Лабораторная работа № 3.
Создание базы данных и ее объектов с помощью команд языка Transact-SQL

Теоретические сведения

Создание базы данных и ее объектов с помощью команд языка Transact-SQL

1. Запустите утилиту SQL Server Management Studio, после чего раскройте на панели Обозреватель объектов (*Object Explorer)* в древовидной структуре папку Базы данных (*Databases)*.

На панели инструментов нажмите кнопку Создать запрос *(New Query)*, что вызовет появление на экране Окна запросов к базе данных. Это окно можно использовать для последовательного формирования в нем сценария (скрипта), который является тестовым файлом с расширением `.sql` и сохраняется на диске. Сценарий представляет собой последовательность команд языка `Transact-SQL`, с помощью которых можно будет впоследствии автоматически сгенерировать базу данных со всеми ее объектами, а также наполнить ее таблицы данными и выполнить ряд других действий. С помощью сценария можно создать такую же базу данных, что была создана средствами графического интерфейса утилиты в преыдущей лабораторной работе.

Создание сценария заключается в том, что каждый раз мы будем добавлять в Окно запросов очередную команду языка `Transact-SQL` (или группу команд, образующих пакет), проверять ее синтаксис и затем выполнять, нажимая на панели инструментов кнопку Выполнить (*Execute)* (или клавишу *F5*). При этом необходимо, чтобы данная команда (или пакет) была выделена подсветкой, иначе будут выполнены все без исключения команды, содержащиеся в Окне запросов.

Если очередная команда (или пакет) будет успешно выполнена, то в окно *Query* можно дописать новую команду (или пакет), выделить подсветкой и далее повторить процесс, описанный выше. В конечном итоге, путем поочередного выполнения команд, будет создана база данных со всеми ее объектами и параллельно в Окне запросов будет сформирован сценарий, который можно сохранить в файле на диске. Теперь с помощью этого сценария можно на любом компьютере, где установлен MS SQL Server, выполнить процедуру создания разработанной базы данных.

Полный синтаксис команд языка `Transact-SQL` занимает вместе с описанием всех аргументов не один десяток страниц. Его можно изучить в литературе или библиотеке *MSDN* (*например,* https://msdn.microsoft.com/ru-ru/library/bb545450.aspx*)*.

Практические задания

Задание 1. Создание новой базы данных с помощью команд языка Transact-SQL

1. *Создание базы данных*. Команда создания базы данных TRADE_XXX, которую нужно вставить в Окно запросов, имеет следующий вид:

```
CREATE DATABASE TRADE_XXX -- Вместо XXX поставьте свою комбинацию
цифр
ON PRIMARY -- Путь к файлам БД уже должен существовать
(NAME = TRADE_XXX,
      FILENAME = 'D:\TRADE__Ivanov\TRADE_XXX.mdf',
      SIZE = 5120KB,
      MAXSIZE = UNLIMITED,
      FILEGROWTH = 1024KB)
LOG ON
(NAME = TRADE_XXX_log,
      FILENAME = 'D:\TRADE_Ivanov\TRADE_XXX_log.ldf',
      SIZE = 2048KB,
      MAXSIZE = 2048GB,
      FILEGROWTH = 10%)
GO
```

Примечание. Команда GO используется для указания конца пакета. Команды, образующие пакет, обрабатываются сервером за один раз, после чего он возвращает клиенту (в нашем случае - пользователю SQL Server Management Studio) результат. После этого клиент может отправить следующий пакет и т.д. В данном случае пакет содержит всего одну команду, предназначенную для создания базы данных.

2. *Подключение к базе данных*.

```
USE TRADE_XXX
GO
```

Примечание. При обращении к объектам текущей базы данных (т.е. выбранной с помощью команды USE) не требуется указание ее имени. Но когда работа ведется в контексте иной базы данных, то необходимо указывать также имя базы данных.

Задание 2. Создание таблиц и индексов базы данных с помощью команд языка Transact-SQL

1. ***Создание таблиц базы данных***

Создайте таблицу Locations:

```
CREATE TABLE Locations(
     LocationID int NOT NULL PRIMARY KEY,
```

```
      Country varchar(20) NOT NULL DEFAULT ('Беларусь'),
      Region varchar(20) NOT NULL,
      City varchar(20) NOT NULL,
      Address varchar(50) NOT NULL,
      Phone char(15) NULL,
-- создание уникального индекса по четырем полям
      constraint IX_Four unique(Country, Region, City, Address)
)
```

Создайте таблицу Clients:

```
CREATE TABLE Clients(
      ClientID int PRIMARY KEY IDENTITY(1,1) NOT NULL,
      ClientName varchar (40) NOT NULL,
      HeadFullName varchar (60) NULL,
      Location int NULL,
-- создание ограничения по внешнему ключу
CONSTRAINT FK_Clients_Locations FOREIGN KEY (Location)
      REFERENCES Locations(LocationID)
ON UPDATE CASCADE
)
GO
```

Создайте таблицу Currency:

```
CREATE TABLE Currency(
      CurrencyID char(3) PRIMARY KEY NOT NULL,
      CurrencyName varchar(30) NOT NULL,
      Rate smallmoney NOT NULL CHECK (Rate>0)
)
```

Создайте таблицу Products:

```
CREATE TABLE Products(
      ProductID int PRIMARY KEY NOT NULL,
      ProductName varchar(50) NOT NULL,
      Measure char(10) NULL,
      Price money NULL CHECK  (Price>=0),
      Currency char(3) NULL,
 CONSTRAINT FK_Products_Currency FOREIGN KEY(Currency)
      REFERENCES dbo.Currency (CurrencyID)
)
GO
```

Создайте таблицу Orders:

```
CREATE TABLE Orders(
      OrderID int IDENTITY(1,1) NOT NULL,
      Client int NOT NULL,
      Product int NOT NULL,
      Quantity numeric(12,3) NULL,
```

```
      OrderDate date NULL DEFAULT getdate(),
      DeliveryDate date NULL DEFAULT getdate()+14,
      PRIMARY KEY ( -- составной первичный ключ
            OrderID ASC,
            Client ASC,
            Product ASC
),
CONSTRAINT FK_Orders_Clients FOREIGN KEY(Client)
      REFERENCES dbo.Clients (ClientID)
      ON UPDATE CASCADE,
CONSTRAINT FK_Orders_Products FOREIGN KEY(Product)
      REFERENCES dbo.Products (ProductID)
      ON UPDATE CASCADE
)
GO
```

2. ***Создание уникальных и неуникальных индексов таблицы***

```
CREATE UNIQUE INDEX UIX_Clients ON Clients(ClientName)
CREATE UNIQUE INDEX UIX_Currency ON Currency(CurrencyName)
CREATE UNIQUE INDEX UIX_Products ON Products(ProductName)
CREATE INDEX IX_Locations ON Locations (Country, City)
CREATE INDEX IX_Products ON Products(ProductName, Measure)
CREATE INDEX IX_Orders ON Orders(OrderDate)
```

3. ***Создайте диаграмму данных для созданной БД***

Диаграмму данных создайте также, как в задании 3.2 лабораторной работы номер 2. Проверьте правильность установленных при создании таблиц связей. Созданная диаграмма должна точно соответствовать Рисунок 44.

Задание 3. Сохранение сценария. Создание базы данных с помощью сценария

1. Сохраните текущий сценарий в файле `D:\FIO\TRADE_XXX\CreateDB_Tables.sql` помощью команды меню Файл – Сохранить *(File - Save as)*...

2. Обновите данные в окне Обозреватель решений (*Object Explorer)*. Для этого в ней нужно выделить папку Базы данных (*Databases)* и в ее контекстном меню выберите команду Обновить (*Refresh)* (или нажать соответствующую кнопку в верхней части этой панели). В результате база данных `TRADE_XXX` станет видимой в окне Обозреватель решений (*Object Explorer)*.

3. Переключитесь на другую базу данных, например, `Master`. Для этого выберите базу данных `Master` в выпадающем списке, расположенном на панели инструментов. Другой вариант - набрать в Окне запросов (*Query)* две команды (`USE Master` и `GO`), выделить их и выполнить.

4. Удалите созданную базу данных TRADE_XXX, поскольку теперь мы ее всегда можем сгенерировать с помощью сценария CreateDB_Tables.sql. Для этого в окне Обозреватель решений (*Object Explorer)* выберите базу данных TRADE_XXX и в ее контекстном меню выполните команду Удалить (*Delete)*.

Замечание. Базу данных невозможно удалить, если предварительно не закрыть существующее соединение или не переключиться на другую базу данных, что и было сделано в предыдущем пункте. Можно обойтись без выполнения предыдущего пункта и сразу удалить текущую базу данных TRADE_XXX, но в этом случае после появления на экране окна Удаление объекта (*Delete Object)* нужно в нижней его части установить флажок Закрыть существующие соединения (*Close Existing Connections)*.

5. Закройте утилиту SQL Server Management Studio.

Задание 4. Подготовка базы данных для ввода данных

1. Запустите утилиту SQL Server Management Studio, в окне Обозреватель решений (*Object Explorer)* в древовидной структуре раскройте папку Базы данных (*Databases)*.

2. С помощью команды меню Файл – Открыть – Файл (*File - Open – File)* загрузите сценарий из файла D:\FIO\TRADE_XXX\CreateDB_Tables.sql в Окно запросов (*Query)*.

Выполните сценарий, нажав на панели инструментов кнопку Выполнить (*Execute)* (или клавишу F5). В результате будет создана база данных TRADE_XXX.

3. Обновите данные в окне Обозреватель решений (*Object Explorer)*. Для этого используйте команду Обновить (*Refresh)* в контекстном меню папки Базы данных (*Databases)* или соответствующую кнопку в верхней части панели. В результате база данных TRADE_XXX станет видимой в окне Обозреватель решений (*Object Explorer).*

Далее продолжите работу с базой данных TRADE_XXX, последовательно добавляя в сценарий, выделяя подсветкой и выполняя приведенные ниже команды или пакеты языка Transact-SQL.

Задание 5. Ввод данных в таблицы базы данных посредством запросов на языке SQL

1. Вставьте данные в таблицу Locations (Рисунок 46):

LocationID	Country	Region	City	Address	Phone
101	Беларусь	Витебская	Полоцк	ул.Лесная, 6	+375172691376
102	Беларусь	Гродненская	Лида	ул.Моховая, 12	*NULL*
103	Беларусь		Минск	ул.Маркса, 24	*NULL*
201	Россия	Московская	Королев	ул.Труда, 8	387-23-04
202	Россия		Москва	ул.Тверская, 25	900-8876
301	Украина		Киев	ул. Крещатик, 14	*NULL*
302	Украина	Львовская	Моршин	ул.Франко, 24	*NULL*

Рисунок 46. Таблица `Locations`.

```
INSERT INTO Locations
           (LocationID, Country, Region, City, Address, Phone)
     VALUES
           (101,'Беларусь','Витебская','Полоцк','ул.Лесная, 6',
      '+375172691376')


INSERT INTO Locations
           (LocationID, Region, City, Address)
     VALUES
           (103,'','Минск','ул.Маркса, 24')


INSERT INTO Locations
           (LocationID, Region, Address, City)
     VALUES
           (103, 'Гродненская', 'ул.Моховая, 12', 'Лида')

INSERT INTO Locations
       VALUES
     (301, 'Украина','', 'Киев', 'ул. Крещатик, 14',   NULL),
      (302, 'Украина','Львовская','Моршин','ул.Франко, 24',   NULL)
```

Замечание. Здесь приведены различные способы использования команды `INSERT`, которые учитывают следующие особенности.

- Если в строке `INSERT INTO` список имен столбцов опущен, то в строке `VALUES` необходимо указывать значения для всех столбцов, которые имеет таблица. В противном случае значения указываются только для тех столбцов, наименование которых фигурируют в строке `INSERT INTO`.
- Несмотря на то, что значение в столбце `Страна` является обязательным (определен как `NOT NULL`), его имя можно не указывать в списке имен столбцов, поскольку для этого столбца определено значение по умолчанию (`'Беларусь'`).
- Если значение поля `Область` не заполняется, то нельзя использовать `NULL`, а нужно задать пустую строку (`''`), поскольку поле `Область` является

обязательным и значение NULL в нем недопустимо.

- Возможно введение данных в несколько строк с помощью одной команды INSERT INTO.
- Все типы текстовые данных должны быть заключены в одиночные кавычки.

2. Вставьте данные в таблицу Clients (Рисунок 47), используя различные варианты команды INSERT INTO:

ClientID	ClientName	HeadFullName	Location
1	ГП "Верас"	Прокушев Станислав Игоревич	202
2	ИП "Темп"	Васько Григорий Терентьевич	101
3	УП "Вера"	Прокуш Станислав Игоревич	102
4	ИП "Темпера"	Вась Григорий Терентьевич	102
5	ОАО "Старт"	Кулагин Василий Петрович	103
6	ОАО "Рога и копыта"	Бендер Остап	301

Рисунок 47. Таблица Clients.

Замечание. В таблице Clients столбец ClientID является автоинкрементным и, поэтому, его значения не приведены и задавать их в запросах не нужно.

3. Вставьте данные в таблицу Currency (Рисунок 48):

CurrencyID	CurrencyName	Rate
BYN	БелРубНов	1,0000
BYR	БелРуб	0,0001
EUR	Евро	2,2900
RUR	Российские рубли	0,0320
USD	Доллары США	1,9350

Рисунок 48. Таблица Currency.

Вставьте данные в таблицу Products (Рисунок 49). Обратите внимание на то, что поле Currency таблицы Products является внешним ключом.

ProductID	ProductName	Measure	Price	Currency
111	Монитор 24"	шт	199,0000	USD
333	Кабель	10м	2,5000	EUR
444	Винчестер HDD 4 TB	шт	299,9000	BYN
555	Компьютер	шт	999,0000	USD
666	Планшет	шт	1205,0000	USD
777	Винчестер HDD 1100GB	шт	119,9900	BYN

Рисунок 49. Таблица Products

4. Вставьте данные в таблицу Orders ().

OrderID	Client	Product	Quantity	OrderDate	DeliveryDate
1	1	111	14,000	2017-04-12	2017-05-03
2	2	444	27,000	2017-05-09	2017-05-25
3	3	777	58,000	2017-04-19	2017-05-03
4	2	111	122,000	2017-04-04	2017-05-03
5	4	444	25,000	2017-04-04	2017-05-10
6	3	111	37,000	2017-04-24	2017-05-10
7	2	555	10,000	2017-06-30	2017-09-25
8	6	777	25,000	2017-05-30	2017-05-15
9	2	111	80,000	2017-04-04	2017-04-14

Рисунок 50. Таблица `Orders`.

Замечание. В таблице `Orders` столбец `OrderID` является автоинкрементным и, поэтому, его значения не приведены.

```
SET DATEFORMAT dmy    -- задаем привычный формат даты день.месяц.год, т.к.
                      --по умолчанию установлен формат год.месяц.день
INSERT INTO Orders    -- год можно задавать как 2-мя, так и 4-мя цифрами
     VALUES (2, 111, 14, '12.04.17', '05.03.17')
```

Задание 6. Создание представления базы данных

1. ***Создание представления базы данных.***

Создайте представление с имеем OrdersCost.

```
CREATE VIEW OrdersCost AS
SELECT dbo.Clients.ClientName, dbo.Products.ProductName,
      dbo.Orders.OrderDate, dbo.Orders.DeliveryDate,
      dbo.Products.Price * dbo.Orders.Quantity AS COST
FROM  dbo.Clients INNER JOIN
       dbo.Locations ON dbo.Clients.Location =
dbo.Locations.LocationID
INNER JOIN
       dbo.Orders ON dbo.Clients.ClientID = dbo.Orders.Client
INNER JOIN
       dbo.Products ON dbo.Orders.Product = dbo.Products.ProductID
ORDER BY dbo.Clients.ClientName, dbo.Products.ProductName DESC
```

Результат выполнения представления OrdersCost показан на Рисунок 51.

ClientName	ProductName	OrderDate	DeliveryDate	COST
ГП "Верас"	Монитор 24"	2017-04-12	2017-05-03	2786,0000000
ИП "Темп"	Винчестер HDD 4 TB	2017-05-09	2017-05-25	8097,3000000
УП "Вера"	Винчестер HDD 1100GB	2017-04-19	2017-05-03	6959,4200000
ИП "Темп"	Монитор 24"	2017-04-04	2017-05-03	24278,0000000
ИП "Темпера"	Винчестер HDD 4 TB	2017-04-04	2017-05-10	7497,5000000
УП "Вера"	Монитор 24"	2017-04-24	2017-05-10	7363,0000000
ИП "Темп"	Компьютер	2017-06-30	2017-09-25	9990,0000000
ОАО "Рога и к...	Винчестер HDD 1100GB	2017-05-30	2017-05-15	2999,7500000
ИП "Темп"	Монитор 24"	2017-04-04	2017-04-14	15920,0000000

Рисунок 51. Результат выполнения представления OrdersCost.

Задание 7. Анализ полученных результатов

1. Сохраните созданный итоговый сценарий в файле `D:\FIO\TRADE_XXX\CreateDB_Insert_Data.sql` с помощью команды меню Файл – Сохранить (*File - Save)* (или соответствующей кнопки на панели инструментов). Далее закройте Окно запросов (*Query)*, содержащее сценарий `CreateDB_Insert_Data.sql`.

2. Удалите базу данных `TRADE_XXX`. Для этого в ее контекстном меню выберите команду Удалить (*Delete)* и затем в появившемся окне Удаление объекта (*Delete Object)* установите флажок Закрыть существующие соединения (*Close Existing Connections)*.

3. С помощью команды меню Файл – Открыть – Файл (*File - Open –File)* загрузите сценарий из файла `CreateDB_Insert_Data.sql`, после чего, нажав на панели инструментов кнопку Выполнить (*Execute)*, создайте базу данных `TRADE_XXX` заново.

4. Обновите данные в окне Обозреватель объектов и сделайте базу данных `TRADE_XXX` видимой.

5. Убедитесь, что с помощью сценария получена база данных `TRADE__XXX` с требуемыми объектами и свойствами.

6. Проведите сравнительный анализ лабораторных работ 1 и 2, т.к. их выполнение привело к получению одного и того же результата - баз данных `TRADE_XXX_1` и `TRADE_XXX` соответственно. Укажите для каждого пункта, связанного с созданием объектов базы данных с помощью графического интерфейса (база данных `TRADE_XXX_1`), соответствующую ему команду языка `Transact-SQL` и, в частности, какие ее фрагменты связаны с установкой тех или иных свойств конкретного объекта базы данных.

7. Удалите созданную базу данных TRADE_XXX.