

## Лекция 8.1.

**Операторы работы с наборами. Объединение результатов двух запросов в один результирующий набор.**

**Оператор UNION [ALL].**

**Пересечение и разность результатов двух запросов.**

**Операторы INTERSECT, EXCEPT.**

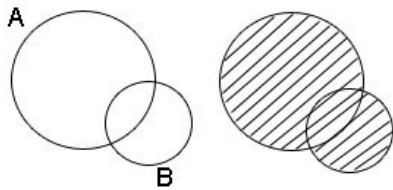
### 1. Оператор UNION [ALL]

Объединением множеств  $A$  и  $B$  называется множество, обозначаемое  $A \cup B$ , состоящее из всех тех и только тех элементов, которые принадлежат или  $A$  или  $B$ , то есть

$$A \cup B = \{x \mid (x \in A) \vee (x \in B)\}.$$

Например, если  $A = \{1, 2, 3, 4\}$ ,  $B = \{1, 3, 5\}$ ,  $C = \{5, 6\}$ , то  
 $A \cup B = \{1, 2, 3, 4, 5\}$ ,  $A \cup C = \{1, 2, 3, 4, 5, 6\}$ ,  $B \cup C = \{1, 3, 5, 6\}$ .

Операции над множествами удобно иллюстрировать фигурами, называемыми **диаграммами Венна** (другое название - **круги Эйлера**). На рисунке ниже слева большим и малым кругами обозначены соответственно множества  $A$  и  $B$ , а справа - результат объединения этих множеств (заштрихованная фигура).



На основе теории множеств создана **концепция реляционных баз данных**, а на основе операций над множествами - **реляционная алгебра** и её операции - используемые в языке запросов к базам данных **SQL**. Операция объединения есть в реляционной алгебре.

## Повтор лекции 2. Раздел 2 Модели данных

1) Объединением двух совместимых по типу отношений  $A$  и  $B$  ( $A \text{ UNION } B$ ) называется отношение с тем же заголовком, как и в отношениях  $A$  и  $B$ , и с телом, состоящим из множества всех кортежей, принадлежащих  $A$  или  $B$  или обоим отношениям.

*Примеры:*

*Исходные отношения:*

A		
CityNo	CityName	RgNo
1	Темиртау	1
2	Сарань	1
3	Балхаш	1

B		
CityNo	CityName	RgNo
2	Сарань	1
3	Балхаш	1
4	Алматы	2

*Объединение:*

A UNION B		
CityNo	CityName	RgNo
1	Темиртау	1
2	Сарань	1
3	Балхаш	1
4	Алматы	2

Объединение – бинарная операция над односхемными отношениями результатом является отношение, которое включает в себя все кортежи обоих исходных отношений без повторов; причём, имена полей односхемных отношений могут быть разными, достаточно, чтобы совпадало количество полей и типы данных соответствующих полей.

*Пример. Для отношений  $R(A,B,C)$  и  $S(A,D,E)$ , объединение  $R \cup S$  будет таким как на рис.:*

Отношение R		
A	B	C
a	b	c
c	a	d
c	b	d

Отношение S		
A	D	E
c	a	e
c	b	d

Объединение R U S		
A	D	F
a	b	c
c	a	d
c	b	d
c	a	e

**Объединением** двух множеств (назовем их А и В) является множество, содержащее все элементы из А и В. Другими словами, в результирующий набор попадают элементы, принадлежащие хотя бы одному из множеств. Закрашенная область представляет результат, возвращаемый оператором работы с наборами.

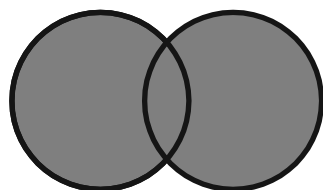


Рис. 6.1. Объединение двух множеств А и В

В языке T-SQL **оператор UNION** объединяет результаты двух входящих запросов. Строка попадает в результирующий набор даже в том случае, если присутствует только с одной стороны. Этот оператор в T-SQL поддерживает оба параметра — **DISTINCT** (по умолчанию) и **ALL**.

Для объединения результатов двух или более запросов в одну таблицу используется команда **UNION**. Команда **UNION** объединяет вывод двух или более запросов в единый набор строк и столбцов и имеет вид:

*Первый запрос*

*UNION [ALL]*

*Второй запрос*

...

Для объединения результатов нескольких запросов с помощью **UNION**, они должны соответствовать следующим требованиям:

- содержать одинаковое количество столбцов;
- типы данных столбцов должны совпадать во всех запросах;
- в промежуточных запросах нельзя использовать сортировку **ORDER BY**.

*Чтобы отсортировать результат объединения, в конце запроса добавляется **ORDER BY**. Названия столбцов в запросах могут отличаться. Поэтому в команде **ORDER BY** указывается название столбцов с первого запроса.*

Если объединяемые наборы содержат в строках идентичные значения, то при объединении повторяющиеся строки удаляются.

*Если необходимо при объединении сохранить повторяющиеся строки, то для этого используется параметр **ALL**.*

*Все запросы выполняются независимо друг от друга, а уже их вывод объединяется.*

*В объединяемых запросах можно использовать одну и ту же таблицу.*

Результатом объединения двух множеств (отношений) A и B ( $A \cup B$ ) будет такое множество (отношение) C, которое включает в себя те и только те элементы, которые есть или во множестве A или во множестве B. Говоря упрощённо, все элементы множества A и множества B, за исключением дубликатов, образующихся за счёт того, что некоторые элементы есть и в первом, и во втором множестве.

R1				R2		
A1	A2	A3		A1	A2	A3
Z7	aa	w11		X8	pp	k21
B7	hh	h15		Q2	ee	h15
X8	pp	w11		X8	pp	w11

*SELECT A1, A2, A3 from R1 UNION SELECT A1, A2, A3 from R2*

R		
A1	A2	A3
Z7	aa	w11
B7	hh	h15
X8	pp	w11
X8	pp	k21
Q2	ee	h15

Оператор UNION подобно inner join или outer join позволяет соединить две таблицы. Но в отличие от inner/outer join объединения соединяют не столбцы разных таблиц, а два однотипных набора в один. Формальный синтаксис объединения:

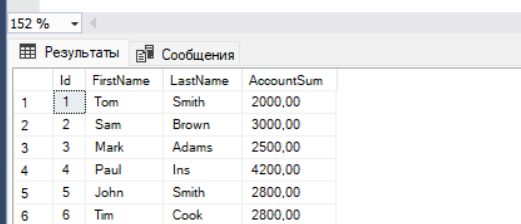
***SELECT\_выражение1  
UNION [ALL] SELECT\_выражение2  
[UNION [ALL] SELECT\_выражениеN]***

Пусть в базе данных будут две отдельные таблицы для клиентов банка (таблица Customers\_MN) и для сотрудников банка (таблица Employees\_MN):

```
1 use master;
2 create database testbasa_MN;
3 use testbasa_MN;
4 CREATE TABLE Customers_MN
5 (
6     Id INT IDENTITY PRIMARY KEY,
7     FirstName NVARCHAR(20) NOT NULL,
8     LastName NVARCHAR(20) NOT NULL,
9     AccountSum MONEY
10 );
11 CREATE TABLE Employees_MN
12 (
13     Id INT IDENTITY PRIMARY KEY,
14     FirstName NVARCHAR(20) NOT NULL,
15     LastName NVARCHAR(20) NOT NULL,
16 );
17
18 INSERT INTO Customers_MN VALUES
19 ('Tom', 'Smith', 2000),
20 ('Sam', 'Brown', 3000),
21 ('Mark', 'Adams', 2500),
22 ('Paul', 'Ins', 4200),
23 ('John', 'Smith', 2800),
24 ('Tim', 'Cook', 2800)
25 INSERT INTO Employees_MN VALUES
26 ('Homer', 'Simpson'),
27 ('Tom', 'Smith'),
28 ('Mark', 'Adams'),
29 ('Nick', 'Svensson')|
```

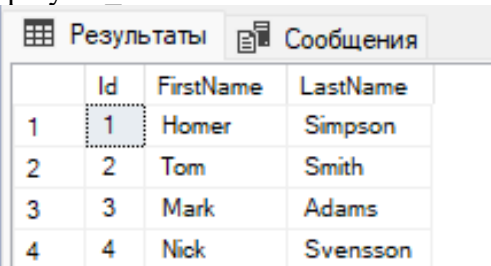
Сообщения

## Customers\_MN



	Id	FirstName	LastName	AccountSum
1	1	Tom	Smith	2000,00
2	2	Sam	Brown	3000,00
3	3	Mark	Adams	2500,00
4	4	Paul	Ins	4200,00
5	5	John	Smith	2800,00
6	6	Tim	Cook	2800,00

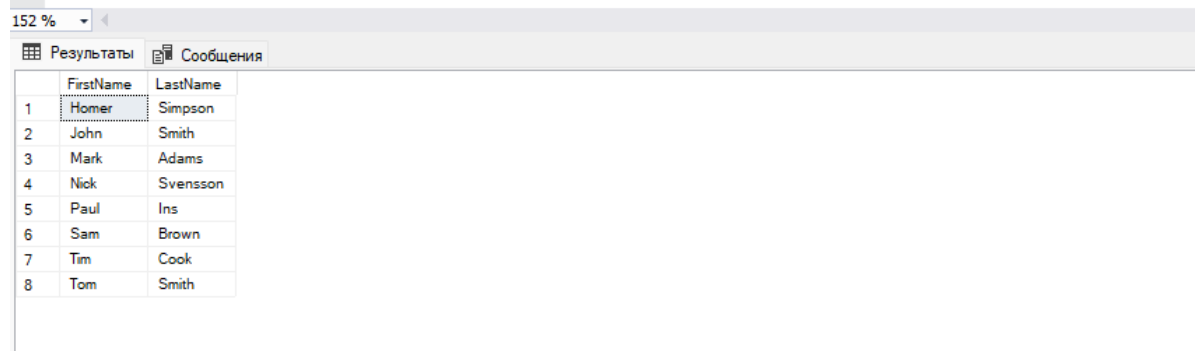
## Employees\_MN



	Id	FirstName	LastName
1	1	Homer	Simpson
2	2	Tom	Smith
3	3	Mark	Adams
4	4	Nick	Svensson

Здесь можем заметить, что обе таблицы, несмотря на наличие различных данных, могут характеризоваться двумя общими атрибутами - именем (FirstName) и фамилией (LastName). Выберем сразу всех клиентов банка и его сотрудников из обеих таблиц

```
33 use testbasa_MN;  
34 SELECT FirstName, LastName  
35 FROM Customers_MN  
36 UNION SELECT FirstName, LastName FROM Employees_MN
```



	FirstName	LastName
1	Homer	Simpson
2	John	Smith
3	Mark	Adams
4	Nick	Svensson
5	Paul	Ins
6	Sam	Brown
7	Tim	Cook
8	Tom	Smith

В данном случае из первой таблицы выбираются два значения - имя и фамилия клиента. Из второй таблицы Employees также выбираются два значения - имя и фамилия сотрудников. То есть при объединении количество выбираемых столбцов и их тип совпадают для обеих выборок.

Оператор работы с множествами UNION (с параметром DISTINCT по умолчанию) **объединяет результаты выполнения двух запросов, удаляя дубликаты**. Если строка содержится в обоих наборах, в конечный результат она попадет только один раз; другими словами, итоговый набор **является полноценным множеством**.

## Customers\_MN

	Id	FirstName	LastName	AccountSum
1	1	Tom	Smith	2000,00
2	2	Sam	Brown	3000,00
3	3	Mark	Adams	2500,00
4	4	Paul	Ins	4200,00
5	5	John	Smith	2800,00
6	6	Tim	Cook	2800,00

## Employees\_MN

	Id	FirstName	LastName
1	1	Homer	Simpson
2	2	Tom	Smith
3	3	Mark	Adams
4	4	Nick	Svensson

При этом названия столбцов объединенной выборки будут совпадать с названиями столбцов первой выборки. И если мы захотим при этом еще произвести сортировку, то в выражениях **ORDER BY** необходимо ориентироваться именно на названия столбцов первой выборки

```

38  --Сортировка
39  SELECT FirstName + ' ' + LastName AS FullName
40  FROM Customers_MN
41  UNION SELECT FirstName + ' ' + LastName AS EmployeeName
42  FROM Employees_MN
43  ORDER BY FullName DESC

```

	FullName
1	Tom Smith
2	Tim Cook
3	Sam Brown
4	Paul Ins
5	Nick Svensson
6	Mark Adams
7	John Smith
8	Homer Simpson

Если же в одной выборке **больше столбцов, чем в другой**, то они не смогут быть объединены. Например, в следующем случае объединение завершится с ошибкой:

```

SELECT FirstName, LastName, AccountSum
FROM Customers
UNION SELECT FirstName, LastName
FROM Employees

```

Также соответствующие **столбцы должны соответствовать по типу**. Так, следующий пример завершится с ошибкой из-за не соответствия по типу данных:

```

SELECT FirstName, LastName
FROM Customers
UNION SELECT Id, LastName
FROM Employees

```

В данном случае первый столбец первой выборки имеет тип *NVARCHAR*, то есть хранит строку. Первый столбец второй выборки - *Id* имеет тип *INT*, то есть хранит число.

Если оба объединяемых набора содержат в строках идентичные значения, то при объединении повторяющиеся строки удаляются.

Например, в случае с таблицами Customers\_MN и Employees\_MN сотрудники банка могут быть одновременно его клиентами и содержаться в обеих таблицах. При объединении в примерах выше всех дублирующиеся строки удалялись.

Если же необходимо при объединении сохранить все, в том числе повторяющиеся строки, то для этого необходимо использовать оператор ALL

```
44
45 --Выберем сразу всех клиентов банка и его сотрудников из обеих таблиц
46 --с дублирующимися строками
47
48 use testbasa_MN;
49 SELECT FirstName, LastName
50 FROM Customers_MN
51 UNION ALL SELECT FirstName, LastName FROM Employees_MN
52
```

152 %

Результаты Сообщения

	FirstName	LastName
1	Tom	Smith
2	Sam	Brown
3	Mark	Adams
4	Paul	Ins
5	John	Smith
6	Tim	Cook
7	Homer	Simpson
8	Tom	Smith
9	Mark	Adams
10	Nick	Svensson

Поскольку оператор UNION ALL не удаляет дубликаты, возвращаемый им результат является **мультимножеством**. Одна и та же строка может присутствовать в результирующем наборе в нескольких экземплярах



152 %

Результаты Сообщения

	Id	FirstName	LastName	AccountSum
1	1	Tom	Smith	2000,00
2	2	Sam	Brown	3000,00
3	3	Mark	Adams	2500,00
4	4	Paul	Ins	4200,00
5	5	John	Smith	2800,00
6	6	Tim	Cook	2800,00

**Объединять выборки можно и из одной и той же таблицы.** Например, в зависимости от суммы на счете клиента нам надо начислять ему определенные проценты:

В данном случае если сумма меньше 3000, то начисляются проценты в размере 10% от суммы на счете. Если на счете больше 3000, то проценты увеличиваются до 30%.

```

55 SELECT FirstName, LastName, AccountSum + AccountSum * 0.1 AS TotalSum
56 FROM Customers_MN WHERE AccountSum < 3000
57 UNION SELECT FirstName, LastName, AccountSum + AccountSum * 0.3 AS TotalSum
58 FROM Customers_MN WHERE AccountSum >= 3000
59
60

```

152 %

Результаты Сообщения

	FirstName	LastName	TotalSum
1	John	Smith	3080.00000
2	Mark	Adams	2750.00000
3	Paul	Ins	5460.00000
4	Sam	Brown	3900.00000
5	Tim	Cook	3080.00000
6	Tom	Smith	2200.00000

**Пример 1.** В базе данных фирмы есть таблица Staff, содержащая данные о сотрудниках фирмы. В ней есть столбцы Salary (размер заработной платы), Job (должность) и Years (длительность трудового стажа). Первый запрос возвращает индивидуальные размеры заработной платы, упорядоченные по должностям:

```
SELECT Name, Job, Salary
FROM STAFF ORDER BY Job
```

Результатом выполнения запроса будет следующая таблица:

Name	Job	Salary
Sanders	Mgr	18357.5
Marenghi	Mgr	17506.8
Pernal	Sales	18171.2
Doctor	Sales	12322.4
Factor	Sales	16228.7

Второй запрос вернёт суммарную заработную плату по должностям. Мы уже готовим этот запрос для соединения с первым, поэтому будем помнить, что условием соединения является равное число столбцов, совпадение их названий, порядка следования и типов данных. Поэтому включаем в таблицу с итогами также столбец Name с произвольным значением 'Z-TOTAL':

```
SELECT 'Z-TOTAL' AS Name, Job, SUM(Salary) AS Salary
FROM STAFF GROUP BY Job
```

Результатом выполнения запроса будет следующая таблица:

Name	Job	Salary
Z-TOTAL	Mgr	35864.3
Z-TOTAL	Sales	46722.3

Теперь объединим запросы при помощи оператора UNION и применим оператору ORDER BY к результату объединения. Группировать следует по двум столбцам: должность (Job) и имя (Name), чтобы строки с итоговыми (суммарными) значениями, в которых значение имени - 'Z-TOTAL', находились ниже строк с индивидуальными значениями. Объединение результатов запросов будет следующим:

```
(SELECT Name, Job, Salary
FROM STAFF)
UNION
(SELECT 'Z-TOTAL' AS Name,
Job, SUM(Salary) AS Salary
FROM STAFF GROUP BY Job)
ORDER BY Job, Name
```

Результатом выполнения запроса с оператором UNION будет следующая таблица, в которой каждая первая строка в каждой группе должностей будет содержать суммарную заработную плату сотрудников, работающих на этой должности:

Name	Job	Salary
Marenghi	Mgr	17506.8
Sanders	Mgr	18357.5
Z-TOTAL	Mgr	35864.3
Doctor	Sales	12322.4
Factor	Sales	16228.7
Pernal	Sales	18171.2
Z-TOTAL	Sales	46722.3

Команда UNION объединяет данные из нескольких таблиц в одну при выборке.

При объединении количество столбцов во всех таблицах должно совпадать, иначе будет ошибка

Имена столбцов будут такие же, как в основной таблице, в которую добавляются данные из других таблиц.

Внимание: если не используется ключевое слово ALL для UNION, все возвращенные строки будут уникальными, так как по умолчанию подразумевается distinct, который удаляет неуникальные значения.

Чтобы отменить такое поведение - нужно указать ключевое слово ALL  
**UNION ALL**

## 2. Пересечение результатов двух запросов.

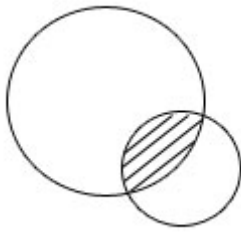
### Оператор NTERSECT

Пересечением множеств  $A$  и  $B$  называется множество, обозначаемое  $A \cap B$  и состоящее из всех тех и только тех элементов, которые принадлежат каждому из множеств  $A$  и  $B$ , то есть

$$A \cap B = \{x \mid (x \in A) \wedge (x \in B)\}.$$

Например, если  $A = \{1, 2, 3, 4\}$ ,  $B = \{1, 3, 5\}$ ,  $C = \{5, 6\}$ , то  $A \cap B = \{1, 3\}$ ,  $A \cap C = \emptyset$ ,  $B \cap C = \{5\}$ .

На рисунке ниже - результат пересечения множеств  $A$  и  $B$  - заштрихованная фигура.



## Повтор лекции 2. Раздел 2 Модели данных

### Пересечение:

*Исходные отношения:*

A		
CityNo	CityName	RgNo
1	Темиртау	1
2	Сарань	1
3	Балхаш	1

B		
CityNo	CityName	RgNo
2	Сарань	1
3	Балхаш	1
4	Алматы	2

A INTERSECT B		
CityNo	CityName	RgNo
2	Сарань	1
3	Балхаш	1

*Пересечение – бинарная операция над односхемными отношениями. Пересечение односхемных отношений  $R$  и  $S$  есть подмножество кортежей, принадлежащих обоим отношениям; причём, имена полей односхемных отношений могут быть разными, достаточно, чтобы совпадало количество полей и типы данных соответствующих полей.*

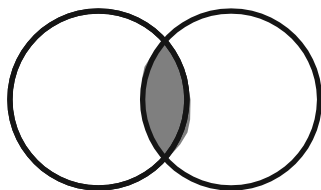
*Пересечение можно выразить через разность так:  $R \cap S = R - (R - S)$ .*

Отношение R		
A	B	C
a	b	c
c	a	d
c	b	d

Отношение S		
A	D	E
g	h	a
a	b	c
c	a	e

Пересечение $R \cap S$		
A	D	F
a	b	c

**Пересечением** двух множеств (назовем их А и В) является множество всех элементов, которые принадлежат как А, так и В.



В языке T-SQL оператор **INTERSECT** возвращает пересечение результирующих наборов, полученных при выполнении двух запросов; в результат попадают только те строки, которые содержатся в обоих входящих наборах. Сначала мы рассмотрим оператор **INTERSECT** (с параметром **DISTINCT** по умолчанию), **затем перейдем к его аналогу, *INTERSECT ALL*, который возвращает мультимножество**

### **Множества (оператор INTERSECT)**

Первым делом оператор **INTERSECT** **убирает дубликаты из двух входящих мультимножеств** и затем возвращает строки, которые содержатся в обоих наборах. Таким образом, если строка попадает в результат, это означает, что она как минимум по одному разу входит в каждое мультимножество.

Команда **INTERSECT** позволяет **найти общие строки в результатах двух запросов**, то есть данный оператор выполняет операцию пересечения множеств. Команда **INTERSECT** имеет следующий вид:

***Первый запрос INTERSECT***

***Второй запрос.***

***Требования к использованию команды INTERSECT такие же, как к командам UNION***

Теперь посмотрим, что получится в результате выполнения этой операции реляционной алгебры и соответствующего ей запроса SQL. Вновь даны два отношения R1 и R2:

R1			R2		
A1	A2	A3	A1	A2	A3
Z7	aa	w11	X8	pp	k21
B7	hh	h15	Q2	ee	h15
X8	pp	w11	X8	pp	w11

*SELECT A1, A2, A3 from R1 INTERSECT SELECT A1, A2, A3 from R2*

Просматриваем все записи в двух отношениях, и обнаруживаем, что и в первом, и во втором отношении есть одна строка - та, которая является третьей и в первом, и во втором отношении.

Получаем новое отношение

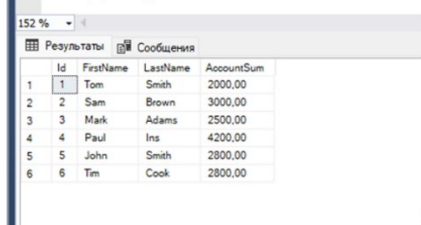
R		
A1	A2	A3
X8	pp	w11

Оператор INTERSECT позволяет найти общие строки для двух выборок, то есть данный оператор выполняет операцию пересечения множеств.

Для его использования применяется следующий формальный синтаксис:

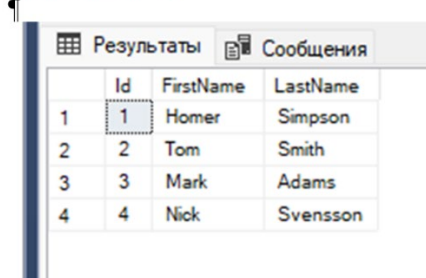
*SELECT\_выражение1*  
*INTERSECT SELECT\_выражение2*

Customers\_MN



	Id	FirstName	LastName	AccountSum
1	1	Tom	Smith	2000,00
2	2	Sam	Brown	3000,00
3	3	Mark	Adams	2500,00
4	4	Paul	Ins	4200,00
5	5	John	Smith	2800,00
6	6	Tim	Cook	2800,00

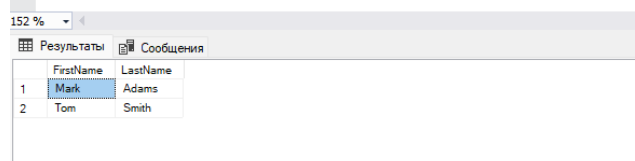
Employees\_MN



	Id	FirstName	LastName
1	1	Homer	Simpson
2	2	Tom	Smith
3	3	Mark	Adams
4	4	Nick	Svensson

**В таблице Customers хранятся все клиенты банка, а в таблице Employees - все его сотрудники. Но сотрудники могут быть одновременно и клиентами банка, поэтому их данные могут храниться сразу в двух таблицах. Найдем всех сотрудников банка, которые одновременно являются его клиентами. То есть нам надо найти общие элементы двух выборок:**

```
1 --операция пересечение
2 use testbasa_MN;
3 SELECT FirstName, LastName
4 FROM Employees_MN
5 INTERSECT SELECT FirstName, LastName
6 FROM Customers_MN
```



	FirstName	LastName
1	Mark	Adams
2	Tom	Smith





**INTERSECT** (*пересечение*) – это оператор Transact-SQL, который выводит одинаковые строки из первого, второго и последующих наборов данных. Другими словами, он выведет только те строки, которые есть как в первом результирующем наборе, так и во втором (третьем и так далее), т.е. происходит пересечение этих строк.

Данный оператор очень полезен, например, тогда, когда необходимо узнать какие строки есть и в первой таблице и во второй (*к примеру, повтор данных*).

Как и у оператора UNION, у INTERSECT есть правила, например, то, что количество полей во всех результирующих наборах должно быть одинаковым, также как и их тип данных.

### 3. Разность результатов двух запросов

#### Операторы EXCEPT

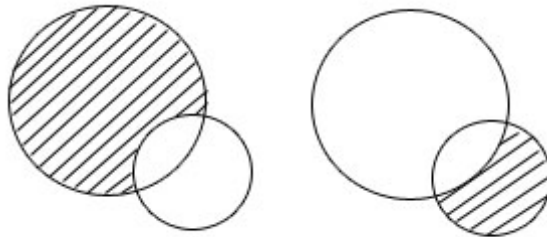
*Разность множеств*

*Разностью множеств  $A$  и  $B$  называется множество, обозначаемое  $A - B$  и состоящее из всех тех и только тех элементов множества  $A$ , которые не являются элементами множества  $B$ , то есть*

$$A - B = \{x \mid (x \in A) \wedge (x \notin B)\}.$$

*Например, если  $A = \{1, 2, 3, 4\}$ ,  $B = \{1, 3, 5\}$ ,  $C = \{5, 6\}$ , то  $A - B = \{2, 4\}$ ,  $A - C = A$ ,  $B - C = \{1, 3\}$ ,  $C - B = \{6\}$ ,  $B - A = \{5\}$ .*

*На рисунке ниже слева - результат разности множеств  $A$  и  $B$ , а справа - результат разности множеств  $B$  и  $A$ .*



## Повтор лекции 2. Раздел 2 Модели данных

### Вычитанием двух совместимых по типу отношений А и В (A MINUS B)

называется отношение с тем же заголовком, как и в отношениях А и В, и с телом, состоящим из множества всех кортежей, принадлежащих отношению А и не принадлежащих отношению В.

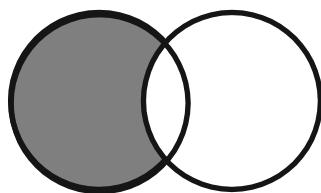
A		
CityNo	CityName	RgNo
1	Темиртау	1
2	Сарань	1
3	Балхаш	1

B		
CityNo	CityName	RgNo
2	Сарань	1
3	Балхаш	1
4	Алматы	2

A MINUS B		
CityNo	CityName	RgNo
1	Темиртау	1

B MINUS A		
CityNo	CityName	RgNo
4	Алматы	2

**Разностью** множеств А и В ( $A - B$ ) является множество элементов, принадлежащих А, но не принадлежащих В. Можете считать, что это все элементы А, кроме тех, которые также входят в В.



В языке T-SQL разность множеств реализована в виде оператора работы с наборами под названием EXCEPT; он принимает два входящих запроса и возвращает строки, которые присутствуют в первом из них, но отсутствуют во втором.

**Рассмотрим оператор EXCEPT (с параметром DISTINCT по умолчанию)**

**Множества (оператор EXCEPT)**

Первым делом оператор EXCEPT убирает дубликаты из двух входящих мультимножеств (превращая их тем самым в настоящие множества), после чего возвращает строки, которые содержатся в первом наборе, но которых нет во втором.

*SELECT\_выражение1*  
*EXCEPT SELECT\_выражение2*

Другими словами, строка, попавшая в конечный результат, встречается в первом мультимножестве как минимум в одном экземпляре, а во втором ее точно нет.

**Оператор EXCEPT, в отличие от предыдущих двух, является ассиметричным, то есть учитывает порядок следования входящих множеств.**

**Здесь те же правила, что и у оператора INTERSECT, т.е. количество столбцов (и их тип) должно быть одинаковым.**

***Таблица Employees содержит данные обо всех сотрудниках банка, а таблица Customers - обо всех клиентах. Но сотрудники банка могут также быть его клиентами. И допустим, нам надо найти всех клиентов банка, которые не являются его сотрудниками:***

Customers\_MN

	Id	FirstName	LastName	AccountSum
1	1	Tom	Smith	2000,00
2	2	Sam	Brown	3000,00
3	3	Mark	Adams	2500,00
4	4	Paul	Ins	4200,00
5	5	John	Smith	2800,00
6	6	Tim	Cook	2800,00

Employees\_MN

	Id	FirstName	LastName
1	1	Homer	Simpson
2	2	Tom	Smith
3	3	Mark	Adams
4	4	Nick	Svensson

```

8 SELECT FirstName, LastName
9 FROM Customers_MN
10 EXCEPT SELECT FirstName, LastName
11 FROM Employees_MN

```

	FirstName	LastName
1	John	Smith
2	Paul	Ins
3	Sam	Brown
4	Tim	Cook

***Подобным образом можно получить всех сотрудников банка, которые не являются его клиентами:***

```

13 SELECT FirstName, LastName
14 FROM Employees_MN
15 EXCEPT SELECT FirstName, LastName
16 FROM Customers_MN

```

	FirstName	LastName
1	Homer	Simpson
2	Nick	Svensson

***EXCEPT полезен тогда, когда необходимо сравнить две таблицы и вывести только те строки первой таблицы, которых нет в другой таблице.***