**1. Введение.**

Целью учебной практики является научиться работать с SQL кодом.

Википедия гласит, что SQL — это декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. Не самое удобоваримое определение. Чтобы понять, о чём вообще речь, разберём его.

Декларативный язык программирования говорит, что должно быть сделано, а не как это необходимо сделать

Выходит, что SQL — это язык программирования, необходимый для написания команд к БД, после выполнения которых она вернёт результат. Результат будет зависеть от команды, написанной на SQL. Как в любом другом языке программирования, в SQL есть операторы для работы с данными, из которых складываются команды. Операторыраспределеныпочетырёмязыкам:

DDL — DataDefinitionLanguage;

DML — DataManipulationLanguage;

DCL — DataControlLanguage;

TCL — TransactionControlLanguage.

**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 «Список жилищных комплексов».**

CREATE TABLE [Список жилищных комлексов] [/\*это четвертая](file:///\\*третья) таблица мной созданная\*/

([Код места] COUNTER CONSTRAINT PrimaryKey PRIMARY KEY,

[Ардрес места] TEXT NOT NULL,

[Тип места] TEXT NOT NULL,

[Тип канализации места] TEXT NOT NULL);

**Таблица «Подачи услуги».**

CREATE TABLE [Подачи услуги] [/\*это](file:///\\*это) третья таблица мной созданная\*/

([Код Подачи] COUNTER CONSTRAINT PrimaryKey PRIMARY KEY,

[Код Клиента] INT NOT NULL,

[Код Услуги] INT NOT NULL,

[Код Места] INT NOT NULL,

[Общая стоимость] MONEY NOT NULL,

[Дата подачи услуги] DATE NOT NULL,

[Дата окончания услуги] DATE NOT NULL,

[Период действия услуги] TEXT NOT NULL,

FOREIGN KEY ([Код Клиента]) REFERENCES Клиенты ([Код Клиента]),

FOREIGN KEY ([Код Услуги]) REFERENCES [Подаваемые услуги] ([Код Услуги]),

FOREIGN KEY ([Код Места]) REFERENCES [Список жилищных комплексов] ([Код Места]));

**Таблица «Подаваемые документы».**

CREATE TABLE [Подаваемы документы] /\*Это вторая таблица которую я создал\*/

([Код услуги] COUNTER CONSTRAINT PrimaryKey PRIMARY KEY,

[Наименование услуги] TEXT NOT NULL,

[Стоимость услуги] MONEY NOT NULL);

**Таблица «Клиенты».**

CREATE TABLE Клиенты /\*Это первая таблица которую я создал\*/

([Код клиента] COUNTER CONSTRAINT PrimaryKey PRIMARY KEY,

[ФИО Клиента] TEXT NOT NULL,

[Место проживания] TEXT NOT NULL,

[Услуги подаваемые в дом] TEXT NOT NULL,

[Год рождения] DATE NOT NULL,

[Номер телефона] INT NOT NULL,

[Паспортные данные] INT NOT NULL,

[Прилагаемые документы] TEXT NOT NULL);

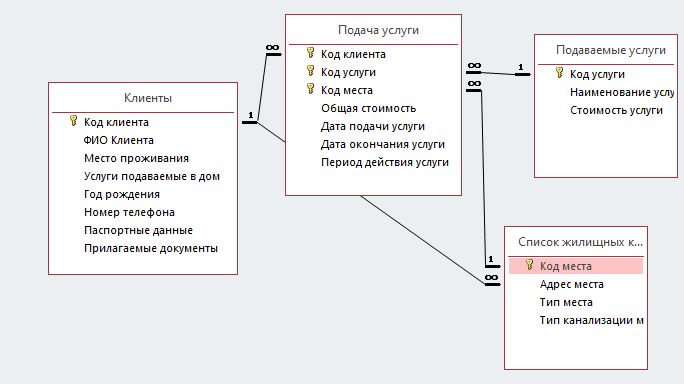


Рисунок №1: схема данных

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

После написания SQLкода для таблиц, я приступил к написанию его же для запросов.

**Запрос «Проверка действия услуги».**

SELECT [Подача услуги].[Дата подачи услуги], [Подача услуги].[Дата окончания услуги], [Подача услуги].[Период действия услуги], [Подача услуги].[Код клиента], [Подача услуги].[Код услуги],[Подача услуги].[Код места]

FROM [Подача услуги]

WHERE ((([Подача услуги].[Период действия услуги])=[Введите сколькок месяцев]));

**Запрос «Проверка подключения услуги».**

SELECT Клиенты.[ФИО Клиента], Клиенты.[Услуги подаваемые в дом], [Подаваемые услуги].[Наименование услуги], [Список жилищных комплексов].[Адрес места]

FROM [Подаваемые услуги], Клиенты INNER JOIN [Список жилищных комплексов] ON Клиенты. [Код клиента] = [Список жилищных комплексов].[Адрес места]

WHERE ((([Подаваемые услуги].[Наименование услуги])=[Введите название услуги]));

После того, как я написал SQLкод для запросов, я приступил к написанию его же для отчётов (рис. №2,3,4).

**Отчёт «Подача услуги».**

SELECT [Подача услуги].[Код Клиента], [Подача услуги].[Код Услуги],

[Подача услуги].[Код Места], [Подача услуги].[Общая стоимость],

[Подача услуги].[Дата подачи услуги], [Подача услуги].[Дата окончания услуги], [Подача услуги].[Период действия услуги]

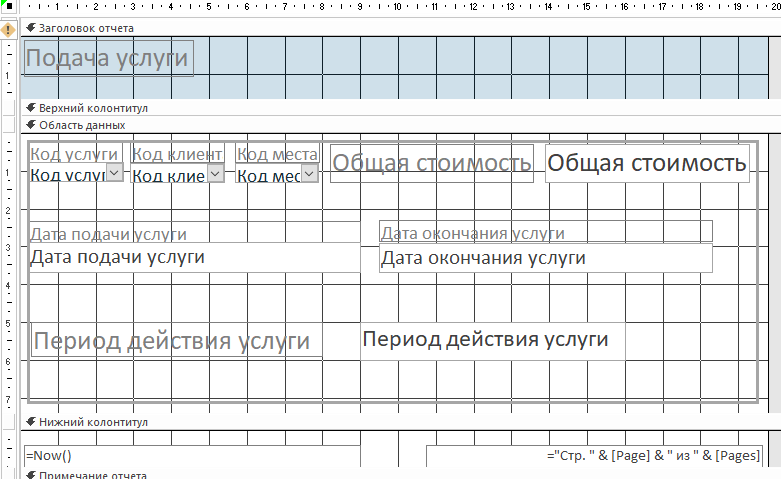
FROM [Подача услуги]; 

Рисунок №2: отчёт «не рекомендую»

**Форма «Клиенты».**

SELECT [Клинты].[ФИО Клиента], [Клиенты].[Местро проживания],

[Клиенты].[Услуги подаваемые в дом], [Клиенты].[Код Подачи]

FROM [Клиенты] INNER JOIN [Клиенты] ON [Клиенты].[Код Клиента] = [Клиенты].[Код Клиента];

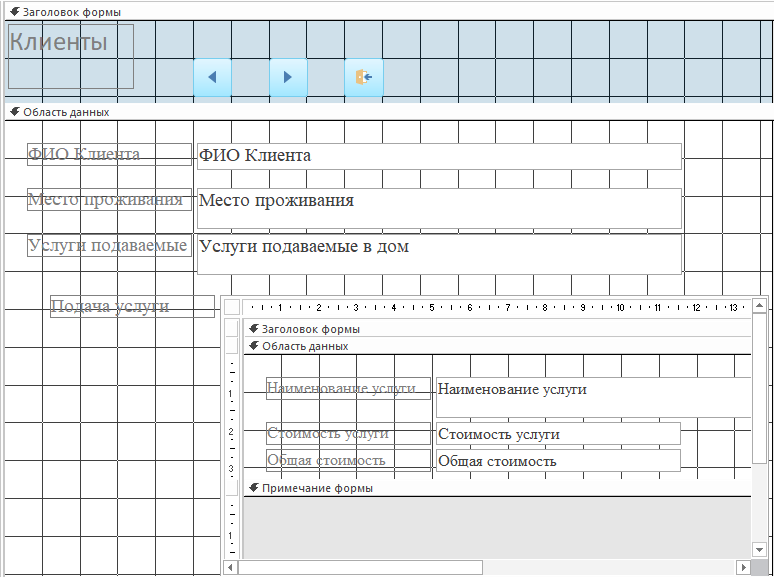


Рисунок №3: Форма «Клиенты».

**Форма «Подача услуги подчиненная форма».**

SELECT [Подаваемые услуги].[Наименование услуги], [Подаваемые услуги].[Стоимость услуги], [Подача услуги].[Общая стоимость], [Подача услуги].[Код клиента]

FROM [Подаваемые услуги] INNER JOIN [Подача услуги] ON [Подаваемые услуги].[Код услуги] = [Подача услуги].[Код услуги];

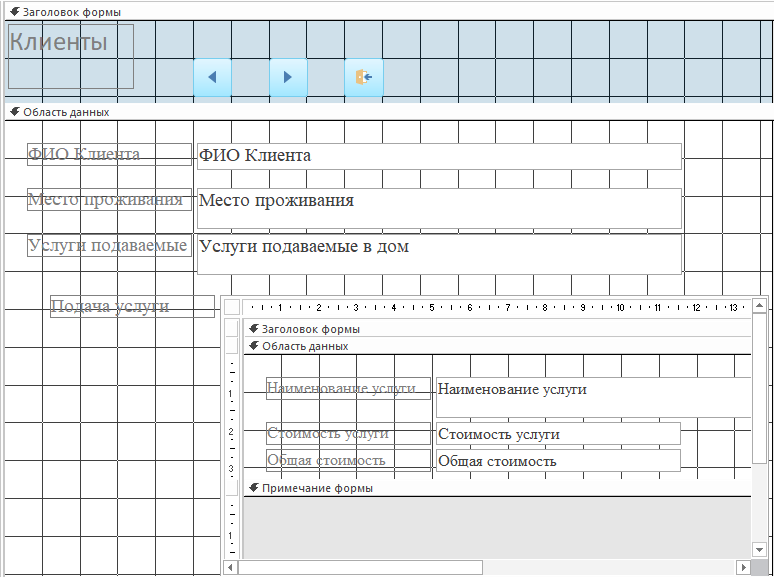


Рисунок № 4: Форма «Подача услуги подчиненная форма».

**3. Расчёт метрик.**

После составления SQL кода я приступил к расчёту метрик

Справочный материал:

**Количественные метрики**

1. Количество строк / Sourcelinesofcode (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 = 55

2. простая метрика стилистики = 18

F=(Nком/Nстр)\*100%

**Метрики Холстеда:**

1. длина программы: N = N1 + N2 = 60

2. объем программы: V = N\*log2(n) = 78,06

3. оценка ее реализации(уровень качества): L= (2\*n2) / (n1\*N2) = 20/400 = 0,05‬

4. трудность ее понимания: Ec=V/ L = 1561,2

5. трудоемкость кодирования: D = (n1 \* N2)(2\*n2) = 1/ L = 20

6. информационное содержание: I = V / D = 3,903

**Метрики Маккейба:**

1. цикломатическое число: Z(G) = e – v + 2p = 2-3+2\*1=1

**Метрики надежности**

1. количество структурных изменений, произведенных с момента проектирования = 5

2. количество ошибок, выявленных в ходе проектирования = 0

3. количество проведенных структурных изменений, необходимых для корректной работы БД = 5

**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.