



ОНЛАЙН-ОБРАЗОВАНИЕ

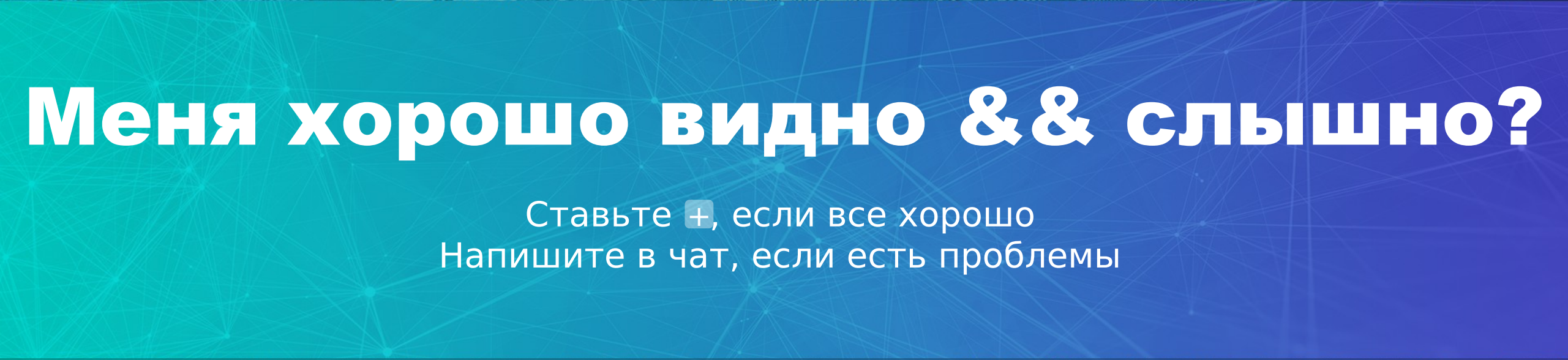
# Онлайн-образование



**Не забыть включить запись!**








# Меня хорошо видно && слышно?

Ставьте ☐+, если все хорошо  
Напишите в чат, если есть проблемы







# Различные виды соединений. Понимание и применение

Курочкин Константин  
«Medindex»



# Правила вебинара



Активно участвуем



Задаем вопрос в чат



Off-topic обсуждаем в Slack `#postgres-dba` или `#general`



Вопросы вижу в чате, могу ответить не сразу

# Маршрут вебинара

Рассмотрим соединения в общем виде



Рассмотрим подробнее различные варианты соединений



Попрактикуемся



Порефлекслируем 😊

# Цели вебинара | После занятия вы сможете

1

Применять различные типы соединений в своих запросах

2

Определять, какой тип соединения вам нужен в зависимости от ситуации

3

Применять операции над множествами

# Смысл | Зачем вам это уметь

1

Для написания запросов с применением различных соединений

2

Для правильного выбора соединения в зависимости от цели

3

Для написания запросов, где необходимо оперировать множествами



# Слайд с заданием

**1** Данные команды эквиваленты? `Inner join` и `join`

**2** Чем отличается `left join` и `right join`?

Если у нас в обеих таблицах, состоящих из одной колонки, по 10 строк с одинаковыми значениями в обеих таблицах, то каково будет количество строк при их прямом соединении?

**3**

# О соединении. Nested loop

```
select A.City_name, Godzilla_attacks  
from A  
JOIN B on A.City_name = B.City_name
```

Table A		Table B		Working Set			
City_name	Country	City_name	Godzilla_attacks	A.City_name	A.Country	B.City_name	B.Godzilla_attacks
Tokyo	Japan	Fukuoka	3				
New York	USA	Nagoya	2				
Fukuoka	Japan	New York	3				
Shanghai	China	Tokai	3				
		Tokyo	13				
		Yokohama	2				

Nested Loop (cost=0.00..6685.01 rows=100000 width=396) (actual time=0.019..121.259 rows=100000 loops=1)

Join Filter: (w.w\_id = s.s\_w\_id)

-> Seq Scan on warehouse1 w (cost=0.00..2.01 rows=1 width=92) (actual time=0.010..0.011 rows=1 loops=1)

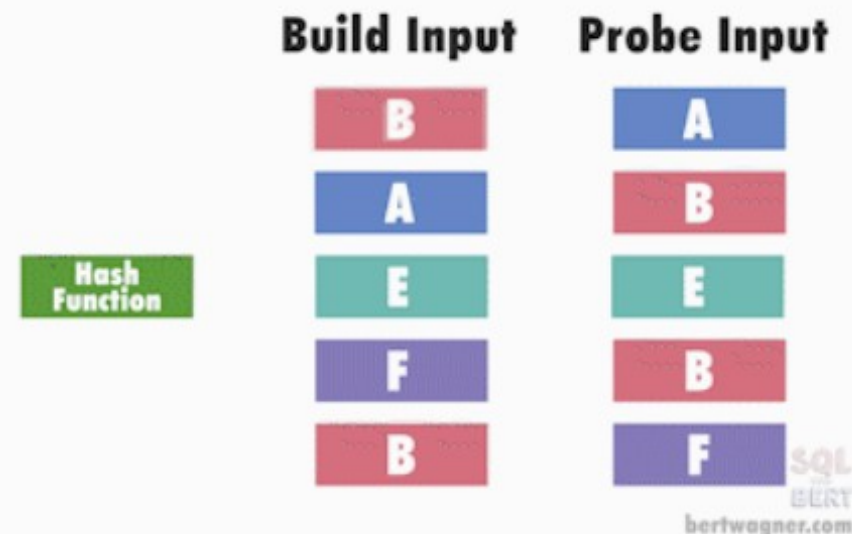
-> Seq Scan on stock1 s (cost=0.00..5433.00 rows=100000 width=304) (actual time=0.004..89.975 rows=100000 loops=1)



# О соединении. Hash join

Hash Join (cost=14.82..31507.11 rows=1000057 width=461)  
(actual time=5.140..957.126 rows=1000000 loops=1)  
Hash Cond: (pa.bid = pb.bid)  
-> Seq Scan on pgbench\_accounts pa (cost=0.00..27754.57  
rows=1000057 width=97)(actual time=1.083..699.670  
rows=1000000 loops=1)  
-> Hash (cost=14.70..14.70 rows=10 width=364)(actual  
time=3.975..3.976 rows=10 loops=1)  
Buckets: 1024 Batches: 1 Memory Usage: 9kB  
-> Index Scan using pgbench\_branches\_pkey on  
pgbench\_branches pb (cost=0.14..14.70 rows=10 width=364)  
(actual time=0.702..3.945 rows=10 loops=1)

## Hash Match Join



# О соединении. Merge join

-> Merge Join (cost=0.29..9.19 rows=100 width=0) (actual time=0.017..0.053 rows=100 loops=1)

Merge Cond: (pt.tid = pt2.tid)

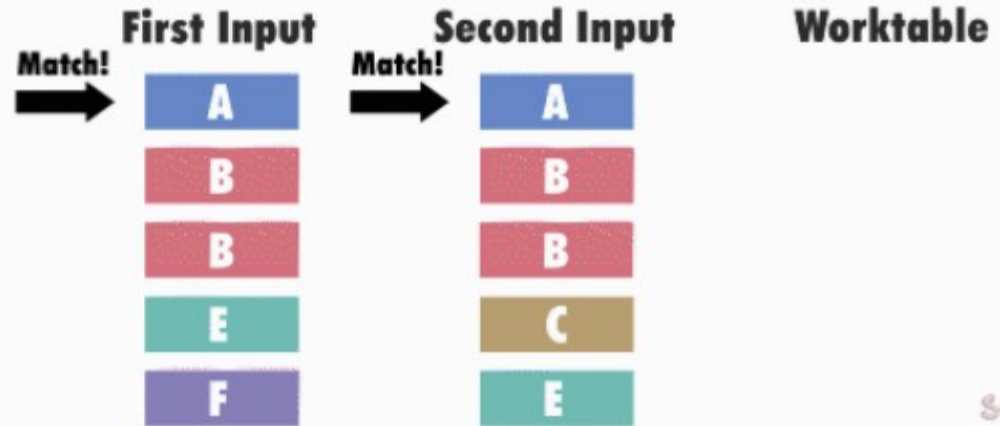
-> Index Only Scan using pgbench\_tellers\_pkey on pgbench\_tellers pt (cost=0.14..3.84 rows=100 width=4) (actual time=0.005..0.013 rows=100 loops=1)

Heap Fetches: 0

-> Index Only Scan using pgbench\_tellers\_pkey on pgbench\_tellers pt2 (cost=0.14..3.84 rows=100 width=4) (actual time=0.010..0.018 rows=100 loops=1)

Heap Fetches: 0

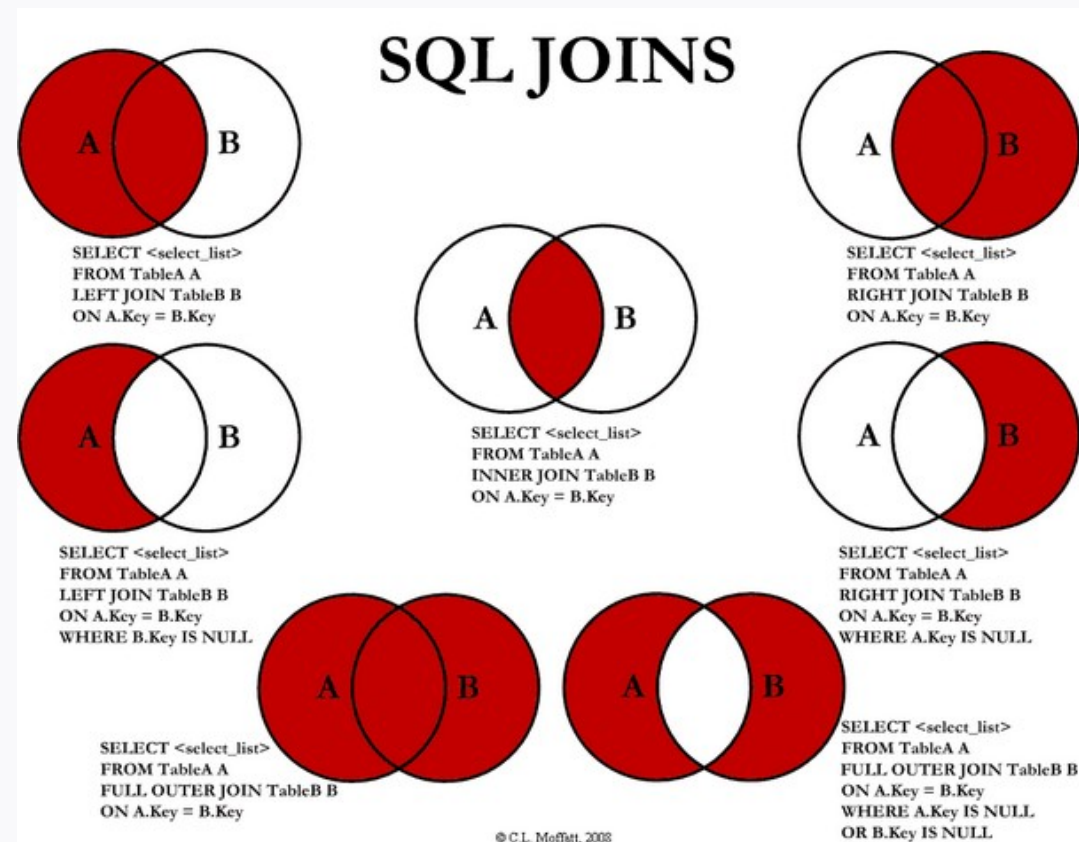
## Merge Join





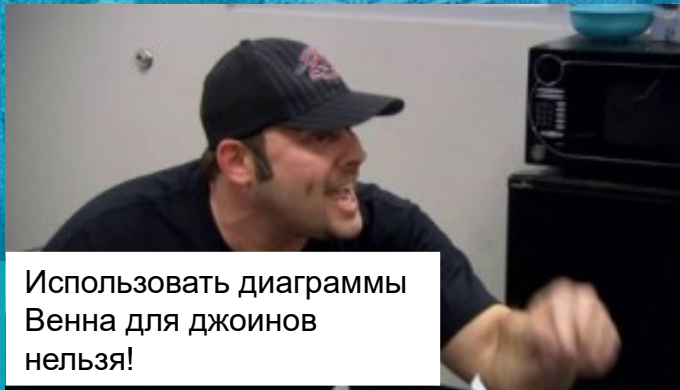
# О соединении

```
SELECT <поля>  
FROM <таблица 1>  
[INNER]  
{LEFT | RIGHT | FULL } [OUTER] JOIN <таблица 2>  
[ON <предикат>]  
[WHERE <предикат>]
```

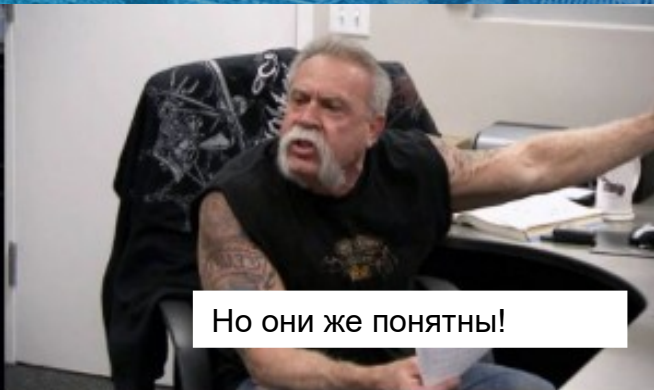


Слова INNER и OUTER необязательны во всех формах. По умолчанию подразумевается INNER (внутреннее соединение), а при указании LEFT, RIGHT и FULL — внешнее соединение.

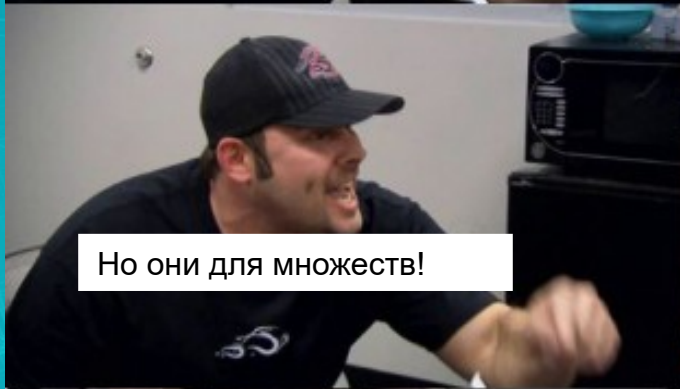




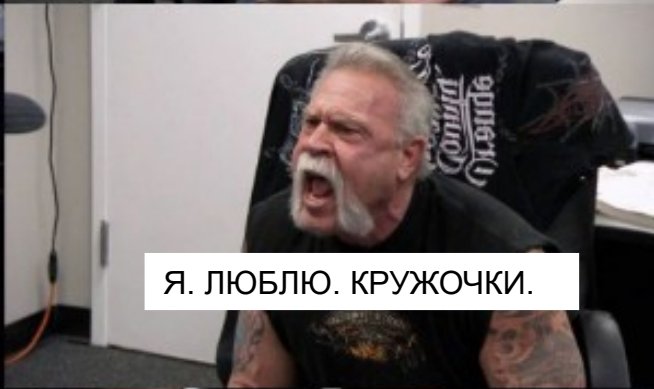
Использовать диаграммы  
Венна для джоинов  
нельзя!



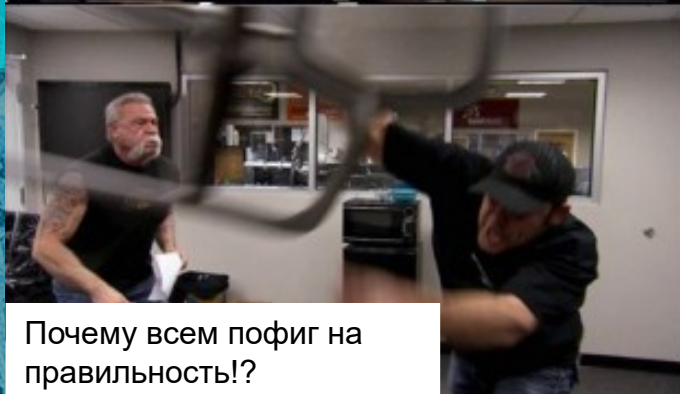
Но они же понятны!



Но они для множеств!



Я. ЛЮБЛЮ. КРУЖОЧКИ.



Почему всем пофиг на  
правильность!?



И мы первые по запросам  
в гугле!

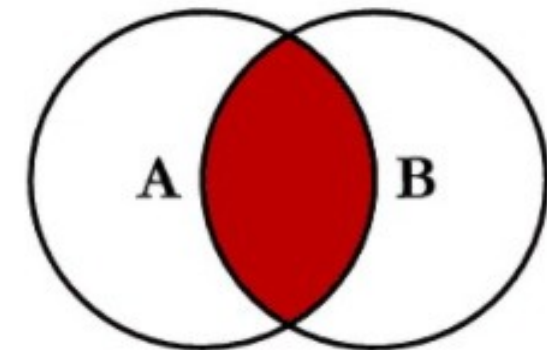


# Прямое соединение



- Если ключи в двух таблицах совпадают, то будет возвращена строка, содержащая колонки из обеих таблиц
- Слово INNER может быть опущено
- Можно писать соединения иначе:

```
SELECT *  
FROM TableA A, TableB B  
WHERE A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```





<https://www.db-fiddle.com/f/kSq5tqj5LyzFMinW3opDJ3/2>

<https://www.db-fiddle.com/f/cy9vkGYgEY7KQhenqRdS9i/11>

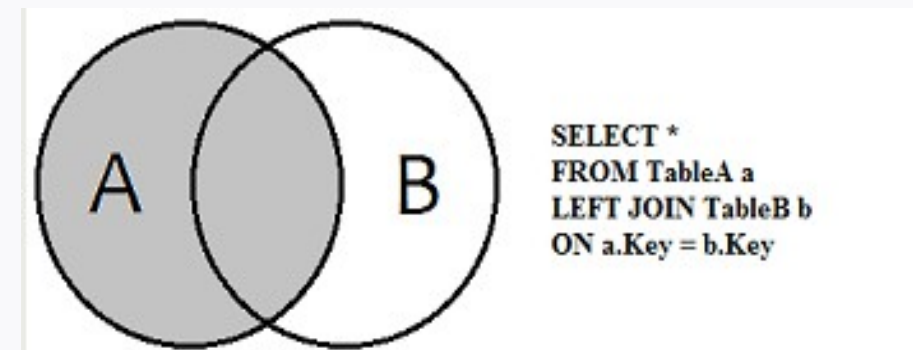
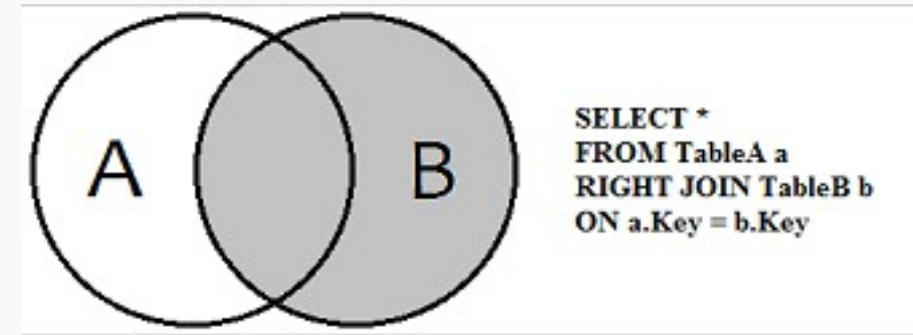




# Левостороннее (правостороннее) соединение



- Если ключи в двух таблицах совпадают, то будет возвращена строка, содержащая колонки из обеих таблиц
- В случае left join: Если для строки из левой таблицы не будет найдено строк с тем же ключом в правой таблице, то вернётся строка из левой таблицы, но в том числе с колонками правой таблицы, в которых будет стоять null
- В случае right join: Если для строки из правой таблицы не будет найдено строк с тем же ключом в левой таблице, то вернётся строка из правой таблицы, но в том числе с колонками левой таблицы, в которых будет стоять null
- Слово OUTER может быть опущено







<https://www.db-fiddle.com/f/cy9vkGYgEY7KQhenqRdS9i/10>

<https://www.db-fiddle.com/f/3oBFfBzb8bpRxbcMEoimzF/4>

<https://www.db-fiddle.com/f/mtaFpNSAN94Yq3hXVYmL4q/3>

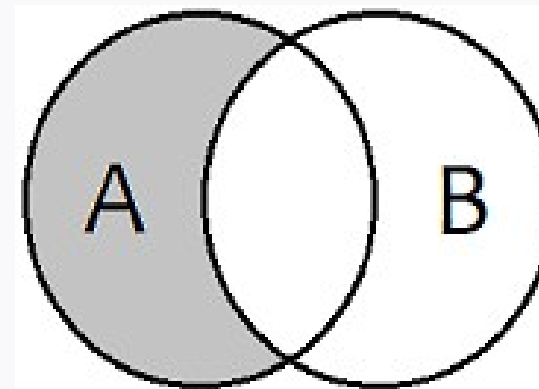




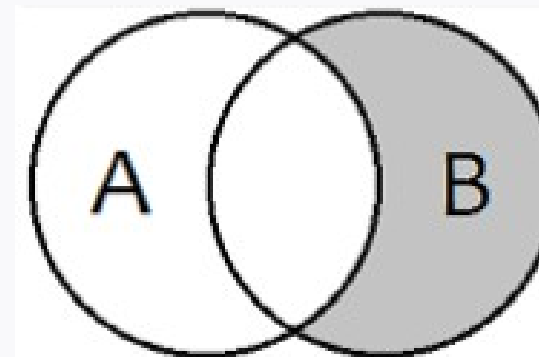
# Левостороннее (правостороннее) соединение



- Если хотим вернуть только те строки, для которых соответствия во второй таблице не нашлось, то необходимо добавить условие:
- В случае left join: *WHERE b.Key is null* (ключ соединения в правой таблице пуст)
- В случае right join: *WHERE a.Key is null* (ключ соединения в левой таблице пуст)



```
SELECT *  
FROM TableA a  
LEFT JOIN TableB b  
ON a.Key = b.Key  
WHERE b.Key IS NULL
```



```
SELECT *  
FROM TableA a  
RIGHT JOIN TableB b  
ON a.Key = b.Key  
WHERE a.Key IS NULL
```





<https://www.db-fiddle.com/f/cy9vkGYgEY7KQhenqRdS9i/6>

<https://www.db-fiddle.com/f/3oBFbBzb8bpRxbcMEoimzF/2>

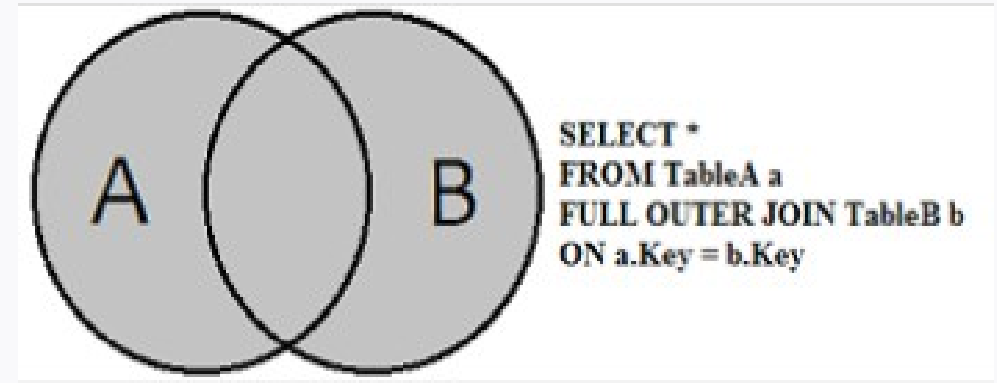




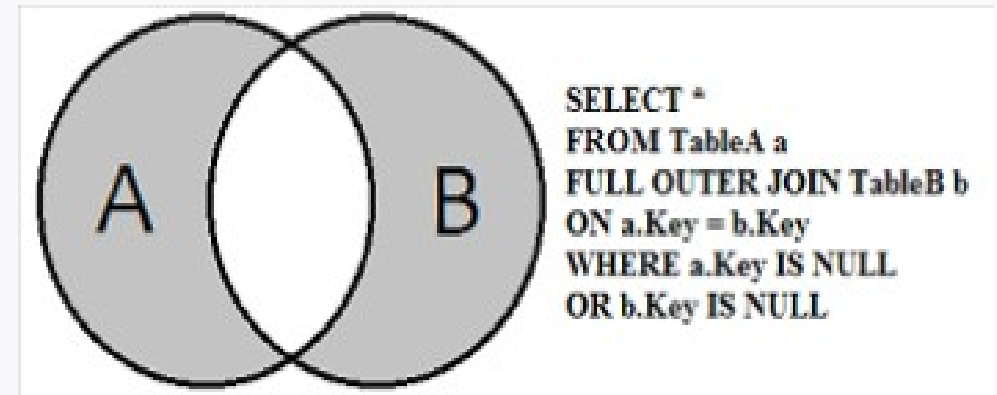
# Полное соединение



- Если ключи в двух таблицах совпадают, то будет возвращена строка, содержащая колонки из обеих таблиц
- Если соответствия не было найдено, то колонки из противоположной таблицы будут пусты



- Можно исключить те строки, для которых было найдено соответствие
- Таким образом будут найдены только те строки, для которых нет соответствия в противоположной таблице







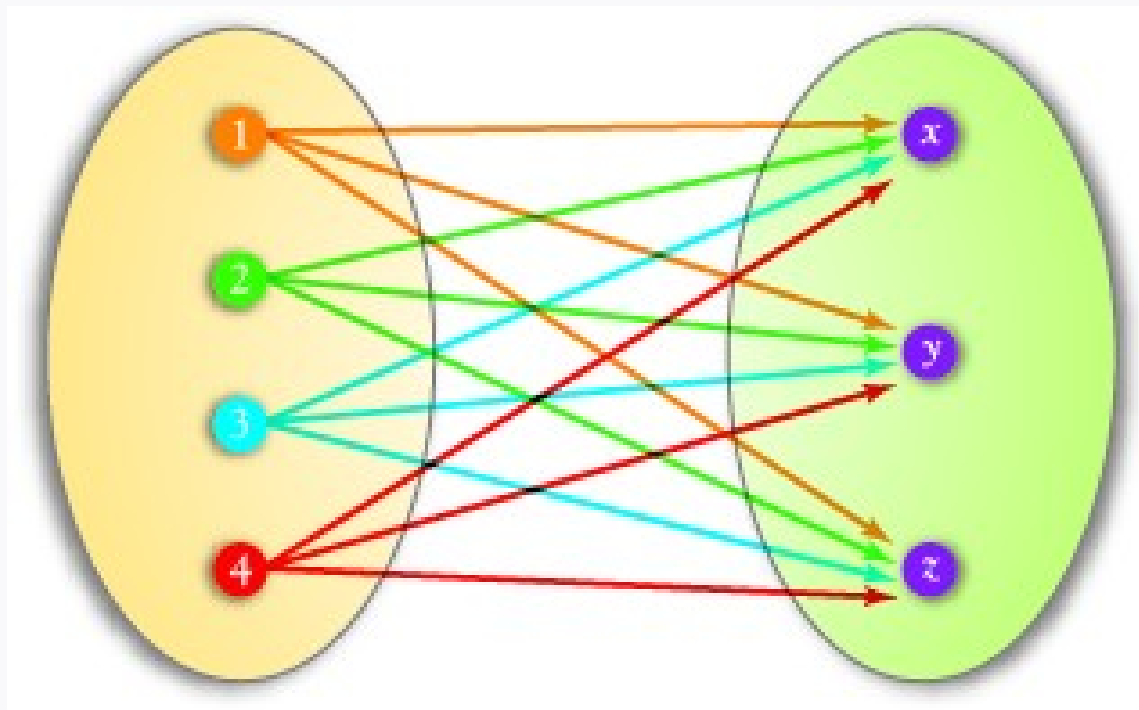
<https://www.db-fiddle.com/f/cy9vkGYgEY7KQhenqRdS9i/7>

<https://www.db-fiddle.com/f/cy9vkGYgEY7KQhenqRdS9i/8>





# Кросс соединение







<https://www.db-fiddle.com/f/95h5i1z93q5bZ5Xwpcc1aP/2>

<https://www.db-fiddle.com/f/95h5i1z93q5bZ5Xwpcc1aP/3>







<https://forms.gle/Bs7XGt4r9cf6CnL48>



# Слайд с заданием

**1** Данные команды эквиваленты? `Inner join` и `join`

**2** Чем отличается `left join` и `right join`?

**3** Если у нас в обеих таблицах, состоящих из одной колонки, по 10 строк с одинаковыми значениями в обеих таблицах, то каково будет количество строк при их прямом соединении?



# Lateral join

```
SELECT <target list>  
FROM <table>  
[INNER]  
{{LEFT | RIGHT | FULL} [OUTER]} JOIN LATERAL  
(<subquery using table.column>) as foo on true;
```

- Можно использовать для возврата первых N строк в рамках группы
- Для джойна с функциями, которые возвращают несколько строк (unnest)

<https://stackoverflow.com/questions/28550679/what-is-the-difference-between-lateral-and-a-subquery-in-postgresql>





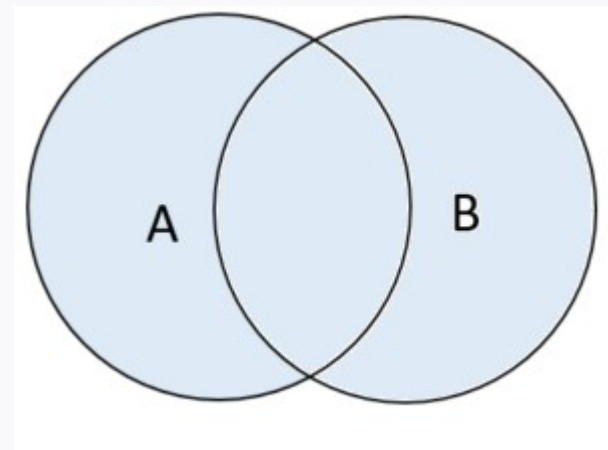
<https://www.db-fiddle.com/f/3LT2kbwWZej9HAVokbdSH8/0>





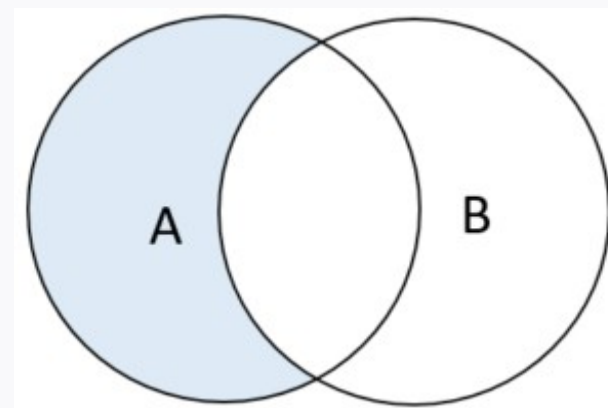
# Операции над множествами

- Объединить
- Можно объединять результаты двух запросов (таблиц) и более
- На выходе будут возвращены строки из всех множеств
- Можно исключать дубликаты (убрав ключевое слово ALL)



```
SELECT <target  
list>  
FROM <table>  
UNION [all]  
SELECT <target  
list>  
FROM <table2>  
...
```

- Исключить
- Можно исключить одно множество из другого
- На выходе будут возвращены строки из верхнего множества, для которых не нашлось соответствия в другом множестве
- Можно исключать дубликаты (убрав ключевое слово ALL)

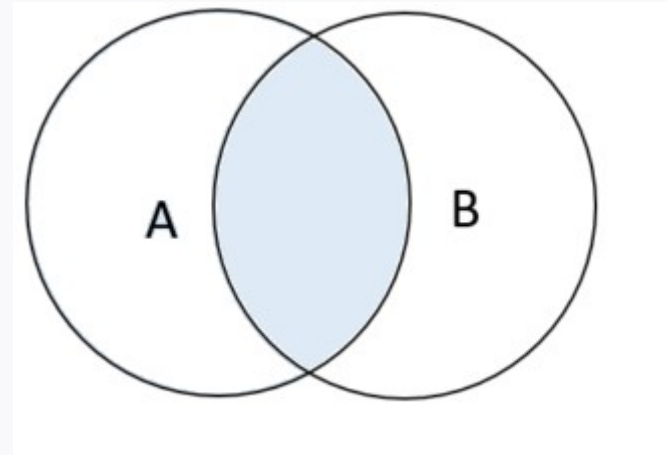


```
SELECT <target  
list>  
FROM <table>  
EXCEPT [all]  
SELECT <target  
list>  
FROM <table2>  
...
```



# Операции над множествами

- Пересечь
- Можно пересечь результаты двух запросов (таблиц) и более
- На выходе будут только те строки, которые полностью совпали
- Можно исключать дубликаты (убрав ключевое слово ALL)



```
SELECT <target  
list>  
FROM <table>  
INTERSECT [all]  
SELECT <target  
list>  
FROM <table2>  
...
```





<https://www.db-fiddle.com/f/qPC43WyZwSJT83EbgLApz7/0>

<https://www.db-fiddle.com/f/qPC43WyZwSJT83EbgLApz7/2>

<https://www.db-fiddle.com/f/qPC43WyZwSJT83EbgLApz7/3>





# Рефлексия



Отметьте самый не раскрытый, по вашему мнению пункт



Какой вариант объяснения джоинов, по вашему мнению, более доступный для понимания?






Заполните, пожалуйста,  
опрос о занятии по ссылке в чате  
<https://otus.ru/polls/51740/>







**До новых встреч!**  
**Приходите на следующие занятия**

**Курочкин Константин**  
«Medindex»