PostgresPro

PG_QUERY_STATE: Всё о внутренней жизни ваших запросов

Соколова Екатерина Витальевна

Почему так долго?

```
WITH RECURSIVE t(n) AS (
        VALUES (1)
        UNION ALL
        SELECT n+1 FROM t
        WHERE n < 1000000000
)
SELECT sum(n) FROM t;
```

Долгое выполнение

```
SELECT my_table.*
FROM some_table,
        some_table AS my_table
GROUP BY my_table.c1;
```

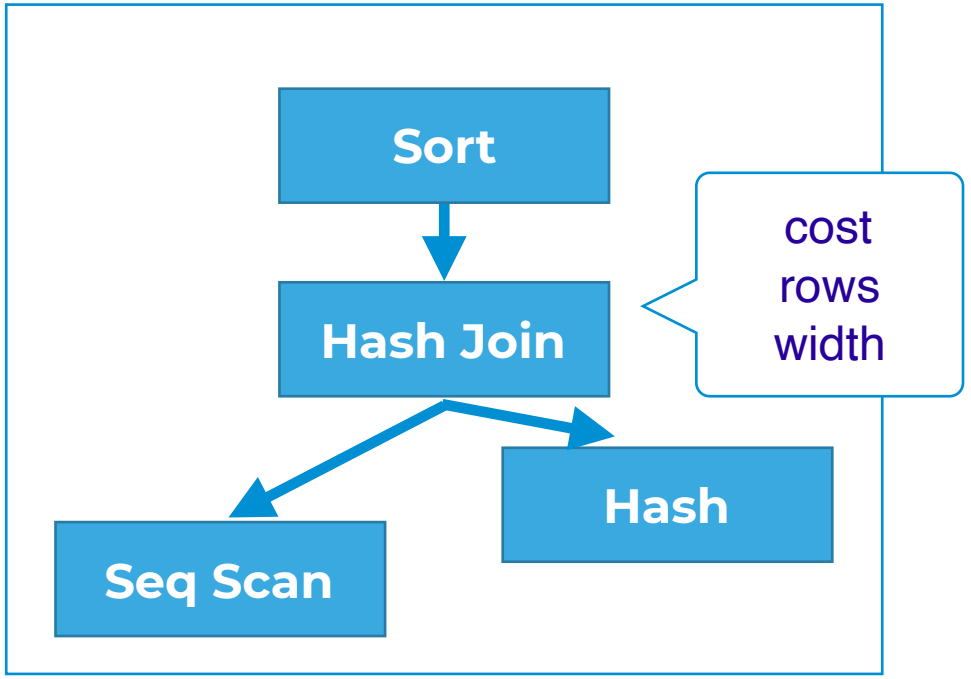
Ошибка в логике построения запроса

Доступная информация о запросе

EXPLAIN | **Выполнение QUERY**

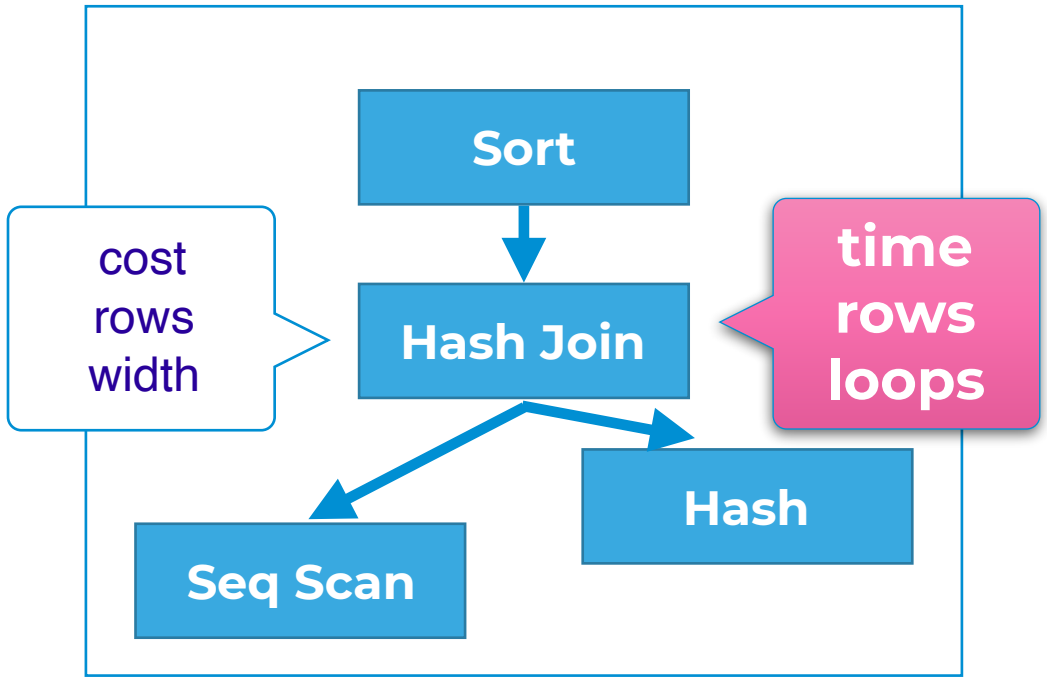
EXPLAIN

- дерево планировщика
- предполагаемое количество обрабатываемых строк в каждом узле плана
- стоимость операций



Доступная информация о запросе

EXPLAIN | Выполнение QUERY | EXPLAIN ANALYSE

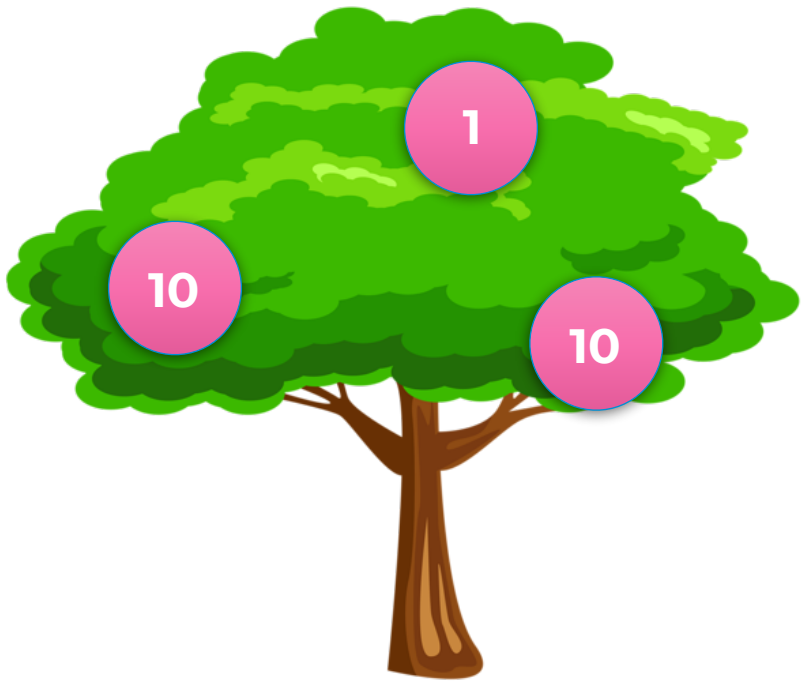


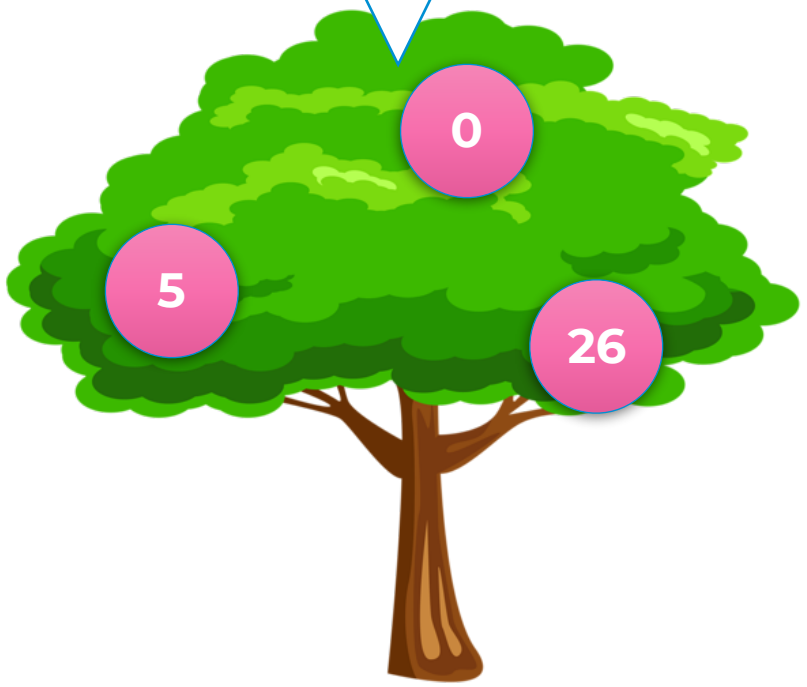
EXPLAIN ANALYSE

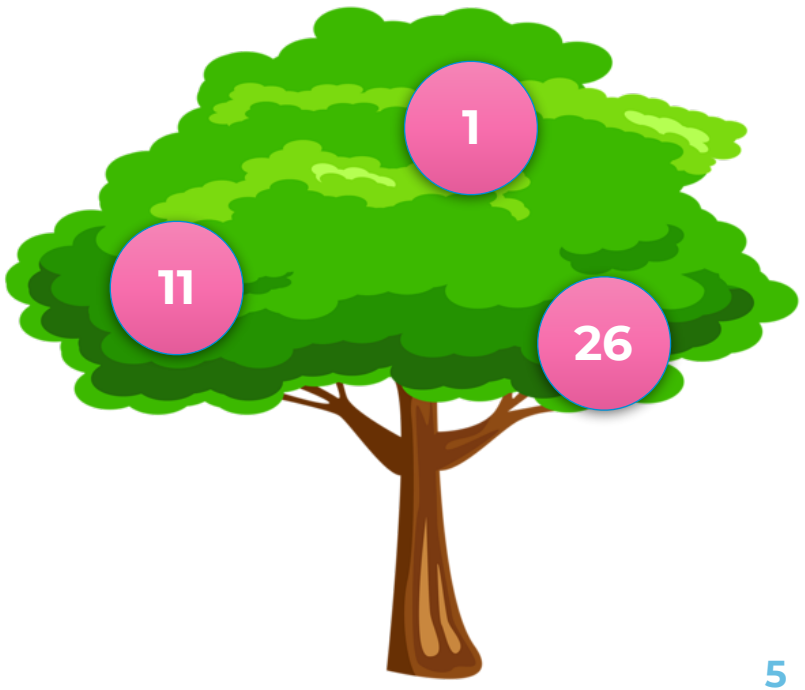
- дерево планировщика
- и предполагавшееся, и реальноеое количество обработанных строк в каждом узле плана
- время выполнения

Что такое pg_query_state?

EXPLAIN | Выполнение QUERY | EXPLAIN ANALYSE







Инструменты мониторинга в реальном времени

pg_stat_activity

pg_stat_statements

pgpro_stats

auto_explain

pg_stat_progress_*

pgpro_pwr

Инструменты мониторинга в реальном времени

pg_stat_activity

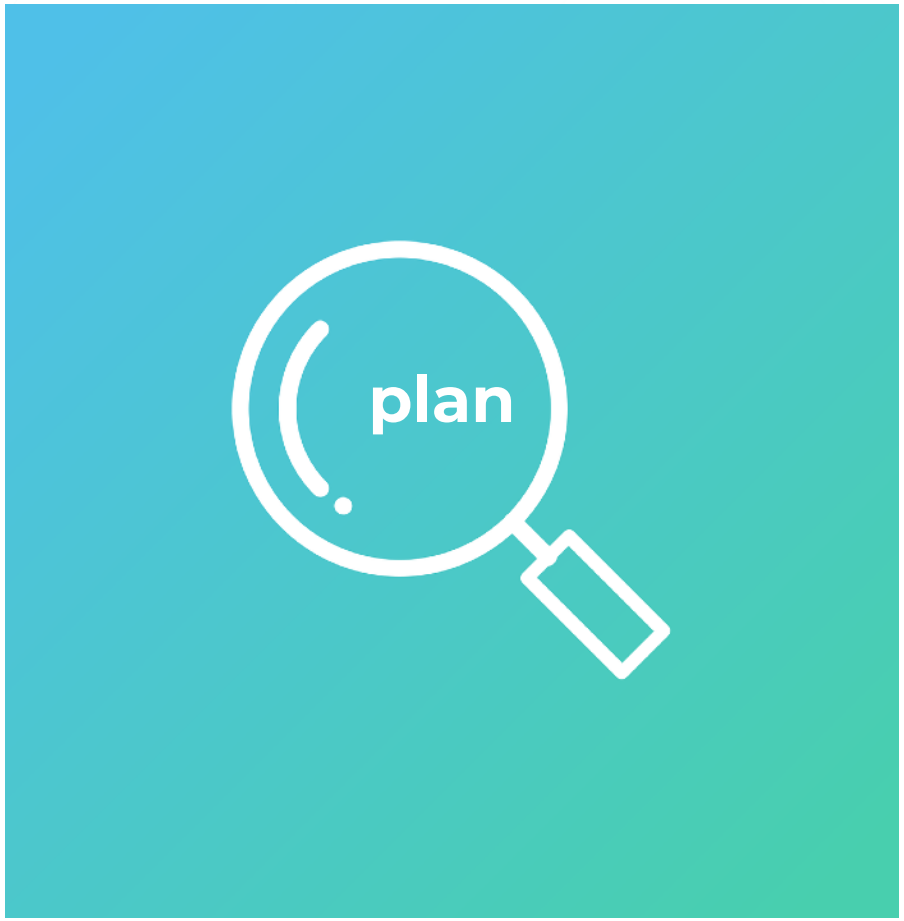
pg_stat_statements

- что происходит внутри запроса?
- впервые запущенный запрос

Инструменты мониторинга в реальном времени

- только если запрос уже выполнялся
- возможность изменения плана при широком разбросе входных данных
- нет информации о текущем состоянии

pgpro_stats



auto_explain

- только после завершения выполнения

sr_plan

- для заранее сохранённых

Инструменты мониторинга в реальном времени

- ANALYZE
- CREATE INDEX
- VACUUM
- CLUSTER
- Base Backup

pg_stat_progress_*

Параметры pg_query_state

```
postgres=# select pg_backend_pid();
pg_backend_pid
----------------
         49265
(1 row)
postgres=# select count(*) from foo;
```

или в другом бэкенде

```
postgres=# SELECT pid FROM pg_stat_activity
                 WHERE query LIKE 'select count%';
   pid   |          query
---------+--------------------------
 49265 | select count(*) from foo;
 49267 | select count(*) from bar;
(2 rows)
```

SELECT / INSERT / UPDATE / DELETE

pid

format
DEFAULT 'text'
xml, json, yaml

verbose
costs
timing
buffers
triggers

DEFAULT FALSE

Вывод pg_query_state

SELECT / INSERT / UPDATE / DELETE

pid

текст запроса
query_text

план запроса
(дерево)
plan

leader_pid

frame_number

```
postgres=# select * from pg_query_state(49265);
  pid   | frame_number |                query_text                |                                  plan                                  | leader_pid
--------+--------------+------------------------------------------+------------------------------------------------------------------------+------------
 49265 |            0 | insert into foo                         +| Insert on foo (Current loop: actual rows=0, loop number=1)            +|
       |              | select generate_series(1,10000000);      |   ->  ProjectSet (Current loop: actual rows=357081, loop number=1)+|
       |              |                                          |         ->  Result (Current loop: actual rows=1, loop number=1)       |
(1 row)
```

Пример
workers > 1

```
postgres=# SET
max_parallel_workers_per_gather = 2;
postgres=# select pg_backend_pid();
 pg_backend_pid
----------------
          49265
(1 row)
postgres=# SELECT count(*) FROM
foo JOIN bar ON foo.c1 = bar.c1;
```

```
postgres=# select pid, query_text, plan, leader_pid from pg_query_state(49265);
-[ RECORD 1 ]+-----------------------------------------------------------------------------------------------
pid          | 49265
query_text   | select count(*) from foo join bar on foo.c1=bar.c1;
plan         | Finalize Aggregate (Current loop: actual rows=0, loop number=1)
             |   ->  Gather (Current loop: actual rows=0, loop number=1)
             |         Workers Planned: 2
             |         Workers Launched: 2
             |         ->  Partial Aggregate (Current loop: actual rows=0, loop number=1)
             |               ->  Nested Loop (Current loop: actual rows=12, loop number=1)
             |                     Join Filter: (foo.c1 = bar.c1)
             |                     Rows Removed by Join Filter: 5673232
             |                     ->  Parallel Seq Scan on foo (Current loop: actual rows=12, loop number=1)
             |                     ->  Seq Scan on bar (actual rows=500000 loops=11)
             |                                       (Current loop: actual rows=173244, loop number=12)
leader_pid   | (null)
-[ RECORD 2 ]+-----------------------------------------------------------------------------------------------
pid          | 49324
query_text   | <parallel query>
plan         | Partial Aggregate (Current loop: actual rows=0, loop number=1)
             |   ->  Nested Loop (Current loop: actual rows=10, loop number=1)
             |         Join Filter: (foo.c1 = bar.c1)
             |         Rows Removed by Join Filter: 4896779
             |         ->  Parallel Seq Scan on foo (Current loop: actual rows=10, loop number=1)
             |         ->  Seq Scan on bar (actual rows=500000 loops=9)
             |                                 (Current loop: actual rows=396789, loop number=10)
leader_pid   | 49265
-[ RECORD 3 ]+-----------------------------------------------------------------------------------------------
pid          | 49323
query_text   | <parallel query>
plan         |  • • •
leader_pid   | 49265
```

Пример

frame_number > 0

```
postgres=# select
pg_backend_pid();
 pg_backend_pid
----------------
          49265
(1 row)

postgres=# select
n_join_foo_bar();
```

```
postgres=# select * from pg_query_state(49265);
-[ RECORD 1 ] +-----------------------------------------------------------------------------------------------------------
pid           | 49265
frame_number | 0
query_text    | select n_join_foo_bar();
plan          | Result (Current loop: actual rows=0, loop number=1)
leader_pid    | (null)
-[ RECORD 2 ] +-----------------------------------------------------------------------------------------------------------
pid           | 49265
frame_number | 1
query_text    | SELECT (select count(*) from foo join bar on foo.c1=bar.c1)
plan          | Result (Current loop: actual rows=0, loop number=1)
              |   InitPlan 1 (returns $0)
              |     ->  Aggregate (Current loop: actual rows=0, loop number=1)
              |           ->  Nested Loop (Current loop: actual rows=51, loop number=1)
              |                 Join Filter: (foo.c1 = bar.c1)
              |                 Rows Removed by Join Filter: 51636304
              |                 ->  Seq Scan on bar (Current loop: actual rows=52, loop number=1)
              |                 ->  Materialize (actual rows=1000000 loops=51)
              |                       (Current loop: actual rows=636355, loop number=52)
              |                       ->  Seq Scan on foo
              |                             (Current loop: actual rows=1000000, loop number=1)
leader_pid    | (null)
```

Установка

github.com/postgrespro/pg_query_state

1) Выполнить в директории модуля:

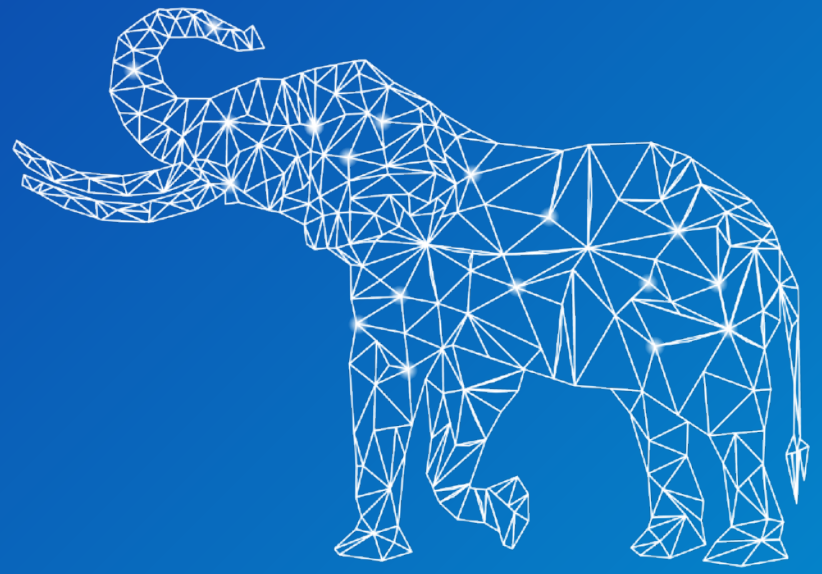
```
make install USE_PGXS=1
```

2) Добавить в postgresql.conf:

```
shared_preload_libraries = 'pg_query_state'
```

3) Выполнить в psql:

```
CREATE EXTENSION pg_query_state;
```



0) Применить runtime_explain.patch и custom_signal.patch

С чем это едят?

Возможности для дальнейшего развития

Представления pg_stat_activity, pg_stat_statements, pg_stat_progress_*

Визуализатор дерева планировщика

IDE для PostgreSQL

Прогресс бар выполнения запроса

Как рассчитать степень выполнения?

Полученное число строк

- Время выполнения
- Стоимость операций
- Количество обработанных узлов
- …

Как рассчитать степень выполнения?

Полученное число строк / ожидаемое число строк

- Если реальное количество превысило ожидаемое, степень выполнения узла (за исключением вершины дерева) = 1
- 100% только после завершения выполнения запроса
- Узлы равноценны, вне зависимости от того, на какой «высоте» они расположены
- Не все узлы стоит учитывать
- Узел Filter имеет состояния «выполнен» (1) или «не выполнен» (0)

Progress bar

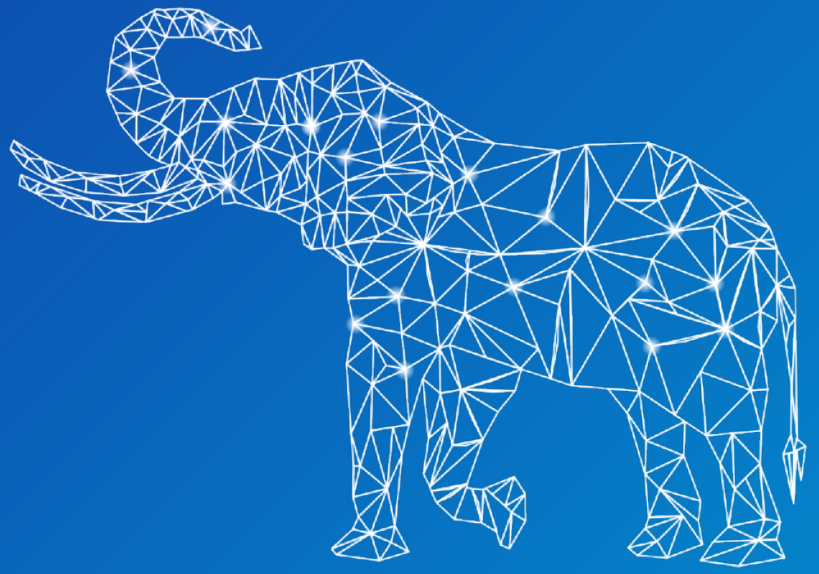
github.com/postgrespro/pg_query_state/tree/progress_bar

1) progress_bar (pid)

   Возвращает действительное число от 0 до 1 — степень завершённости запроса

2) progress_bar_visual (pid, delay)

   Через каждые delay секунд печатает степень завершённости запроса



Спасибо за внимание

Вопросы?

Соколова Екатерина

e.sokolova@postgrespro.ru

github.com/postgrespro/pg_query_state

PostgresPro

www.postgrespro.ru