

Лекция 10

Модификация данных. Добавление, изменение, удаление данных. Дополнительные инструкции и предложения для модификации таблиц T-SQL (TRUNCATE TABLE; MERGE; OUTPUT)

10.1 Добавление данных

Повтор Лабораторная 5.3

INSERT [INTO] имя_таблицы [(список_столбцов)] VALUES (значение1, значение2, ... значениеN)

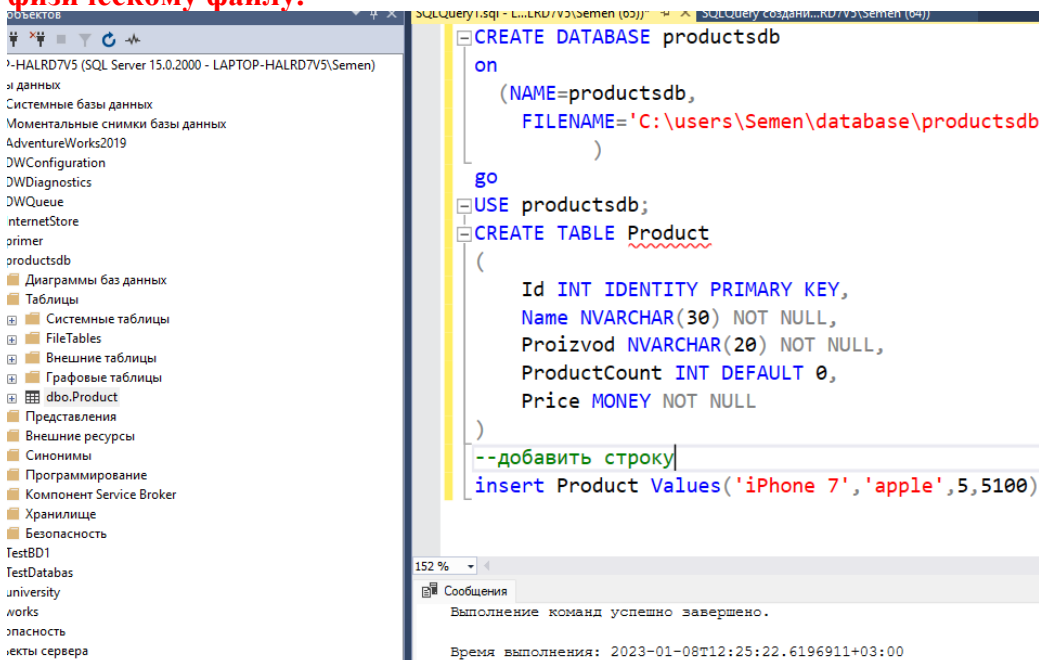
Вначале идет выражение **INSERT INTO**, затем в скобках можно указать список столбцов через запятую, в которые надо добавлять данные, и в конце после слова **VALUES** скобках перечисляют добавляемые для столбцов значения.

INTO Необязательное ключевое слово, которое можно использовать между ключевым словом **INSERT** и целевой таблицей.

При использовании данной формы оператора **INSERT** список **VALUES** должен содержать количество значений, равное количеству полей таблицы. Причем тип данных каждого из значений, указываемых в списке **VALUES**, должен совпадать с типом данных поля, соответствующего этому значению.

Значения, относящиеся к символьным типам и датам, должны быть заключены в апострофы. В списке значений может также использоваться значение **NULL**.

Задание 1. Создадим следующую базу данных, указав свой путь к физическому файлу.



Стоит учитывать, что значения для столбцов **в скобках после ключевого слова VALUES передаются по порядку их объявления.**

Например, в выражении **CREATE TABLE** выше можно увидеть, что первым столбцом идет **Id**. Но так как для него задан атрибут **IDENTITY**, то значение этого столбца автоматически генерируется, и его можно не указывать.

Второй столбец представляет **Name**, поэтому первое значение - строка "iPhone 7" будет передано именно этому столбцу. Второе значение - строка "Apple" будет передана третьему столбцу **Proizvod** и так далее.

Задание 2. При вводе значений можно указать непосредственные столбцы, в которые будут добавляться значения:

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the server hierarchy, including the 'product2sdb' database. The right pane shows the SQL script for creating the database and table, and inserting data.

```
CREATE DATABASE product2sdb
ON
(
    (NAME=product2sdb,
     FILENAME='C:\users\Semen\database\product2sdb.mdf'
    )
)
GO
USE product2sdb;
CREATE TABLE Product
(
    Id INT IDENTITY PRIMARY KEY,
    PName NVARCHAR(30) NOT NULL,
    Proizvod NVARCHAR(20) NOT NULL,
    ProductCount INT DEFAULT 0,
    Price MONEY NOT NULL
)
--добавить строку
insert Product Values('iPhone 7','apple',5,5100)
--добавить строку, указав непосредственные столбцы, в которые будут добавляться значения
insert Product (PName,Price,Proizvod)
Values('iPhone 6',6100,'apple')
```

The bottom pane shows the results of the query, displaying a table with 4 rows and 5 columns: Id, PName, Proizvod, ProductCount, and Price.

Id	PName	Proizvod	ProductCount	Price
1	iPhone 7	apple	5	5100.00
2	iPhone 7	apple	5	5100.00
3	iPhone 7	apple	5	5100.00
4	iPhone 6	apple	0	6100.00

Здесь значение указывается только для трех столбцов. Причем теперь значения передаются в порядке следования столбцов:

Список полей в операторе INSERT может иметь произвольный порядок, не зависящий от порядка, по которому задаются поля при создании таблицы.

Однако список значений должен соответствовать порядку, в котором указаны поля, связанные с этими значениями.

При выполнении данного оператора во все остальные поля будет занесено значение NULL. Естественно, что поля, которые не указываются в круглых скобках после имени таблицы, не должны иметь ограничения **NOT NULL**, иначе попытка выполнения оператора INSERT окажется неудачной.

Задание 3. Добавить сразу несколько строк:

--добавление еще трех столбцов

```
INSERT INTO Product
```

```
VALUES
```

```
('iPhone 6', 'Apple', 3, 36000),
```

```
('Galaxy S8', 'Samsung', 2, 46000),
```

```
('Galaxy S8 Plus', 'Samsung', 1, 56000)
```

Получим в итоге:

Внешние таблицы
Графовые таблицы
dbo.Product
Представления
Внешние ресурсы
Синонимы
Программирование
Компонент Service Broker
Хранилище
Безопасность
productsdb
TestBD1
TestDatabas
university
works
Безопасность
Объекты сервера
Репликация
PolyBase
Управление
Профилировщик XEvent

Id	PName	Proizvod	ProductCount	Price
1	iPhone 7	apple	5	5100,00
2	iPhone 7	apple	5	5100,00
3	iPhone 7	apple	5	5100,00
4	iPhone 6	apple	0	6100,00
5	iPhone 6	Apple	3	36000,00
6	Galaxy S8	Samsung	2	46000,00
7	Galaxy S8 Plus	Samsung	1	56000,00

В данном случае в таблицу будут добавлены три строки.

Задание 4. При добавлении мы можем указать, чтобы для столбца использовалось значение по умолчанию с помощью ключевого слова DEFAULT или значение NULL:

```
insert Product (PName,Proizvod,ProductCount,Price)
```

```
Values('Mi6', 'Xiaomi', DEFAULT, 28000)
```

Id	PName	Proizvod	ProductCount	Price
1	iPhone 7	apple	5	5100,00
2	iPhone 7	apple	5	5100,00
3	iPhone 7	apple	5	5100,00
4	iPhone 6	apple	0	6100,00
5	iPhone 6	Apple	3	36000,00
6	Galaxy S8	Samsung	2	46000,00
7	Galaxy S8 Plus	Samsung	1	56000,00
8	Mi6	Xiaomi	0	28000,00

Если все столбцы имеют атрибут DEFAULT, определяющий значение по умолчанию, или допускают значение NULL, то можно для всех столбцов вставить значения по умолчанию:

```
INSERT INTO Product
```

```
DEFAULT VALUES
```

10.2 Обновление данных. Команда UPDATE

Для изменения данных в записях таблицы используется оператор UPDATE

Инструкция UPDATE используется для модифицирования строк таблицы. Эта инструкция имеет следующую общую форму:

```
UPDATE tab_name  
{SET column_1 = {expression | DEFAULT | NULL} [...n]}  
[FROM tab_name1 [...n]]  
[WHERE condition]
```

Строки таблицы *tab_name* выбираются для изменения **в соответствии с условием** в предложении WHERE.

Значения столбцов каждой модифицируемой строки **изменяются** с помощью предложения SET инструкции UPDATE, которое соответствующему столбцу присваивает выражение (обычно) или константу.

Если предложение **WHERE** отсутствует, то инструкция **UPDATE** модифицирует все строки таблицы.

Для изменения уже имеющихся строк в таблице применяется команда UPDATE. Она имеет следующий формальный синтаксис:

UPDATE имя_таблицы
SET столбец1 = значение1, столбец2 = значение2, ... столбецN = значениеN
[FROM выборка AS псевдоним_выборки]
[WHERE условие_обновления]

Например, увеличим у всех товаров цену на 5000:

```
USE product2sdb;  
SELECT * FROM Product;  
UPDATE Product  
SET Price = Price + 5000  
SELECT * FROM Product
```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'product2sdb' database is selected under the 'Диаграммы баз данных' (Database Diagrams) folder. The 'Product' table is highlighted under the 'Таблицы' (Tables) folder. The main window displays the 'Product' table data in a grid. The columns are 'Id', 'PName', 'Proizvod', 'ProductCount', and 'Price'. The data is as follows:

Id	PName	Proizvod	ProductCount	Price
1	iPhone 7	apple	5	10100.00
2	iPhone 7	apple	5	10100.00
3	iPhone 7	apple	5	10100.00
4	iPhone 6	apple	0	11100.00
5	iPhone 6	Apple	3	41000.00
6	Galaxy S8	Samsung	2	51000.00
7	Galaxy S8 Plus	Samsung	1	61000.00
8	Mi6	Xiaomi	0	33000.00

Используем критерий, и изменим название производителя с "Samsung" на "Samsung I":

```
USE product2sdb;  
SELECT * FROM Product;  
UPDATE Product  
SET Proizvod = 'Samsung I'  
WHERE Proizvod = 'Samsung'  
  
SELECT * FROM Product
```

```
USE product2sdb;  
SELECT * FROM Product;  
UPDATE Product  
SET Proizvod = 'Samsung I'  
WHERE Proizvod = 'Samsung'  
  
SELECT * FROM Product
```

52 %

Результаты Сообщения

	Id	PName	Proizvod	ProductCount	Price
1	1	iPhone 7	apple	5	15100,00
2	3	iPhone 7	apple	5	15100,00
3	5	iPhone 7	apple	5	15100,00
4	6	iPhone 6	apple	0	16100,00
5	7	iPhone 6	Apple	3	46000,00
6	8	Galaxy S8	Samsung	2	56000,00
7	9	Galaxy S8 Plus	Samsung	1	66000,00
8	11	Mi6	Xiaomi	0	38000,00

	Id	PName	Proizvod	ProductCount	Price
1	1	iPhone 7	apple	5	15100,00
2	3	iPhone 7	apple	5	15100,00
3	5	iPhone 7	apple	5	15100,00
4	6	iPhone 6	apple	0	16100,00
5	7	iPhone 6	Apple	3	46000,00
6	8	Galaxy S8	Samsung I	2	56000,00
7	9	Galaxy ...	Samsung I	1	66000,00
8	1...	Mi6	Xiaomi	0	38000,00

Более сложный запрос - заменим у поля Proizvod значение "Apple" на "Apple I." в первых 2 строках:

```
USE productssdb;  
SELECT * FROM Product;  
UPDATE Product  
SET Proizvod = 'Apple I'  
FROM  
(SELECT TOP 2 * FROM Product WHERE Proizvod='Apple') AS Selected1  
WHERE Product.id= Selected1.id  
SELECT * FROM Product
```

С помощью подзапроса после ключевого слова FROM производится выборка первых двух строк, в которых Proizvod ='Apple'. Для этой выборки будет определен псевдоним Selected. Псевдоним указывается после оператора AS.

Далее идет условие обновления Product.Id = Selected1.Id. То есть фактически мы имеем дело с двумя таблицами - Product и Selected1(которая является производной от Product). В Selected1 находится две первых строки, в которых Manufacturer='Apple'. В Product - вообще все строки. И обновление производится только для тех строк, которые есть в выборке Selected1. То есть если в таблице Product десятки товаров с производителем Apple, то обновление коснется только двух первых из них.

```

USE productssab;
SELECT * FROM Product;
UPDATE Product
SET Proizvod = 'Apple I'
FROM
(SELECT TOP 2 * FROM Product WHERE Proizvod='Apple') AS Sel1
WHERE Product.id= Selected1.id
SELECT * FROM Product

```

152 %

Результаты Сообщения

	Id	PName	Proizvod	ProductCount	Price
1	1	iPhone 7	apple	5	15100,00
2	3	iPhone 7	apple	5	15100,00
3	5	iPhone 7	apple	5	15100,00
4	6	iPhone 6	apple	0	16100,00
5	7	iPhone 6	Apple	3	46000,00
6	8	Galaxy S8	Samsung I	2	56000,00
7	9	Galaxy S8 Plus	Samsung I	1	66000,00
8	11	Mi6	Xiaomi	0	38000,00

	Id	PName	Proizvod	ProductCount	Price
1	1	iPhone 7	Apple I	5	15100,00
2	3	iPhone 7	Apple I	5	15100,00
3	5	iPhone 7	apple	5	15100,00
4	6	iPhone 6	apple	0	16100,00
5	7	iPhone 6	Apple	3	46000,00
6	8	Galaxy S8	Samsung I	2	56000,00
7	9	Galaxy ...	Samsung I	1	66000,00
8	1...	Mi6	Xiaomi	0	38000,00

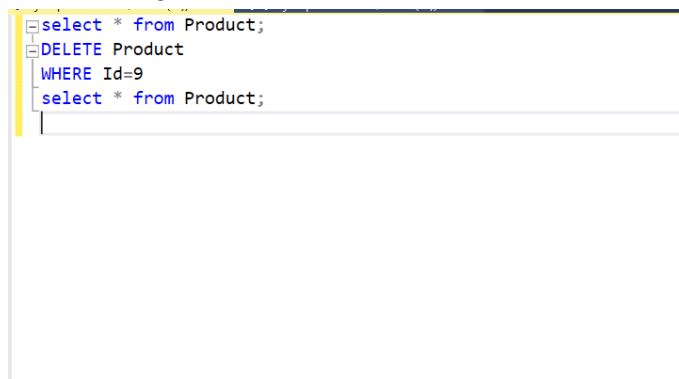
10.3 Удаление данных. Команда DELETE

Для удаления применяется команда **DELETE**:

DELETE [FROM] имя_таблицы WHERE условие_удаления

Например, удалим строки, у которых id равен 9:

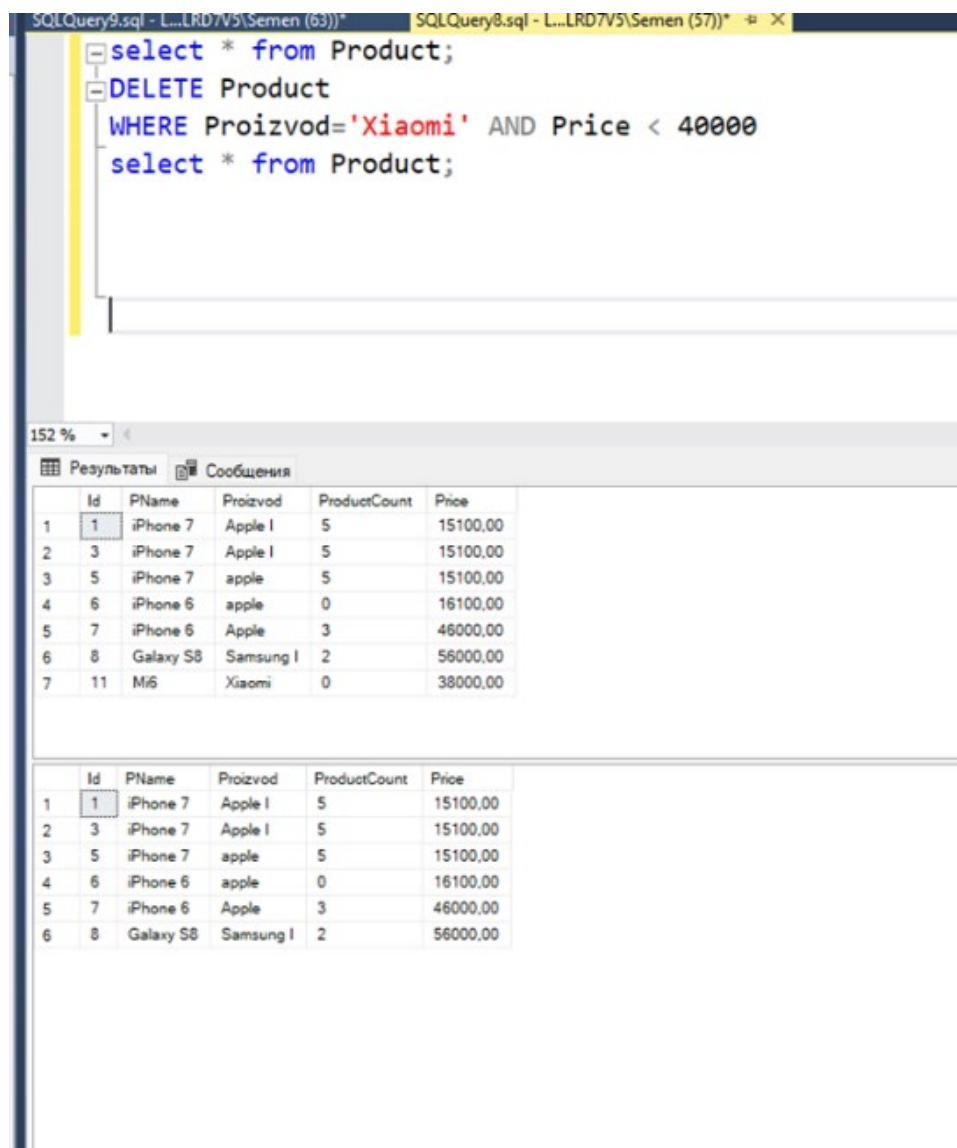
```
select * from Product;  
DELETE Product  
WHERE Id=9
```



```
select * from Product;  
DELETE Product  
WHERE Id=9  
select * from Product;
```

Id	PName	Proizvod	ProductCount	Price
1	iPhone 7	Apple I	5	15100,00
3	iPhone 7	Apple I	5	15100,00
5	iPhone 7	apple	5	15100,00
6	iPhone 6	apple	0	16100,00
7	iPhone 6	Apple	3	46000,00
8	Galaxy S8	Samsung I	2	56000,00
11	M6	Xiaomi	0	38000,00

Или удалим все товары, производителем которых является Xiaomi и которые имеют цену меньше 40000:



The screenshot shows a SQL Server Enterprise Manager window with two tabs. The active tab, 'SQLQuery9.sql - L...LRD7V5\Сemen (63))', contains the following SQL script:

```
select * from Product;  
DELETE Product  
WHERE Proizvod='Xiaomi' AND Price < 40000  
select * from Product;
```

Below the query editor, the 'Results' pane displays the data after the query execution. The zoom level is set to 152%. The results are shown in a table with 6 columns: Id, PName, Proizvod, ProductCount, and Price. The table contains 6 rows of data, with the first row (Id 1) selected.

	Id	PName	Proizvod	ProductCount	Price
1	1	iPhone 7	Apple I	5	15100,00
2	3	iPhone 7	Apple I	5	15100,00
3	5	iPhone 7	apple	5	15100,00
4	6	iPhone 6	apple	0	16100,00
5	7	iPhone 6	Apple	3	46000,00
6	8	Galaxy S8	Samsung I	2	56000,00

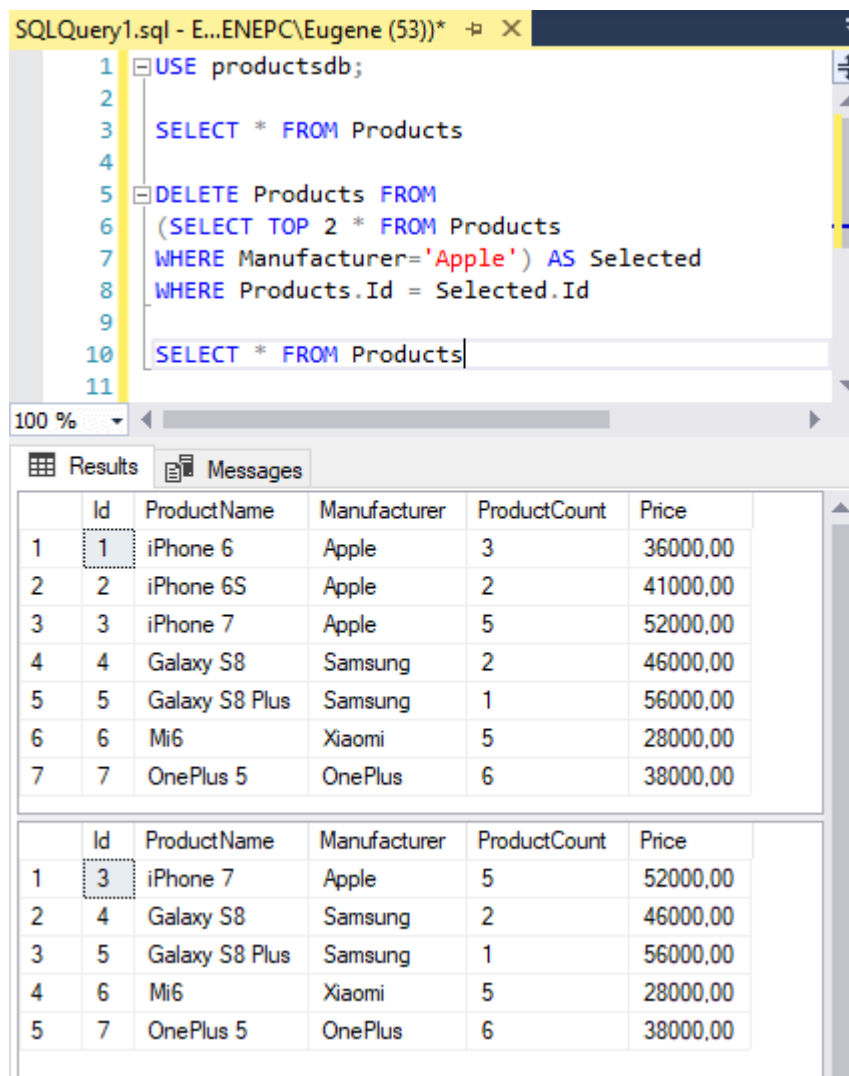
Below the first table, a second table is shown, which is identical to the first one, representing the state of the 'Product' table after the deletion of the Xiaomi product with a price less than 40000.

	Id	PName	Proizvod	ProductCount	Price
1	1	iPhone 7	Apple I	5	15100,00
2	3	iPhone 7	Apple I	5	15100,00
3	5	iPhone 7	apple	5	15100,00
4	6	iPhone 6	apple	0	16100,00
5	7	iPhone 6	Apple	3	46000,00
6	8	Galaxy S8	Samsung I	2	56000,00

Более сложный пример - удалим первые два товара, у которых производитель - Apple:

```
1 DELETE Products FROM
2 (SELECT TOP 2 * FROM Products
3 WHERE Manufacturer='Apple') AS Selected
4 WHERE Products.Id = Selected.Id
```

После первого оператора FROM идет выборка двух строк из таблицы Products. Этой выборке назначается псевдоним Selected с помощью оператора AS. Далее устанавливаем условие, что если Id в таблице Products имеет то же значение, что и Id в выборке Selected, то строка удаляется.



The screenshot shows a SQL query window with the following text:

```
1 USE productsdb;
2
3 SELECT * FROM Products
4
5 DELETE Products FROM
6 (SELECT TOP 2 * FROM Products
7 WHERE Manufacturer='Apple') AS Selected
8 WHERE Products.Id = Selected.Id
9
10 SELECT * FROM Products
11
```

Below the query window, the 'Results' tab is active, displaying two tables. The first table shows the initial state of the 'Products' table, and the second table shows the state after the deletion of the first two rows where the manufacturer is 'Apple'.

	Id	ProductName	Manufacturer	ProductCount	Price
1	1	iPhone 6	Apple	3	36000,00
2	2	iPhone 6S	Apple	2	41000,00
3	3	iPhone 7	Apple	5	52000,00
4	4	Galaxy S8	Samsung	2	46000,00
5	5	Galaxy S8 Plus	Samsung	1	56000,00
6	6	Mi6	Xiaomi	5	28000,00
7	7	OnePlus 5	OnePlus	6	38000,00

	Id	ProductName	Manufacturer	ProductCount	Price
1	3	iPhone 7	Apple	5	52000,00
2	4	Galaxy S8	Samsung	2	46000,00
3	5	Galaxy S8 Plus	Samsung	1	56000,00
4	6	Mi6	Xiaomi	5	28000,00
5	7	OnePlus 5	OnePlus	6	38000,00

Если необходимо вовсе удалить все строки вне зависимости от условия, то условие можно не указывать:

1 DELETE Products

Общий вид оператора удаления DELETE

DELETE

[FROM]

**{ table_name WITH (< table_hint_limited > [...n])
| view_name
| rowset_function_limited
}**

[FROM { < table_source > } [,...n]]

[WHERE

**{ < search_condition >
| { [CURRENT OF
 { [GLOBAL] cursor_name }
 | cursor_variable_name
 }
| }
}**

]

Минимальная команда удаления выглядит следующим образом:

DELETE tbPeoples

Эта команда удаляет все строки из таблицы tbPeoples. Те же самые действия можно выполнить с помощью вызова команды:

DELETE FROM tbPeoples

Отличие этой команды в том, что мы добавили ключевое слово FROM, которое в данной команде может опускаться.

Чтобы ограничить количество удаляемых строк, используется секция WHERE. Она используется точно так же, как и в запросах SELECT или UPDATE.

Пример. Удалим все строки, в которых поле "vcFamil" содержит нулевое значение. Это делается с помощью следующего запроса:

DELETE FROM tbPeoples

WHERE vcFamil is NULL

С помощью секции WHERE мы требуем, чтобы сервер удалил только те записи, в которых поле "vcFamil" содержит нулевое значение.

Пример. Следующий запрос удаляет запись, в которой первичный ключ "idPeoples" содержит значение -22:

DELETE FROM tbPeoples

WHERE idPeoples = -22

Пример. Предыдущие два запроса можно было бы выполнить одной командой:

DELETE FROM tbPeoples

WHERE vcFamil is NULL

OR idPeoples = -22

В данном случае удаляются записи, в которых или фамилия не заполнена, или ключевое поле равно -22.

Теперь рассмотрим удаление из связанных таблиц. Допустим, что нам необходимо удалить номера телефонов определенного работника. Для этого выполняем следующий запрос:

```
DELETE pn  
FROM tbPhoneNumbers pn, inserted i  
WHERE pn.idPeoples=i.idPeoples  
AND vcFamil = Фамилия
```

Нельзя удалять сразу из двух таблиц, может быть только одна. Так как в этом примере связываются две таблицы, то после оператора DELETE необходимо явно указать, из какой из двух происходит удаление.

Но у MS SQL Server есть вариант решения этой проблемы - TRUNCATE TABLE.

Ее можно использовать при удалении из таблицы всех записей, и при этом обращения к журналу будут сведены к минимуму. Для удаления всех записей из таблицы Товары нужно использовать следующий запрос:

TRUNCATE TABLE Имя_Таблицы

В результате этого из таблицы будут удалены все записи, а в журнал будет записано только информация о том, что все удалено.

С помощью оператора TRUNCATE TABLE можно удалять только все записи из таблицы, потому что нет секции WHERE. К тому же, нельзя удалять из таблицы, которая связана с помощью внешнего ключа.

Есть еще одно очень важное отличие между операторами DELETE и TRUNCATE TABLE.

Если удалить все записи из таблицы с помощью DELETE, а затем добавить новую запись, то значение автоматически увеличиваемого поля будет увеличиваться дальше, **т.е. счетчик не будет сброшен.**

А после удаления с помощью TRUNCATE TABLE, значение счетчика сбрасывается и если после этого добавить строку, то значение автоматически увеличиваемого поля начнет свой отсчет с единицы. Учитывайте этот эффект при использовании определенного оператора удаления данных.