# Лекция 7.

# Соединение нескольких таблиц в запросе. Операции с множествами. Соединение таблиц.

## Неявное соединение таблиц.

## Явное соединение таблиц (JOIN).

## Внутреннее соединение INNER [OUTER] JOIN.

## Внешнее левое соединение LEFT [OUTER] JOIN.

## Внешнее правое соединение RIGHT [OUTER] JOIN.

## Полное внешнее соединение FULL [OUTER] JOIN.

## Перекрестное соединение CROSS JOIN. Группировка в соединениях.

### 1. Общие сведения

SQL Server выполняет операции сортировки, пересечения, объединения и поиска различий при помощи технологий хэш-соединений и сортировки в оперативной памяти.

SQL Server реализует операции логического соединения в соответствии с синтаксисом Transact-SQL:

**Перекрестное соединение**
**внутреннее соединение,**
**левое внешнее соединение.**
**Правое внешнее соединение**
**Полное внешнее соединение**

**С помощью соединения можно получать данные из двух или нескольких таблиц на основе логических связей между ними. Соединения позволяют указать, как в SQL Server должны использоваться данные из одной таблицы для выбора строк из другой таблицы.**

Соединение определяет **способ связывания двух таблиц** в запросе следующим образом:

**для каждой таблицы указываются столбцы, используемые в соединении. В типичном условии соединения указывается внешний ключ из одной таблицы и связанный с ним ключ из другой таблицы;**

**Указывается логический оператор (например, = или <>), используемый для сравнения значений из столбцов.**

**Соединения выражаются логически с помощью синтаксиса Transact-SQL.**

**Выделяют следующие виды соединения, каждому из которых соответствует своя форма оператора JOIN:**

**CROSS JOIN — перекрестное или декартово соединение**
**[INNER] JOIN — естественное или внутреннее соединение**
**LEFT [OUTER] JOIN — левое внешнее соединение**
**RIGHT [OUTER] JOIN — правое внешнее соединение**
**FULL [OUTER] JOIN — полное внешнее соединение**

**Внутренние соединения можно задавать в предложениях FROM и WHERE.**

**Внешние соединения и перекрестные соединения можно задавать только в предложении FROM.**

**Условия соединения сочетаются с условиями поиска WHERE и HAVING для управления строками, выбранными из базовых таблиц, на которые ссылается предложение FROM.**

Табличный оператор JOIN принимает на вход две таблицы. Он может выполнять **операции трех различных видов соединения: перекрестного, внутреннего и внешнего.** Все они состоят из разных логических этапов обработки.

Перекрестное соединение включает только декартово произведение.

Внутреннее состоит из декартового произведения и фильтрации.

Внешнее соединение, в дополнение к внутреннему, добавляет внешние строки.

**Внешнее соединение**

**Внешнее соединение позволяет в отличие от внутреннего извлечь не только строки с одинаковыми значениями соединяемых столбцов, но и строки без совпадений из одной или обеих таблиц.**

**Выделяют три вида внешних соединений:**

1. **левое внешнее соединение — в результирующий набор попадают все строки из таблицы с левой стороны оператора сравнения (независимо от того имеются ли совпадающие строки с правой стороны), а из таблицы с правой стороны — только строки с совпадающими значениями столбцов. При этом если для строки из левой таблицы нет соответствий в правой таблице, значениям строки в правой таблице будут присвоены NULL**
2. **правое внешнее соединение — аналогично левому внешнему соединению, но таблицы меняются местами**
3. **полное внешнее соединение — композиция левого и правого внешнего соединения: результирующий набор состоит из всех строк обеих таблиц. Если для строки одной из таблиц нет соответствующей строки в другой таблице, всем ячейкам строки второй таблицы присваивается значение NULL**

## 7.2 Соединение таблиц. Неявное соединение таблиц

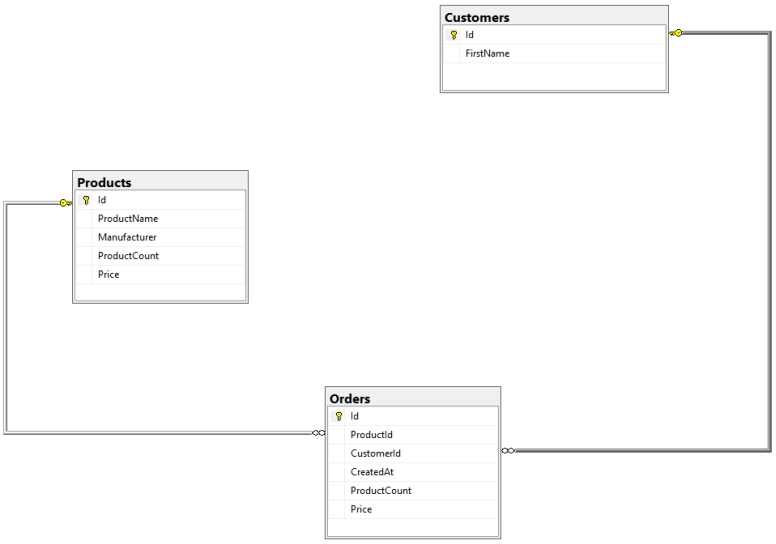
**Возможно также выполнить соединение и без оператора JOIN с помощью инструкции WHERE используя столбцы соединения, но этот синтаксис считается неявным.**

**Неявное соединение производится на основе простой выборки неявно путем сведения данных**

Для сведения данных из разных таблиц мы можем использовать стандартную команду SELECT. Допустим, у нас есть следующие таблицы, которые связаны между собой связями:

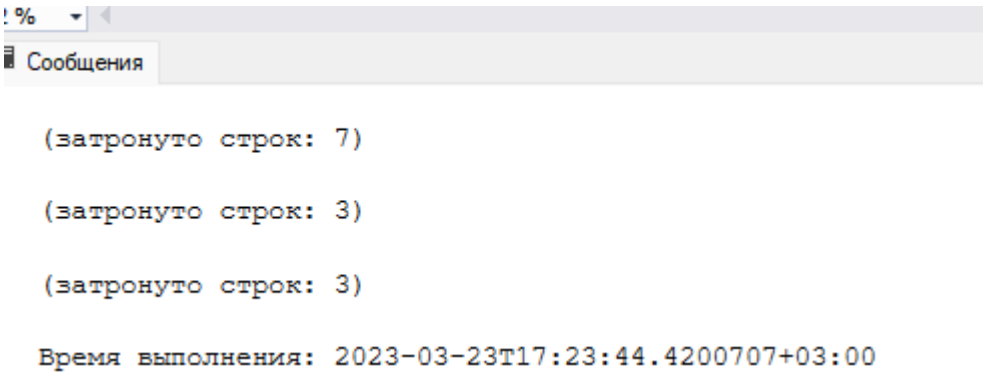
```
USE testbasa;

CREATE TABLE Products
(
    Id INT IDENTITY PRIMARY KEY,
    ProductName NVARCHAR(30) NOT NULL,
    Manufacturer NVARCHAR(20) NOT NULL,
    ProductCount INT DEFAULT 0,
    Price MONEY NOT NULL
);
CREATE TABLE Customers
(
    Id INT IDENTITY PRIMARY KEY,
    FirstName NVARCHAR(30) NOT NULL
);
CREATE TABLE Orders
(
    Id INT IDENTITY PRIMARY KEY,
--ON DELETE CASCADE используется  для автоматического удаления строк
--из дочерней таблицы при удалении строк из родительской таблицы
    ProductId INT NOT NULL REFERENCES Products(Id) ON DELETE CASCADE,
    CustomerId INT NOT NULL REFERENCES Customers(Id) ON DELETE CASCADE,
    CreatedAt DATE NOT NULL,
    ProductCount INT DEFAULT 1,
    Price MONEY NOT NULL
);
```

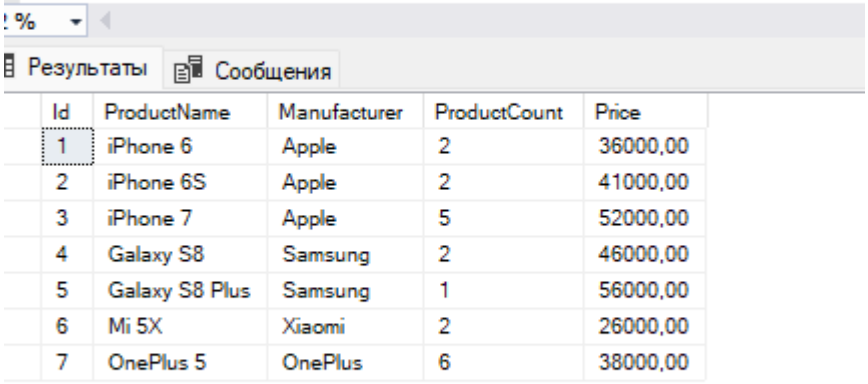
Здесь таблицы Products и Customers связаны с таблицей Orders связью один ко многим. Таблица Orders в виде внешних ключей ProductId и CustomerId содержит ссылки на столбцы Id из соответственно таблиц Products и Customers. Также она хранит количество купленного товара (ProductCount) и и по какой цене он был куплен (Price). И кроме того, таблица также хранит в виде столбца CreatedAt дату покупки.




**Введем данные:**

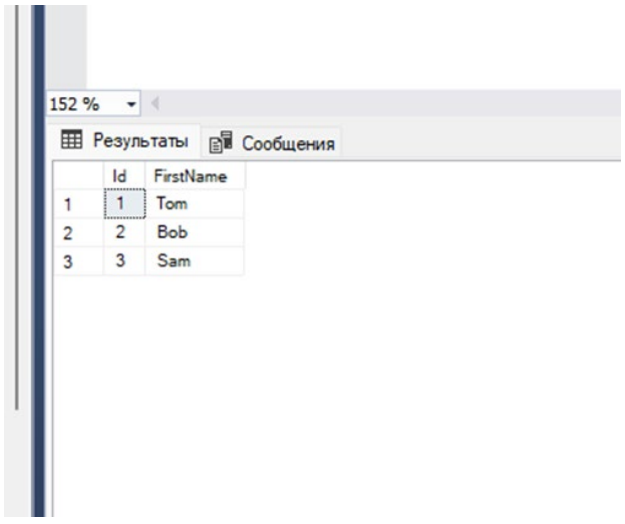
```
USE testbasa;
INSERT INTO Products
VALUES ('iPhone 6', 'Apple', 2, 36000),
('iPhone 6S', 'Apple', 2, 41000),
('iPhone 7', 'Apple', 5, 52000),
('Galaxy S8', 'Samsung', 2, 46000),
('Galaxy S8 Plus', 'Samsung', 1, 56000),
('Mi 5X', 'Xiaomi', 2, 26000),
('OnePlus 5', 'OnePlus', 6, 38000)

INSERT INTO Customers VALUES ('Tom'), ('Bob'),('Sam')

INSERT INTO Orders
VALUES
(
    (SELECT Id FROM Products WHERE ProductName='Galaxy S8'),
    (SELECT Id FROM Customers WHERE FirstName='Tom'),
    '2017-07-11',
    2,
    (SELECT Price FROM Products WHERE ProductName='Galaxy S8')
),
(
    (SELECT Id FROM Products WHERE ProductName='iPhone 6S'),
    (SELECT Id FROM Customers WHERE FirstName='Tom'),
    '2017-07-13',
    1,
    (SELECT Price FROM Products WHERE ProductName='iPhone 6S')
),
(
    (SELECT Id FROM Products WHERE ProductName='iPhone 6S'),
    (SELECT Id FROM Customers WHERE FirstName='Bob'),
    '2017-07-11',
    1,
    (SELECT Price FROM Products WHERE ProductName='iPhone 6S')
    )

 (Ctrl)
```



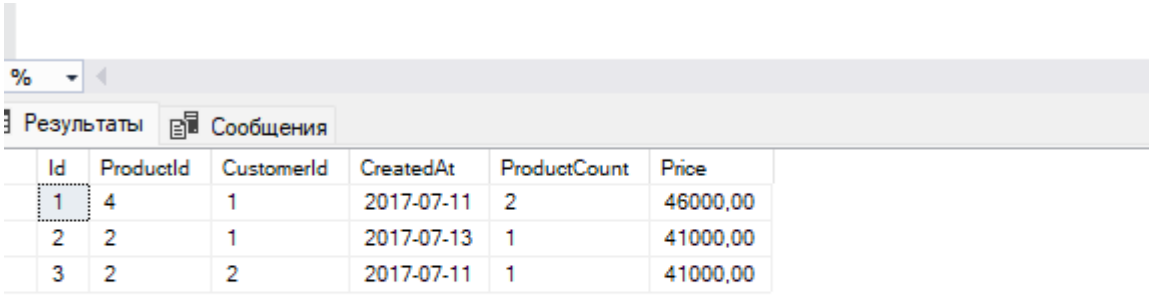

| Id | ProductName | Manufacturer | ProductCount | Price |
|---|---|---|---|---|
| 1 | iPhone 6 | Apple | 2 | 36000,00 |
| 2 | iPhone 6S | Apple | 2 | 41000,00 |
| 3 | iPhone 7 | Apple | 5 | 52000,00 |
| 4 | Galaxy S8 | Samsung | 2 | 46000,00 |
| 5 | Galaxy S8 Plus | Samsung | 1 | 56000,00 |
| 6 | Mi 5X | Xiaomi | 2 | 26000,00 |
| 7 | OnePlus 5 | OnePlus | 6 | 38000,00 |

Customers



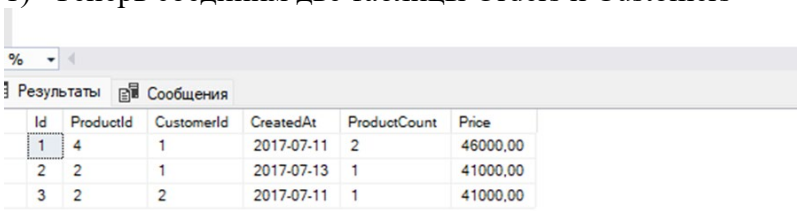

| | Id | FirstName |
|---|---|---|
| 1 | 1 | Tom |
| 2 | 2 | Bob |
| 3 | 3 | Sam |

Orders

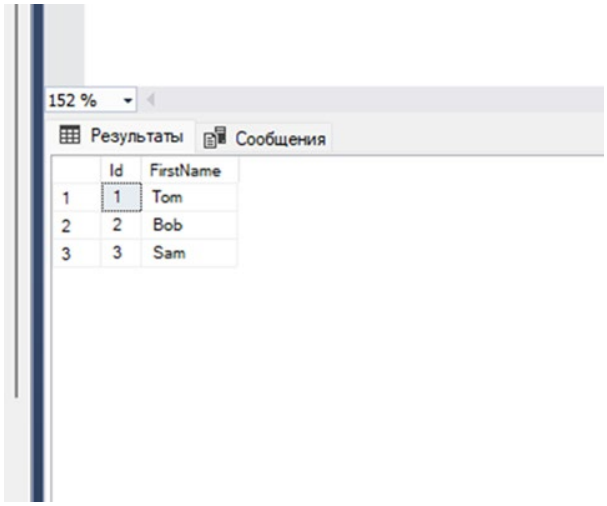



| Id | ProductId | CustomerId | CreatedAt | ProductCount | Price |
|---|---|---|---|---|---|
| 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 |
| 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 |
| 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 |

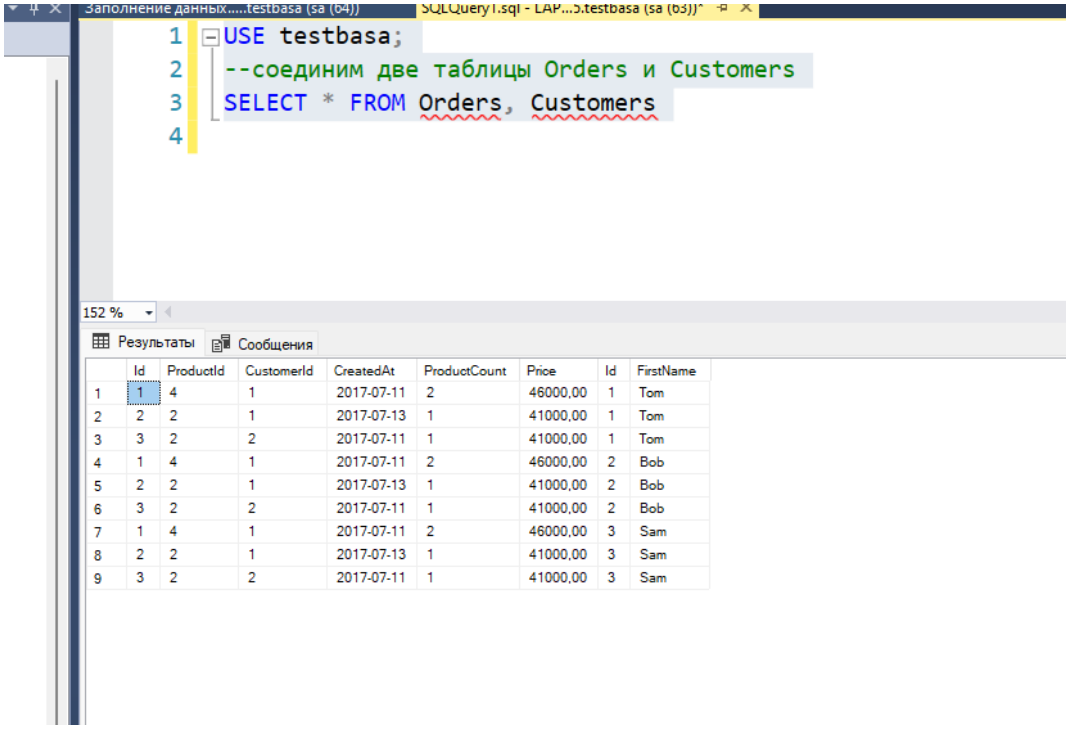
1) Теперь соединим две таблицы Orders и Customers



| Id | ProductId | CustomerId | CreatedAt | ProductCount | Price |
|---|---|---|---|---|---|
| 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 |
| 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 |
| 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 |



| | Id | FirstName |
|---|---|---|
| 1 | 1 | Tom |
| 2 | 2 | Bob |
| 3 | 3 | Sam |



```
1  USE testbasa;
2  --соединим две таблицы Orders и Customers
3  SELECT * FROM Orders, Customers
4
```

| | Id | ProductId | CustomerId | CreatedAt | ProductCount | Price | Id | FirstName |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 | 1 | Tom |
| 2 | 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 | 1 | Tom |
| 3 | 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 | 1 | Tom |
| 4 | 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 | 2 | Bob |
| 5 | 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 | 2 | Bob |
| 6 | 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 | 2 | Bob |
| 7 | 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 | 3 | Sam |
| 8 | 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 | 3 | Sam |
| 9 | 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 | 3 | Sam |

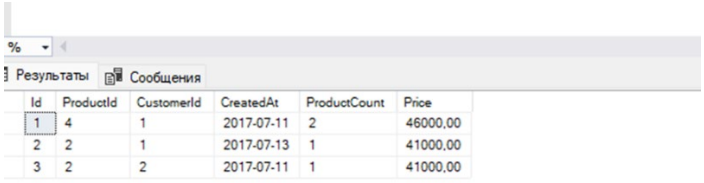
То есть в данном случае **мы получаем прямое (декартово) произведение двух групп.**

2) **Но вряд ли это тот результат, который хотелось бы видеть.** Тем более каждый заказ из Orders связан с конкретным покупателем из Customers, а не со всеми возможными покупателями.

Чтобы решить задачу, необходимо использовать **выражение WHERE и фильтровать строки при условии, что поле CustomerId из Orders соответствует полю Id из Customers:**

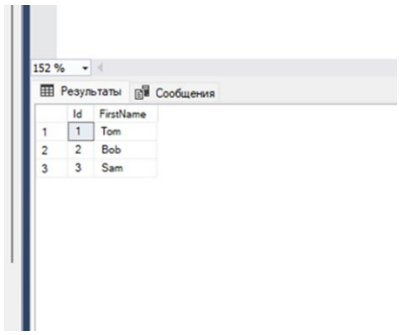
**Исходные данные:**

Orders



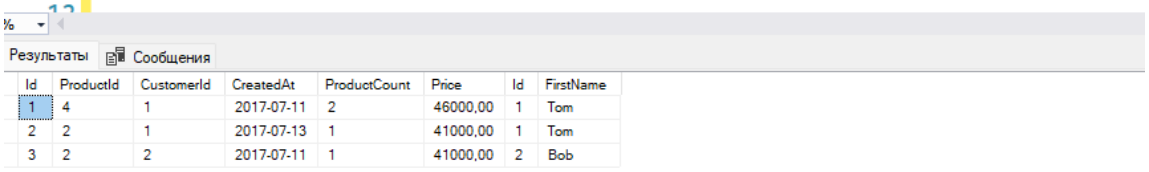
| Id | ProductId | CustomerId | CreatedAt | ProductCount | Price |
|---|---|---|---|---|---|
| 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 |
| 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 |
| 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 |

CustomerId



| Id | FirstName |
|---|---|
| 1 | Tom |
| 2 | Bob |
| 3 | Sam |

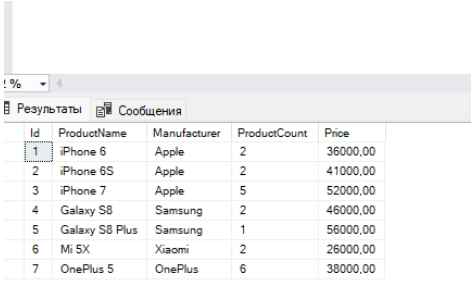
```
8  --использовать выражение WHERE и фильтровать строки при условии,
9  --что поле CustomerId из Orders соответствует полю Id из Customers:
10 SELECT * FROM Orders, Customers
11 WHERE Orders.CustomerId = Customers.Id
```



| Id | ProductId | CustomerId | CreatedAt | ProductCount | Price | Id | FirstName |
|---|---|---|---|---|---|---|---|
| 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 | 1 | Tom |
| 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 | 1 | Tom |
| 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 | 2 | Bob |

3) Теперь объединим данные **по трем таблицам Orders, Customers и Products**. То есть получим **все заказы и добавим информацию по клиенту и связанному товару**

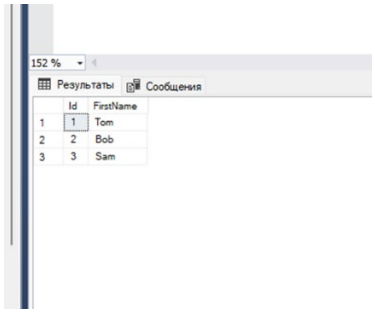
**Исходные данные:**



Результаты Сообщения

| Id | ProductName | Manufacturer | ProductCount | Price |
|---|---|---|---|---|
| 1 | iPhone 6 | Apple | 2 | 36000,00 |
| 2 | iPhone 6S | Apple | 2 | 41000,00 |
| 3 | iPhone 7 | Apple | 5 | 52000,00 |
| 4 | Galaxy S8 | Samsung | 2 | 46000,00 |
| 5 | Galaxy S8 Plus | Samsung | 1 | 56000,00 |
| 6 | Mi 5X | Xiaomi | 2 | 26000,00 |
| 7 | OnePlus 5 | OnePlus | 6 | 38000,00 |

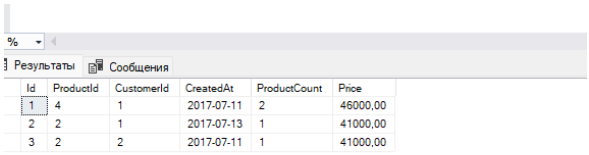
Customers



152 % Результаты Сообщения

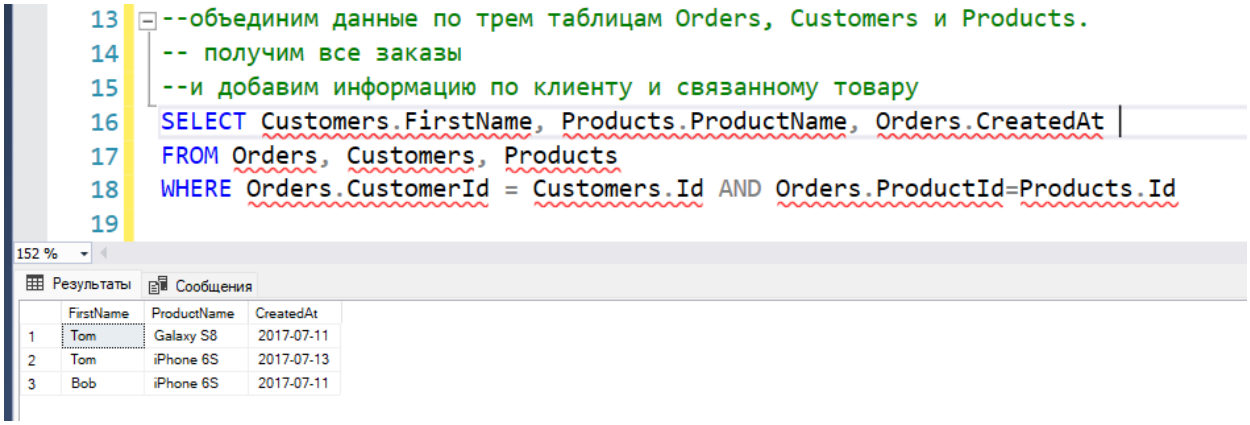
| | Id | FirstName |
|---|---|---|
| 1 | 1 | Tom |
| 2 | 2 | Bob |
| 3 | 3 | Sam |

Orders



Результаты Сообщения

| Id | ProductId | CustomerId | CreatedAt | ProductCount | Price |
|---|---|---|---|---|---|
| 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 |
| 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 |
| 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 |



```
13  --объединим данные по трем таблицам Orders, Customers и Products.
14  -- получим все заказы
15  --и добавим информацию по клиенту и связанному товару
16  SELECT Customers.FirstName, Products.ProductName, Orders.CreatedAt
17  FROM Orders, Customers, Products
18  WHERE Orders.CustomerId = Customers.Id AND Orders.ProductId=Products.Id
19
```

152 % Результаты Сообщения

| | FirstName | ProductName | CreatedAt |
|---|---|---|---|
| 1 | Tom | Galaxy S8 | 2017-07-11 |
| 2 | Tom | iPhone 6S | 2017-07-13 |
| 3 | Bob | iPhone 6S | 2017-07-11 |

**Поскольку надо соединить три таблицы, то применяются как минимум два условия. Ключевой таблицей остается Orders, из которой извлекаются все заказы, а затем к ней подсоединяется данные по клиенту по условию Orders.CustomerId = Customers.Id и данные по товару по условию Orders.ProductId=Products.Id**

4) **Поскольку в данном случае названия таблиц сильно увеличивают код, то мы его можем сократить за счет использования псевдонимов таблиц:**

```
-- использования псевдонимов таблиц:
SELECT C.FirstName, P.ProductName, O.CreatedAt
FROM Orders AS O, Customers AS C, Products AS P
WHERE O.CustomerId = C.Id AND O.ProductId=P.Id
```

Результаты | Сообщения

| | FirstName | ProductName | CreatedAt |
|---|---|---|---|
| 1 | Tom | Galaxy S8 | 2017-07-11 |
| 2 | Tom | iPhone 6S | 2017-07-13 |
| 3 | Bob | iPhone 6S | 2017-07-11 |

**Если необходимо при использовании псевдонима выбрать все столбцы из определенной таблицы, то можно использовать звездочку:**

```
--при использовании псевдонима выбрать все столбцы из определенной таблицы,
--то можно использовать звездочку

SELECT C.FirstName, P.ProductName, O.*
FROM Orders AS O, Customers AS C, Products AS P
WHERE O.CustomerId = C.Id AND O.ProductId=P.Id
```

Результаты | Сообщения

| | FirstName | ProductName | Id | ProductId | CustomerId | CreatedAt | ProductCount | Price |
|---|---|---|---|---|---|---|---|---|
| 1 | Tom | Galaxy S8 | 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 |
| 2 | Tom | iPhone 6S | 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 |
| 3 | Bob | iPhone 6S | 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 |

2. **Для явного соединения данных из двух таблиц применяется оператор JOIN. Общий формальный синтаксис применения оператора INNER JOIN**

```
SELECT столбцы
FROM таблица1
   [INNER] JOIN таблица2
   ON условие1
   [[INNER] JOIN таблица3
   ON условие2]
```

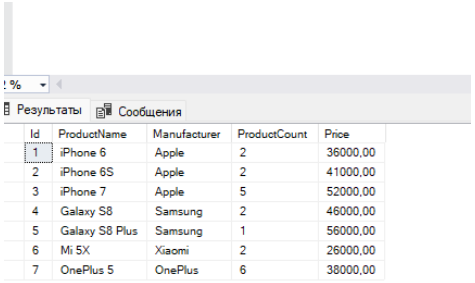
После оператора JOIN идет название второй таблицы, из которой надо добавить данные в выборку. *Перед JOIN может использоваться необязательное ключевое слово INNER. Его наличие или отсутствие ни на что не влияет.*

Затем после ключевого слова ON указывается условие соединения. Это условие устанавливает, как две таблицы будут сравниваться. В большинстве случаев для соединения применяется первичный ключ главной таблицы и внешний ключ зависимой таблицы.

**2.1 Используя JOIN, выберем все заказы и добавим к ним информацию о товарах:**

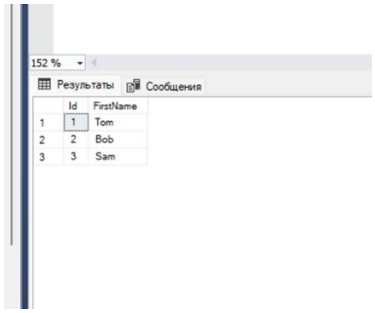
**1)**

**Исходные данные:**



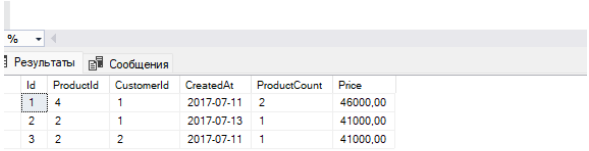
| Id | ProductName | Manufacturer | ProductCount | Price |
|---|---|---|---|---|
| 1 | iPhone 6 | Apple | 2 | 36000,00 |
| 2 | iPhone 6S | Apple | 2 | 41000,00 |
| 3 | iPhone 7 | Apple | 5 | 52000,00 |
| 4 | Galaxy S8 | Samsung | 2 | 46000,00 |
| 5 | Galaxy S8 Plus | Samsung | 1 | 56000,00 |
| 6 | Mi 5X | Xiaomi | 2 | 26000,00 |
| 7 | OnePlus 5 | OnePlus | 6 | 38000,00 |

Customers



| Id | FirstName |
|---|---|
| 1 | Tom |
| 2 | Bob |
| 3 | Sam |

Orders



| Id | ProductId | CustomerId | CreatedAt | ProductCount | Price |
|---|---|---|---|---|---|
| 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 |
| 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 |
| 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 |

```
use testbasa;
--выберем все заказы и добавим к ним информацию о товарах
SELECT Orders.CreatedAt, Orders.ProductCount, Products.ProductName
FROM Orders
JOIN Products ON Products.Id = Orders.ProductId
```

| CreatedAt | ProductCount | ProductName |
|---|---|---|
| 2017-07-11 | 2 | Galaxy S8 |
| 2017-07-13 | 1 | iPhone 6S |
| 2017-07-11 | 1 | iPhone 6S |

Поскольку таблицы могут содержать столбцы с одинаковыми названиями, то при указании столбцов для выборки указывается их полное имя вместе с именем таблицы, например, "Orders.ProductCount"

2)

```
--используя псевдонимы, мы можем сократить код:
SELECT O.CreatedAt, O.ProductCount, P.ProductName
FROM Orders AS O
JOIN Products AS P
ON P.Id = O.ProductId
```

Результаты | Сообщения

| | CreatedAt | ProductCount | ProductName |
|---|---|---|---|
| | 2017-07-11 | 2 | Galaxy S8 |
| | 2017-07-13 | 1 | iPhone 6S |
| | 2017-07-11 | 1 | iPhone 6S |

**3) Подобным образом мы можем присоединять и другие таблицы. Например, добавим к заказу информацию о покупателе из таблицы Customer**

```
--информацию о покупателе из таблицы Customer
SELECT Orders.CreatedAt, Customers.FirstName, Products.ProductName
FROM Orders
JOIN Products ON Products.Id = Orders.ProductId
JOIN Customers ON Customers.Id=Orders.CustomerId
```

Результаты | Сообщения

| | CreatedAt | FirstName | ProductName |
|---|---|---|---|
| 1 | 2017-07-11 | Tom | Galaxy S8 |
| 2 | 2017-07-13 | Tom | iPhone 6S |
| 3 | 2017-07-11 | Bob | iPhone 6S |

**4) Благодаря соединению таблиц мы можем использовать их столбцы для фильтрации выборки или ее сортировки:**

```
--Благодаря соединению таблиц мы можем использовать их столбцы для фильтрации выборки
или ее сортировки

SELECT Orders.CreatedAt, Customers.FirstName, Products.ProductName
FROM Orders
JOIN Products ON Products.Id = Orders.ProductId
JOIN Customers ON Customers.Id=Orders.CustomerId
WHERE Products.Price < 45000
ORDER BY Customers.FirstName
```

Результаты | Сообщения

| | CreatedAt | FirstName | ProductName |
|---|---|---|---|
| 1 | 2017-07-11 | Bob | iPhone 6S |
| 2 | 2017-07-13 | Tom | iPhone 6S |

## 5) Условия после ключевого слова ON могут быть более сложными по составу

```
31  --Условия после ключевого слова ON могут быть более сложными по составу
32  SELECT Orders.CreatedAt, Customers.FirstName, Products.ProductName
33  FROM Orders
34  JOIN Products ON Products.Id = Orders.ProductId AND Products.Manufacturer='Apple'
35  JOIN Customers ON Customers.Id=Orders.CustomerId
36  ORDER BY Customers.FirstName
```

Результаты | Сообщения

| CreatedAt | FirstName | ProductName |
|---|---|---|
| 2017-07-11 | Bob | iPhone 6S |
| 2017-07-13 | Tom | iPhone 6S |

3. **MS SQL Server также поддерживает внешнее соединение или outer join**.

**В отличие от inner join внешнее соединение возвращает все строки одной или двух таблиц, которые участвуют в соединении**.

Outer Join имеет следующий формальный синтаксис:

***SELECT столбцы***
***FROM таблица1***
***{LEFT\RIGHT\FULL} [OUTER] JOIN таблица2 ON условие1***
***[{LEFT\RIGHT\FULL} [OUTER] JOIN таблица3 ON условие2]...***

Перед оператором JOIN указывается одно из ключевых слов **LEFT, RIGHT или FULL**, которые определяют тип соединения:

**LEFT: выборка будет содержать все строки из первой или левой таблицы**

**RIGHT: выборка будет содержать все строки из второй или правой таблицы**

**FULL: выборка будет содержать все строки из обоих таблиц**

*Также перед оператором JOIN может указываться ключевое слово OUTER, но его применение необязательно. Далее после JOIN указывается присоединяемая таблица, а затем идет условие соединения.*

***LEFT [OUTER] JOIN — левое внешнее соединение***
***RIGHT [OUTER] JOIN — правое внешнее соединение***
***FULL [OUTER] JOIN — полное внешнее соединение***

*Внутренние соединения можно задавать в предложениях FROM и WHERE.*

*Внешние соединения и перекрестные соединения можно задавать только в предложении FROM.*

*Условия соединения сочетаются с условиями поиска WHERE и HAVING для управления строками, выбранными из базовых таблиц, на которые ссылается предложение FROM.*

*Табличный оператор JOIN принимает на вход две таблицы. Он может выполнять* ***операции трех различных видов соединения: перекрестного, внутреннего и внешнего.*** *Все они состоят из разных логических этапов обработки.*

*Перекрестное соединение включает только декартово произведение.*
*Внутреннее состоит из декартового произведения и фильтрации.*
***Внешнее соединение, в дополнение к внутреннему, добавляет внешние строки.***

***Внешнее соединение***

***Внешнее соединение позволяет в отличие от внутреннего извлечь не только строки с одинаковыми значениями соединяемых столбцов, но и строки без совпадений из одной или обеих таблиц.***

***Выделяют три вида внешних соединений:***

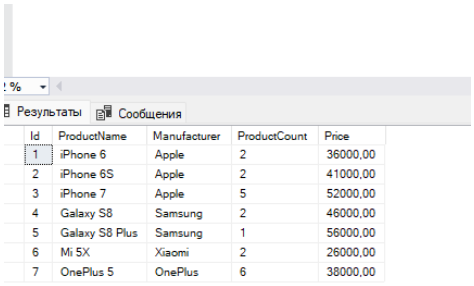
***левое внешнее соединение — в результирующий набор попадают все строки из таблицы с левой стороны оператора сравнения (независимо от того имеются ли совпадающие строки с правой стороны), а из таблицы с правой стороны — только строки с совпадающими значениями столбцов. При этом если для строки из левой таблицы нет соответствий в правой таблице, значениям строки в правой таблице будут присвоены NULL***

***правое внешнее соединение — аналогично левому внешнему соединению, но таблицы меняются местами***

***полное внешнее соединение — композиция левого и правого внешнего соединения: результирующий набор состоит из всех строк обеих таблиц. Если для строки одной из таблиц нет соответствующей строки в другой таблице, всем ячейкам строки второй таблицы присваивается значение NULL***

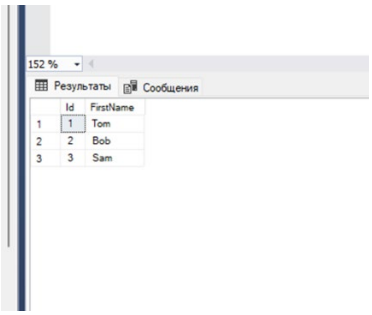
1) Соединим таблицы Orders и Customers

**Исходные данные:**



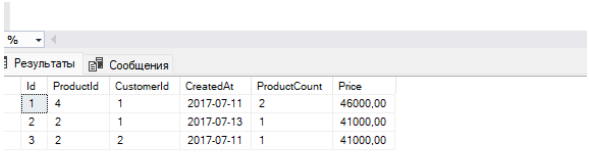
| Id | ProductName | Manufacturer | ProductCount | Price |
|---|---|---|---|---|
| 1 | iPhone 6 | Apple | 2 | 36000,00 |
| 2 | iPhone 6S | Apple | 2 | 41000,00 |
| 3 | iPhone 7 | Apple | 5 | 52000,00 |
| 4 | Galaxy S8 | Samsung | 2 | 46000,00 |
| 5 | Galaxy S8 Plus | Samsung | 1 | 56000,00 |
| 6 | Mi 5X | Xiaomi | 2 | 26000,00 |
| 7 | OnePlus 5 | OnePlus | 6 | 38000,00 |

Customers

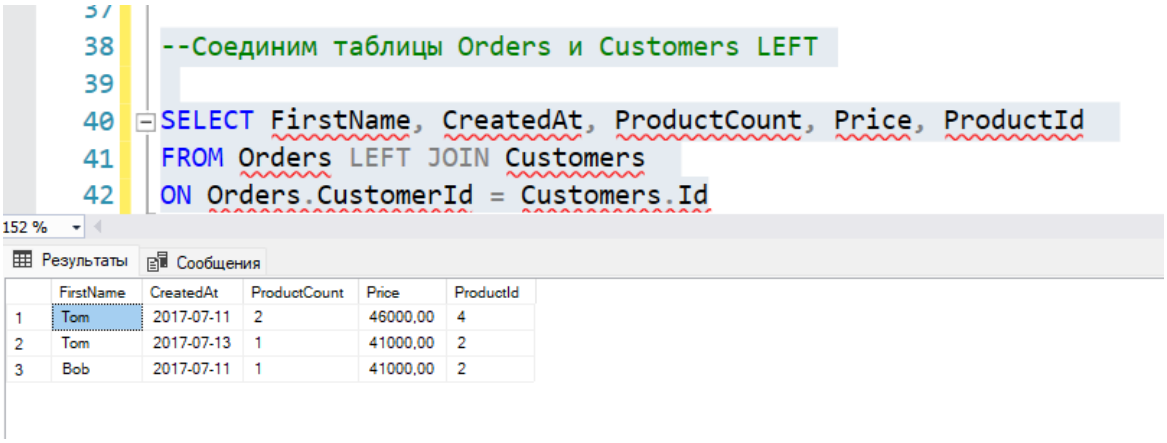


| Id | FirstName |
|---|---|
| 1 | Tom |
| 2 | Bob |
| 3 | Sam |

Orders



| Id | ProductId | CustomerId | CreatedAt | ProductCount | Price |
|---|---|---|---|---|---|
| 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 |
| 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 |
| 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 |



```
--Соединим таблицы Orders и Customers LEFT

SELECT FirstName, CreatedAt, ProductCount, Price, ProductId
FROM Orders LEFT JOIN Customers
ON Orders.CustomerId = Customers.Id
```

| | FirstName | CreatedAt | ProductCount | Price | ProductId |
|---|---|---|---|---|---|
| 1 | Tom | 2017-07-11 | 2 | 46000,00 | 4 |
| 2 | Tom | 2017-07-13 | 1 | 41000,00 | 2 |
| 3 | Bob | 2017-07-11 | 1 | 41000,00 | 2 |

Таблица Orders является первой или левой таблицей, а таблица Customers - правой таблицей. Поэтому, так как здесь используется выборка по левой таблице, то вначале будут выбираться все строки из Orders, а затем к ним по условию Orders.CustomerId = Customers.Id будут добавляться связанные строки из Customers.

По вышеприведенному результату может показаться, что левостороннее соединение аналогично INNER Join, но это не так.

**Inner Join объединяет строки из дух таблиц при соответствии условию. Если одна из таблиц содержит строки, которые не соответствуют этому условию, то данные строки не включаются в выходную выборку**.

*Inner join – это объединение когда выводятся все записи из одной таблицы и все соответствующие записи из другой таблице, а те записи которых нет в одной или в другой таблице выводиться не будут, т.е. только те записи которые соответствуют ключу.*

**Left Join выбирает все строки первой таблицы и затем присоединяет к ним строки правой таблицы.**

К примеру, возьмем таблицу Customers и добавим к покупателям информацию об их заказах:

```
45  -- INNER JOIN
46  SELECT FirstName, CreatedAt, ProductCount, Price
47  FROM Customers JOIN Orders
48  ON Orders.CustomerId = Customers.Id
49
50  --LEFT JOIN
51  SELECT FirstName, CreatedAt, ProductCount, Price
52  FROM Customers LEFT JOIN Orders
53  ON Orders.CustomerId = Customers.Id
```

152 %

Результаты | Сообщения

| | FirstName | CreatedAt | ProductCount | Price |
|---|---|---|---|---|
| 1 | Tom | 2017-07-11 | 2 | 46000,00 |
| 2 | Tom | 2017-07-13 | 1 | 41000,00 |
| 3 | Bob | 2017-07-11 | 1 | 41000,00 |

| | FirstName | CreatedAt | ProductCount | Price |
|---|---|---|---|---|
| 1 | Tom | 2017-07-11 | 2 | 46000,00 |
| 2 | Tom | 2017-07-13 | 1 | 41000,00 |
| 3 | Bob | 2017-07-11 | 1 | 41000,00 |
| 4 | Sam | NULL | NULL | NULL |

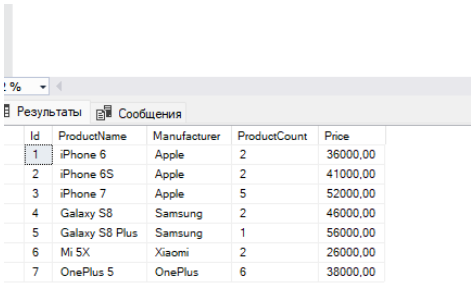
***LEFT JOIN – это объединение данных по левому ключу, т.е. допустим, мы объединяем две таблицы по left join, и это значит что все данные из второй таблицы подтянутся к первой, а в случае отсутствия ключа выведется NULL значения, другими словами выведутся все данные из левой таблицы и все данные по ключу из правой таблицы.***

***2 Изменим в примере выше тип соединения на правостороннее:***

*Теперь будут выбираться все строки из Customers, а к ним уже будет присоединяться связанные по условию строки из таблицы Orders:*

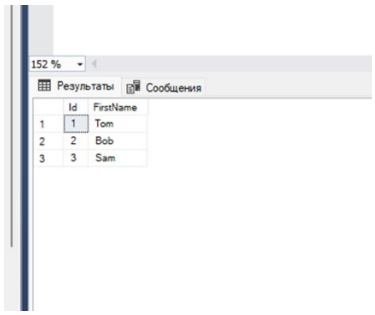
***RIGHT JOIN – это такое же объединение как и Left join только будут выводиться все данные из правой таблицы и только те данные из левой таблицы в которых есть ключ объединения.***

**Исходные данные:**



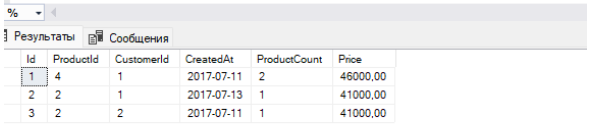
| Id | ProductName | Manufacturer | ProductCount | Price |
|---|---|---|---|---|
| 1 | iPhone 6 | Apple | 2 | 36000,00 |
| 2 | iPhone 6S | Apple | 2 | 41000,00 |
| 3 | iPhone 7 | Apple | 5 | 52000,00 |
| 4 | Galaxy S8 | Samsung | 2 | 46000,00 |
| 5 | Galaxy S8 Plus | Samsung | 1 | 56000,00 |
| 6 | Mi 5X | Xiaomi | 2 | 26000,00 |
| 7 | OnePlus 5 | OnePlus | 6 | 38000,00 |

Customers

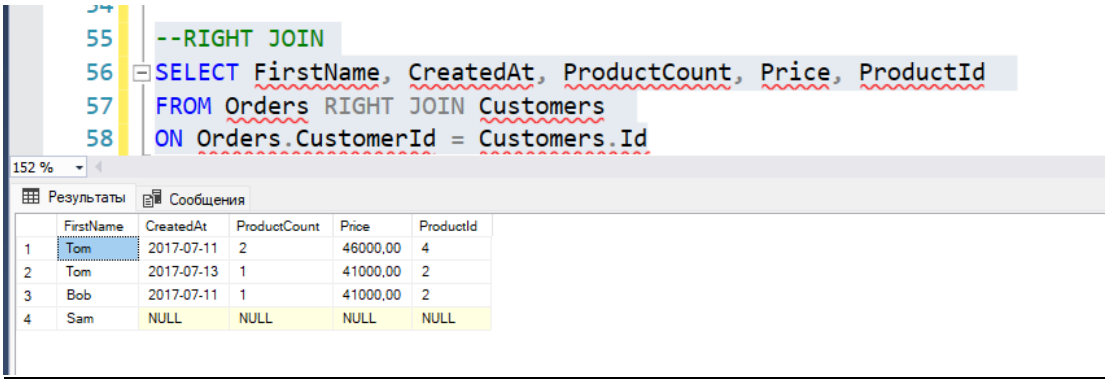


| Id | FirstName |
|---|---|
| 1 | Tom |
| 2 | Bob |
| 3 | Sam |

Orders



| Id | ProductId | CustomerId | CreatedAt | ProductCount | Price |
|---|---|---|---|---|---|
| 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 |
| 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 |
| 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 |



```
55  --RIGHT JOIN
56  SELECT FirstName, CreatedAt, ProductCount, Price, ProductId
57  FROM Orders RIGHT JOIN Customers
58  ON Orders.CustomerId = Customers.Id
```

| | FirstName | CreatedAt | ProductCount | Price | ProductId |
|---|---|---|---|---|---|
| 1 | Tom | 2017-07-11 | 2 | 46000,00 | 4 |
| 2 | Tom | 2017-07-13 | 1 | 41000,00 | 2 |
| 3 | Bob | 2017-07-11 | 1 | 41000,00 | 2 |
| 4 | Sam | NULL | NULL | NULL | NULL |

Поскольку один из покупателей из таблицы Customers не имеет связанных заказов из Orders, то соответствующие столбцы, которые берутся из Orders, будут иметь значение NULL.

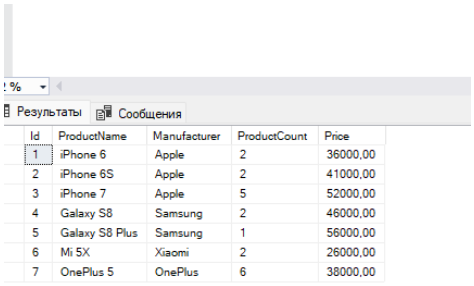
## 4. Объединение SQL CROSS JOIN

**Cross Join или перекрестное соединение создает набор строк, где каждая строка из одной таблицы соединяется с каждой строкой из второй таблицы.**

**CROSS JOIN – это объединение SQL по которым каждая строка одной таблицы объединяется с каждой строкой другой таблицы.**

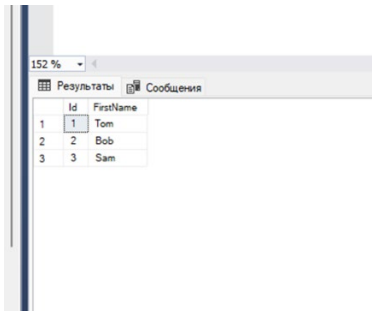
Например, соединим таблицу заказов Orders и таблицу покупателей Customers

**Исходные данные:**



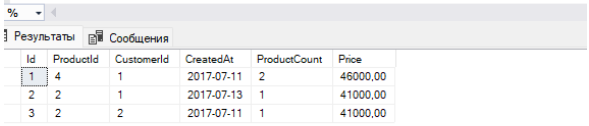
| Id | ProductName | Manufacturer | ProductCount | Price |
|---|---|---|---|---|
| 1 | iPhone 6 | Apple | 2 | 36000,00 |
| 2 | iPhone 6S | Apple | 2 | 41000,00 |
| 3 | iPhone 7 | Apple | 5 | 52000,00 |
| 4 | Galaxy S8 | Samsung | 2 | 46000,00 |
| 5 | Galaxy S8 Plus | Samsung | 1 | 56000,00 |
| 6 | Mi 5X | Xiaomi | 2 | 26000,00 |
| 7 | OnePlus 5 | OnePlus | 6 | 38000,00 |

Customers



| Id | FirstName |
|---|---|
| 1 | Tom |
| 2 | Bob |
| 3 | Sam |

Orders



| Id | ProductId | CustomerId | CreatedAt | ProductCount | Price |
|---|---|---|---|---|---|
| 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 |
| 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 |
| 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 |

```
60  --соединим таблицу заказов Orders и таблицу покупателей Customers
61  SELECT * FROM Orders CROSS JOIN Customers
```

| | Id | ProductId | CustomerId | CreatedAt | ProductCount | Price | Id | FirstName |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 | 1 | Tom |
| 2 | 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 | 1 | Tom |
| 3 | 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 | 1 | Tom |
| 4 | 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 | 2 | Bob |
| 5 | 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 | 2 | Bob |
| 6 | 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 | 2 | Bob |
| 7 | 1 | 4 | 1 | 2017-07-11 | 2 | 46000,00 | 3 | Sam |
| 8 | 2 | 2 | 1 | 2017-07-13 | 1 | 41000,00 | 3 | Sam |
| 9 | 3 | 2 | 2 | 2017-07-11 | 1 | 41000,00 | 3 | Sam |