**1. Введение.**

Целью учебной практики является научиться работать с SQL кодом.

SQL (Structured Query Language) — язык структурированных запросов, с помощью него пишутся специальные запросы (так называемые SQL инструкции) к базе данных с целью получения данных из базы данных или для манипулирования этими данными. Это своего рода набор инструкций — какие данные извлечь из таблиц, как представить и в какой последовательности.

**2. План отчёта**

1. Создание таблиц и связей между ними через SQL код.
2. Создание запросов и отчётов через SQLкод.
3. Расчёт метрик.
4. Вывод по проделанной работе.

**1. Создание таблиц и связей между ними через SQL код.**

После завершения создания БД, я начал писать SQL код для таблиц (рис. №1).

**Таблица автомобили:**

/\*Это первая таблица, которую я создал/

CREATE TABLE Автомобили

(Код COUNTER PRIMARY KEY,

Марка TEXT(50),

[Дата изготовления] DATE,

[Кол-во пробега] SMALLINT,

Цвет TEXT(50),

[Тип автомобиля] TEXT(50)

FOREIN KEY (“Код”) REFERENCES Автомобили (“Код”));

**Таблица Список работ:**

/\*Это вторая таблица, которую я создал/

CREATE TABLE Список работ

(Код COUNTER PRIMARY KEY,

[Вид работ] TEXT(50),

Цена MONEY,

Время TEXT(50)

FOREIN KEY (“Код”) REFERENCES Список работ (“Код”));

**Таблица Места техобслуживания:**

/\*Это третья таблица, которую я создал/

CREATE TABLE Места

(Код COUNTER PRIMARY KEY,

Адрес TEXT(50),

[Тип места] TEXT(50)

FOREIN KEY (“Код”) REFERENCES Места техобслуживания (“Код”));

**Таблица Завершение:**

/\*Это четвёртая и последняя таблица, которую я создал/

CREATE TABLE Завершениеt

(Код COUNTER PRIMARY KEY,

Автомобиль INT,

Работа INT,

[Дата принятия автомобиля] DATE,

[Дата сдачи автомобиля] DATE,

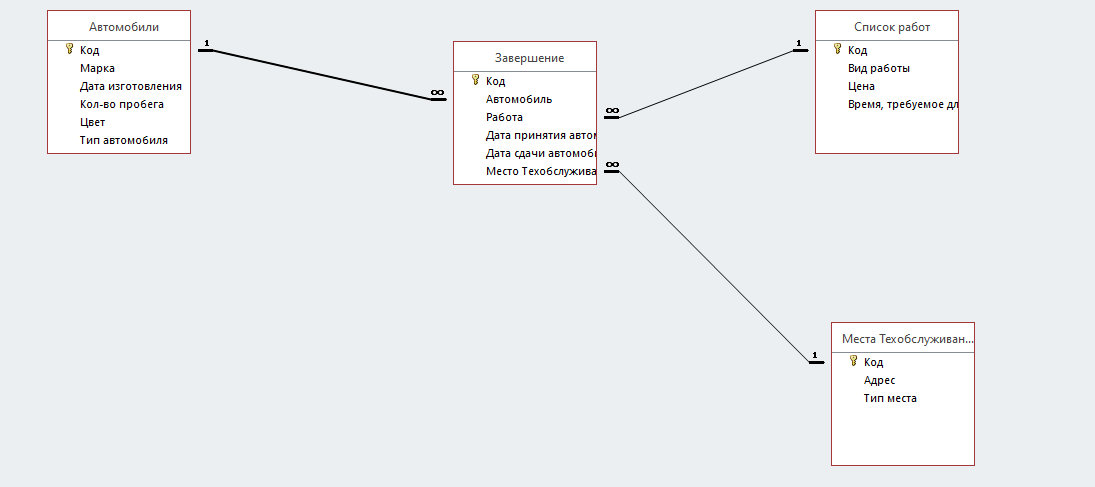
[Место техобслуживания] INT); 

Рисунок №1: схема данных

**2. Создание запросов и отчётов через SQL код.**

**Запрос Грузовые автомобили:**

/\*Это первый запрос, который я создал, он выводит только грузовые автомобили/

SELECT Автомобили.Код, Автомобили.Марка, Автомобили.[Дата изготовления], Автомобили.[Кол-во пробега], Автомобили.Цвет, Автомобили.[Тип автомобиля]

FROM Автомобили

WHERE (((Автомобили.[Тип автомобиля])="Грузовой"));

**Запрос Зелёные авто:**

/\*Это второй запрос, который я создал, он выводит только автомобили зелёного цвета/

SELECT Автомобили.Код, Автомобили.Марка, Автомобили.[Дата изготовления], Автомобили.[Кол-во пробега], Автомобили.Цвет, Автомобили.[Тип автомобиля]

FROM Автомобили

WHERE (((Автомобили.Цвет)="Зелёный"));

**Запрос Поиск по марки:**

/\*Это третий запрос, который я создал, он выводит по первой букве опр. Марку /

SELECT Автомобили.Код, Автомобили.Марка, Автомобили.[Дата изготовления], Автомобили.[Кол-во пробега], Автомобили.Цвет, Автомобили.[Тип автомобиля]

FROM Автомобили

WHERE (((Автомобили.Марка) Like [Введите 1 букву марки] & "\*"));

**Запрос Пробег больше 150000:**

/\*Это четвёртый запрос, который я создал, он вывод только автомобили пробег которых больше 15000/

SELECT Автомобили.Код, Автомобили.Марка, Автомобили.[Дата изготовления], Автомобили.[Кол-во пробега], Автомобили.Цвет, Автомобили.[Тип автомобиля]

FROM Автомобили

WHERE (((Автомобили.[Кол-во пробега])>150000));

**Запрос Работа по опр. Марки:**

/\*Это пятый запрос, который я создал, он выводит ремонты по автомобилям опр. Марки /

SELECT Автомобили.Цвет, Автомобили.Марка, Завершение.Работа, Завершение.[Место Техобслуживания]

FROM Автомобили INNER JOIN Завершение ON Автомобили.Код = Завершение.Автомобиль

WHERE (((Автомобили.Марка)=[Введите марку автомобиля]));

**Запрос Ремонты:**

/\*Это шестой запрос, он просто показывает какие ремонты будут по автомобилям/

SELECT Завершение.Автомобиль, Sum([Список работ].Цена) AS Цена, Завершение.[Дата принятия автомобиля], Завершение.[Дата сдачи автомобиля]

FROM [Список работ] INNER JOIN Завершение ON [Список работ].Код = Завершение.Работа

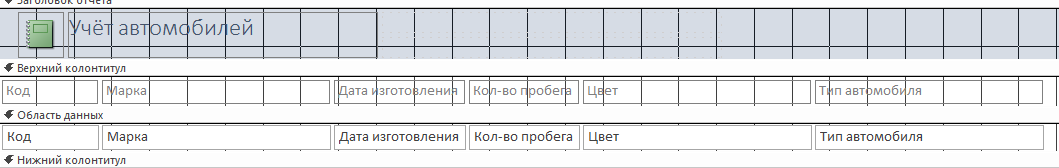
GROUP BY Завершение.Автомобиль, Завершение.[Дата принятия автомобиля], Завершение.[Дата сдачи автомобиля];

**Отчёт Автомобили:**

/\*Это первый отчёт, который я создал, он показывает информацию об автомобилях(учёт)/

SELECT Автомобили.Код, Автомобили.Марка, Автомобили.[Дата изготовления], Автомобили.[Кол-во пробега], Автомобили.Цвет, Автомобили.[Тип автомобиля]

FROM Автомобили;

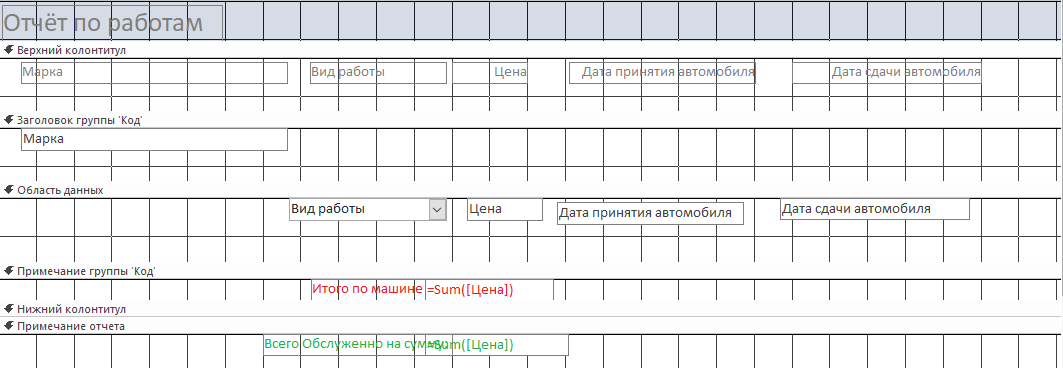


**Отчёт Отчёт по работам:**

/\*Это второй отчёт, который я создал, он показывает какие работы проделывались на опр. Машине и считает финальную стоимость работ/

SELECT Автомобили.Марка, [Список работ].[Вид работы], [Список работ].Цена, Завершение.[Дата принятия автомобиля], Завершение.[Дата сдачи автомобиля], Автомобили.Код

FROM [Список работ] INNER JOIN (Автомобили INNER JOIN Завершение ON Автомобили.Код = Завершение.Автомобиль) ON [Список работ].Код = Завершение.Работа;



**3. Расчёт метрик.**

**Справочный материал:**

**Количественные метрики**

1. -количество строк / Source lines of code (SLOC);
2. -простая метрика стилистики / процент комментариев

F=(Nком/Nстр)\*100%, где

* Nком - количество комментариев в программе;
* Nстр - количество строк или операторов исходного текста.

**Метрики Холстеда:**

1. -длина программы: N = N1 + N2;
2. -объем программы: V = N\*log2(n);
3. -оценка ее реализации (уровень качества): L= (2\*n2) / (n1\*N2);
4. -трудность ее понимания: Ec= V/ L;
5. -трудоемкость кодирования: D = (n1 \* N2)(2\*n2) = 1/ L;
6. -информационное содержание (умственные затраты на создание): I = V / D,

Где:

1. n = n1 + n2 (словарь программы);
2. n1 - число уникальных операторов программы, включая символы - разделители, имена процедур и знаки операций (словарь операторов);
3. n2 - число уникальных операндов программы (словарь операндов);
4. N1 - общее число операторов в программе;
5. N2 - общее число операндов в программе;

*Чем отличается оператор от операнда?*

* +, \*, /, - это операторы
* x, у, z, 999, -25, number1 - это операнды

**Метрики Маккейба:**

1. -цикломатическое число (цикломатическая сложность графа программы):

Z(G) = e – v + 2p,

Где:

1. e - число дуг ориентированного графа G;
2. v - число вершин;
3. p - число компонентов связности графа.

Число компонентов связности графа можно рассматривать как количество дуг, которые необходимо добавить для преобразования графа в сильно связный.

Cильно связным называется граф, любые две вершины которого взаимно достижимы. Для графов корректных программ, т. е. графов, не имеющих недостижимых от точки входа участков и "висячих" точек входа и выхода, сильно связный граф, как правило, получается путем замыкания дугой вершины, обозначающей конец программы, на вершину, обозначающую точку входа в эту программу.

**Метрики надежности** (кол-во ошибок и дефектов)

1. -количество структурных изменений, произведенных с момента проектирования;
2. -количество ошибок, выявленных в ходе проектирования;
3. -количество проведенных структурных изменений, необходимых для корректной работы БД.

**Выполнение:**

**Количественные метрики**

1.количество строк = 80

2. простая метрика стилистики = 15

F=(Nком/Nстр)\*100%

**Метрики Холстеда:**

1.длина программы: N = N1 + N2 = 59

2. объем программы: V = N\*log2(n) = 1.08

3. оценка ее реализации (уровень качества): L= (2\*n2) / (n1\*N2) = 24 / 96 = 0.25

4. трудность ее понимания: Ec=V/ L = 4.32

5. трудоемкость кодирования: D = (n1 \* N2)(2\*n2) = 1/ L = 4

6. информационное содержание: I = V / D = 0.27

**Метрики Маккейба:**

1. цикломатическое число: Z(G) = e – v + 2p = 49 – 14 + 2 = 37

**Метрики надежности**

1.количество структурных изменений, произведенных с момента проектирования = 2

2. количество ошибок, выявленных в ходе проектирования = 0

3. количество проведенных структурных изменений, необходимых для корректной работы БД = 0

**4. Вывод по проделанной работе.**

В данной практике я научился следующему:

1. Создавать таблицы и связи между ними через SQL код.
2. Создавать запросы и отчёты через SQLкод.
3. Рассчитывать метрики.