOn.

# 2 ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1 Формирование требований

Требования к БД

Хранение и возможность редактировать информацию определенными пользователями

**2.1.1 Основные сущности базы данных**

В БД будут храниться следующие сущности:

* Работник - просмотр и изменение информации в БД.
* Автомобиль – тут будут хранится информация о автомобилях.
* Клиент - тут будут хранится информация о клиентах
* Продажи – здесь будут хранится сведенья о продажах

|  |  |
| --- | --- |
| **Сущность** | **Характеристики** |
| Работник | Код работника  Фамилия  Имя  Отчества  Должность  Паспортные данные  Номер телефона |
| Автомобиль | Марка автомобиля  Страна производства  Год производства  Цвет  Модель автомобиля  Цена  Код автомобиля |
| Клиент | Код клиента  Фамилия  Имя  Отчества  Телефон |
| Продажа | Код продажи  Код клиента  Код работника  Код работника  Дата продажи |

**2.1.2 Определение API для взаимодействия с базой данных**

|  |  |
| --- | --- |
| Метод | Аргументы |
| Изменить работника | Код работника  Фамилия  Имя  Отчества  Должность  Паспортные данные  Номер телефона |
| Изменить автомобиль | Марка автомобиля  Страна производства  Год производства  Цвет  Модель автомобиля  Цена  Код автомобиля |
| Изменить клиента | Код клиента  Фамилия  Имя  Отчества  Телефон |
| Изменить продажу | Дата продажи |

## 2.2 Проектирование базы данных

(Сюда содержимое практической работы №4)

#### 2.2.1 Логическая схема базы данных

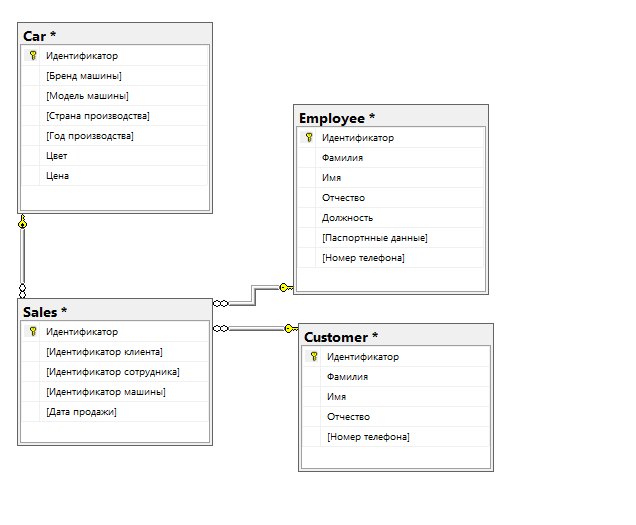


Рисунок 2.1 – Логическая схема базы данных

#### 2.2.2 Физическая схема базы данных

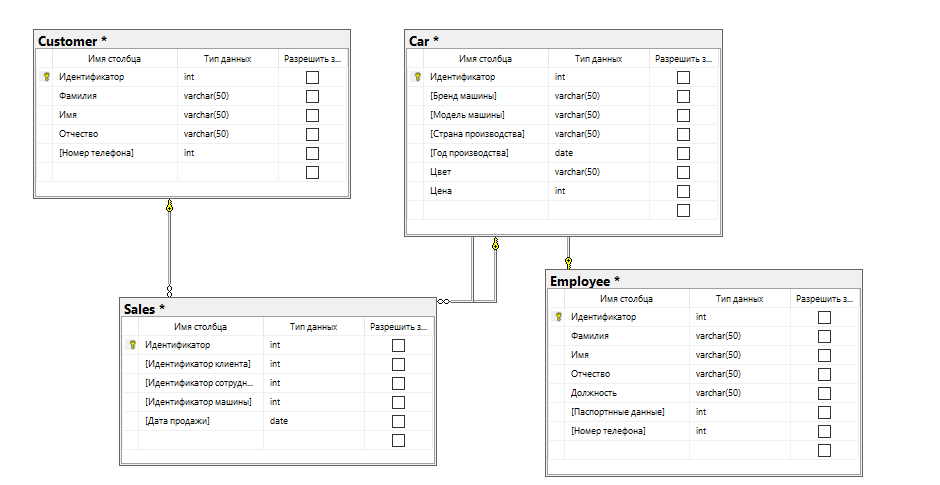


Рисунок 2.2 – Физическая схема базы данных

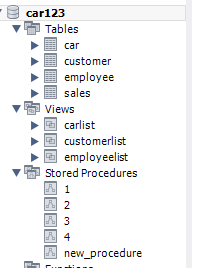
**2.2.3 Определение типовых запросов к объектам базы данных**

|  |  |  |
| --- | --- | --- |
| **Таблица** | **Запрос** | **Пример** |
| Employee | Список Работников | SELECT \* FROM ‘Employee’ |
| Добавить работника | CREATE DEFINER=`root`@`localhost` PROCEDURE `2`()  INSERT INTO employee (surname, name, middle\_name, position  , passport\_data, phone\_number) VALUES ("Антонов", "Антон", "Антонович", "менеджер", "343435353" , "723456789") |
| Изменить работника | UPDATE employee SET surname = (@surname, ‘surname‘) WHERE employee\_id =1 |
| Удалить работника | CREATE DEFINER=`root`@`localhost` PROCEDURE `4`()  DELETE FROM employee WHERE employee\_id=1 |
| Car | Список машин | SELECT \* FROM ‘Car’ |
| Добавить машину | INSERT INTO ‘Car’ (‘Car’, ‘car\_model’, ‘country\_of\_origin’, ‘production\_year’, ‘colour’, ‘price’) VALUES (‘Жигули’, ‘XS MAX’, ‘МАСКВА’, ’10.10.2021’, ’red’, ‘1000’) |
| Изменить машину | UPDATE ‘Car’ SET production\_year = IsNull(@production\_year, 10.10.2021), user\_id= IsNull(@user\_id, 2) WHERE production\_year =11.10.2021 |
| Удалить машину | DELETE FROM ‘Car’ WHERE file\_id=1 |
| Customer | Список клиентов | SELECT \* FROM ‘Customer’ |
| Добавить клиента | INSERT INTO ‘Customer’ (‘surname, ‘name’, ‘middle\_name’, ‘phone\_number’) VALUES (‘Греш’, ‘ Арпек’, ‘Грегов’, ‘+792324423’) |
| Изменить клиента | UPDATE ‘Customer’ SET phone\_number = IsNull(@phone\_number, ‘+792324423), user\_id= IsNull(@user\_id, 2) WHERE customer \_id =1 |
| Удалить клиента | DELETE FROM ‘Folder’ WHERE customer \_id =1 |
|  |  |  |

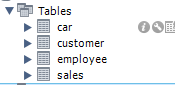
**2.2.4 Определение процедур и функций API**

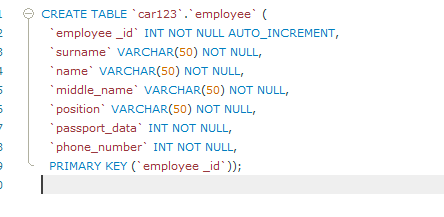
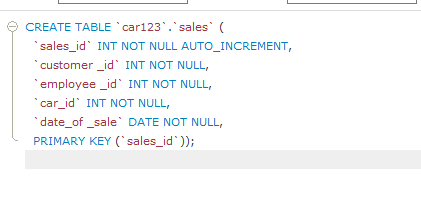
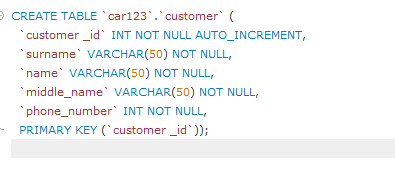
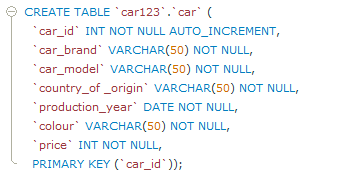
|  |  |
| --- | --- |
| **API функции** | **Описание API функции** |
| changeCustomer (customer \_id(int, PK, AI)  surname (varchar50, NN)  name (varchar50, NN)  middle\_name (varchar50, NN)  phone\_number (int, NN) | Изменяет все данные клиента |
| changeCar (car\_brand(varchar50, NN)  car\_model (varchar50, NN)  country\_of \_origin (varchar50, NN)  production\_year (date, NN)  colour (varchar50, NN)  price (money50 NN))) | Изменяет все данные машины |
| changecar\_brand(car\_id (int, PK, AI), car\_brand(varchar50,NN)) | Изменить бренд определенной машины |
| change Employee (employee \_id (int, PK, AI)  surname (varchar50, NN)  name (varchar50, NN)  middle\_name(varchar50,NN)  position (varchar50, NN)  passport\_data (int, NN)  phone\_number (int, NN) | Изменяет все данные работника |

## 2.3 Разработка базы данных

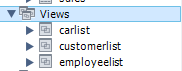


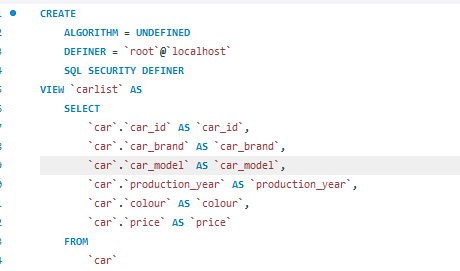
#### 2.3.1 Разработанные таблицы

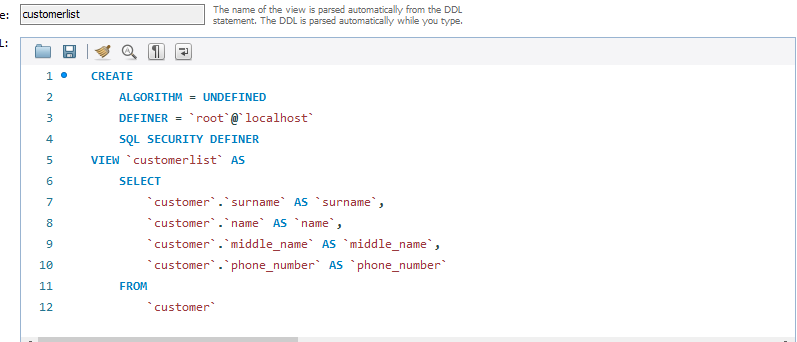


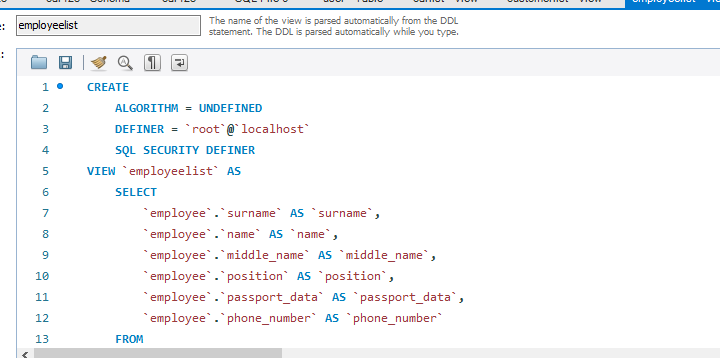


#### 2.3.2 Разработанные представления





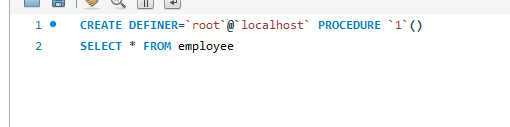




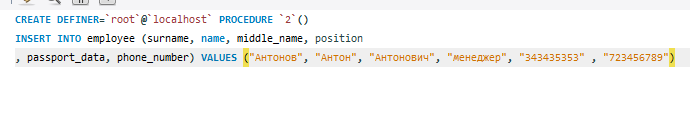
#### 2.3.3 Разработанные процедуры и функции

Процедуры на примере одной таблицы

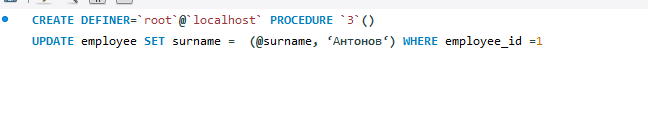
1 Вывод всей информации



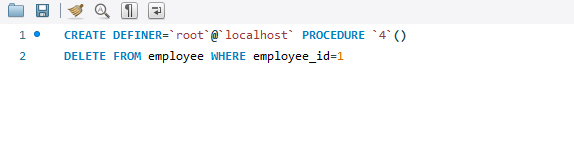
2 Ввод информации



3 Изменение информации в таблице



4 Удаление строк в таблице



# 3 ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ

(Не трогайте, вам потом самим скажут что сюда вставлять)

# ЗАКЛЮЧЕНИЕ

В результате выполнения ВКР была разработана БД «Автосалон»

В процессе выполнения ВКР были выполнены следующие задачи:

1) Разработка БД

2) Разработка функций БД

3) Изучение среды

4) Создание БД

# СПИСОК ЛИТЕРАТУРЫ

1) Основные команды SQL. [Электронный ресурс]. — URL: https://tproger.ru/translations/sql-recap/ (Дата обращения: 25.11.2021)

2) Оператор SQL PRIMARY KEY. [Электронный ресурс]. — URL: http://2sql.ru/novosti/sql-primary-key/ (Дата обращения: 25.11.2021)

3) Представления. [Электронный ресурс]. — URL: https://docs.microsoft.com/ru-ru/sql/relational-databases/views/views?view=sql-server-ver15 (Дата обращения: 25.11.2021)