saby

Полезные SQL-конструкции в PostgreSQL

о синтаксическом «сахаре» и «ленивом» исполнении кода

спикер

Кирилл Боровиков

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

PG BootCamp Russia 2024 Kazan

PGBootCamp.ru

saby

Кирилл Боровиков

- 20+ лет в разработке, 15+ лет с PostgreSQL
- «Тензор» - сотни проектов, тысячи разработчиков
- технический директор, ведущий архитектор БД
- 150 хабрапубликаций (@Kilor)
- explain.tensor.ru – анализируем ваши планы

saby

О чем поговорим?

- SQL – это просто! или нет?..
- как писать меньше кода
- зачем писать больше кода

```
ARRAY                  CASE
WITH ORDINALITY        coalesce / nullif
IN / ANY / ALL         UNION ALL
ROW                    LATERAL
TABLE
```

saby

SQL – декларативный язык

вы описываете, что хотите получить

но СУБД лучше «знает», как это сделать

какие индексы использовать, в каком порядке соединять таблицы и накладывать условия, ...

saby

SQL – декларативный язык

некоторые СУБД принимают «подсказки»

demo

Worksheet | Query Builder

```
1  SELECT /* a comment */
2         /*+ full(e) */
3         /*+ full(d) */
4         e.empno,
5         e.ename,
6         d.dname
7    FROM emp e
8    LEFT JOIN dept d
9      ON d.deptno = e.deptno
10  WHERE empno > 7900;
```

PostgreSQL –

но всегда готов рассказать,

как конкретно он планирует

выполнять ваш запрос

(или уже как-то выполнил 😈)

saby

```
Query Text: explain (analyze, buffers, costs off)
SELECT * FROM pg_class WHERE (oid, relname) = (
  SELECT oid, relname FROM pg_class WHERE relkind = 'r' LIMIT 1
);

Index Scan using pg_class_relname_nsp_index on pg_class (actual time=0.049..0.050 rows=1 loops=1)
  Index Cond: (relname = $1)
  Filter: (oid = $0)
  Buffers: shared hit=4
  InitPlan 1 (returns $0,$1)
    ->  Limit (actual time=0.019..0.020 rows=1 loops=1)
          Buffers: shared hit=1
          ->  Seq Scan on pg_class pg_class_1 (actual time=0.015..0.015 rows=1 loops=1)
                Filter: (relkind = 'r'::"char")
                Rows Removed by Filter: 5
                Buffers: shared hit=1
```

saby

saby

explain.tensor.ru

на что смотрим: время, RRbF, buffers

Курс «PostgreSQL для начинающих»: #4 – Анализ запросов (ч.1 – как и зачем читать планы)

saby

ARRAY

конструктор массива

https://postgrespro.ru/docs/postgresql/16/sql-expressions#SQL-SYNTAX-ARRAY-CONSTRUCTORS

saby

«Размножение» строк

```sql
SELECT
  i
, unnest(ARRAY[0, 1])
, unnest(ARRAY[2 * i, -2 * i - 1])
FROM
  generate_series(0, 3) i;
```

i integer	unnest integer	unnest integer
0	0	0
0	1	-1
1	0	2
1	1	-3
2	0	4
2	1	-5
3	0	6
3	1	-7

saby

Пересечение массивов

```
SELECT
  ARRAY(
    SELECT
      unnest('{1,2,3,4,5}'::integer[])
  INTERSECT
    SELECT
      unnest('{2,3,5,7,11}'::integer[])
  );
```

```
array
integer[]
{3,5,2}
```

saby

Объединение массивов («гармошка»)

```
SELECT
  ARRAY(
    SELECT
      unnest(v::integer[])
    FROM
      (
        VALUES ('{1,2,3}'), ('{4,5}'), ('{6}')
      ) T(v)
  );
```

```
array
integer[]
{1,2,3,4,5,6}
```

```
SELECT
  '{1,2,3}'::integer[] ||
  '{4,5}'::integer[] ||
  '{6}'::integer[];
```

saby

Объединение массивов (*)

```
SELECT
  string_to_array(      -- превратили строку в массив
    string_agg(         -- соединили строки
      array_to_string( -- превратили массивы в строки
        v::integer[]
      , ','), ','), ',')
FROM
  (
    VALUES ('{1,2,3}'), ('{4,5}'), ('{6}')
  ) T(v);
```

saby

WITH ORDINALITY

нумератор строк

https://postgrespro.ru/docs/postgresql/16/queries-table-expressions#QUERIES-TABLEFUNCTIONS

saby

Нумерация элементов

```
SELECT
   *
, row_number() OVER() ord
FROM
   unnest('{1,2,4,8,16}'::integer[]) id;
```

id	ord
integer	bigint
1	1
2	2
4	3
8	4
16	5

#	node, ms	tree, ms	rows	WindowAgg	Function Scan on unnest id
		.022	5	итоговые результаты	
0	.012	.022	5	WindowAgg (actual time=0.015..0.022 rows=5 loops=1)	
1	.010		5	-> Function Scan on unnest id (actual time=0.009..0.010 rows=5 loops=1)	

saby

Нумерация элементов

```
SELECT
   *
FROM
   unnest('{1,2,4,8,16}'::integer[])
     WITH ORDINALITY T(id, ord);
```

id	ord
integer	bigint
1	1
2	2
4	3
8	4
16	5

#	node, ms	tree, ms	rows	Function Scan on unnest t
		.010	5	итоговые результаты
0	.010		5	Function Scan on unnest t (actual time=0.009..0.010 rows=5 loops=1)

saby

«Связывание» массивов

```
SELECT
  v1
, v2
FROM
  unnest('{1,2,3,4}'::integer[])
    WITH ORDINALITY T1(v1, rn)
LEFT JOIN
  unnest('{5,6}'::integer[])
    WITH ORDINALITY T2(v2, rn)
      USING(rn);
```

v1 integer	v2 integer
1	5
2	6
3	
4	

PostgreSQL Antipatterns: сизифов JOIN массивов

saby

«Связывание» массивов

#	node, ms	tree, ms	rows	Merge Left Join	Function Scan on unnest t1	Materialize	Function Scan on u
		.022	4	> итоговые результаты			
0	.007	.022	4	Merge Left Join (actual time=0.018..0.022 rows=4 loops=1) Merge Cond: (t1.rn = t2.rn)			
1	.010		4	-> Function Scan on unnest t1 (actual time=0.010..0.010 rows=4 loops=1)			
2	.002	.005	2	-> Materialize (actual time=0.004..0.005 rows=2 loops=1)			
3	.003		2	-> Function Scan on unnest t2 (actual time=0.002..0.003 rows=2 loops=1)			

saby

«Связывание» массивов (*)

```
SELECT
  v1
, v2
FROM
  unnest(
    '{1,2,3,4}'::integer[]
  , '{5,6}'::integer[]
  ) T(v1, v2);
```

v1	v2
integer	integer
1	5
2	6
3	
4	

```
# node, ms  tree, ms  rows   Function Scan on t
        .012         4   > итоговые результаты
0       .012         4   Function Scan on t (actual time=0.011..0.012 rows=4 loops=1)
```

saby

IN

наличие в списке

https://postgrespro.ru/docs/postgresql/16/functions-comparisons#FUNCTIONS-COMPARISONS-IN-SCALAR

saby

Замена OR-цепочек

```
v = 2 OR
v = 3 OR
v = 5
```

```
v IN (2, 3, 5)
```

saby

Замена OR-цепочек

```
SELECT
  *
FROM
  pg_class
WHERE
  relname = 'pg_class' OR
  relname = 'pg_index';
```

```
SELECT
  *
FROM
  pg_class
WHERE
  relname IN (
    'pg_class'
  , 'pg_index'
  );
```

saby

Замена OR-цепочек

saby

Проблемы с NULL

```
v = 2 OR
v = 3 OR
v IS NULL
```

```
v IN (2, 3, NULL)
```

```
v : NULL -> TRUE
```

```
v : NULL -> NULL
```

saby

Проблемы с NULL: «невозможное» значение

```
v = 2 OR
v = 3 OR
v IS NULL
```

```
coalesce(v, -1)
  IN (2, 3, -1)
```

```
v : NULL -> TRUE
```

```
v : NULL -> TRUE
```

saby

ANY

«любой из»

https://postgrespro.ru/docs/postgresql/16/functions-comparisons#FUNCTIONS-COMPARISONS-ANY-SOME

saby

Обобщение IN

```
v IN (2, 3, 5)
```

```
v = ANY(ARRAY[2,3,5])
```

```
expr op ANY(arrayExpr)
```

saby

Передача параметров

```
v = ANY(ARRAY[2,3,5])
```

```
v = ANY(
  '{2,3,5}'::integer[]
)
```

```
v IN ($1, $2, $3)
-- $1 = 2
-- $2 = 3
-- $3 = 5
```

```
v = ANY($1::integer[])
-- $1 = '{2,3,5}'
```

PostgreSQL Antipatterns: передача наборов и выборок в SQL

saby

Набор LIKE-шаблонов

```sql
SELECT
  *
FROM
  pg_class
WHERE
  relname LIKE ANY(
    '{pg_publication%,pg_subscription%}'::text[]
  );
```

oid	relname	...
oid	name	
6100	pg_subscription	
6102	pg_subscription_rel	
6104	pg_publication	
6106	pg_publication_rel	
6110	pg_publication_oid_index	
...		

SQL HowTo: делаем из мухи слона (алгоритм Ли)

saby

ALL

«каждый из»

https://postgrespro.ru/docs/postgresql/16/functions-comparisons#FUNCTIONS-COMPARISONS-ALL

saby

Уникализация массива

```
v = ANY(ARRAY[2,3,5])
-- равно любому
```

```
v <> ALL(ARRAY[2,3,5])
-- не равно ни одному
```

saby

Замена AND-цепочек

```
v LIKE 'м_х_' AND
v LIKE '_у_а' AND
v LIKE 'му%'
```

```
v LIKE ALL (
  '{м_х_,_у_а,му%}'::text[]
)
```

saby

Набор LIKE-шаблонов

```
SELECT
   *
FROM
   pg_class
WHERE
   relname LIKE ALL( -- ~ '^pg_class.*_index$'
     '{pg\_class%,%\_index}'::text[]
   );
```

oid	relname	...
oid	name	
2662	pg_class_oid_index	
2663	pg_class_relname_nsp_index	
3455	pg_class_tblspc_relfilenode_index	

saby

ROW

конструктор записи

https://postgrespro.ru/docs/postgresql/16/sql-expressions#SQL-SYNTAX-ROW-CONSTRUCTORS

saby

Сравнение сконструированных строк

```
SELECT
  ('pg_class'::regclass, 1) IN (
    SELECT
      attrelid, attnum
    FROM
      pg_attribute
  );
```

https://postgrespro.ru/docs/postgresql/16/functions-subquery#FUNCTIONS-SUBQUERY-IN

saby

Перебор по индексу

```sql
SELECT
  *
FROM
  pg_attribute
WHERE
  (
    attrelid = 'pg_class'::regclass AND -- если таблица есть,
    attnum > 0                          -- надо взять "первое" ее поле
  ) OR
  attrelid > 'pg_class'::regclass       -- если вдруг нету - "следующую" таблицу
ORDER BY
  attrelid, attnum
LIMIT 1;
```

regclass, regnamespace, ...

saby

Перебор по индексу

```
SELECT                                          * шаг алгоритма «DISTINCT для бедных»
  *
FROM
  pg_attribute
WHERE
  (attrelid, attnum) > ('pg_class'::regclass, 0)
ORDER BY
  attrelid, attnum
LIMIT 1;
```

PostgreSQL Antipatterns: убираем медленные и ненужные сортировки

saby

Перебор по индексу

#	node, ms	tree, ms	rows	RRbF				sh.ht
		.066	1	164		Limi	Index Scan using pg_attribute_relid_attnum_index on pg_attribute	
							итоговые результаты	14
								112KB
0	.002	.066	1				Limit (actual time=0.066..0.066 rows=1 loops=1)	
							Buffers: shared hit=14	
1	.064		1	164	99.4%	IR IC	-> Index Scan using pg_attribute_relid_attnum_index on pg_attribute (actual time=0.064..0.064 rows=1 loops=1)	14
							Filter: (((attrelid = '1259'::oid) AND (attnum > 0)) OR (attrelid > '1259'::oid))	112KB
							Rows Removed by Filter: 164	
							Buffers: shared hit=14	

#	node, ms	tree, ms	rows				sh.ht
		.028	1		Limit	Index Scan using pg_attribute_relid_attnum_index on pg_attribute	
						итоговые результаты	3
							24KB
0	.002	.028	1			Limit (actual time=0.027..0.028 rows=1 loops=1)	
						Buffers: shared hit=3	
1	.026		1			-> Index Scan using pg_attribute_relid_attnum_index on pg_attribute (actual time=0.026..0.026 rows=1 loops=1)	3
						Index Cond: (ROW(attrelid, attnum) > ROW('1259'::oid, 0))	24KB
						Buffers: shared hit=3	

saby

TABLE

вывод выборки

https://postgrespro.ru/docs/postgresql/16/sql-select#SQL-TABLE

saby

Базовый синтаксис

```
SELECT
    *
FROM
    имя_таблицы;
```

```
TABLE имя_таблицы;
```

можно:

- WITH
- UNION / INTERSECT / EXCEPT
- ORDER BY
- LIMIT / OFFSET / FETCH
- FOR (блокировка)

нельзя:

- WHERE
- GROUP BY
- HAVING

saby

Отладка выборок

```
SELECT
   *
FROM
   (
      ...
   ) T;
```

```
WITH T AS (
   ...
)
TABLE T;
```

saby

Фиксация выборки

```
WITH T AS (
   TABLE pg_class
)
   (TABLE T LIMIT 1)
UNION ALL
   (TABLE T LIMIT 1 OFFSET 100);
```

oid	relname	...
oid	name	
112	pg_foreign_data_wrapper_oid_index	
2699	pg_trigger_tgconstraint_index	

1-я и 101-я записи из одного набора

saby

Фиксация выборки

#	node, ms	tree, ms	rows	Seq Scan on pg_class / Li / CTE Scan on t / Limit	sh.ht
		.075	2	итоговые результаты	4 32KB
0	.001	.075	2	Append (actual time=0.031..0.075 rows=2 loops=1) Buffers: shared hit=4	
1				CTE t	
2	.040		101	-> Seq Scan on pg_class (actual time=0.026..0.040 rows=101 loops=1) Buffers: shared hit=4	4 32KB
3	.002	.031	1	-> Limit (actual time=0.030..0.031 rows=1 loops=1) Buffers: shared hit=1	
4	.027	.029	1	-> CTE Scan on t (actual time=0.029..0.029 rows=1 loops=1) Buffers: shared hit=1	
5	.005	.043	1	CC -> Limit (actual time=0.043..0.043 rows=1 loops=1) Buffers: shared hit=3	
6		.038	101	-> CTE Scan on t t_1 (actual time=0.001..0.038 rows=101 loops=1) Buffers: shared hit=3	

saby

Подстановка значения

```
WITH T AS (
  SELECT ARRAY( ... ) -- 1 строка, 1 столбец
)
SELECT
  unnest((TABLE T));  -- скобки вокруг!
```

saby

Подстановка по «словарю»

```
WITH dict AS (
  SELECT
    k
  , k * random() v
  FROM
    generate_series(1, 1e5, 100) k
)
SELECT
  k
, v
, k * random() v2
FROM
  generate_series(1, 1e4) k
NATURAL LEFT JOIN
  dict;
```

k	v	v2
numeric	double precision	double precision
1	0.49262788461257645	0.33890315293108797
2		0.4251541072815863
...		
101	40.91576145838801	76.33713131812358
...		

saby

Подстановка по «словарю»

#	node, ms	tree, ms	rows	
				Merge Left Join · Functic · Sort · Function Scan on ger · CT
		12.536	10′000	итоговые результаты
0	7.602	12.536	10′000	Merge Left Join (actual time=4.327..12.536 rows=10′000 loops=1) Merge Cond: (k.k = dict.k)
1				CTE dict
2	.742		1′000	-> Function Scan on generate_series k_1 (actual time=0.154..0.742 rows=1′000 loops=1)
3	1.801	3.711	10′000	-> Sort (actual time=3.110..3.711 rows=10′000 loops=1) Sort Key: k.k Sort Method: quicksort Memory: 541kB
4	1.910		10′000	-> Function Scan on generate_series k (actual time=1.446..1.910 rows=10′000 loops=1)
5	.174	1.223	101	-> Sort (actual time=1.212..1.223 rows=101 loops=1) Sort Key: dict.k Sort Method: quicksort Memory: 64kB
6	.307	1.049	1′000	-> CTE Scan on dict (actual time=0.156..1.049 rows=1′000 loops=1)

saby

Подстановка по «словарю»

```
CREATE EXTENSION hstore;
```

```
WITH dict AS (
  SELECT
    hstore( -- формируем словарь
      array_agg(k)::text[]
    , array_agg(k * random())::text[]
    )
  FROM
    generate_series(1, 1e5, 100) k
)
SELECT
  k
, ((TABLE dict) -> k::text)::double precision v -- извлекаем из словаря
, k * random() v2
FROM
  generate_series(1, 1e4) k;
```

saby

Подстановка по «словарю»

#	node, ms	tree, ms	rows	Function Scan on generate_series k	Aggregate	Fi
		10.041	10′000	итоговые результаты		
0	8.904	10.041	10′000	Function Scan on generate_series k (actual time=2.594..10.041 rows=10′000 loops=1)		
1				CTE dict		
2	.930	1.129	1	-> Aggregate (actual time=1.129..1.129 rows=1 loops=1)		
3	.199		1′000	-> Function Scan on generate_series k_1 (actual time=0.148..0.199 rows=1′000 loops=1)		
4				InitPlan 2 (returns $1)		
5	.008	1.137	1	-> CTE Scan on dict (actual time=1.136..1.137 rows=1 loops=1)		

saby

Подстановка по «словарю»

Метод			время, ms	%
JOIN			12.536	+24.8
hstore	/	->	10.041	---
jsonb_object	/	->>	11.836	+17.8
json_object	/	->>	1434.771	143x

PostgreSQL Antipatterns: ударим словарем по тяжелому JOIN

saby

CASE

значение по условию

https://postgrespro.ru/docs/postgresql/16/functions-conditional#FUNCTIONS-CASE

saby

Изоляция сложных условий простыми

```sql
SELECT * FROM pg_class
WHERE
  (relkind, relnamespace) = ('r', 'pg_catalog'::regnamespace) AND
  (
    SELECT count(*) FROM pg_index WHERE indrelid = pg_class.oid
  ) > 1 -- таблицы с более чем 1 индексом
ORDER BY
  relname
LIMIT 16;
```

saby

Изоляция сложных условий простыми

saby

Изоляция сложных условий простыми

```
SELECT * FROM pg_class
WHERE
  CASE WHEN (relkind, relnamespace) = ('r', 'pg_catalog'::regnamespace) THEN
    EXISTS(
      SELECT NULL FROM pg_index WHERE indrelid = pg_class.oid
      LIMIT 1 OFFSET 1
    )
  END
ORDER BY
  relname
LIMIT 16;
```

1. <простое условие> AND <сложный запрос>
   -> CASE WHEN <простое условие> THEN <сложный запрос>

2. count(*) > N
   -> EXISTS(... LIMIT 1 OFFSET N)

PostgreSQL Antipatterns: редкая запись долетит до середины JOIN

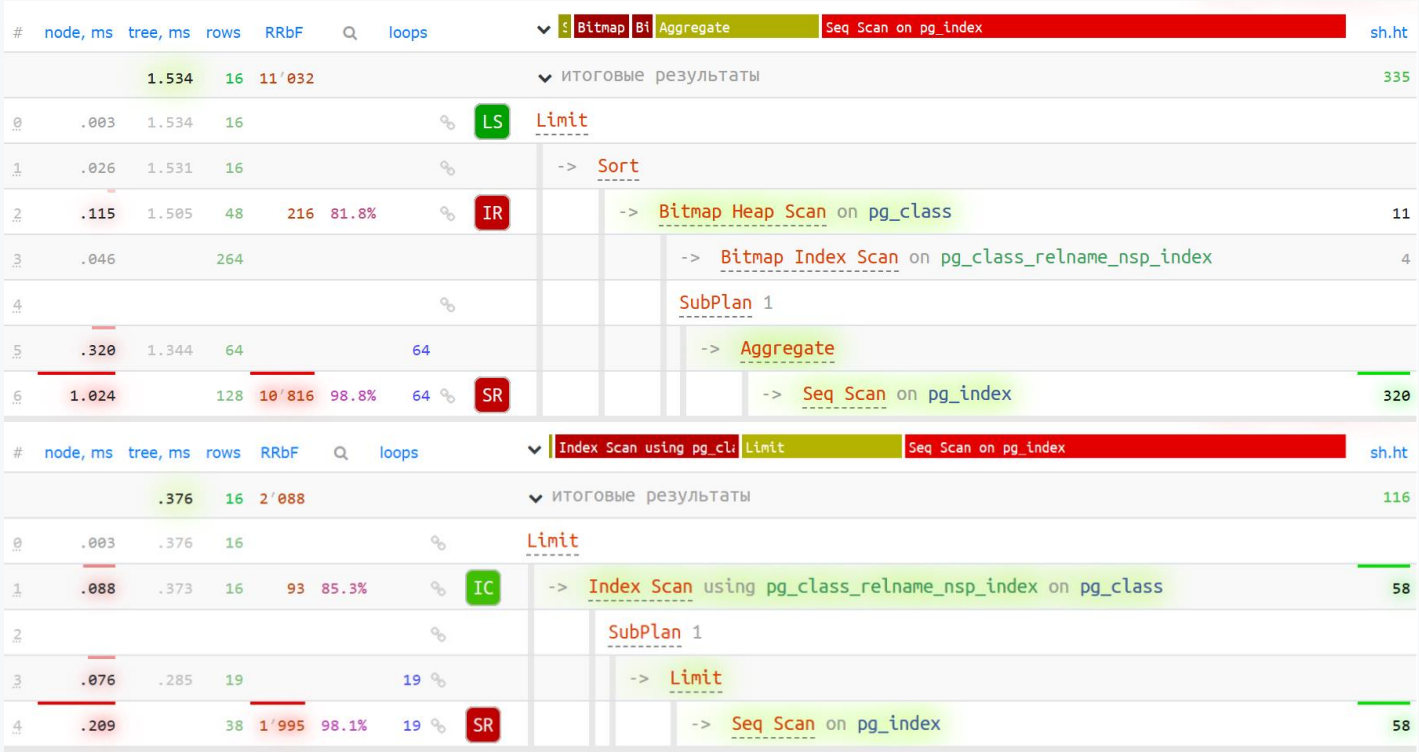
saby

Изоляция сложных условий простыми

#	node, ms	tree, ms	rows	RRbF	loops	Index Scan using pg_class_relname_nsp_ / Limit / Seq Scan on pg_index	sh.ht
		.376	16	2'088		итоговые результаты	116 928KB
0	.003	.376	16			Limit (actual time=0.088..0.376 rows=16 loops=1) Buffers: shared hit=116	
1	.088	.373	16	93	85.3%	IC -> Index Scan using pg_class_relname_nsp_index on pg_class (actual time=0.087..0.373 rows=16 loops=1) Filter: CASE WHEN ((relkind = 'r'::"char") AND (relnamespace = '11'::oid)) THEN (SubPlan 1) ELSE NULL::boolean END Rows Removed by Filter: 93 Buffers: shared hit=116	58 464KB
2						SubPlan 1	
3	.076	.285	19		19	-> Limit (actual time=0.015..0.015 rows=1 loops=19) Buffers: shared hit=58	
4	.209		38	1'995	98.1% 19	SR -> Seq Scan on pg_index (actual time=0.009..0.011 rows=2 loops=19) Filter: (indrelid = pg_class.oid) Rows Removed by Filter: 105 Buffers: shared hit=58	58 464KB

saby

Изоляция сложных условий простыми



saby

Порядок вычислений

```
SELECT
  CASE
    WHEN random() < 0.5 THEN (SELECT 1)
    WHEN random() < 0.5 THEN (SELECT 2)
    ELSE (SELECT 3)
  END;
```

saby

Порядок вычислений

#	node, ms	tree, ms	rows	loops	Result
		.006	1		итоговые результаты
0	.006		1		Result (actual time=0.006..0.006 rows=1 loops=1)
1					InitPlan 1 (returns $0)
2			n/e		-> Result (never executed)
3					InitPlan 2 (returns $1)
4			n/e		-> Result (never executed)
5					InitPlan 3 (returns $2)
6			1		-> Result (actual time=0.000..0.000 rows=1 loops=1)

saby

Порядок вычислений

```
SELECT
  CASE
    WHEN i % 2 <> 0 THEN (SELECT i / 2)
    WHEN i % 3 <> 0 THEN (SELECT i / 3)
    WHEN i % 5 <> 0 THEN (SELECT i / 5)
  END
FROM
  generate_series(1, 1000) i;
```

saby

Порядок вычислений

#	node, ms	tree, ms	rows	loops	Function Scan on generate_series i
		.508	1′000		итоговые результаты
0	.508		1′000		Function Scan on generate_series i (actual time=0.086..0.508 rows=1′000 loops=1)
1					SubPlan 1
2		500	500		-> Result (actual time=0.000..0.000 rows=1 loops=500)
3					SubPlan 2
4		334	334		-> Result (actual time=0.000..0.000 rows=1 loops=334)
5					SubPlan 3
6		133	133		-> Result (actual time=0.000..0.000 rows=1 loops=133)

saby

Порядок OR-цепочки

```
(SELECT ...A) OR (SELECT ...B) OR (SELECT ...C)
```

```
CASE
  WHEN (SELECT ...A) THEN TRUE
  WHEN (SELECT ...B) THEN TRUE -- NOT (...A)
  WHEN (SELECT ...C) THEN TRUE -- NOT (...A & B)
END
```

PostgreSQL Antipatterns: вычисление условий в SQL

saby

Порядок AND-цепочки

```
(SELECT ...A) AND (SELECT ...B) AND (SELECT ...C)
```

```
CASE
  WHEN (SELECT ...A) THEN
    CASE
      WHEN (SELECT ...B) THEN
        (SELECT ...C)
    END
END
```

saby

Вынос условия

```
CASE
   WHEN (SELECT ...A) = 1 THEN 'one'
   WHEN (SELECT ...A) = 2 THEN 'two'
   WHEN (SELECT ...A) = 3 THEN 'three'
END
```

* а вдруг внутри не-STABLE функция?..
clock_timestamp

saby

Вынос условия

```
CASE (SELECT ...A)
  WHEN 1 THEN 'one'
  WHEN 2 THEN 'two'
  WHEN 3 THEN 'three'
END
```

saby

Вынос условия (*)

```
CASE (SELECT ...A)
  WHEN 1 THEN 'one'
  WHEN 2 THEN 'two'
  WHEN 3 THEN 'three'
END
```

```
'{"1":"one","2":"two","3":"three"}'::json
  ->> (SELECT ...A)::text
```

saby

coalesce

первый не-NULL

https://postgrespro.ru/docs/postgresql/16/functions-conditional#FUNCTIONS-COALESCE-NVL-IFNULL

saby

Замена CASE

```
SELECT
  CASE WHEN a IS NOT NULL THEN a ELSE b END
FROM
  (
    SELECT
      (SELECT CASE WHEN random() < 0.5 THEN 1 END) a
      -- в половине случаев тут NULL
    , (SELECT 2) b
  ) T;
```

saby

Замена CASE

#	node, ms	tree, ms	rows	Subquery Scan on t	Result	Result
		.008	1	итоговые результаты		
0	.002	.008	1	Subquery Scan on t (actual time=0.007..0.008 rows=1 loops=1)		
1	.004	.006	1	-> Result (actual time=0.006..0.006 rows=1 loops=1)		
2				InitPlan 1 (returns $0)		
3	.002		1	-> Result (actual time=0.002..0.002 rows=1 loops=1)		
4				InitPlan 2 (returns $1)		
5			1	-> Result (actual time=0.000..0.000 rows=1 loops=1)		

saby

Замена CASE

```
CASE
  WHEN a IS NOT NULL
    THEN a
  ELSE b
END
```

```
coalesce(a, b)
```

saby

Замена CASE

```
SELECT
  coalesce(a, b)
FROM
  (
    SELECT
      (SELECT CASE WHEN random() < 0.5 THEN 1 END) a
      -- в половине случаев тут NULL
    , (SELECT 2) b
  ) T;
```

saby

Замена CASE

```
SELECT
  coalesce(
    (SELECT CASE WHEN random() < 0.5 THEN 1 END)
  , (SELECT 2)
  );
```

saby

Замена CASE

#	node, ms	tree, ms	rows	loops	Result	Result
		.007	1		итоговые результаты	
0	.004	.007	1		Result (actual time=0.006..0.007 rows=1 loops=1)	
1					InitPlan 1 (returns $0)	
2	.003		1		-> Result (actual time=0.003..0.003 rows=1 loops=1)	
3					InitPlan 2 (returns $1)	
4				n/e	-> Result (never executed)	

saby

Замена CASE (*)

```
CASE
  WHEN EXISTS(
    SELECT ...A
  ) THEN TRUE
  ELSE FALSE
END
```

```
coalesce(
  EXISTS(SELECT ...A)
, FALSE
)
```

```
EXISTS(SELECT ...A)
```

PostgreSQL Antipatterns: бесполезный CASE

saby

nullif

NULL при равенстве

https://postgrespro.ru/docs/postgresql/16/functions-conditional#FUNCTIONS-NULLIF

saby

Замена CASE

```
CASE
   WHEN a = 1 THEN NULL
   ELSE a
END
```

```
nullif(a, 1)
```

PostgreSQL Antipatterns: бесполезный CASE #2

saby

UNION ALL

объединение выборок

https://postgrespro.ru/docs/postgresql/16/queries-union

saby

Замена OR-цепочек

```
SELECT
  *
FROM
  pg_class
WHERE
  relname = 'pg_class' OR
  relname = 'pg_index';
```

saby

Замена OR-цепочек

#	node, ms	tree, ms	rows	Bitmap Heap Scan on pg_class / BitmapOr / Bitmap Index Scan on pg_class_relname_nsp_index / Bitmap Inde	sh.ht
		.026	2	итоговые результаты	5 40KB
0	.006	.026	2	Bitmap Heap Scan on pg_class (actual time=0.025..0.026 rows=2 loops=1) Recheck Cond: ((relname = 'pg_class'::name) OR (relname = 'pg_index'::name)) Heap Blocks: exact=1 Buffers: shared hit=5	1 8KB
1	.002	.020	BO	-> BitmapOr (actual time=0.019..0.020 rows=0 loops=1) Buffers: shared hit=4	
2	.016		1	-> Bitmap Index Scan on pg_class_relname_nsp_index (actual time=0.016..0.016 rows=1 loops=1) Index Cond: (relname = 'pg_class'::name) Buffers: shared hit=2	2 16KB
3	.002		1	-> Bitmap Index Scan on pg_class_relname_nsp_index (actual time=0.002..0.002 rows=1 loops=1) Index Cond: (relname = 'pg_index'::name) Buffers: shared hit=2	2 16KB

saby

Замена OR-цепочек

```
  SELECT * FROM pg_class WHERE relname = 'pg_class'
UNION ALL
  SELECT * FROM pg_class WHERE relname = 'pg_index';
```

saby

Противоположные условия

```
SELECT
  (
    SELECT (SELECT random() * +i) WHERE i % 2 = 0
  UNION ALL
    SELECT (SELECT random() * -i) WHERE i % 2 = 1
  )
FROM
  generate_series(0, 7) i;
```

?column?
double precision
0
-0.3293574548567477
1.0861763249741205
-0.40808150003692956
3.583351632216316
-3.6140284792073496
4.20011484910809
-5.95659560756944

saby

Противоположные условия

#	node, ms	tree, ms	rows	loops	Function Scan on generate_series i	Append	Result	Result
		.030	8		> итоговые результаты			
0	.014	.030	8		Function Scan on generate_series i (actual time=0.017..0.030 rows=8 loops=1)			
1					SubPlan 3			
2	.008	.016	8	8	-> Append (actual time=0.002..0.002 rows=1 loops=8)			
3	.004	.008	8	8	-> Result (actual time=0.001..0.001 rows=1 loops=8) One-Time Filter: ((i.i % 2) = 0)			
4					InitPlan 1 (returns $1)			
5	.004		4	4	-> Result (actual time=0.001..0.001 rows=1 loops=4)			
6			8	8	-> Result (actual time=0.000..0.000 rows=1 loops=8) One-Time Filter: ((i.i % 2) = 1)			
7					InitPlan 2 (returns $2)			
8			4	4	-> Result (actual time=0.000..0.000 rows=1 loops=4)			

saby

Приоритет выборок

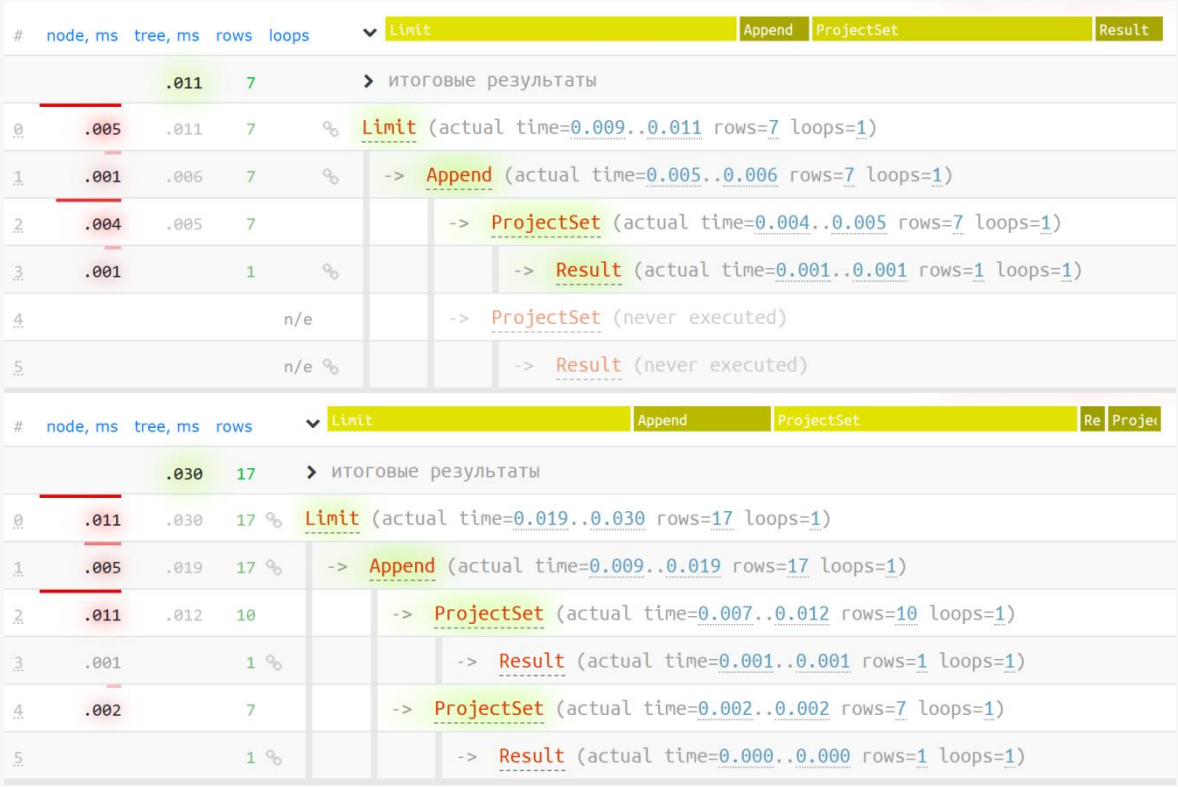
```
(
  SELECT generate_series(1, 10)
UNION ALL
  SELECT generate_series(-10, -1)
)
LIMIT random() * 20;
```

generate_series
integer
1
2
3
4
5
6
7

generate_series
integer
1
2
3
4
5
6
7
8
9
10
-10
-9
-8
-7
-6
-5
-4

saby

Приоритет выборок



saby

Приоритет первого найденного

```
(
  (SELECT ... LIMIT 1)
UNION ALL
  (SELECT ... LIMIT 1)
)
LIMIT 1;
```

saby

LATERAL

упорядочение выборок

https://postgrespro.ru/docs/postgresql/16/queries-table-expressions#QUERIES-LATERAL

saby

Вложенный «цикл»

```
SELECT
  a, b
FROM
  (
    SELECT generate_series(1, 4)
  ) X(a)
JOIN
  (
    SELECT generate_series(1, 4)
  ) Y(b)
    ON b <= a;
```

a	b
integer	integer
1	1
2	1
2	2
3	1
3	2
3	3
4	1
4	2
4	3
4	4

saby

Вложенный «цикл»

```
SELECT
  a, b
FROM
  (
    SELECT generate_series(1, 4)
  ) X(a)
, LATERAL
  (
    SELECT generate_series(1, a)
  ) Y(b);
```

Ключевое слово LATERAL может предварять вложенный запрос SELECT в списке FROM. Оно позволяет **обращаться** в этом вложенном SELECT **к столбцам** элементов FROM, **предшествующим** ему в списке FROM. (Без LATERAL все вложенные подзапросы SELECT обрабатываются независимо и не могут ссылаться на другие элементы списка FROM.)

saby

Вложенный «цикл»

#	node, ms	tree, ms	rows	RRbF	loops	Nested Loop	ProjectSet	Result	Materialize	ProjectSet
		.022	10	6		> итоговые результаты				
0	.008	.022	10	6	37.5%	Nested Loop (actual time=0.014..0.022 rows=10 loops=1) Join Filter: ((generate_series(1, 4)) <= (generate_series(1, 4))) Rows Removed by Join Filter: 6				
1	.004	.006	4			-> ProjectSet (actual time=0.005..0.006 rows=4 loops=1)				
2	.002		1			-> Result (actual time=0.001..0.002 rows=1 loops=1)				
3	.005	.008	16		4	-> Materialize (actual time=0.002..0.002 rows=4 loops=4)				
4	.003		4			-> ProjectSet (actual time=0.001..0.003 rows=4 loops=1)				
5			1			-> Result (actual time=0.000..0.000 rows=1 loops=1)				

#	node, ms	tree, ms	rows	loops	Nested Loop	ProjectSet	Result	ProjectSet
		.011	10		> итоговые результаты			
0	.003	.011	10		Nested Loop (actual time=0.005..0.011 rows=10 loops=1)			
1	.003	.004	4		-> ProjectSet (actual time=0.003..0.004 rows=4 loops=1)			
2	.001		1		-> Result (actual time=0.001..0.001 rows=1 loops=1)			
3	.004		12	4	-> ProjectSet (actual time=0.000..0.001 rows=3 loops=4)			
4			4	4	-> Result (actual time=0.000..0.000 rows=1 loops=4)			

saby

Вложенный «цикл»

```
SELECT
  a, b
FROM
  generate_series(1, 4) X(a)
, generate_series(1, a) Y(b);
```

Слово **LATERAL** можно также добавить перед вызовом функции в списке **FROM**, но в этом случае оно **будет избыточным**, так как **выражения с функциями могут ссылаться на предыдущие элементы** списка **FROM** в любом случае.

#	node, ms	tree, ms	rows	loops	Nested Loop	Function Scan on generate_series x	Function Scan on generate_series y
		.022	10		итоговые результаты		
0	.007	.022	10		Nested Loop (actual time=0.011..0.022 rows=10 loops=1)		
1	.007		4		-> Function Scan on generate_series x (actual time=0.007..0.007 rows=4 loops=1)		
2	.008		12	4	-> Function Scan on generate_series y (actual time=0.001..0.002 rows=3 loops=4)		

saby

Связанные записи

```
SELECT DISTINCT ON(relname) -- вывести первый из индексов по таблице
  *
FROM
  pg_class
JOIN
  pg_index
    ON pg_index.indrelid = pg_class.oid
WHERE
  (relkind, relnamespace) = ('r', 'pg_catalog'::regnamespace)
ORDER BY
  relname, indexrelid
LIMIT 16;
```

saby

Связанные записи

#	node, ms	tree, ms	rows	RRbF		Sort	Hash Join	Seq	H	Seq Scan on pg_c	sh.ht
		.402	16	363		итоговые результаты					19
0	.003	.402	16			Limit					
1	.007	.399	16			-> Unique					
2	.148	.392	37			-> Sort					
3	.104	.244	124			-> Hash Join					
4	.030		171			-> Seq Scan on pg_index					5
5	.012	.110	64			-> Hash					
6	.098		64	363	85.0%	SR -> Seq Scan on pg_class					14

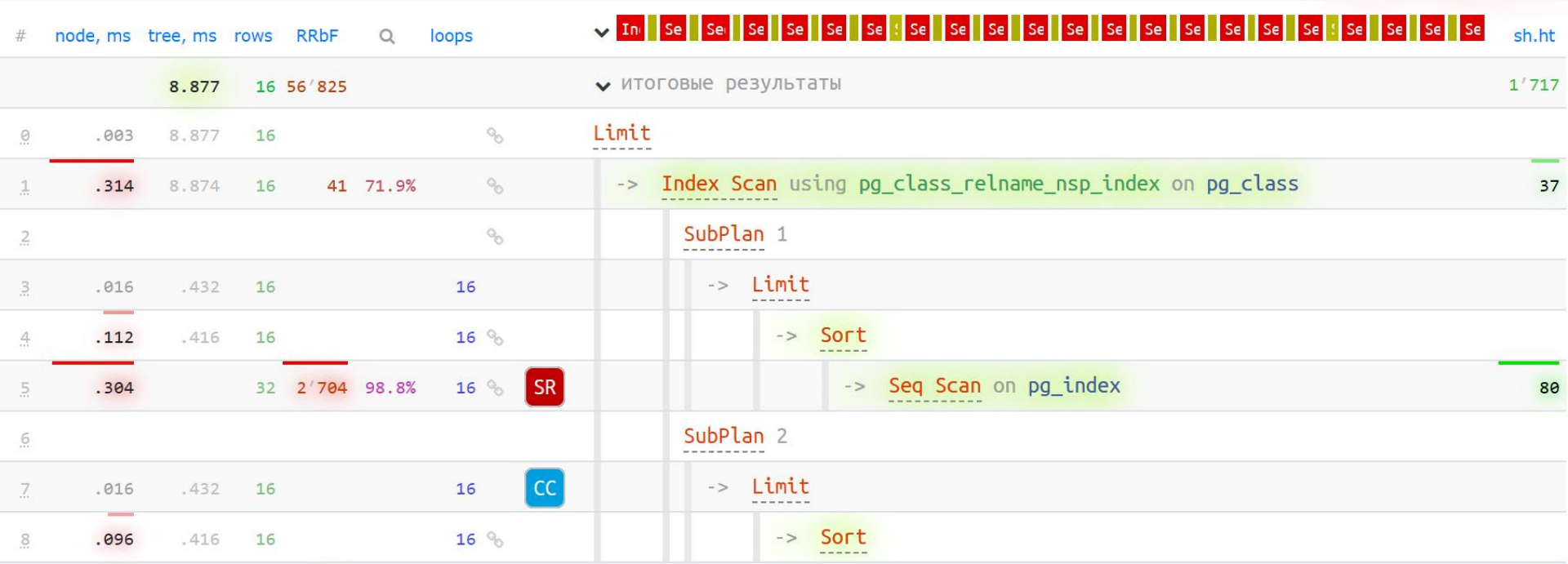
saby

Связанные записи

```
SELECT
  *
, (
    SELECT
      pg_index -- возвращаем цельную запись таблицы
    FROM
      pg_index
    WHERE
      indrelid = pg_class.oid
    ORDER BY
      indexrelid
    LIMIT 1
  ).* -- «разворачиваем» запись «в поля»
FROM
  pg_class
WHERE
  (relkind, relnamespace) = ('r', 'pg_catalog'::regnamespace)
ORDER BY
  relname
LIMIT 16;
```

saby

Связанные записи



PostgreSQL Antipatterns: «где-то я тебя уже видел...»

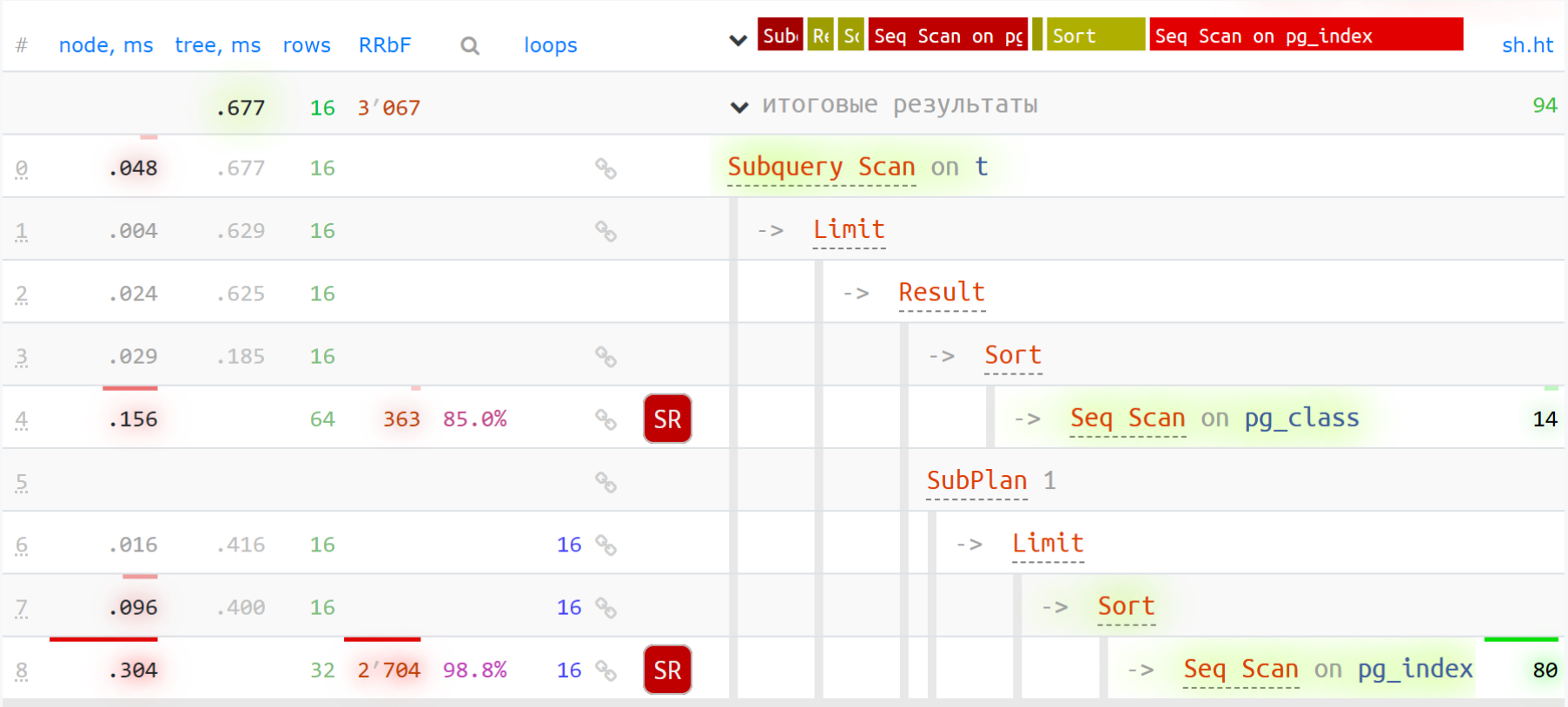
saby

Связанные записи

```
WITH T AS (
  SELECT
    pg_class   -- возвращаем цельную запись таблицы
  , (
      ...
    ) pg_index -- возвращаем единой записью, без «разворота»
  FROM
    pg_class
  WHERE
    (relkind, relnamespace) = ('r', 'pg_catalog'::regnamespace)
  ORDER BY
    relname
  LIMIT 16
)
SELECT
  (pg_class).* -- «разворачиваем» только тут
, (pg_index).*
FROM
  T;
```

saby

Связанные записи



saby

Связанные записи

```
SELECT
  *
FROM
  pg_class
, LATERAL (
    SELECT
      *
    FROM
      pg_index
    WHERE
      indrelid = pg_class.oid
    ORDER BY
      indexrelid
    LIMIT 1
  )
WHERE
  (relkind, relnamespace) = ('r', 'pg_catalog'::regnamespace)
ORDER BY
  relname
LIMIT 16;
```

saby

Связанные записи

#	node, ms	tree, ms	rows	RRbF	loops	Nested Lo	Index Scan using pg_cl	Limit	Sort	Bitmap Heap	Bitmap Inde	sh.ht
		.282	16	41		итоговые результаты						73
0	.004	.282	16			Limit						
1	.026	.278	16			-> Nested Loop						
2	.060		16	41	71.9%	-> Index Scan using pg_class_relname_nsp_index on pg_class						37
3	.016	.192	16		16	-> Limit						
4	.112	.176	16		16	-> Sort						
5	.032	.064	32		16	-> Bitmap Heap Scan on pg_index						20
6	.032		32		16	-> Bitmap Index Scan on pg_index_indrelid_index						16

saby

Связанные записи

Метод	время, ms	%
JOIN + DISTINCT ON	0.402	+42.5
(record).*	8.877	31.5x
CTE + (record).*	0.677	2.4x
LATERAL	0.282	---

saby

PG BootCamp Russia 2024 Kazan

PGBootCamp.ru

Появились вопросы?

Кирилл Боровиков

Компания «Тензор», технический директор

kilor@tensor.ru