Оглавление

Соединения (JOIN)

В теории множеств существует три наиболее общих операции:

- **Пересечение** – используется для выделения общих элементов в двух или более множествах.
- **Разность** – Используется для поиска элементов, которые имеются в одном множестве, но отсутствуют в другом.
- **Объединение** – Используется для объединения двух или более множеств.

Пересечение множеств

Пересечение множеств — это множество, которому принадлежат те, и только те элементы, которые одновременно принадлежат всем данным множествам.

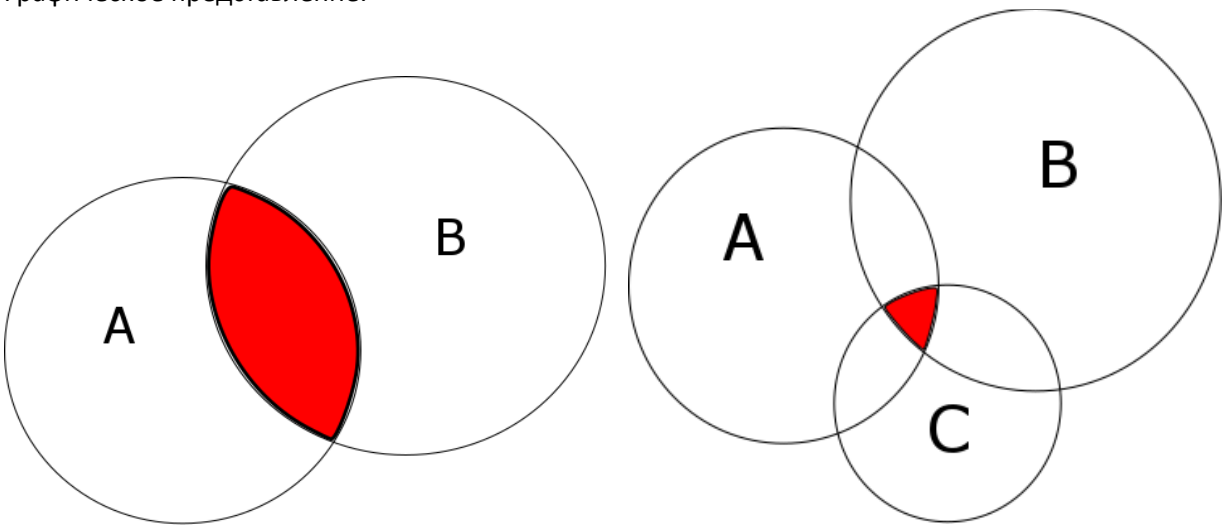
Рассмотрим два множества целых чисел:

A = {1, 5, 8, 9, 32, 55, 78}
B = {3, 7, 8, 22, 55, 99}

Пересечением множеств A и B будет множество

C = {8, 55}

Графическое представление:



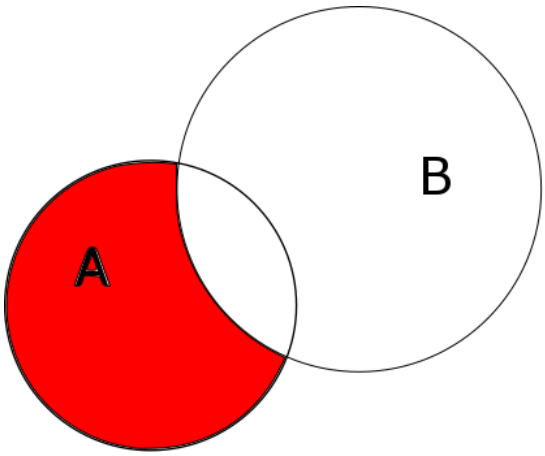
Разность множеств

Разностью множеств A и B (A – B) будет множество **C** в которое входят все элементы первого множества, не входящие во второе множество.

A = {1, 5, 8, 9, 32, 55, 78}
B = {3, 7, 8, 22, 55, 99}
C = {1, 5, 9, 32, 78},

A разностью B – A будет

D = {3, 7, 22, 99}



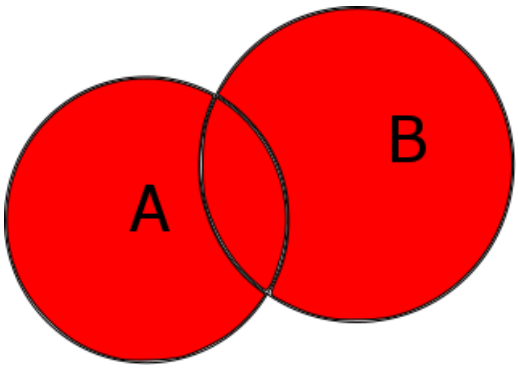
Объединение множеств

Объединением множеств A и B называется множество **C** содержащее все элементы множества **A и B.**

A = {1, 5, 8, 9, 32, 55, 78}
B = {3, 7, 8, 22, 55, 99}

Объединением множеств A и B будет множество

C = {1, 3, 5, 7, 8, 9, 22, 32, 55, 78, 99}



Операции с множествами, состоящими из сложных элементов (на примере пересечения)

Если множество состоит из сложных элементов, то в пересечение попадают элементы, у которых совпадают все атрибуты:

Картофель	Вода	Говядина	Горошек
Картофель	Бульон из говядины	Говядина	Капуста
Паста	Вода	Тофу	Горошек в стручках
Паста	Вода	Свинина	Лук

Рис	Куриный бульон	Баранина	Лук
Паста	Отвар из овощей	Тофу	Горошек в стручках
Картофель	Вода	Говядина	Горошек
Бобы	Вода	Свинина	Лук

Результат:

Картофель	Вода	Говядина	Горошек

В SQL подобная операция реализуется с помощью оператора INTERSECT. Однако эта операция, наряду с рядом других «теоретико-множественных» операций, реализована не во всех серверах и имеет существенное ограничение: таблицы, над которыми выполняется эта операция, должны иметь одинаковую структуру (как и в случае с UNION).

Операции соединения в SQL

Операция JOIN отличается от операций UNION, INTERSECT и EXCEPT тем, что не требует совпадения структуры элементов множеств, которыми она оперирует. Операция JOIN, для своего выполнения, требует (в наиболее распространенном случае) лишь совпадения значений некоторых столбцов из нескольких разнородных таблиц. В более общем случае для выполнения операции соединения требуется истинность предиката, который определяет соединение.

Например, имея множество элементов конструктора Lego и множество фруктов можно построить их соединение, основанное на совпадении цветов, т.е. отобрать элементы конструктора и фрукты, например, красного цвета.

С помощью соединения можно получать данные из двух или нескольких таблиц на основе ***логических связей*** между ними. Соединения указывают, как SQL Server должен использовать 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 первичный ключ из другой таблицы;
- указывается логический оператор (например, = или <>,) для сравнения значений столбцов.

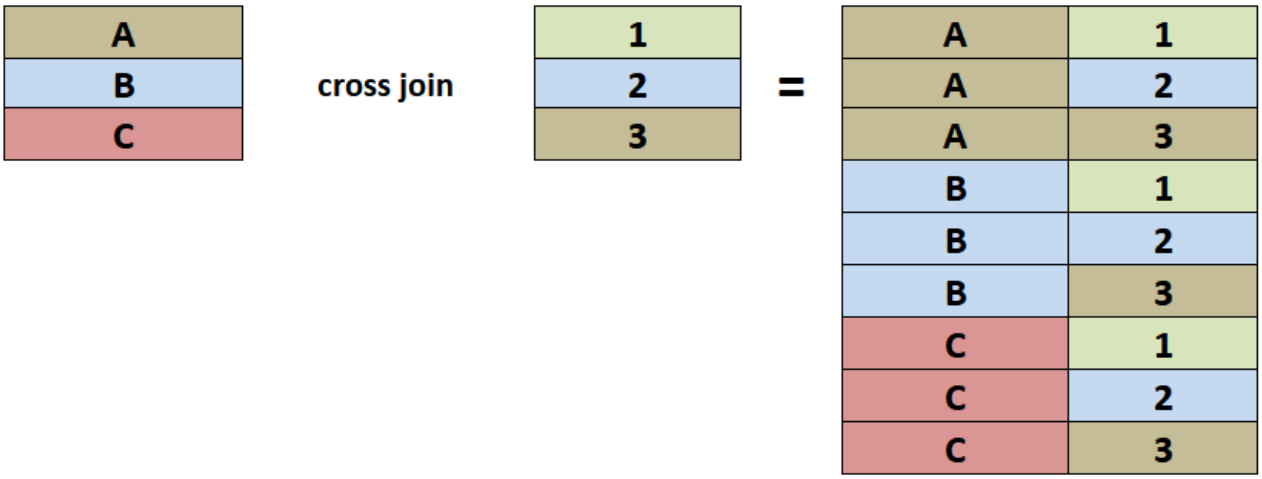
Ниже приведен упрощенный синтаксис соединения с использованием предложения FROM стандарта ISO:

FROM ***first_table*** *join_type* ***second_table*** [ON (*join_condition*)]

То, что условия соединения задаются в предложении FROM, помогает отделить их от условий поиска, которые могут быть заданы в предложении WHERE. Объединение рекомендуется задавать именно таким способом.

Аргумент join_type задает тип соединения: ***внутренний, внешний или перекрестный***.
Аргумент join_condition определяет предикат, который будет вычисляться для каждой пары соединяемых строк.

Перекрестное соединение (cross join)

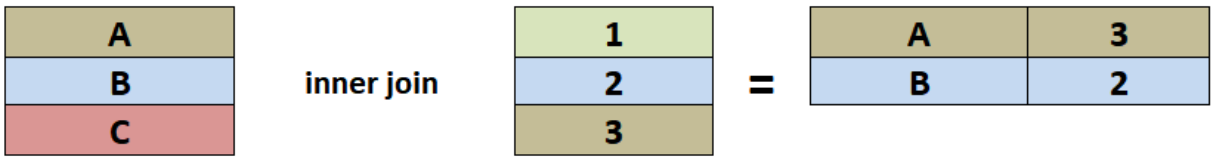


Перекрестное соединение содержит все возможные комбинации строк первой таблицы со строками второй таблицы. Такое соединение называется еще ***декартовым произведением***. Размер результирующего набора декартова произведения, вычисляется как произведение количества строк в первой таблице на количество строк во второй таблице.

При этом если одна из таблиц пустая (не содержит ни одной строки), то результат декартового произведения таблиц будет пустая таблица.

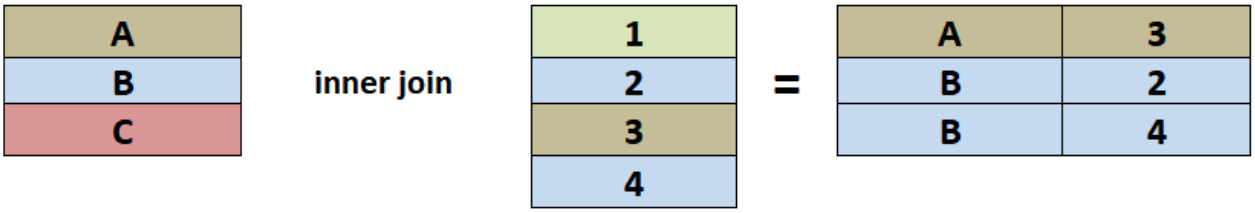
```
select
       s.CITY
       , u.CITY
from STUDENTS s
       cross join UNIVERSITIES u
```

Внутреннее соединения (inner join)



Для выполнения этого соединения SQL сервер логически объединяет каждую строку таблицы указанной слева от оператора JOIN с каждой строкой таблицы указанной справа (***декартово произведение***). Затем к результату применяется критерий, указанный в условии ON (вычисляется join condition), чтобы отобрать результирующие записи. Если для пары записей предикат истинен, то пара записей попадает в результирующую выборку.

В результирующей таблице записи из таблиц участвующих в соединении будут соединяться столько раз, сколько раз найдется совпадение:



Пример. Отобрать фамилии всех студентов, получивших оценку "хорошо", вместе с названиями предметов, по которым получена оценка.

```
select
       subjects.name
       , surname
       , mark
from exam_marks
       inner join students on students.id = exam_marks.student_id
       inner join subjects on subjects.id = exam_marks.subj_id and exam_marks.mark = 4
```

inner join является соединением «по умолчанию» и, поэтому, запись inner join эквивалентна просто join.

Таким образом, внутреннее соединение есть декартово соединение, к которому применена фильтрация.

Так как при выполнении внутреннего соединения результирующая выборка фильтруется, то количество записей в результирующей выборке внутреннего соединения может быть меньше количества записей в перекрестном соединении.

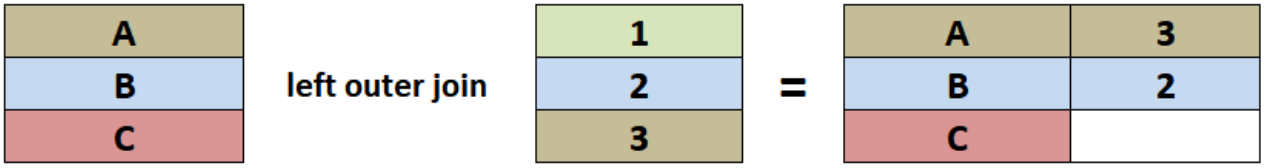
Следующие запросы SQL дают одинаковый результат:

```
select
       sb.name
       , st.surname
       , m.mark
from exam_marks m
       inner join students st on st.id = m.student_id
       inner join subjects sb on sb.id = m.subj_id and m.mark = 4;

select
       sb.name
       , st.surname
       , m.mark
from exam_marks m
       cross join students st
       cross join subjects sb
where st.id = m.student_id and sb.id = m.subj_id and m.mark = 4;

select
       sb.name
       , st.surname
       , m.mark
from exam_marks m, students st, subjects sb
where st.id = m.student_id and sb.id = m.subj_id and m.mark = 4;
```

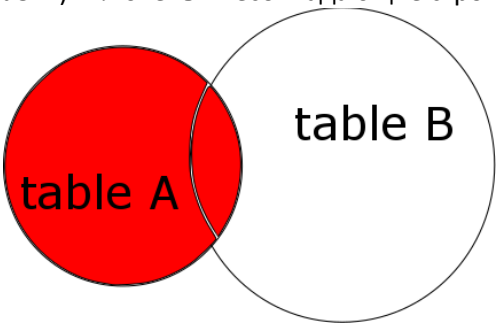
Внешнее соединение (outer join), разность

Внешнее соединение включает строки из левой и правой таблиц , указанных в соединении, даже если они не имеют связанных строк в объединенной таблице. Вы можете создать три варианта внешнего соединения, чтобы указать какие из несовпадающих строк, должны быть включены:

Левое внешнее соединение (left outer join)



Все строки из первой именованной таблицы («левая» таблица, которая отображается слева в разделе JOIN) включены. Несовпадающие строки в правой таблице не отображаются.



Пример. Отобрать фамилии всех студентов и оценки, которые они получили.

```
select
	s.surname
	,s.name
	,m.mark
from STUDENTS as s
inner join EXAM_MARKS m on s.id = m.student_id
```

В результате получим 120 строк. Однако в результирующую выборку не попадут студенты, которые еще не сдавали экзамены, и для которых еще нет оценок.

Получаем список студентов, которые еще не сдали экзамен

```
select * from STUDENTS where id not in (select student_id from EXAM_MARKS);
```

Выполняем проверку

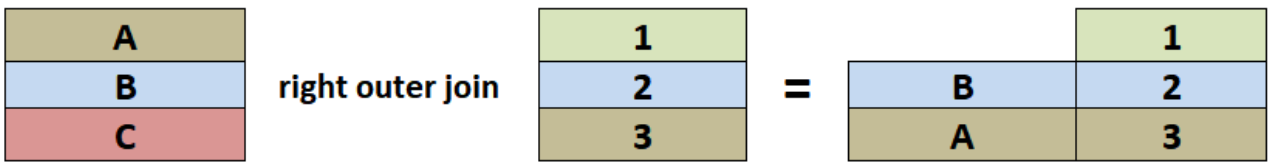
```
select
	s.surname
	,s.name
	,m.mark
from STUDENTS as s
	inner join EXAM_MARKS as m on s.id = m.student_id
where
	s.id not in (select student_id from EXAM_MARKS)
```

Для того, что бы получить полный список студентов применяем внешнее соединение:

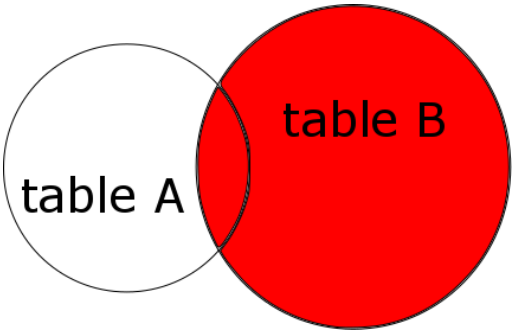
```
select
	s.surname
	,s.name
	,m.mark
from STUDENTS as s
left outer join EXAM_MARKS as m on s.id = m.student_id
```

Результат – 123 строки.

Правое внешнее соединение (right outer join)



Все строки во второй названной таблице («правая» таблица, которая появляется справа в разделе JOIN) включены. Несовпадающие строки в левой таблице не включаются.

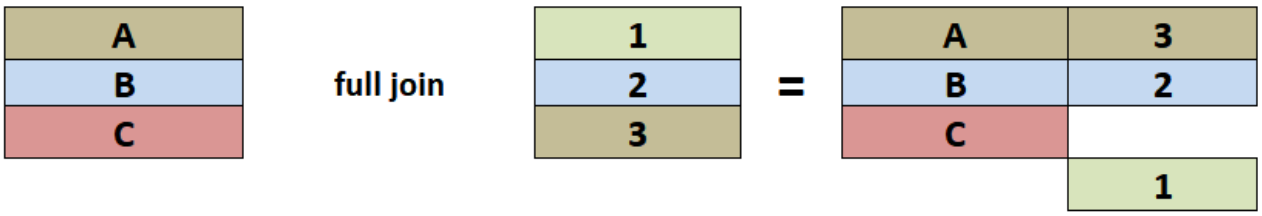


Пример. Запрос

```
select
       s.surname
       ,s.name
       ,m.mark
from STUDENTS as s
right outer join EXAM_MARKS as m on s.id = m.student_id
```

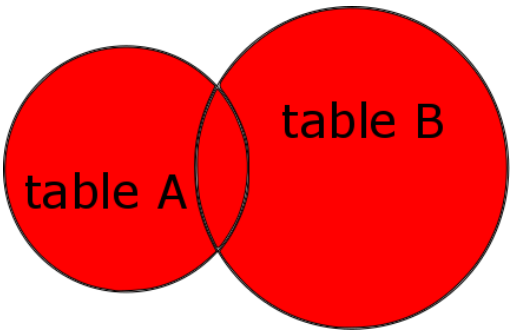
выберет только тех студентов, у которых есть оценки.

Полное внешнее соединение (full outer join)



Все строки во всех соединенных таблицах включены, независимо от того, совпадают они или нет.

full outer join не является ни «левым» ни «правым» соединением. В него включены все строки как левой, так и правой таблицы. Когда отсутствуют строки, выполняющие условие с левой стороны JOIN, то в наборе результата «справа» будут присутствовать значения NULL. Наоборот, когда отсутствуют строки выполняющие условие с правой стороны JOIN, то значения NULL будут присутствовать в результирующем наборе «слева».




```
select
       *
from STUDENTS as s full outer join UNIVERSITIES as u on s.univ_id = u.id
```

Пример. Найти студентов, которые учатся в городе, в котором не проживают

```
select
       s.id
       ,s.surname
       ,s.name
       ,s.city
       ,s.univ_id
from STUDENTS as s
where s.city <> (select city from UNIVERSITIES where id = s.univ_id);
```

А теперь при помощи соединения добавив название университета и названия города, в котором данный университет расположен:

```
select
       s.id
       ,s.surname
       ,s.name
       ,s.city
       ,s.univ_id
       ,u1.id
       ,u1.name
       ,u1.city
from STUDENTS as s inner join UNIVERSITIES as u1 on
       (u1.id = s.univ_id and u1.city <> s.city);
```

Соединение с вложенными запросами

Вложенным запросом называется запрос, помещаемый в инструкцию SELECT, INSERT, UPDATE или DELETE или в другой вложенный запрос. Подзапрос может быть использован везде, где разрешены выражения.

Примеры.

Показать список студентов (фамилию, имя, курс и название университета в котором студент обучается):

```
select
       surname
       ,name
       ,course
       ,(select name from UNIVERSITIES where id = STUDENTS.univ_id)
from STUDENTS;
```

С использованием соединения:

```
select
       STUDENTS.surname
       ,STUDENTS.name
       ,STUDENTS.course
       ,UNIVERSITIES.name
from STUDENTS left outer join UNIVERSITIES on STUDENTS.univ_id = UNIVERSITIES.id;
```

Найти идентификаторы, фамилии и стипендии студентов, получающих стипендию выше средней на курсе, на котором они учатся.

```
select
	stud.*
from students as stud where stipend > (
	select avg(stipend) from students where stud.course=course)
```

А показать среднее значение стипендии?

```
select
	s.*
	, s_avg.stipend
from students s,
	(select
		course
		, avg(stipend) as stipend
	 from students group by course) s_avg
where s.course=s_avg.course and s.stipend > s_avg.stipend
```

Эквивалентное выражение с использованием нового синтаксиса оператора JOIN

```
select
	s.*
	, s_avg.stipend
from students as s
	inner join (
		select
			course
			, avg(stipend) as stipend
		from students group by course) as s_avg
		on s.course=s_avg.course and s.stipend > s_avg.stipend
```

Показать всех студентов и их максимальные оценки:

```
select
	s.name
	,s.surname
	,t.mmark
from STUDENTS as s left outer join (
				select student_id,max(mark) as mmark
				from EXAM_MARKS
				group by STUDENT_ID
			) t on s.id = t.student_id
```

Theta (θ) Join

Тета-соединение объединяет кортежи из разных отношений, если они удовлетворяют условию тета. Условие соединения обозначается символом θ.

Equi join

Когда Тета- соединение использует только оператор сравнения на равенство, оно называется **equi join**.

Natural Join

Относительно удобно и не поддерживается всеми серверами. При **natural join** сравниваются столбцы таблиц и для них выполняется **equi join**, которое может быть внутренним или внешним. Чтобы использовать это объединение, вы должны следовать определенному соглашению об именах (имена столбцов в разных таблицах должны совпадать).

Self join

Это обычное соединение, но в нем таблица соединяется сама с собой. Пример «электронная сваха»:

```
select
       sm.gender
       ,sm.name
       ,sm.surname
       ,sm.birthday
       ,sf.gender
       ,sf.name
       ,sf.surname
       ,sf.birthday
from STUDENTS as sm inner join STUDENTS as sf on
       year(sf.birthday) < year(sm.birthday)
       and sf.gender = 'f' and sm.gender = 'm'
       and sf.univ_id = sm.univ_id;
```

Полезные ссылки:

http://datareview.info/article/8-sposobov-obedineniya-join-tablic-v-sql-chast-1/

http://datareview.info/article/8-sposobov-obedineniya-join-tablic-v-sql-chast-2/

http://www.developingthefuture.net/types-of-joins-in-sql/

http://www.realcoding.net/articles/rabota-s-mnozhestvami-v-transact-sql.html

https://blog.codinghorror.com/a-visual-explanation-of-sql-joins/

https://www.codeproject.com/articles/33052/visual-representation-of-sql-joins

https://www.w3resource.com/sql/joins/sql-joins.php