**Лабораторная работа № 3. T-SQL – язык реляционной базы данных**

**КУРНОСЕНКО СОФЬЯ**

Язык реляционной базы данных в **SQL Server** называется **Transact-SQL** (**T-SQL**). Операторы языка делятся на несколько групп, основными из которых является язык определения данных (**Data Definition Language**, **DDL**) и язык манипулирования данными (**Data Manipulation Language**, **DML**).

Язык **DDL** содержит три обобщенных оператора: **create** *объект* (создание объекта базы данных), **alter** *объект* (изменение характеристик объекта) и **drop** *объект* (удаление существующего объекта). Эти опера­торы создают, изменяют и удаляют объекты базы данных, такие как сама база данных, таблицы, столбцы и индексы.

Язык **DML** содержит операторы, которые манипулируют данными, осуществляя выборку информации (**SELECT**), добавление (**INSERT**), удаление (**DELETE**) и изменение (**UPDATE**). При записи операторов можно использовать на клавиатуре любой регистр.

→ ***Теория (моя)***

**SQL и T-SQL**

**SQL** (Structured Query Language) – язык структурированных запросов, с помощью данного языка пишутся специальные запросы к базе данных с целью получения этих данных и манипулирования ими.

Язык SQL – это стандарт, он реализован во всех реляционных базах данных, но у каждой СУБД есть расширение этого стандарта, так называемый диалект языка SQL.

**T-SQL** – это расширение языка SQL, реализованное в Microsoft SQL Server.

T-SQL – это сокращенное название языка, а полное название Тransact-SQL, т.е. транзакционный SQL.

T-SQL обладает всеми возможностями языка SQL, однако предназначен он для решения задач программирования, не связанных с реляционными данными. Иными словами, на T-SQL Вы можете полноценно программировать, используя переменные, циклы, условные конструкции и другие возможности. T-SQL позволяет нам реализовывать сложные алгоритмы бизнес-логики в виде хранимых процедур и функций, тем самым создавая так называемые «программы» внутри базы данных.

Кроме этого, T-SQL упрощает написание SQL запросов за счет огромного количества встроенных системных функций, а также позволяет администрировать SQL Server и получать системную информацию с помощью системных представлений, процедур и функций.

**Операторы SQL**

С точки зрения реализации язык SQL представляет собой набор операторов, которые делятся на определенные группы и у каждой группы есть свое назначение. В сокращенном виде эти группы называются DDL, DML, DCL и TCL.

* **DDL** - Data Definition Language - язык определения данных
* **DML** - Data Manipulation Language - язык манипулирования данными
* **DCL** - Data Control Language - язык управления данными
* **TCL** - Transaction Control Language - язык управления транзакциями

**DDL – Data Definition Language**

Data Definition Language (DDL) – это группа операторов определения данных. Операторы DDL создают, изменяют и удаляют объекты базы данных, такие как сама база данных, таблицы, столбцы и индексы.

DDL включает операторы:

▫ CREATE – используется для создания объектов базы данных;

▫ ALTER – используется для изменения объектов базы данных (добавить столбец, поменять тип данных столбца и т.д.);

▫ DROP – используется для удаления объектов базы данных.

**DML – Data Manipulation Language**

Data Manipulation Language (DML) – это группа операторов для манипуляции данными. С помощью этих операторов мы можем добавлять, изменять, удалять и выгружать данные из базы, т.е. манипулировать ими.

Операторы DML предназначены *для работы со строками таблиц*.

В эту группу входят самые распространённые операторы языка SQL:

▫ SELECT – осуществляет выборку данных;

▫ INSERT – добавляет новые данные;

▫ UPDATE – изменяет существующие данные;

▫ DELETE – удаляет данные.

**DCL – Data Control Language**

Data Control Language (DCL) – группа операторов определения доступа к данным. Иными словами, это операторы для управления разрешениями, с помощью них мы можем разрешать или запрещать выполнение определенных операций над объектами базы данных.

Операторы DCL предназначены для управления процессом авторизации.

**Авторизация** – это процедура проверки разрешений на выполнение определенных операций.

**Принципал** – это объект сервера или БД, которому может быть выдано разрешение на выполнение операции, а также отобрано или запрещено разрешение.

Сюда входят:

▫ GRANT – предоставляет пользователю или группе разрешения на определённые операции с объектом;

▫ REVOKE – отзывает выданные разрешения;

▫ DENY– задаёт запрет, имеющий приоритет над разрешением.

**TCL – Transaction Control Language**

Transaction Control Language (TCL) – группа операторов для управления транзакциями. **Транзакция** – это несколько DML-операторов, которые либо все выполняются, либо все не выполняются.

Группа операторов TCL предназначена как раз для реализации и управления транзакциями. Сюда можно отнести:

▫ BEGIN TRANSACTION – служит для определения начала транзакции;

▫ COMMIT TRANSACTION – отмечает конец успешной транзакции;

▫ ROLLBACK TRANSACTION – откатывает все изменения, сделанные в контексте текущей транзакции;

▫ SAVE TRANSACTION – устанавливает промежуточную точку сохранения внутри транзакции.

**ЗАДАНИЯ**

*1. Удалить базу данных* ***X\_MyBASE****, созданную с помощью команд* ***Server Management Studio*** *и вновь создать с помощью языка T****-****SQL.*

→ ***Теория (методичка)***

Чтобы создать скрипт на языке **SQL** надо в окне **Редактор запросов**, которое открывается после нажатия кнопки **Создать запрос** на панели инструментов, набрать операторы языка и нажать кнопку **Выполнить** или клавишу <**F5**>.

Оператор CREATE предназначен для создания объектов БД. Например, создать базу данных с именем **ПРОДАЖИ** можно с помощью операторов:

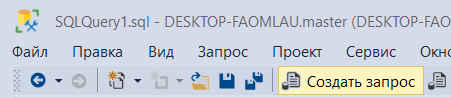
**USE master;**

**CREATE database ПРОДАЖИ;**

Затем надо нажать кнопку **Обновить (Refresh)**.

→ ***Выполнение заданий***

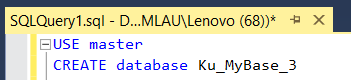
Выбираю *Создать запрос* на панели инструментов:



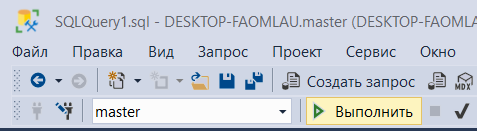
В редакторе запросов ввожу такой запрос:

USE master

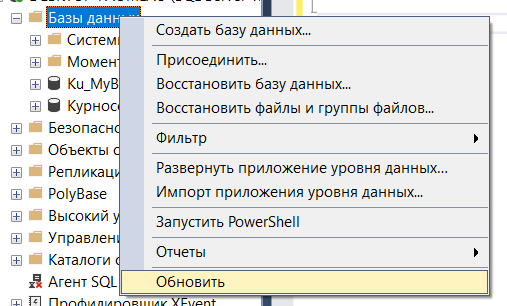
CREATE database Ku\_MyBase\_3



Выполняю:



Обновляю базы данных:



Только что созданная БД появилась в списке:



*2. Разработать сценарии для создания в базе данных* ***X\_MyBASE*** *нужных таблиц.*

*Использовать ограничения целостности.*

*Установить связи между полями.*

*Просмотреть структуры таблиц с помощью команды* ***Проект*** *(****Design****) в контекстном меню таблиц.*

→ ***Теория (методичка)***

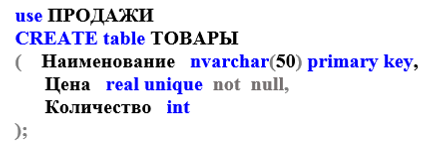
Для сохранения скрипта используется **Файл** / **Сохранить как** (**File** / **Save As**).

При создании таблиц используются различные ограничения. Ограничения, накладываемые на столбцы таблиц баз данных, предотвращают появление данных, не соответствующих предварительно заданным свойствам таблиц. Эти ограничения называются ограничениями целостности. В таблице приведено описание ограничений целостности.

|  |  |
| --- | --- |
| Ограничение | Действие ограничения целостности |
| **data type**  тип данных | Предотвращает появление в столбце значений, не соответствующих типу данных |
| **not null**  запрет значений **null** | Предотвращает появление в столбце значений null |
| **default**  знач. по умолчанию | Устанавливает значение в столбце по умолчанию при выполнении операции INSERT |
| **primary key**  первичный ключ | Предотвращает появление в столбце повторяющихся значений и пустого значения |
| **foreign key**  внешний ключ | Устанавливает связь между таблицей со столбцом, имеющим свойство **foreign key** и таблицей, имеющей столбец со свойством **primary key**; |
| **unique**  уникальное значение | Не допускает пустые и повторяющиеся значения, не может быть использовано для связи с полем другой таблицы |
| **check**  проверка значений | Предотвращает появление в столбце значения, не удовлетворяющего логическому условию |

Ниже записаны скрипты для создания таблиц с ограничениями целостности в базе данных **ПРОДАЖИ** и установления связей между полями таблиц. Здесь связи определяются при создании таблицы **Заказы**.

Создание таблицы **Товары**:



Создание таблицы **Заказчики**:

**CREATE TABLE Заказчики**

**( Наименование\_фирмы nvarchar(20) primary key,**

**Адрес nvarchar(50),**

**Расчетный\_счет nvarchar(20)**

**) ;**

Создание таблицы **Заказы**:

**CREATE TABLE Заказы**

**( Номер\_заказа int primary key**

**Наименование\_товара nvarchar(20) foreign key references**

**Товары (Наименование),**

**Цена\_продажи real,**

**Количество int,**

**Дата\_поставки date,**

**Заказчик nvarchar(20) foreign key references**

**Заказчики(Наименование\_фирмы)**

**)**

→ ***Выполнение заданий***

В редакторе запросов ввожу такой запрос:

USE Ku\_MyBase\_3

CREATE TABLE Водители (

Код\_водителя int PRIMARY KEY,

Фамилия nvarchar(50),

Имя nvarchar(50),

Отчество nvarchar(50),

Стаж smallint CONSTRAINT experience\_positive CHECK(Стаж > 0),

);

CREATE TABLE Маршруты (

Код\_маршрута int PRIMARY KEY,

Название nvarchar(50),

[Дальность, км] float CONSTRAINT distance\_positive CHECK([Дальность, км] > 0),

);

CREATE TABLE Перевозки(

Код\_перевозки int PRIMARY KEY,

Код\_маршрута int FOREIGN KEY REFERENCES Маршруты(Код\_маршрута),

Дата\_отправки datetime,

Дата\_возвращения datetime,

);

CREATE TABLE [Водители на перевозках] (

Код\_перевозки int FOREIGN KEY REFERENCES Перевозки(Код\_перевозки),

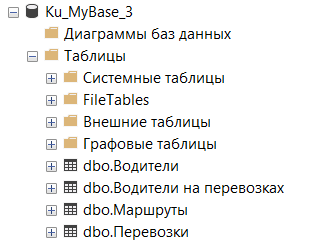
Код\_водителя int FOREIGN KEY REFERENCES Водители(Код\_водителя),

[Оплата, USD] smallmoney CONSTRAINT payment\_positive CHECK([Оплата, USD] > 0),

PRIMARY KEY (Код\_перевозки, Код\_водителя),

);

Обновляю БД Ku\_MyBase\_3:

******

*3. Опробовать процедуру внесения изменения в структуру одной из таблиц с помощью оператора* ***ALTER*** *добавив столбец.*

*С использованием* ***ALTER*** *добавить некоторые ограничения целостности.*

*Просмотреть новую структуру и удалить добавленный столбец*

→ ***Теория (методичка)***

Для модификации существующих объектов базы данных или сервера применяется оператор ALTER. Добавление столбца в таблицу:

**ALTER Table Товары ADD Дата\_поступления date;**

Этот оператор можно записать в тот же скрипт, но выполнить следует, не повторяя предыдущих операторов, для чего оператор надо выделить и нажать **Выполнить** (**Execute**). Можно использовать клавишу **F5**.

Обновить базу данных с помощью кнопки **Обновить** (**Refresh**).

Если значение атрибута равно по умолчанию некоторому сочетанию символов, то используется ограничение **Default** и через пробел в одинарных кавычках нужные символы. Для атрибута может быть введено ограничение **check**. Например, ограничения добавляются с помощью **ALTER**:

**ALTER Table STUDENT ADD POL nchar(1) default 'м' check (POL in ('м', 'ж'));**

Атрибут **POL** может принимать только значения **м** или **ж**.

Для удаления объектов сервера или БД предназначен оператор DROP.

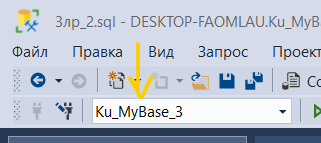
Удаление столбца:

**ALTER Table Товары DROP Column Дата\_поступления;**

Удаление таблицы **ТОВАРЫ**: **DROP table ТОВАРЫ**

→ ***Выполнение заданий***

Выбираем БД, в пределах которой совершаются действия (контекст выполнения выбираем, короче):

******

***НЕ ЗАБЫВАЕМ ПРО КОНТЕКСТ ВЫПОЛНЕНИЯ ИНСТРУКЦИЙ В ЭТОМ И ПОСЛЕДУЮЩИХ ЗАДАНИЯХ.***

В скрипт предыдущего задания добавляю следующие инструкции для создания столбца и добавления ограничений:

--создаем столбец "Пол" в таблице "Водители", задаем ему дефолтное значение и ограничение

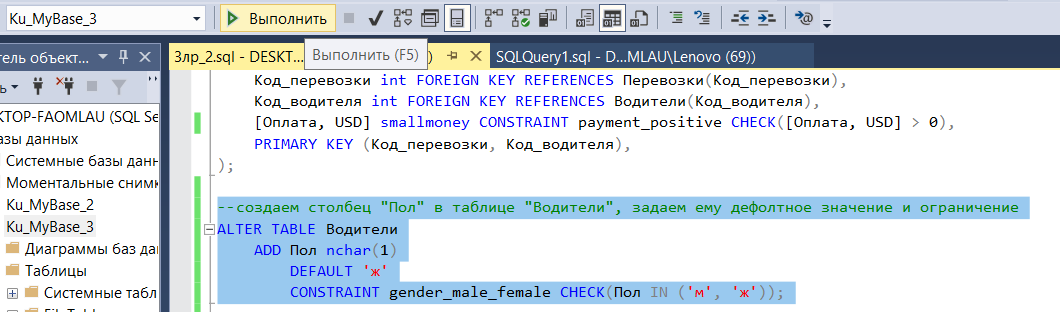
ALTER TABLE Водители

ADD Пол nchar(1)

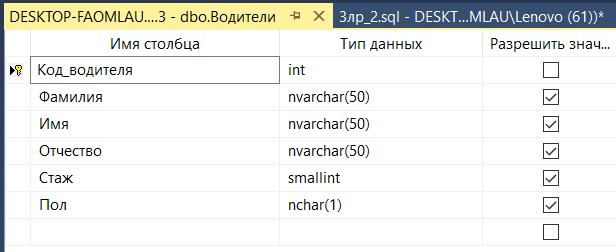
DEFAULT 'ж'

CONSTRAINT gender\_male\_female CHECK(Пол IN ('м', 'ж'));

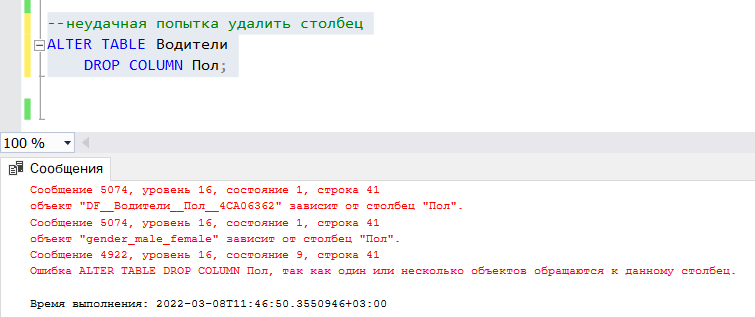
Далее выделяю эти строки и запускаю скрипт (таким образом, выполнятся только выделенные инструкции):



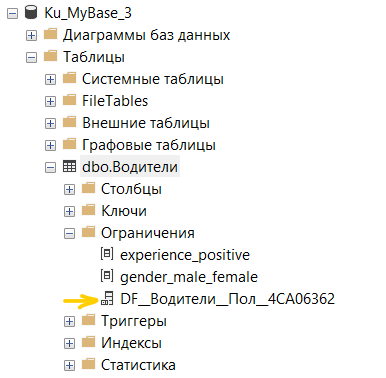
Обновляю БД и теперь таблица «Водители» имеет следующую структуру:



Попытка удалить новосозданный столбец:



Первое сообщение говорит нам об ограничении, которое было создано СУБД автоматически. Ограничения для данной таблицы мы можем просмотреть в обозревателе объектов:



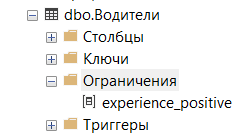
Второе сообщение говорит об ограничении («gender\_male\_female»), которое мы создали сами.

Удалим эти ограничения:

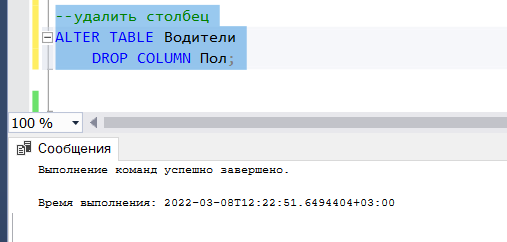
ALTER TABLE Водители

DROP CONSTRAINT DF\_\_Водители\_\_Пол\_\_4CA06362, gender\_male\_female;

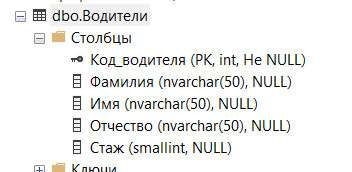
Поле обновления раздела «Ограничения» видим, что ограничения удалены:



Теперь столбец удаляется:



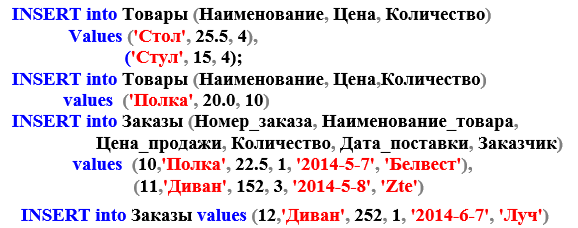
После обновления таблицы «Водители»:



*4. С помощью оператора* ***INSERT*** *заполнить все таблицы информацией.*

→ ***Теория (методичка)***

Оператор **INSERT** используется для ввода информации в таблицу.



→ ***Выполнение заданий***

INSERT INTO Водители

VALUES (1, 'Хмельная', 'Ольга', 'Викторовна', 5),

(2, 'Поляруш', 'Иван', 'Андреевич', 2);

INSERT INTO Маршруты

VALUES (1, 'Гомель - Минск', 310),

(2, 'Брест - Витебск', 627.4),

(3, 'Минск - Могилев', 197.8);

INSERT INTO Перевозки

VALUES (1, 1, '20211112', '20211113'),

(2, 2, '20220212', '20220213'),

(3, 3, '20220308', '20220309');

INSERT INTO [Водители на перевозках]

VALUES (1, 1, 200.0000),

(1, 2, 160.0000),

(2, 2, 180.0000);

Важно соблюдать порядок заполнения таблиц, в зависимости от первичных и внешних ключей.

Строки брать в одинарные кавычки.

Дату записывать в формате YYYYMMDD.

*5. Вывести все строки и столбцы одной из таблиц.*

*Написать оператор* ***SELECT****, выбирающий все строки для двух столбцов таблицы.*

*Подсчитать количество строк в таблице.*

*Опробовать запросы на поиск информации с использованием* ***Where, Distinct,******Top.***

→ ***Теория (методичка)***

**SELECT** − основной оператор для поиска информации в базе данных. Чтобы вывести всю информацию из таблицы **Товары**, можно записать:

**SELECT \* From Товары;**

Для выбора содержимого двух столбцов (**Наименование, Цена**) служит оператор:

**SELECT Наименование, Цена From Товары;**

Чтобы подсчитать количество строк в таблице можно использовать следующую форму оператора:

**SELECT count(\*) From Товары;**

Вывод наименований товаров, цена которых меньше 200, при этом столбец результатов озаглавлен **Дешевые товары**:



Если наименование поля содержит символ пробела, то оно заключается в квадратные скобки.

Секция **DISTINCT** позволяет не выводить повторяющиеся строки, опция **TOP** ограничивает количество результирующих строк.

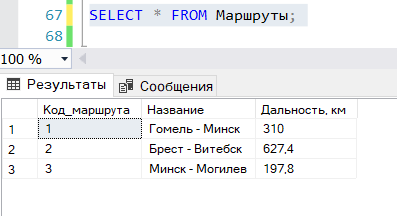
Вывод пяти строк столбцов **Наименование\_товара** и **Цена\_продажи** из таблицы **Заказы**. Результаты отсортированы по убыванию, повторяющиеся строки не выводятся:



→ ***Выполнение заданий***

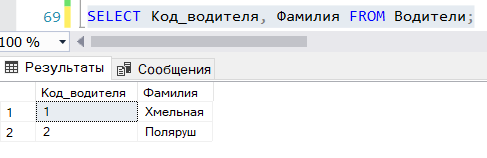
Выбор всех строк и столбцов таблицы «Маршруты»:

SELECT \* FROM Маршруты;

****

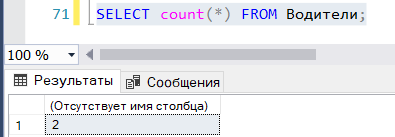
Выбор всех строк двух столбцов таблицы «Водители»:

SELECT Код\_водителя, Фамилия FROM Водители;

****

Подсчет количества строк в таблице «Водители»:

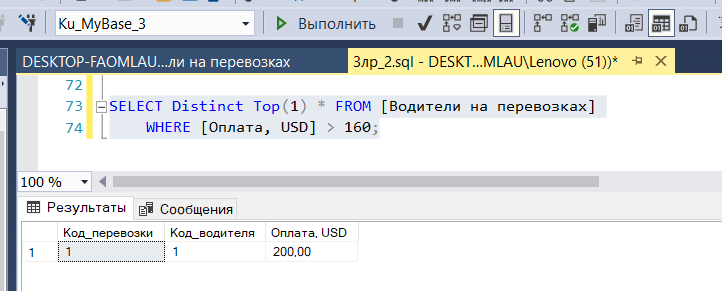
SELECT count(\*) FROM Водители;

****

Из таблицы «Водители на перевозках» выбираем все столбцы, при этом записи в столбцах должны быть уникальны, столбец «Оплата, USD» должен иметь значение больше 160, но в выводе отобразится только первая стока, удовлетворяющая перечисленным условиям:

SELECT Distinct Top(1) \* FROM [Водители на перевозках]

WHERE [Оплата, USD] > 160;



*6. Опробовать процедуру внесения изменения в содержимое одной из таблиц с помощью оператора* ***UPDATE****.*

*Проверить результат, используя* ***Select****.*

→ ***Теория (методичка)***

Для изменения строк таблицы используется оператор **UPDATE**.

Например, в примере, приведенном ниже, содержимое столбца **Количество** заменяется на число 1: **UPDATE ТОВАРЫ set Количество = 1;**

Содержимое столбца **Цена** увеличивается на 1 для товара, наименование которого − **Стол**:

**UPDATE ТОВАРЫ set Цена = Цена+1 Where Наименование = 'Стол';**

Чтобы удалить одну или несколько строк используется оператор **DELETE**. В примере, приведенном ниже, удаляется строка из таблицы **Товары**, в которой значение столбца **Наименование** равно **Стул**.

**DELETE from ТОВАРЫ Where Наименование = 'Стул';**

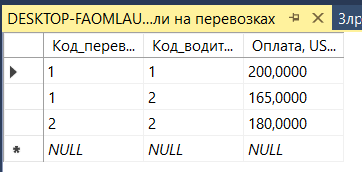
→ ***Выполнение заданий***

UPDATE [Водители на перевозках]

SET [Оплата, USD] = 165

WHERE Код\_перевозки = 1 AND Код\_водителя = 2;

В результате:

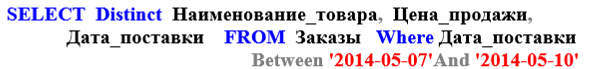
****

*7. Разработать запросы на поиск информации с использованием различных предикатов.*

→ ***Теория (методичка)***

В секции WHERE, помимо операций сравнения, логических связок **AND**, **OR** и **NOT**, могут использоваться предикаты (выражения, значениями которых могут быть только «истина» и «ложь»).

Предикат BETWEEN позволяет выбирать строки с заданным диапазоном значений. Вывод наименования, цены и даты поставки товаров, которые должны быть доставлены между **2014-05-07** до **2014-05-10**:



Ключевое слово LIKE осуществляет сравнение полей с заданным шаблоном, при этом в шаблоне символ **%** обозначает любое количество любых символов:

Вывод товаров, наименования которых начинаются с буквы **С**:



Ключевое слово IN отбирает строки, в которых значение анализируемого поля совпадает с одним из значений, записанных справа от IN в круглых скобках.

Вывод наименований товаров, цена продажи которых равна 15 или 210. Повторяющиеся наименования не выводятся:

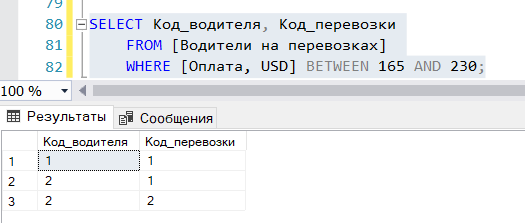


***→ Выполнение заданий***

SELECT Код\_водителя, Код\_перевозки

FROM [Водители на перевозках]

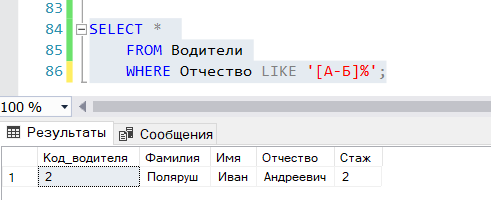
WHERE [Оплата, USD] BETWEEN 165 AND 230;



SELECT \*

FROM Водители

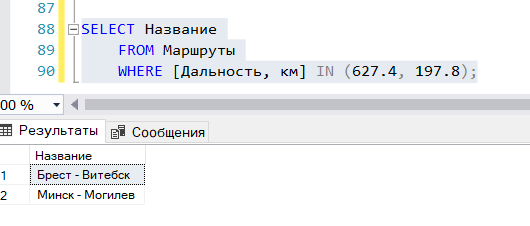
WHERE Отчество LIKE '[А-Б]%';



SELECT Название

FROM Маршруты

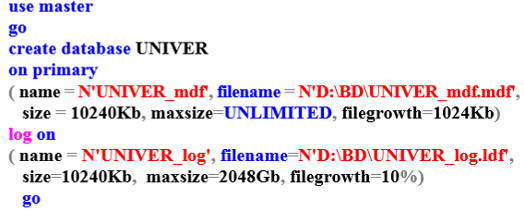
WHERE [Дальность, км] IN (627.4, 197.8);



*8. Внести изменения в сценарий создания базы данных* ***X\_MyBASE*** *с тем, чтобы файлы размещались в определенных местах памяти.*

→ ***Теория (методичка)***

БД представляет собой набор файлов операционной системы трех типов: первичный файл (расширение **mdf**), вторичные файлы (**ndf**) и файлы журнала транзакций (**log**). Все файлы БД, кроме файлов журнала транзакций, распределены по файловым группам. Файловые группы – это проименованный набор файлов БД.



Файловая группа, называемая первичной, является обязательной. Для ее обозначения используются ключевые слова **ON PRIMARY**. В примере создается база **UNIVER**, файл **UNIVER\_mdf.mdf** с первоначальным размером 10240Кb, максимальным размером неограниченным, приращением 1024Кb. Файл располагается на диске **D** в папке **BD** в первичной файловой группе.

Журнал транзакций в операторе **CREATE DATABASE** описывается отдельно в секции, обозначенной ключевыми словами **LOG ON**. Файл **UNIVER\_log.ldf** с первоначальным размером 10240Кb, максимальным размером 2048Gb, приращением 10%, располагается на диске **D** в папке **BD** в журнале транзакций. Логическое имя задается ключевым словом **name**.

Файловые группы используются в основном для упрощения администрирования БД. Существует операторы (например, копирования и восстановления БД), позволяющие рассматривать файловую группу как единое целое и выполнять операции не для каждого файла по отдельности, а сразу для файловой группы.

Пример с использованием вторичной файловой группой:

**use master**

**create database UNIVER on primary**

**( name = N'UNIVER\_mdf', filename = N'D:\BD\UNIVER\_mdf.mdf',**

**size = 10240Kb, maxsize=UNLIMITED, filegrowth=1024Kb),**

**( name = N'UNIVER\_ndf', filename = N'D:\BD\UNIVER\_ndf.ndf',**

**size = 10240KB, maxsize=1Gb, filegrowth=25%),**

**filegroup FG1**

**( name = N'UNIVER\_fg1\_1', filename = N'D:\BD\UNIVER\_fgq-1.ndf',**

**size = 10240Kb, maxsize=1Gb, filegrowth=25%),**

**( name = N'UNIVER\_fg1\_2', filename = N'D:\BD\UNIVER\_fgq-2.ndf',**

**size = 10240Kb, maxsize=1Gb, filegrowth=25%)**

**log on**

**( name = N'UNIVER\_log', filename=N'D:\BD\UNIVER\_log.ldf',**

**size=10240Kb, maxsize=2048Gb, filegrowth=10%)**

В примере файлы (кроме файла журнала транзакций) распределены по двум файловым группам: в первичной файловой группе находятся один первичный и один вторичный файлы; во вторичной файловой группе с именем FG1 – два вторичных файла.

→ ***Выполнение заданий***

Создаю новую БД:

CREATE DATABASE Ku\_MyBase\_3\_8

ON PRIMARY (name='Ku\_MyBase\_3\_mdf', filename='D:\DB\Ku\_MyBase\_3\_mdf.mdf',

size=10240Kb, maxsize=unlimited, filegrowth=1024Kb),

FILEGROUP laba\_3\_8 (name='Ku\_MyBase\_3\_ndf', filename='D:\DB\Ku\_MyBase\_3\_ndf.ndf',

size=10240Kb, maxsize=500Mb, filegrowth=64Kb)

LOG ON (name='Ku\_MyBase\_3\_log', filename='D:\DB\Ku\_MyBase\_3\_log.ldf',

size=10240Kb, maxsize=500Mb, filegrowth=64Kb)

*9. Разместить таблицы базы данных* ***X\_MyBASE*** *в файловых группах.*

***→ Теория (методичка)***

Пример создания таблицы **AUDITORIUM**, предназначенной для хранения данных об аудиторном фонде вуза, в файловой группе **FG1**:

**CREATE TABLE AUDITORIUM**

**( AUDITORIUM char(20) primary key,**

**AUDITORIUM\_TYPE char(10) foreign key references**

**AUDITORIUM\_TYPE(AUDITORIUM\_TYPE),**

**AUDITORIUM\_CAPACITY int default 1**

**check (AUDITORIUM\_CAPACITY between 1 and 300),**

**AUDITORIUM\_NAME varchar(50)**

**) on FG1;**

***→ Выполнение заданий***

В БД, созданной в предыдущем задании, создаю таблицу, которую добавляю в файловую группу:

USE Ku\_MyBase\_3\_8;

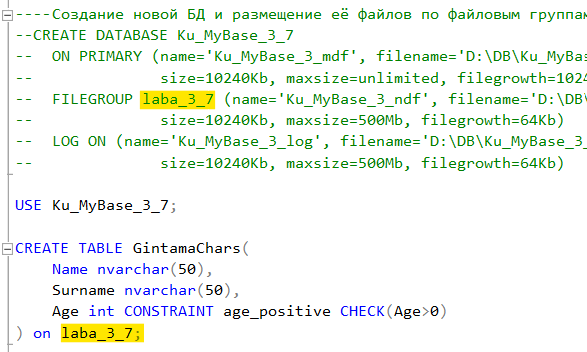
CREATE TABLE GintamaChars(

Name nvarchar(50),

Surname nvarchar(50),

Age int CONSTRAINT age\_positive CHECK(Age>0)

) on laba\_3\_8;

****

*10.**Запустить утилиту* ***SQLCMD*** *с параметром, позволяющим вывести в окно консоли краткую инструкцию о применении утилиты.*

***→ Теория (методичка)***

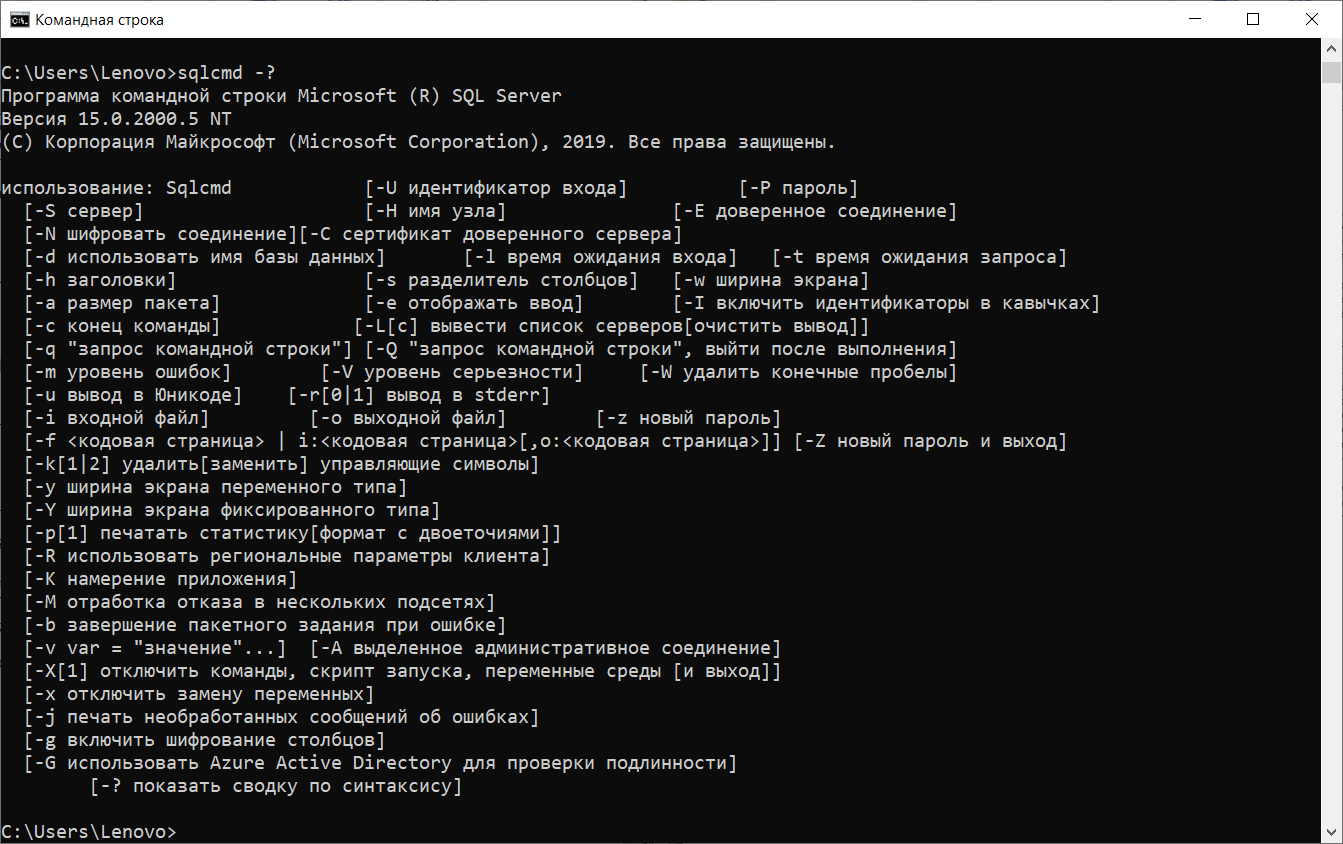
Команды **Transact-SQL** можно вводить и выполнять с помощью утилиты **SQLCMD**. Для этого надо выбрать **Пуск** / **Выполнить** (**Командная строка**).

В поле ввода появившегося окна следует ввести **cmd**, затем команду **sqlcmd** и можно вводить операторы языка. Выход из данного режима – команда **exit**.

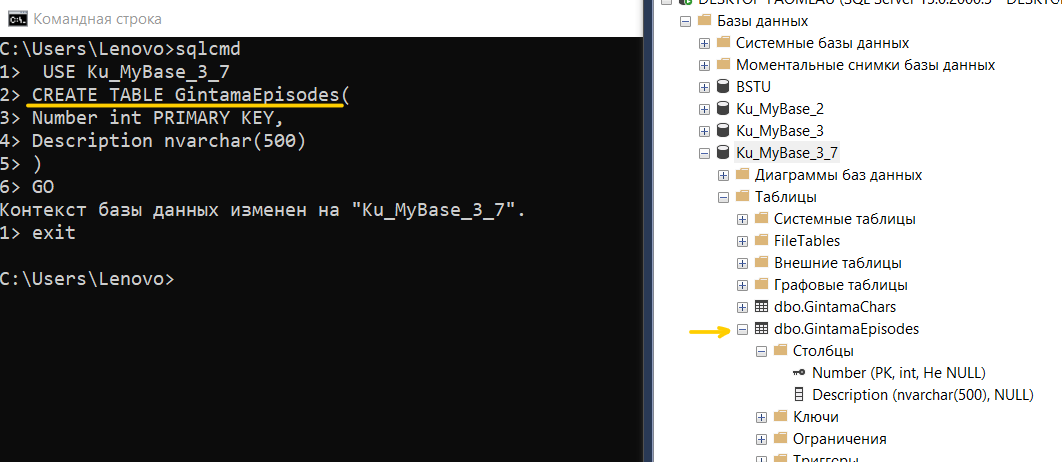
***→ Выполнение заданий***

Запуск cmd.

Ввод команды *sqlcmd -?*:



***→ Теория (моя)***

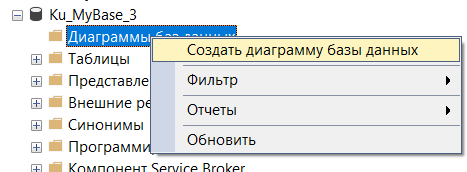
******

11. С помощью **SSMS** просмотреть диаграмму базы данных **X\_MyBASE** и убедиться, что все требуемые внешние ключи (**foreign key**) отражены на диаграмме.

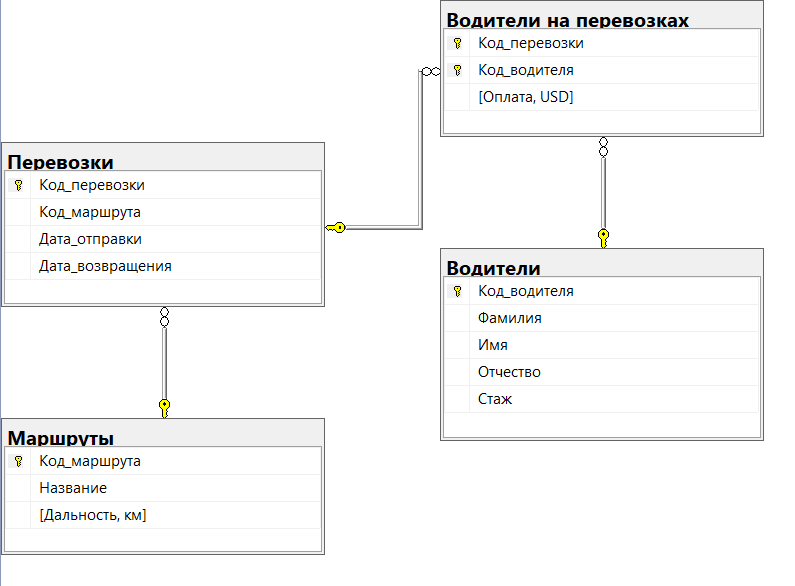
Опробовать различные операторы языка T-SQL на примере базы данных **X\_MyBASE**.

***→ Выполнение заданий***

Выбираем «Создать диаграмму базы данных» в контекстном меню БД:

****

В итоге установленные в коде связи визуализируются:

****