

Лабораторная работа 3.3

Создание таблиц и обеспечение целостности данных.

Пакеты. Команда GO

Создание таблиц. Команда *CREATE TABLE*. Первичный и внешний ключи. Резервные ключи. Ограничения *PRIMARY KEY*, *NULL*, *NOT NULL*, *UNIQUE*, *FOREIGN KEY*, *CHECK*. Составной первичный ключ. столбец идентификаторов (*IDENTITY*). Указание значений по умолчанию для столбца (*DEFAULT*). Установка ограничений на уровне столбца/таблицы. Имена ограничений (*CONSTRAINT*). Обеспечение целостности данных: целостность таблицы, ссылочная целостность. Внешние ключи.

1. Теоретические сведения

1.1 Рассмотрим общий синтаксис команды **CREATE TABLE**

Полный синтаксис:

<https://learn.microsoft.com/Ru-Ru/sql/t-sql/statements/create-table-transact-sql?view=azuresqldb-current>

Простой синтаксис **CREATE TABLE** (общий, если не используются параметры):

CREATE TABLE

```
{ database_name.schema_name.table_name | schema_name.table_name | table_name }  
( { <column_definition> } [ , ... n ] )
```

[;]

Аргументы

database_name

Имя базы данных, в которой создается таблица. Параметр *database_name* должен указывать имя существующей базы данных. Если не указано, в качестве *database_name* по умолчанию выбирается текущая база данных. Имя входа для текущего соединения должно быть связано с идентификатором пользователя, существующего в базе данных, указанной аргументом *database_name*, а этот пользователь должен обладать разрешениями *CREATE TABLE*.

schema_name

Имя схемы, которой принадлежит новая таблица.

table_name⁹⁺

Имя новой таблицы. Имена таблиц должны соответствовать правилам для идентификаторов.

Аргумент **table name** может состоять не более чем из 128 символов, за исключением имен локальных временных таблиц (имена с префиксом из одного символа решетки (#)), длина которых не должна превышать 116 символов.

2. Атрибуты и ограничения

2.1 **PRIMARY KEY**

В таблице возможно наличие только одного ограничения по первичному ключу. **Ограничение первичного ключа (*PRIMARY KEY*) обеспечивает уникальность строк и запрещает хранить значения *NULL* в соответствующих столбцах.**

Если первичный ключ состоит из нескольких полей, то эти поля необходимо перечислить в скобках через запятую. Все поля, которые входят в первичный ключ, должны иметь характеристику *NOT NULL*. В этом случае, при попытке добавить новую строку, в которой значения в этих полях будут одинаковы с любой из уже существующих строк таблицы, база данных возвратит ошибку и не сможет добавить такую строку.

Всегда создавайте ключевое поле. В MS SQL Server для этого можно использовать автоматически увеличиваемое поле, которое не требует дополнительных расходов на сопровождение. В настоящее время Microsoft не рекомендует использовать в качестве ключа автоматически увеличиваемые поля. Да, эти поля достаточно просты в сопровождении, но не всегда эффективны, потому что автоматически увеличиваемое поле нельзя редактировать.

Сейчас рекомендуется для ключа создавать поле типа **GUID (Global Unique Identifier, Глобально Уникальный Идентификатор)**. Значения полей этого типа можно редактировать и достаточно просто сгенерировать самостоятельно. Ограничение, которое обеспечивает целостность сущностей для указанного столбца или столбцов с помощью уникального индекса. **Можно создать только одно ограничение *PRIMARY KEY* для таблицы.**

2.2 IDENTITY

Указывает, что новый столбец является столбцом идентификаторов. Столбцы идентификаторов обычно используются с ограничением PRIMARY KEY для поддержания уникальности идентификаторов строк в таблице. Свойство IDENTITY может назначаться столбцам типа tinyint, smallint, int, bigint, decimal(p, 0) или numeric(p, 0). *Для каждой таблицы можно создать только один столбец идентификаторов. Ограниченные значения по умолчанию и ограничения DEFAULT не могут использоваться в столбце идентификаторов.* Необходимо указать как начальное значение, так и приращение, или же не указывать ничего. **Если ничего не указано, применяется значение по умолчанию (1,1).**

2.3. UNIQUE

Ограничение, которое обеспечивает целостность сущностей для указанного столбца или столбцов с помощью уникального индекса. **Ограничение UNIQUE обеспечивает уникальность строк и реализует такую концепцию реляционной модели, как резервные ключи.** В отличие от первичных ключей в одной таблице возможно присутствие нескольких ограничений UNIQUE. Кроме того, уникальность распространяется на все столбцы, включая те, которые могут иметь пустые значения. В соответствии со спецификацией SQL столбцы с ограничением UNIQUE поддерживают разные отметки типа NULL (как будто они отличаются). Однако язык T-SQL, реализованный в SQL Server, запрещает дублировать эти отметки (любые два значения NULL не могут быть равными). В таблице может быть несколько ограничений UNIQUE.

2.4. Правила допустимости значения NULL в рамках определения таблицы

Допустимость значений NULL для столбца определяет, разрешает ли столбец использовать такие значения для своих данных. Значение NULL не является нулевым или пустым. NULL означает, что никаких данных не вводилось либо значение NULL было указано явно. Обычно оно подразумевает, что значение неизвестно или неприменимо. **Рекомендуется всегда явно определять столбец как NULL или NOT NULL для невычисляемых столбцов или, если используется пользовательский тип данных, разрешать, чтобы для столбца применялась возможность, установленная для этого типа по умолчанию.**

2.5 DEFAULT

Указывает значение, присваиваемое столбцу в случае отсутствия явно заданного значения при вставке. **Столбец может иметь только определение DEFAULT.**

Ограничение DEFAULT может содержать значения констант, функции, стандартные функции без параметров SQL или значение NULL. Определения DEFAULT (вместе с инструкциями INSERT и UPDATE) позволяют указать значение по умолчанию, используемое, если значение не задано. Кроме констант, определения DEFAULT могут включать функции.

Определения DEFAULT нельзя создавать для столбцов с типом данных timestamp или столбцов со свойством IDENTITY.

Определения DEFAULT нельзя создавать для столбцов с псевдонимами типов данных, если такой тип привязан к определенному по умолчанию объекту.

2.6 CHECK

Ограничение, обеспечивающее целостность домена путем ограничения возможных значений, которые могут быть введены в столбец или столбцы. Ограничения CHECK в вычисляемых столбцах должны быть также помечены как PERSISTED. **Столбец может содержать любое количество ограничений CHECK, а условие может включать несколько логических выражений, соединенных операторами AND и OR.** При указании нескольких ограничений CHECK для столбца их проверка производится в порядке создания.

Ограничение CHECK уровня столбца может ссылаться только на ограничиваемый столбец, а ограничение CHECK уровня таблицы — только на столбцы этой таблицы.

Если для столбца или столбцов задано правило либо одно или несколько ограничений CHECK, применяются все ограничения.

Ограничения CHECK нельзя определять для столбцов типов text, ntext или image.

2.7 CONSTRAINT

Необязательное ключевое слово, указывающее на начало определения ограничения PRIMARY KEY, NOT NULL, UNIQUE, FOREIGN KEY или CHECK.

2.8 FOREIGN KEY REFERENCES

Ограничение, которое обеспечивает ссылочную целостность данных в этом столбце или столбцах. Ограничения FOREIGN KEY требуют, чтобы каждое значение в столбце существовало в соответствующем связанном столбце или столбцах в связанной таблице.

Ограничения FOREIGN KEY могут ссылаться только на таблицы в пределах той же базы данных на том же сервере.

!!Столбцы, участвующие в связи по внешнему ключу, должны иметь одинаковую длину и масштаб.

Ограничения FOREIGN KEY могут ссылаться только на столбцы, являющиеся ограничениями PRIMARY KEY или UNIQUE в связанной таблице.

Ограничения FOREIGN KEY не применяются к временным таблицам.

Если столбцу, имеющему ограничение внешнего ключа, задается значение, отличное от NULL, такое же значение должно существовать и в указываемом столбце; в противном случае будет возвращено сообщение о нарушении внешнего ключа. Если не указаны исходные столбцы, ограничения FOREIGN KEY применяются к предшествующему столбцу.

Предложение REFERENCES ограничения внешнего ключа на уровне столбца может содержать только один ссылочный столбец. Этот столбец должен принадлежать к тому же типу данных, что и столбец, для которого определяется ограничение.

Предложение REFERENCES ограничения внешнего ключа на уровне таблицы должно содержать такое же число ссылочных столбцов, какое содержится в списке столбцов в ограничении. Тип данных каждого ссылочного столбца должен также совпадать с типом соответствующего столбца в списке столбцов. Ссылочные столбцы должны быть указаны в том же порядке, который использовался при указании столбцов первичного ключа или уникального ограничения в упоминаемой таблице.

Если частью внешнего ключа или ключа, на который указывает ссылка, является столбец типа timestamp, ключевые слова CASCADE, SET NULL и SET DEFAULT указывать нельзя.

Компонент Database Engine не имеет стандартного предела на количество ограничений FOREIGN KEY, содержащихся в таблице, ссылающейся на другие таблицы, или на количество ограничений FOREIGN KEY в других таблицах, ссылающихся на указанную таблицу. Тем не менее фактическое количество ограничений FOREIGN KEY, доступных для использования, ограничивается конфигурацией оборудования, базы данных и приложения. Рекомендуется, чтобы таблица содержала не более 253 ограничений FOREIGN KEY, а также чтобы на нее ссылалось не более 253 ограничений FOREIGN KEY.

Выражения ON DELETE и ON UPDATE

ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }

Определяет операцию, которая производится над строками создаваемой таблицы, если эти строки имеют ссылочную связь, а строка, на которую имеются ссылки, удаляется из родительской таблицы.

Параметр по умолчанию — NO ACTION.

NO ACTION Компонент Компонент Database Engine формирует ошибку, и выполняется откат операции удаления строки из родительской таблицы.

CASCADE Если из родительской таблицы удаляется строка, соответствующие ей строки удаляются и из ссылающейся таблицы.

SET NULL Все значения, составляющие внешний ключ, при удалении соответствующей строки родительской таблицы устанавливаются в NULL. Для выполнения этого ограничения внешние ключевые столбцы должны допускать значения NULL.

SET DEFAULT Все значения, составляющие внешний ключ, при удалении соответствующей строки родительской таблицы устанавливаются в значение по умолчанию. Для выполнения этого ограничения все внешние ключевые столбцы должны иметь определения по умолчанию. Если столбец допускает значения NULL и значение по умолчанию явно не определено, значением столбца по умолчанию становится NULL.

ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }

Указывает, какое действие совершается над строками в изменяемой таблице, когда эти строки имеют ссылочную связь и строка родительской таблицы, на которую указывает ссылка, обновляется. Параметр по умолчанию — NO ACTION.

NO ACTION Компонент Компонент Database Engine возвращает ошибку, а обновление строки родительской таблицы откатывается.

CASCADE Соответствующие строки обновляются в ссылающейся таблице, если эта строка обновляется в родительской таблице.

SET NULL Всем значениям, составляющим внешний ключ, присваивается значение NULL, когда обновляется соответствующая строка в родительской таблице. Для выполнения этого ограничения внешние ключевые столбцы должны допускать значения NULL.

SET DEFAULT Всем значениям, составляющим внешний ключ, присваивается их значение по умолчанию, когда обновляется соответствующая строка в родительской таблице. Для выполнения этого ограничения все внешние ключевые столбцы должны иметь определения по умолчанию. Если столбец допускает значения NULL и значение по умолчанию явно не определено, значением столбца по умолчанию становится NULL.

2. Практическая часть. Атрибуты и ограничения столбцов и таблиц)

2.1. Создать БД TestBD1

2.2. В базе данных TestBD1 создадим таблицу Klient

Первые два столбца представляют идентификатор клиента и его возраст и имеют тип *INT*, то есть будут хранить числовые значения. Следующие два столбца представляют имя и фамилию клиента и имеют тип *NVARCHAR(20)*, то есть представляют строку *UNICODE* длиной не более 20 символов. Последние два столбца *Email* и *Phone* представляют адрес электронной почты и телефон клиента и имеют тип *VARCHAR(30/20)* - они также хранят строку, но не в кодировке *UNICODE*.

```
1 Create table Klient
2 (
3   Id int,
4   Age int,
5   NName NVARCHAR(20),
6   Fio NVARCHAR(20),
7   Email VARCHAR(30),
8   Phone VARCHAR(20)
9 )
```

2.3 PRIMARY KEY

При создании таблицы желательно, чтобы она имела уникальный столбец или же совокупность столбцов, которая уникальна для каждой ее строки – по данному уникальному значению можно однозначно идентифицировать запись. Такое значение называется первичным ключом таблицы. Первичный ключ уникально идентифицирует строку в таблице. В качестве первичного ключа необязательно должны выступать столбцы с типом *int*, они могут представлять любой другой тип.

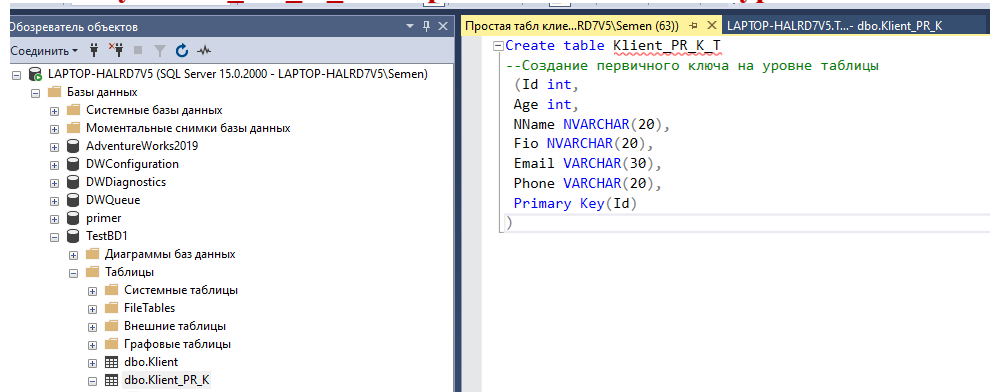
Ограничение первичного ключа (PRIMARY KEY) обеспечивает уникальность строк и запрещает хранить значения NULL в соответствующих столбцах. Каждый уникальный набор значений в атрибутах, составляющих ограничение, может появиться в таблице только один раз — не более чем в одной строке. СУРБД пресекает любые попытки определить ограничения первичного ключа для столбцов, которые способны содержать пустые значения.

С помощью выражения **PRIMARY KEY** столбец можно сделать первичным ключом:

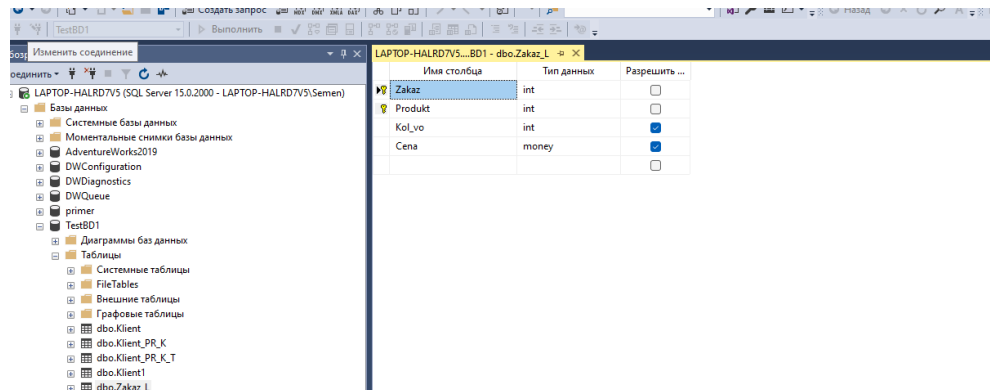
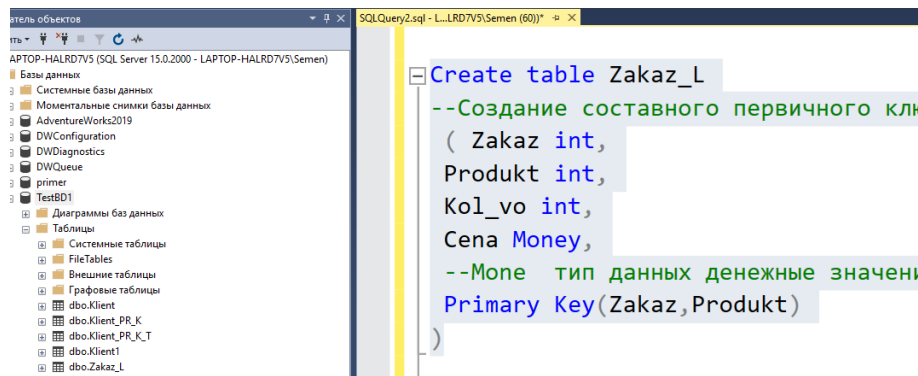
Создать таблицу Klient_PR_K1 с первичным ключом на уровне столбца:

```
use TestBD1
Create table Klient_PR_K1
(
  Id int Primary Key,
  Age int,
  NName NVARCHAR(20),
  Fio NVARCHAR(20),
  Email VARCHAR(30),
  Phone VARCHAR(20)
)
```

Создать таблицу Klient_PR_K_T с первичным ключом на уровне таблицы:



Создать таблицу Zakaz_L с составным первичным ключом (если сразу два столбца должны уникально идентифицировать строку в таблице)



2.4 IDENTITY

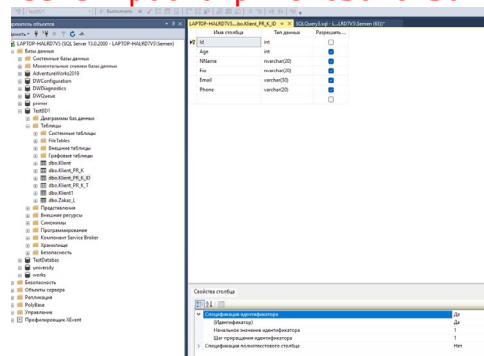
Атрибут **IDENTITY** позволяет сделать столбец идентификатором. Этот атрибут может назначаться для столбцов **числовых типов INT, SMALLINT, BIGINT, TINYINT, DECIMAL и NUMERIC**. При добавлении новых данных в таблицу SQL Server будет инкрементировать на единицу значение этого столбца у последней записи. Как правило, в роли идентификатора выступает тот же столбец, который является первичным ключом.

Также можно использовать полную форму атрибута: **IDENTITY(seed, increment)**. Здесь параметр **seed** указывает на начальное значение, с которого будет начинаться отсчет. А параметр **increment** определяет, насколько будет увеличиваться следующее значение. По умолчанию атрибут использует следующие значения: **IDENTITY(1, 1)**. То есть отсчет начинается с 1.

Создать таблицу Klient_PR_K_ID

```
Create table Klient PR K ID
(
  Id int Primary Key Identity,
  Age int,
  NName NVARCHAR(20),
  Fio NVARCHAR(20),
  Email VARCHAR(30),
  Phone VARCHAR(20)
)
```

Через SSMS просмотрите свойство столбца Id



Также можно использовать полную форму атрибута: `IDENTITY(seed, increment)`. Здесь параметр `seed` указывает на начальное значение, с которого будет начинаться отсчет. А параметр `increment` определяет, насколько будет увеличиваться следующее значение. По умолчанию атрибут использует следующие значения: `IDENTITY(1, 1)`. То есть отсчет начинается с 1. А последующие значения увеличиваются на единицу. Но мы можем это поведение переопределить.

Например:

`Id INT IDENTITY (2, 3)`

Также следует учитывать, что в таблице только один столбец должен иметь такой атрибут. Автоматически увеличиваемое поле имеет следующие ограничения:

В таблице может быть только одна Identity колонка;

Она должна использоваться с целочисленными типами данных, дробные типы запрещены;

Значение в этом поле не может обновляться.

Не разрешаются нулевые значения. Колонка всегда должна быть заполнена, и при этом, значение заносится только сервером, вы не можете на него повлиять.

2.5 UNIQUE

Ограничение `UNIQUE` обеспечивает уникальность строк и реализует такую концепцию реляционной модели, **как резервные ключи**. В отличие от первичных ключей в одной таблице возможно присутствие нескольких ограничений `UNIQUE`.

Если мы хотим, чтобы столбец имел только уникальные значения, то для него можно определить атрибут `UNIQUE`.

Создать таблицу `Klient_PR_K_ID_UN` с уникальными ключами на уровне столбцов

```
Create table Klient_PR_K_ID_UN
(
  Id int Primary Key Identity,
  Age int,
  NName NVARCHAR(20),
  Fio NVARCHAR(20),
  Email VARCHAR(30) UNIQUE,
  Phone VARCHAR(20) UNIQUE
)
```

В данном случае столбцы, которые представляют электронный адрес и телефон, будут иметь уникальные значения. И мы не сможем добавить в таблицу две строки, у которых значения для этих столбцов будут совпадать.

Также мы можем определить этот атрибут на уровне таблицы:

Создать таблицу `Klient_PR_K_ID_UN1` с уникальными ключами на уровне таблицы

```
Create table Klient_PR_K_ID_UN1
-- Уникальные значения
(
  Id int Primary Key Identity,
  Age int,
  NName NVARCHAR(20),
  Fio NVARCHAR(20),
  Email VARCHAR(30),
  Phone VARCHAR(20),
  UNIQUE (Email, Phone)
)
```

2.6 NULL и NOT NULL

Чтобы указать, может ли столбец принимать значение `NULL`, при определении столбца ему можно задать атрибут `NULL` или `NOT NULL`. Если этот атрибут явным образом не будет использован, то по умолчанию столбец будет допускать значение `NULL`. Исключением является тот

случай, когда столбец выступает в роли первичного ключа - в этом случае по умолчанию столбец имеет значение NOT NULL.

2.7 DEFAULT

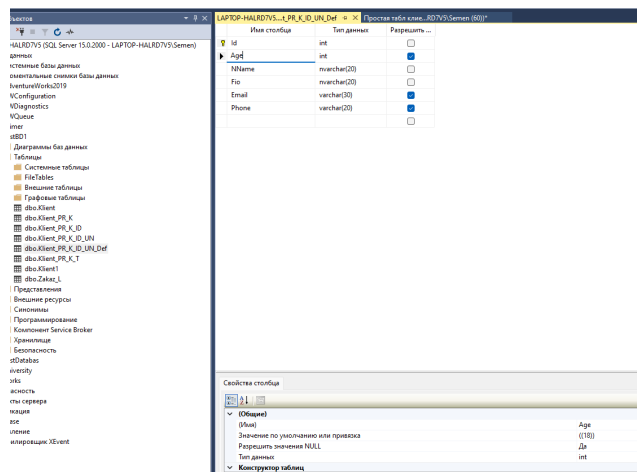
Атрибут **DEFAULT** определяет значение по умолчанию для столбца. Если при добавлении данных для столбца не будет предусмотрено значение, то для него будет использоваться значение по умолчанию.

Ограничение **DEFAULT** связывается с определенным атрибутом. Выражение используется в качестве значения по умолчанию для пустых атрибутов строки, добавляемой в таблицу.

Создать следующую табл. Klient_PR_K_ID_UN_Def

```
Create table Klient PR K ID UN Def
--Уникальные значения
(Id int Primary Key Identity,
Age int Default 18,
NName NVARCHAR(20) Not Null,
Fio NVARCHAR(20) Not Null,
Email VARCHAR(30) UNIQUE,
Phone VARCHAR(20) UNIQUE)

```



2.7 CHECK

Ключевое слово **CHECK** задает ограничение для диапазона значений, которые могут храниться в столбце. Для этого после слова **CHECK** указывается в скобках условие, которому должен соответствовать столбец или несколько столбцов. Например, возраст клиентов не может быть меньше 0 или больше 100:

Здесь также указывается, что столбцы Email и Phone не могут иметь пустую строку в качестве значения (пустая строка **не** эквивалентна значению NULL).

Для соединения условий используется ключевое слово **AND**. Условия можно задать в виде операций сравнения больше (>), меньше (<), не равно (!=).

Ограничение CHECK позволяет определить предикат о том, что перед изменением или добавлением в таблицу строка должна пройти проверку на соответствие.

Создать следующую таблицу Klient_PR_K_ID_UN_Def_CH, где ограничения на уровне таблицы

```

Create table Klient PR K ID UN Def CH
--Уникальные значения,
--значения по умолчанию, ограничения значений
(Id int Primary Key Identity,
Age int Default 18 Check (Age>0 And Age<100),
NName NVARCHAR(20) Not Null,
Fio NVARCHAR(20) Not Null,
Email VARCHAR(30) UNIQUE Check (Email!=''),
Phone VARCHAR(20) UNIQUE Check (Phone!=''))
)

Create table Klient PR K ID UN Def CH
--Уникальные значения,
--значения по умолчанию, ограничения значений
(Id int Primary Key Identity,
Age int Default 18,
NName NVARCHAR(20) Not Null,
Fio NVARCHAR(20) Not Null,
Email VARCHAR(30) UNIQUE,
Phone VARCHAR(20) UNIQUE,
Check (Age>0 And Age<100) AND (Email!='') AND (Phone!=''))
)

```

Применение ограничений CHECK

В следующем примере показано ограничение, применяемое к значениям, вводимым в столбец CreditRating таблицы Vendor. Ограничение не имеет имени.

CHECK (CreditRating >= 1 and CreditRating <= 5)

В этом примере показано именованное ограничение вводимых в столбец таблицы символьных данных по шаблону.

```

CONSTRAINT CK_emp_id CHECK (
    emp_id LIKE '[A-Z][A-Z][A-Z][1-9][0-9][0-9][0-9][FM]'
    OR emp_id LIKE '[A-Z]-[A-Z][1-9][0-9][0-9][0-9][0-9][FM]'
)

```

В этом примере указывается, что значения должны входить в заданный список или соответствовать заданному шаблону.

```

CHECK (
    emp_id IN ('1389', '0736', '0877', '1622', '1756')
    OR emp_id LIKE '99[0-9][0-9]'
)

```

2.8 Оператор CONSTRAINT. Установка имени ограничений.

С помощью ключевого слова **CONSTRAINT** можно задать имя для ограничений. В качестве ограничений могут использоваться **PRIMARY KEY, UNIQUE, DEFAULT, CHECK**.

Имена ограничений можно задать на уровне столбцов. Они указываются после **CONSTRAINT** перед атрибутами:

Ограничения могут носить произвольные названия, но, как правило, для применяются следующие префиксы:

- "PK_" - для PRIMARY KEY
- "FK_" - для FOREIGN KEY
- "CK_" - для CHECK
- "UQ_" - для UNIQUE
- "DF_" - для DEFAULT

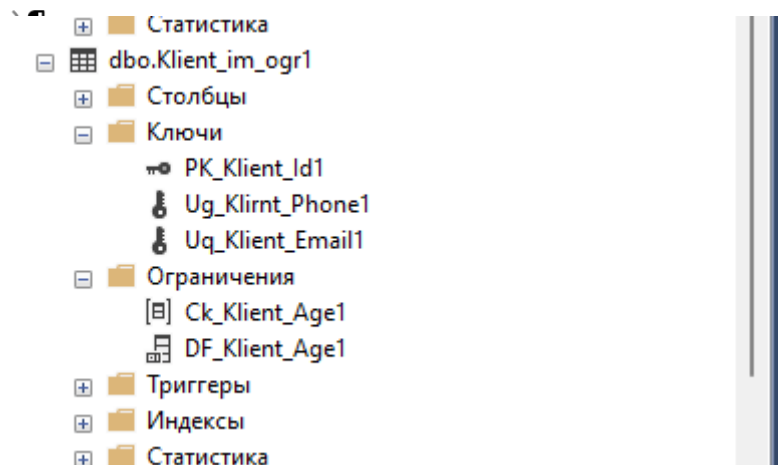
Обычно наименования ограничения для столбца составляется как
Соответствующий Префикс_имя Таблицы_имя Столбца

Создать следующие таблицы Klient_im_ogr и Klient_im_ogr1


```

Create table Klient_im_ogr
--Установка имени ограничений
.(Id int Constraint PK_Klient_Id Primary Key Identity,
Age int
Constraint DF_Klient_Age Default 18
Constraint Ck_Klient_Age Check (Age>0 And Age<100),
NName NVARCHAR(20) Not Null,
Fio NVARCHAR(20) Not Null,
Email VARCHAR(30) Constraint Uq_Klient_Email UNIQUE,
Phone VARCHAR(20) Constraint Ug_Klirnt_Phone UNIQUE
)
||
Create table Klient_im_ogr1
--Установка имени ограничений
.(Id int Identity,
Age int Constraint DF_Klient_Age1 Default 18,
NName NVARCHAR(20) Not Null,
Fio NVARCHAR(20) Not Null,
Email VARCHAR(30),
Phone VARCHAR(20),
Constraint Ck_Klient_Age1 Check (Age>0 And Age<100),
Constraint PK_Klient_Id1 Primary Key (Id),
Constraint Uq_Klient_Email1 UNIQUE (Email),
Constraint Ug_Klirnt_Phone1 UNIQUE (Phone))

```



В принципе необязательно задавать имена ограничений, при установке соответствующих атрибутов SQL Server автоматически определяет их имена. Но, зная имя ограничения, мы можем к нему обращаться, например, для его удаления.

2.9 Внешние ключи

Внешние ключи применяются для установки связи между таблицами. Внешний ключ устанавливается для столбцов из зависимой, подчиненной таблицы, и указывает на один из столбцов из главной таблицы. Хотя, как правило, внешний ключ указывает на первичный ключ из связанной главной таблицы, но это необязательно должно быть непременным условием. Внешний ключ также может указывать на какой-то другой столбец, который имеет уникальное значение.

Общий синтаксис установки внешнего ключа на уровне столбца:

```

[FOREIGN KEY] REFERENCES главная_таблица (столбец_главной_таблицы)
[ON DELETE {CASCADE|NO ACTION}]
[ON UPDATE {CASCADE|NO ACTION}]

```

Для создания ограничения внешнего ключа на уровне столбца после ключевого слова **REFERENCES** указывается имя связанной таблицы и в круглых скобках имя связанного столбца, на который будет указывать внешний ключ. Также обычно добавляются ключевые слова FOREIGN KEY, но в принципе их необязательно указывать. После выражения REFERENCES идет выражение ON DELETE и ON UPDATE.

Общий синтаксис установки внешнего ключа на уровне таблицы:

FOREIGN KEY (столбец1, столбец2, ... столбецN) REFERENCES главная_таблица (столбец_главной_таблицы1, столбец_главной_таблицы2, ... столбец_главной_таблицыN)
[ON DELETE {CASCADE|NO ACTION}]
[ON UPDATE {CASCADE|NO ACTION}]

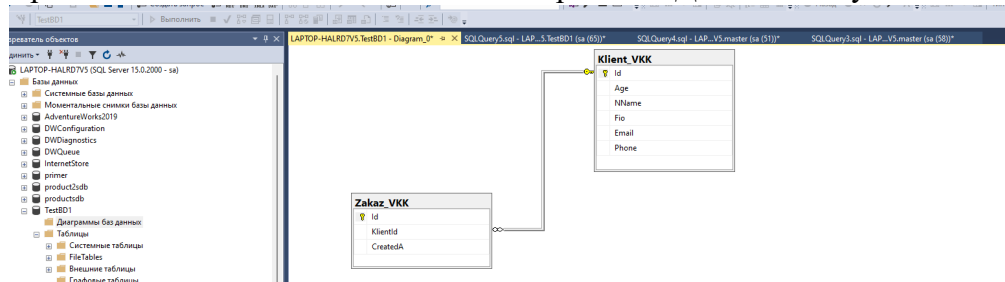
Задание : определите две таблицы и свяжите их посредством внешнего ключа:

```

--
Create table Klient_VKK
-- Уникальные значения
-- значения по умолчанию
-- таблица для демонстрации внешнего ключа
(Id int Primary Key Identity,
Age int Default 18,
NName NVARCHAR(20) Not Null,
Fio NVARCHAR(20) Not Null,
Email VARCHAR(30) UNIQUE,
Phone VARCHAR(20) UNIQUE
)
Create table Zakaz_VKK
-- таблица для демонстрации внешнего ключа на уровне столбца
(Id int Primary Key Identity,
KlientId int REFERENCES Klient_VKK(ID),
CreatedA Date
)

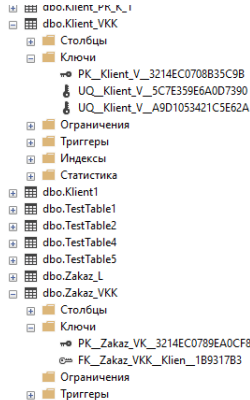
```

Средствами SSMS можно создать диаграммы БД соответствующих таблиц



Здесь определены таблицы *Klient_VKK* и *Zakaz_VKK*. *Klient_VKK* является *главной* и представляет клиента. *Zakaz_VKK* является *зависимой* и представляет заказ, сделанный клиентом. Эта таблица через столбец *CustomerId* связана с таблицей *Customers* и ее столбцом *Id*. То есть столбец *KlientId* является внешним ключом, который указывает на столбец *Id* из таблицы *Klient_VKK*.

Здесь явно имена ограничений по первичному и внешнему ключу не задавали.



!!Определение внешнего ключа на уровне таблицы выглядело бы следующим образом:

```
||
Create table Zakaz_VKK2
-- таблица для определения внешнего ключа на уровне таблицы
(Id int Primary Key Identity,
KlientId int,
CreatedA Date,
Foreign Key (KlientId) REFERENCES Klient_VKK(ID)
.)
```

С помощью оператора **CONSTRAINT** можно задать имя для ограничения внешнего ключа. Обычно это имя начинается с префикса "FK_":

```
||
Create table Zakaz_VKK2
-- таблица для определения внешнего ключа на уровне таблицы
(Id int Primary Key Identity,
KlientId int,
CreatedA Date,
CONSTRAINT FK_Zakaz_Klient Foreign Key (KlientId) REFERENCES Klient_VKK(ID)
.)
```

ON DELETE и ON UPDATE

С помощью выражений **ON DELETE** и **ON UPDATE** можно установить действия, которые выполняться соответственно при удалении и изменении связанной строки из главной таблицы. И для определения действия мы можем использовать следующие опции:

- **CASCADE**: автоматически удаляет или изменяет строки из зависимой таблицы при удалении или изменении связанных строк в главной таблице.
- **NO ACTION**: предотвращает какие-либо действия в зависимой таблице при удалении или изменении связанных строк в главной таблице. То есть фактически какие-либо действия отсутствуют.
- **SET NULL**: при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение **NULL**.
- **SET DEFAULT**: при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение по умолчанию, которое задается с помощью атрибуты **DEFAULT**. Если для столбца не задано значение по умолчанию, то в качестве него применяется значение **NULL**.

Каскадное удаление

По умолчанию, если на строку из главной таблицы по внешнему ключу ссылается какая-либо строка из зависимой таблицы, то мы не сможем удалить эту строку из главной таблицы. Вначале нам необходимо будет удалить все связанные строки из зависимой таблицы.

И если при удалении строки из главной таблицы необходимо, чтобы были удалены все связанные строки из зависимой таблицы, то применяется каскадное удаление, то есть опция **CASCADE**:

```
Create table Zakaz_VKK2
-- таблица для определения внешнего ключа на уровне таблицы+каскадное удаление
(Id int Primary Key Identity,
KlientId int,
CreatedA Date,
Foreign Key (KlientId) REFERENCES Klient_VKK(ID) ON DELETE CASCADE
```

Аналогично работает выражение **ON UPDATE CASCADE**. При изменении значения первичного ключа автоматически изменится значение связанного с ним внешнего ключа. Но так как первичные ключи, как правило, изменяются очень редко, да и в принципе не рекомендуется использовать в качестве первичных ключей столбцы с изменяемыми значениями, **то на практике выражение ON UPDATE используется редко.**

Установка NULL

При установке для внешнего ключа опции **SET NULL** необходимо, чтобы столбец внешнего ключа допускал значение NULL:

```
Create table Zakaz_VKK2
-- таблица для определения внешнего ключа на уровне таблицы+каскадное удаление
(Id int Primary Key Identity,
 KlientId int,
 CreatedA Date,
 Foreign Key (KlientId) REFERENCES Klient_VKK(ID) ON DELETE SET NULL
)
```

Установка значения по умолчанию

```
Create table Zakaz_VKK2
-- таблица для определения внешнего ключа на уровне таблицы+каскадное удаление
(Id int Primary Key Identity,
 KlientId int,
 CreatedA Date,
 Foreign Key (KlientId) REFERENCES Klient_VKK(ID) ON DELETE SET DEFAULT
)
```

!!Резюме: Внешний ключ отвечает за ссылочную целостность. Это ограничение определяется для одного или нескольких атрибутов в так называемой ссылающейся таблице и указывает на атрибуты потенциального ключа (ограничения PRIMARY KEY или UNIQUE) в таблице, на которую ссылаются (ее также называют ссылочной). Стоит заметить, что это может быть одна и та же таблица. Внешний ключ нужен для того, чтобы его атрибуты принимали только те значения, которые существуют в ссылочных столбцах.

3. Пакеты. Команда GO

Совместить в одном скрипте несколько команд, в этом случае отдельные наборы команд называются пакетами (batch).

Каждый пакет состоит из одного или нескольких SQL-выражений, которые выполняются как одно целое. В качестве сигнала завершения пакета и выполнения его выражений служит команда GO.

Смысл разделения SQL-выражений на пакеты состоит в том, что одни выражения должны успешно выполниться до запуска других выражений. Например, при добавлении таблиц мы должны бы уверены, что была создана база данных, в которой мы собираемся создать таблицы.

Выполнить: Например, определим следующий скрипт:

