

PostgreSQL для начинающих

#3: Сложные SELECT

Кирилл Боровиков / Компания «Тензор», технический директор / explain.tensor.ru, sbis.ru

SELECT – ЭТО СЛОЖНО...

```
[ WITH [ RECURSIVE ] запрос_WITH [ , ... ] ]  
SELECT [ ALL | DISTINCT [ ON ( выражение [ , ... ] ) ] ]  
    [ * | выражение [ [ AS ] имя_результата [ , ... ] ]  
    [ FROM элемент_FROM [ , ... ] ]  
    [ WHERE условие ]  
    [ GROUP BY [ ALL | DISTINCT ] элемент_группирования [ , ... ] ]  
    [ HAVING условие ]  
    [ WINDOW имя_окна AS ( определение_окна ) [ , ... ] ]  
    [ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] выборка ]  
    [ ORDER BY выражение [ ASC | DESC | USING оператор ] [ NULLS { FIRST | LAST } ] [ , ... ] ]  
    [ LIMIT { число | ALL } ]  
    [ OFFSET начало [ ROW | ROWS ] ]  
    [ FETCH { FIRST | NEXT } [ число ] { ROW | ROWS } { ONLY | WITH TIES } ]  
    [ FOR { UPDATE | NO KEY UPDATE | SHARE | KEY SHARE } [ OF имя_таблицы [ , ... ] ] [ NOWAIT | SKIP LOCKED ] [ ... ] ]
```

<https://postgrespro.ru/docs/postgresql/15/sql-select>

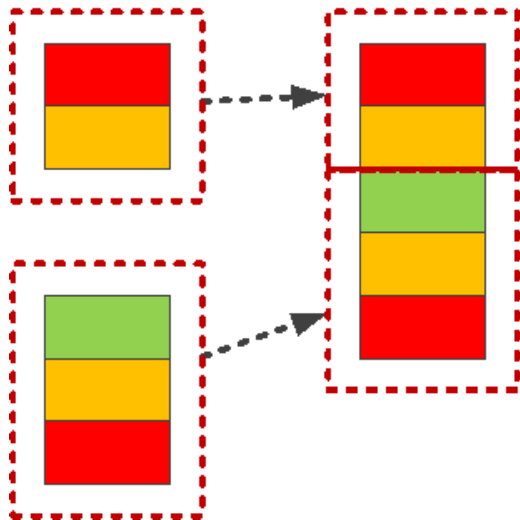
<https://postgrespro.ru/docs/postgresql/15/queries>

UNION

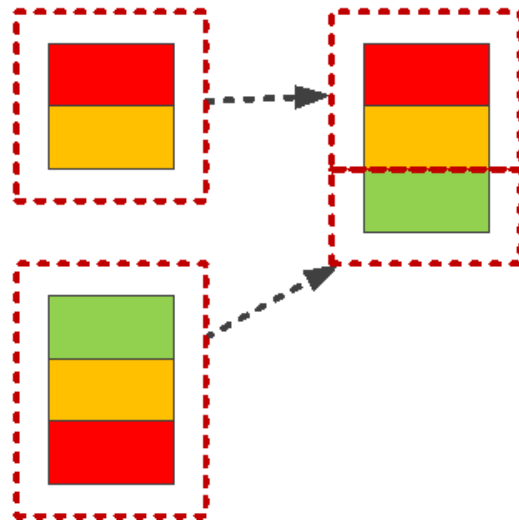
onepamop_SELECT UNION [ALL | DISTINCT] *onepamop_SELECT*

<https://postgrespro.ru/docs/postgresql/15/sql-select#SQL-UNION>

UNION ALL



UNION



UNION

VALUES

(1, 2)
, (1, 2)

UNION

VALUES

(3, 4)
, (1, 2);

column1	column2
integer	integer
1	2
3	4

VALUES

(1, 2)
, (1, 2)

UNION ALL

VALUES

(3, 4)
, (1, 2);

column1	column2
integer	integer
1	2
1	2
3	4
1	2

X UNION Y == X UNION DISTINCT Y == DISTINCT (X UNION ALL Y)

UNION

VALUES

(1, 2)

UNION

VALUES

(3);

ERROR: each UNION query must have the **same number of columns**

VALUES

(1, 2)

UNION

VALUES

(3, NULL);

ERROR: UNION **types** integer and text cannot be matched

VALUES

(1, 2)

UNION

VALUES

(3, NULL::integer)

, (3, NULL::integer);

column1	column2
integer	integer
1	2
3	

-- все NULL одинаковы!

INTERSECT | EXCEPT

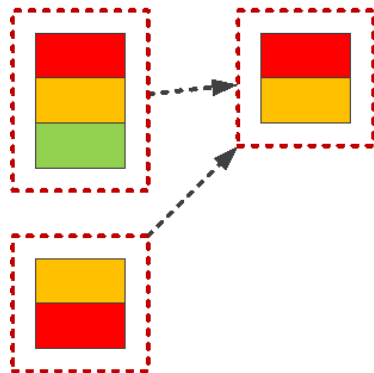
onepamop_SELECT INTERSECT [ALL | DISTINCT] *onepamop_SELECT*

<https://postgrespro.ru/docs/postgresql/15/sql-select#SQL-INTERSECT>

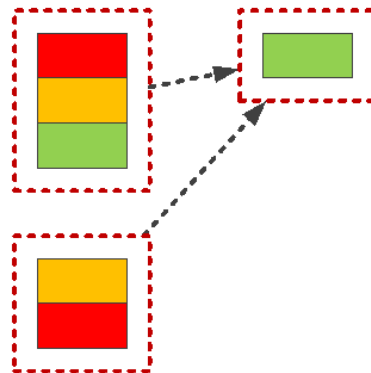
onepamop_SELECT EXCEPT [ALL | DISTINCT] *onepamop_SELECT*

<https://postgrespro.ru/docs/postgresql/15/sql-select#SQL-EXCEPT>

INTERSECT



EXCEPT



INTERSECT | EXCEPT

INTERSECT ALL == $\min(m, n)$

EXCEPT ALL == $\max(m - n, 0)$

UNION -> + -> low

EXCEPT -> - -> low

INTERSECT -> * -> high

A UNION B INTERSECT C EXCEPT D

A + B * C - D

A UNION (B INTERSECT C) EXCEPT D

WITH (CTE)

```
[ WITH [ RECURSIVE ] запрос_WITH [, ...] ]  
имя_CTE [ (имя_столбца, ...) ] AS ( -- Common Table Expression  
    { SELECT | TABLE | VALUES |  
      { INSERT | UPDATE | DELETE } ... RETURNING ...  
    } |  
    {  
        нерекурсивная_часть  
    UNION [ ALL | DISTINCT ]  
        рекурсивная_часть  
    }  
)
```

<https://postgrespro.ru/docs/postgresql/15/sql-select#SQL-WITH>

WITH (CTE)

```
WITH v AS (  
    VALUES  
        (1, 2)  
)  
TABLE v  
UNION ALL  
TABLE v;
```

column1	column2
integer	integer
1	2
1	2

```
WITH v(x, y) AS (  
    SELECT  
        1 a  
        , 2 b  
)  
TABLE v  
UNION ALL  
TABLE v;
```

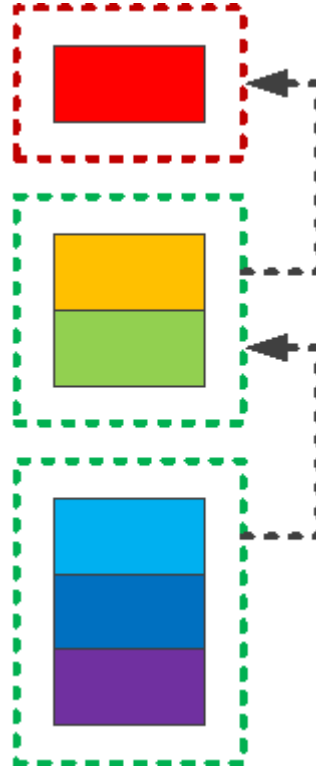
x	y
integer	integer
1	2
1	2

WITH RECURSIVE

```
WITH RECURSIVE fib(i, a, b) AS (  
    VALUES(0, 0, 1) -- затравка  
    UNION ALL  
    SELECT          -- шаг рекурсии  
        i + 1  
        , greatest(a, b)  
        , a + b  
    FROM  
        fib          -- обращение к себе  
    WHERE  
        i < 10       -- условие продолжения  
)  
TABLE fib;
```

i	a	b
integer	integer	integer
0	0	1
1	1	1
2	1	2
3	2	3
4	3	5
5	5	8
6	8	13
7	13	21
8	21	34
9	34	55
10	55	89

WITH RECURSIVE



WITH RECURSIVE

```
WITH RECURSIVE exp(i, n) AS (  
    VALUES(0, 1)  
    UNION ALL  
    SELECT  
        i + 1  
        , unnest(ARRAY[n * 2, n * 2])  
    FROM  
        exp  
    WHERE  
        i < 2  
)  
TABLE exp;
```



i	n
integer	integer
0	1
1	2
1	2
2	4
2	4
2	4
2	4

WINDOW / OVER

WINDOW *имя_окна* **AS** (*определение_окна*) [, ...]

[*имя_существующего_окна*]

[**PARTITION BY** *выражение* [, ...]]

[**ORDER BY** *выражение* [**ASC** | **DESC** | **USING оператор**] [**NULLS** { **FIRST** | **LAST** }] [, ...]]

[*предложение_рамки*]

рамка:

{ **RANGE** | **ROWS** | **GROUPS** } *начало_рамки* [*исключение_рамки*]

{ **RANGE** | **ROWS** | **GROUPS** } **BETWEEN** *начало_рамки* **AND** *конец_рамки* [*исключение_рамки*]

начало/конец	исключение
--------------	------------

UNBOUNDED PRECEDING	EXCLUDE CURRENT ROW
----------------------------	----------------------------

<i>смещение</i> PRECEDING	EXCLUDE GROUP
----------------------------------	----------------------

CURRENT ROW	EXCLUDE TIES
--------------------	---------------------

<i>смещение</i> FOLLOWING	EXCLUDE NO OTHERS
----------------------------------	--------------------------

UNBOUNDED FOLLOWING	
----------------------------	--

<https://postgrespro.ru/docs/postgresql/15/sql-select#SQL-WINDOW>

WINDOW / OVER

```
WITH RECURSIVE sum(i, s) AS (  
    VALUES(1, 1)  
    UNION ALL  
    SELECT  
        i + 1  
        , s + (i + 1)  
    FROM  
        sum  
    WHERE  
        i < 10  
)  
TABLE sum;
```

```
SELECT  
    i  
    , sum(i) OVER(ORDER BY i) s -- "OKHO"  
FROM  
    generate_series(1, 10) i;
```

$$1 = 1 \cdot$$

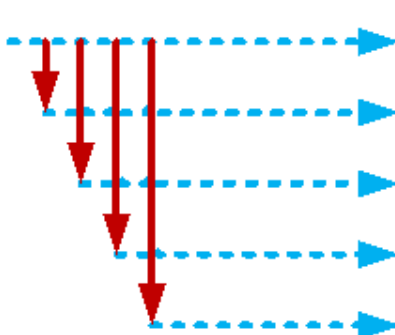
$$3 = 1 + 2 \therefore$$

$$6 = 1 + 2 + 3 \therefore \therefore$$

$$10 = 1 + 2 + 3 + 4 \therefore \therefore \therefore$$

WINDOW / OVER

i	ORDER BY	agg	sum(i)
1			1
2			3
3			6
4			10
5			15
6			21
7			28
8			36
9			45
10			55



OVER(ORDER BY i)

==

OVER(ORDER BY i

ROWS BETWEEN

UNBOUNDED PRECEDING AND

CURRENT ROW)

WINDOW / OVER

```
SELECT
```

```
  i
```

```
, sum(i) OVER() s
```

```
FROM
```

```
  generate_series(1, 10) i;
```

i integer	s bigint
1	55
2	55
3	55
4	55
5	55
6	55
7	55
8	55
9	55
10	55

```
OVER()
```

```
==
```

```
OVER(
```

```
  ROWS BETWEEN
```

```
    UNBOUNDED PRECEDING AND
```

```
    UNBOUNDED FOLLOWING)
```

Математик, физик, инженер и программист доказывают одну и ту же теорему: все нечетные числа больше двух — простые.

Математик: "3 — простое, 5 — простое, 7 — простое, 9 — не простое. Это контрпример, значит, теорема неверна".

Физик, с карандашом и бумагой: "3, 5 и 7 — простые, 9 — ошибка эксперимента, 11 — простое"

Инженер, взяв в руки калькулятор: "3 — простое, 5 — простое, 7 — простое, 9 — приблизительно простое, 11 — тоже простое"

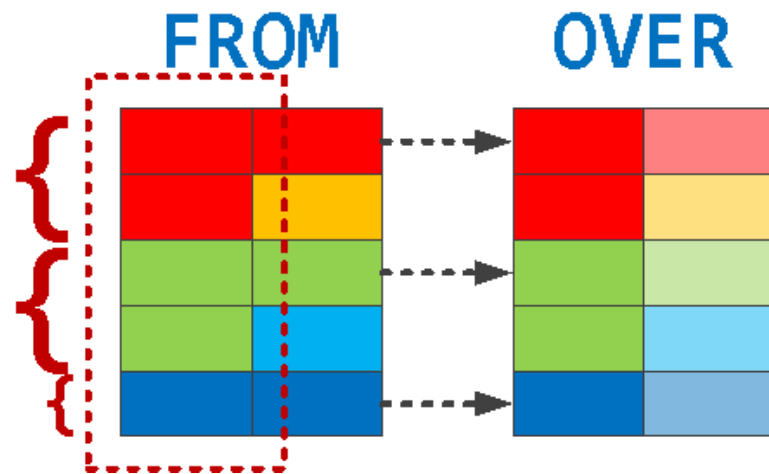
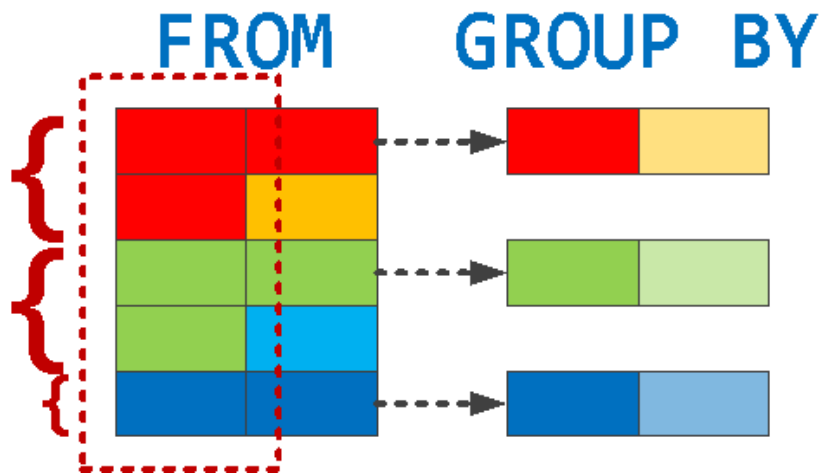
Программист написал программу и смотрит на экран: "1 — простое, 1 — простое, 1 — простое, 1 — простое: Да все они простые!"

WINDOW / OVER

```
SELECT
  i
, row_number() OVER w r
, sum(i) OVER(w
  ROWS BETWEEN 1 PRECEDING
  AND 1 FOLLOWING -- скользящее окно
  EXCLUDE CURRENT ROW
) s
FROM
  generate_series(0, 9) i
WINDOW -- тут все окна
  w AS (PARTITION BY i / 5 ORDER BY i);
```

i	r	s
integer	bigint	bigint
0	1	1 -- i / 5 == 0
1	2	2
2	3	4 -- 1(p) + 3(f)
3	4	6
4	5	3
5	1	6 -- i / 5 == 1
6	2	12
7	3	14
8	4	16
9	5	8 -- 8(p) + NULL(f)

WINDOW / OVER



WINDOW / OVER

```
SELECT DISTINCT ON(i / 5) -- сегмент
    i / 5
, sum(i) OVER(PARTITION BY i / 5) s
FROM
    generate_series(0, 9) i;
```

```
SELECT
    i / 5 -- ключ группировки
, sum(i) s
FROM
    generate_series(0, 9) i
GROUP BY
    1;
```

?column?	s
integer	bigint
0	10
1	35

GROUP BY

```
агрегатная_функция ([ ALL | DISTINCT ] выражение [ , ... ]  
    [ ORDER BY предложение_order_by ] )  
    [ FILTER ( WHERE условие_фильтра ) ]
```

```
агрегатная_функция ( * )  
    [ FILTER ( WHERE условие_фильтра ) ]
```

```
агрегатная_функция ( [ выражение [ , ... ] ] ) WITHIN GROUP (ORDER BY предложение_order_by )  
    [ FILTER ( WHERE условие_фильтра ) ]
```

<https://postgrespro.ru/docs/postgresql/15/sql-expressions#SYNTAX-AGGREGATES>

<https://postgrespro.ru/docs/postgresql/15/functions-aggregate>

GROUP BY

SELECT

```
    string_agg(i::text, ',') soa
, string_agg(i::text, ',' ORDER BY i DESC) sod -- сортировка всегда в конце
, string_agg(i::text, ',') FILTER(WHERE i % 2 = 0) sf
, string_agg(DISTINCT (i % 3)::text, ',') sd
```

FROM

```
    generate_series(0, 9) i;
```

soa text	sod text	sf text	sd text
0,1,2,3,4,5,6,7,8,9	9,8,7,6,5,4,3,2,1,0	0,2,4,6,8	0,1,2

GROUP BY

```
SELECT
    percentile_cont(ARRAY[0.5, 0.9, 0.95])
        WITHIN GROUP(ORDER BY i)
, rank(1.5)
        WITHIN GROUP(ORDER BY i)
FROM
    generate_series(0, 10) i;
```

percentile_cont		rank
double precision[]		bigint
{5,9,9.5}		3

FROM

FROM элемент_FROM [, ...]

[**ONLY**] имя_таблицы [*] [[**AS**] псевдоним [(псевдоним_столбца [, ...])]]

[**TABLESAMPLE** метод_выборки (аргумент [, ...]) [**REPEATABLE** (затравка)]]

[**LATERAL**] (выборка) [**AS**] псевдоним [(псевдоним_столбца [, ...])]

имя_запроса_WITH [[**AS**] псевдоним [(псевдоним_столбца [, ...])]]

[**LATERAL**] имя_функции ([аргумент [, ...]])

[**WITH ORDINALITY**] [[**AS**] псевдоним [(псевдоним_столбца [, ...])]]

[**LATERAL**] имя_функции ([аргумент [, ...]]) [**AS**] псевдоним (определение_столбца [, ...])

[**LATERAL**] имя_функции ([аргумент [, ...]]) **AS** (определение_столбца [, ...])

[**LATERAL**] **ROWS FROM**(имя_функции ([аргумент [, ...]]) [**AS** (определение_столбца [, ...])] [, ...])

[**WITH ORDINALITY**] [[**AS**] псевдоним [(псевдоним_столбца [, ...])]]

соединение

<https://postgrespro.ru/docs/postgresql/15/sql-select#SQL-FROM>

WITH ORDINALITY

```
SELECT
```

```
  *
```

```
FROM
```

```
  generate_series(1, 10, 2) i;
```

```
i
```

```
integer
```

```
  1
```

```
  3
```

```
  5
```

```
  7
```

```
  9
```

```
SELECT
```

```
  *
```

```
, row_number() OVER() ord
```

```
FROM
```

```
  generate_series(1, 10, 2) i;
```

```
i
```

```
integer
```

```
  1
```

```
  3
```

```
  5
```

```
  7
```

```
  9
```

```
| ord
```

```
bigint
```

```
  1
```

```
  2
```

```
  3
```

```
  4
```

```
  5
```


WITH ORDINALITY

```
SELECT
    *
FROM
    generate_series(1, 10, 2)
    WITH ORDINALITY T(i, ord);
```

```
SELECT
    *
FROM
    generate_series(1, 10, 2)
    WITH ORDINALITY;
```

```
SELECT
    *
, row_number() OVER() ord
FROM
    generate_series(1, 10, 2) i;
```

generate_series	ordinality
integer	bigint
1	1
3	2
5	3
7	4
9	5

JOIN

соединение

элемент_FROM, элемент_FROM

элемент_FROM **CROSS JOIN** *элемент_FROM*

элемент_FROM {

[**INNER**]

| **LEFT** [**OUTER**]

| **RIGHT** [**OUTER**]

| **FULL** [**OUTER**]

} **JOIN** *элемент_FROM*

{ **ON** *условие_соединения* | **USING** (*столбец_соединения* [, ...]) [**AS** *псевдоним_использования_соединения*] }

элемент_FROM **NATURAL** { [**INNER**] | **LEFT** | **RIGHT** | **FULL** } **JOIN** *элемент_FROM*

<https://postgrespro.ru/docs/postgresql/15/sql-select#SQL-FROM>

CROSS JOIN

```
SELECT
  *
FROM
  (
    VALUES
      (1, 2)
    , (3, 4)
    , (5, 6)
  ) X(a, b)
CROSS JOIN -- или «через запятую» - X, Y
  (
    VALUES
      (1, 8)
    , (3, 10)
    , (3, 12)
    , (7, 14)
  ) Y(a, c)
```

a	b	a	c
integer	integer	integer	integer
1	2	1	8
1	2	3	10
1	2	3	12
1	2	7	14
3	4	1	8
3	4	3	10
3	4	3	12
3	4	7	14
5	6	1	8
5	6	3	10
5	6	3	12
5	6	7	14

INNER JOIN

```
SELECT
  *
FROM
  (
    VALUES
      (1, 2)
    , (3, 4)
    , (5, 6)
  ) X(a, b)
JOIN -- или INNER JOIN
  (
    VALUES
      (1, 8)
    , (3, 10)
    , (3, 12)
    , (7, 14)
  ) Y(a, c)
  USING(a); -- или ON-условие
```

a	b	c
integer	integer	integer
1	2	8 -- (1,2) x (1,8)
3	4	10 -- (3,4) x (3,10)
3	4	12 -- (3,4) x (3,12)

```
X INNER JOIN Y ON cond == X CROSS JOIN Y WHERE cond
```

```
X INNER JOIN Y USING(a) == X CROSS JOIN Y WHERE X.a = Y.a
```

```
X INNER JOIN Y ON TRUE == X CROSS JOIN Y
```

LEFT JOIN

```
SELECT
  *
FROM
  (
    VALUES
      (1, 2)
    , (3, 4)
    , (5, 6)
  ) X(a, b)
LEFT JOIN -- или LEFT OUTER JOIN
  (
    VALUES
      (1, 8)
    , (3, 10)
    , (3, 12)
    , (7, 14)
  ) Y(a, c)
  USING(a); -- или ON-условие
```

a	b	c	
integer	integer	integer	
1	2	8	-- (1,2) x (1,8)
3	4	10	-- (3,4) x (3,10)
3	4	12	-- (3,4) x (3,12)
5	6		-- (5,6) x (NULL)

```
X LEFT JOIN Y ON cond
WHERE Y IS DISTINCT FROM NULL
==
X INNER JOIN Y ON cond
```

RIGHT JOIN

```
SELECT
  *
FROM
  (
    VALUES
      (1, 2)
    , (3, 4)
    , (5, 6)
  ) X(a, b)
RIGHT JOIN -- или RIGHT OUTER JOIN
  (
    VALUES
      (1, 8)
    , (3, 10)
    , (3, 12)
    , (7, 14)
  ) Y(a, c)
  USING(a); -- или ON-условие
```

a	b	c	
integer	integer	integer	
1	2	8	-- (1,2) x (1,8)
3	4	10	-- (3,4) x (3,10)
3	4	12	-- (3,4) x (3,12)
7		14	-- (NULL) x (7,14)

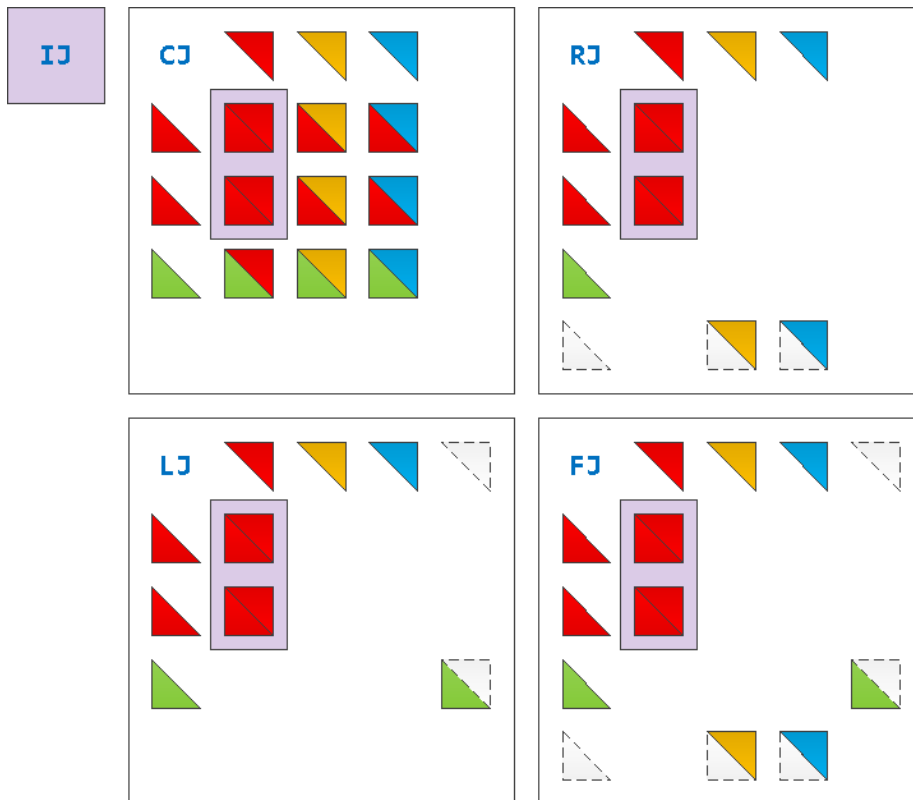
```
X RIGHT JOIN Y ON cond
==
Y LEFT JOIN X ON cond
```

FULL JOIN

```
SELECT
  *
FROM
  (
    VALUES
      (1, 2)
    , (3, 4)
    , (5, 6)
  ) X(a, b)
FULL JOIN -- или FULL OUTER JOIN
  (
    VALUES
      (1, 8)
    , (3, 10)
    , (3, 12)
    , (7, 14)
  ) Y(a, c)
  USING(a); -- или ON-условие
```

a	b	c	
integer	integer	integer	
1	2	8	-- (1,2) x (1,8)
3	4	10	-- (3,4) x (3,10)
3	4	12	-- (3,4) x (3,12)
5	6		-- (5,6) x (NULL)
7		14	-- (NULL) x (7,14)

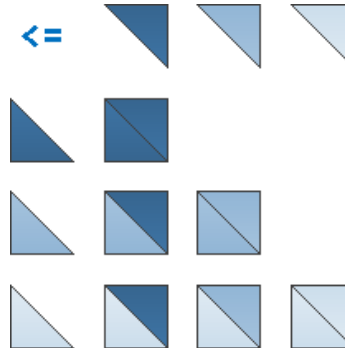
JOIN



JOIN

```
SELECT
  *
FROM
  (
    VALUES
      (1, 1)
    , (2, 2)
    , (3, 3)
  ) X(a, b)
INNER JOIN
  (
    VALUES
      (1, 1)
    , (2, 2)
    , (3, 3)
  ) Y(a, c)
ON X.a <= Y.a;
```

a	b	c
integer	integer	integer
1	1	1
1	1	2
1	1	3
2	2	2
2	2	3
3	3	3



NATURAL

```
X(a, b, x) NATURAL ??? JOIN Y(a, b, y)
```

```
X(a, b, x) ??? JOIN Y(a, b, y)  
    USING(a, b)
```

LATERAL

```
SELECT
```

```
    *
```

```
FROM
```

```
    (
```

```
        VALUES(1)
```

```
    ) X(i)
```

```
, (
```

```
    VALUES(i + 1)
```

```
    ) Y(j);
```

```
ERROR: column "i" does not exist
LINE 8:     VALUES(i + 1)
```

```
SELECT
```

```
    *
```

```
FROM
```

```
    (
```

```
        VALUES(1)
```

```
    ) X(i)
```

```
, LATERAL (
```

```
    VALUES(i + 1)
```

```
    ) Y(j);
```

i		j
integer		integer
1		2

CASE

CASE

WHEN *условие* **THEN** *результат*

[**WHEN** ... **THEN** ...]

[**ELSE** *результат*]

END

CASE *выражение*

WHEN *значение* **THEN** *результат*

[**WHEN** ... **THEN** ...]

[**ELSE** *результат*]

END

<https://postgrespro.ru/docs/postgresql/15/functions-conditional#FUNCTIONS-CASE>

CASE

```
SELECT
  i
, CASE i % 2
    WHEN 0 THEN 'even'
    WHEN 1 THEN 'odd'
  END v1
, CASE
    WHEN i % 15 = 0 THEN 'foobar'
    WHEN i % 3 = 0 THEN 'foo'
    WHEN i % 5 = 0 THEN 'bar'
  END v2
FROM
  generate_series(0, 15) i;
```

i	v1	v2
integer	text	text
0	even	foobar
1	odd	
2	even	
3	odd	foo
4	even	
5	odd	bar
6	even	foo
7	odd	
8	even	
9	odd	foo
10	even	bar
11	odd	
12	even	foo
13	odd	
14	even	
15	odd	foobar

COALESCE

COALESCE(*значение* [, ...])

Функция COALESCE возвращает первый попавшийся аргумент, отличный от NULL. Если же все аргументы равны NULL, результатом тоже будет NULL.

<https://postgrespro.ru/docs/postgresql/15/functions-conditional#FUNCTIONS-COALESCE-NVL-IFNULL>

COALESCE

```
SELECT
  i
, coalesce(
  CASE i % 2
    WHEN 0 THEN 'even'
  END
, CASE
  WHEN i % 15 = 0 THEN 'foobar'
  WHEN i % 3 = 0 THEN 'foo'
  WHEN i % 5 = 0 THEN 'bar'
  END
)
FROM
  generate_series(0, 15) i;
```

i	coalesce
integer	text
0	even
1	
2	even
3	foo
4	even
5	bar
6	even
7	
8	even
9	foo
10	even
11	
12	even
13	
14	even
15	foobar

NULLIF

NULLIF(*значение1*, *значение2*)

Функция NULLIF выдаёт значение NULL, если значение1 равно значению2; в противном случае она возвращает значение1. Это может быть полезно для реализации обратной операции к COALESCE.

<https://postgrespro.ru/docs/postgresql/15/functions-conditional#FUNCTIONS-NULLIF>

NULLIF

```
SELECT
  i
, nullif(
    i % 2
  , i % 3
)
FROM
  generate_series(0, 15) i;
```

i	nullif
integer	integer
0	-- i % 2 -> 0 == 0 <- i % 3
1	
2	0 -- i % 2 -> 0 == 2 <- i % 3
3	1
4	0
5	1
6	
7	
8	0
9	1
10	0
11	1
12	
13	
14	0
15	1

GREATEST | LEAST

GREATEST(*значение* [, ...])

LEAST(*значение* [, ...])

Функции GREATEST и LEAST выбирают наибольшее или наименьшее значение из списка выражений. Все эти выражения должны приводиться к общему типу данных, который станет типом результата. Значения NULL в этом списке игнорируются, так что результат выражения будет равен NULL, только если все его аргументы равны NULL.

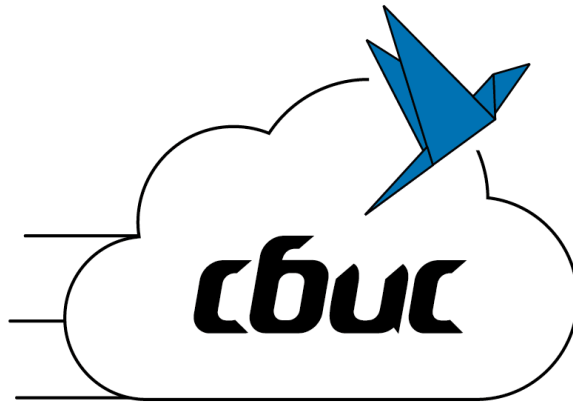
<https://postgrespro.ru/docs/postgresql/15/functions-conditional#FUNCTIONS-GREATEST-LEAST>

SELECT – ЭТО СЛОЖНО...

```
[ WITH [ RECURSIVE ] запрос_WITH [ , ... ] ]  
SELECT [ ALL | DISTINCT [ ON ( выражение [ , ... ] ) ] ]  
    [ * | выражение [ [ AS ] имя_результата [ , ... ] ]  
    [ FROM элемент_FROM [ , ... ] ]  
    [ WHERE условие ]  
    [ GROUP BY [ ALL | DISTINCT ] элемент_группирования [ , ... ] ]  
    [ HAVING условие ]  
    [ WINDOW имя_окна AS ( определение_окна ) [ , ... ] ]  
    [ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] выборка ]  
    [ ORDER BY выражение [ ASC | DESC | USING оператор ] [ NULLS { FIRST | LAST } ] [ , ... ] ]  
    [ LIMIT { число | ALL } ]  
    [ OFFSET начало [ ROW | ROWS ] ]  
    [ FETCH { FIRST | NEXT } [ число ] { ROW | ROWS } { ONLY | WITH TIES } ]  
    [ FOR { UPDATE | NO KEY UPDATE | SHARE | KEY SHARE } [ OF имя_таблицы [ , ... ] ] [ NOWAIT | SKIP LOCKED ] [ ... ] ]
```

<https://postgrespro.ru/docs/postgresql/15/sql-select>

<https://postgrespro.ru/docs/postgresql/15/queries>



Спасибо за внимание!

Боровиков Кирилл

kilor@tensor.ru / <https://n.sbis.ru/explain>

sbis.ru / tensor.ru