PG BootCamp Russia 2025 Ekaterinburg

PGBootCamp.ru

Работа с запросами с точки зрения DBA

Александр Никитин. DBA Team.

Немного обо мне

- С базами начал работать где-то в 2001 г. (MS FoxPro, FireBird, Oracle, PostgreSQL).
- DBA с 2014 года.
- С 2020 г. выступаю на крупных конференциях.
- В консалтинге с 2021 года.

Содержание

- Инструменты
- Примеры использования
- Выводы

Инструменты

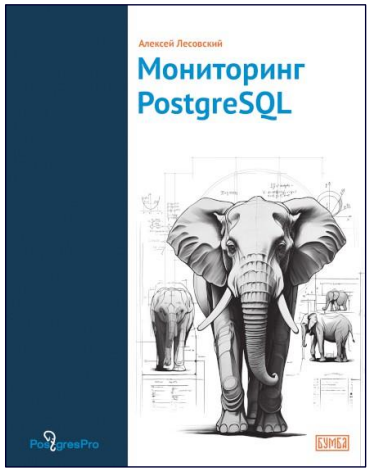
Инструменты

pg_stat_statements – расширение (что это?) PostgreSQL

psql: alter system set shared_preload_libraries='pg_stat_statements';

psql: create extension pg_stat_statements;

Рестарт PostgreSQL



https://www.postgresql.org/docs/17/pgstatstatements.html

https://postgrespro.ru/education/books/monitoring

Инструменты

https://github.com/dataegret/pg-utils - набор утилит DBA

```
git clone https://github.com/dataegret/pg-utils ~/stuff
```

Утилиты

Доклад

Инструменты

psql -f ~/stuff/sql/globals/query_stat_total_N.sql

```
total time:     25:42:36 (IO: 0.99%)
total queries:  77,966,682 (unique: 2,103)
report for all databases, version 0.9.5 @ PostgreSQL 11.7
tracking top 10000 queries, utilities on, logging 50ms+ queries
```

Инструменты

psql -f ~/stuff/sql/globals/query_stat_total_N.sql

```
 total time:      25:42:36 (IO: 0.99%)
 total queries:   77,966,682 (unique: 2,103)
 report for all databases, version 0.9.5 @ PostgreSQL 11.7
 tracking top 10000 queries, utilities on, logging 50ms+ queries

========================================================
pos:1    total time: 11:07:00 (43.2%, CPU: 43.7%, IO: 0.0%)       calls: 4,434 (0.01%)
    avg_time: 9025.74ms (IO: 0.0%)
 user: user1      db: db_name    rows: 12,817 (0.00%)     query:
 SELECT tote_location, COUNT(tote_location) FROM (SELECT * FROM (SELECT B.tote_name,
 B.tote_state, (CASE B.tote_location
     WHEN $1….
```

Инструменты

```
2024-09-29 00:00:00.686 MSK 105770 user@db_name from 10.13.19.81 [vxid:53/174763983
 txid:0] [SELECT] LOG:  duration: 100.494 ms  execute S_17: select totetransp0_.id as
 id1_1_, totetransp0_.created as created2_1_, totetransp0_.modified as modified3_1_,
 totetransp0_.version as version4_1_, totetransp0_.active_date as active_d5_1_,
 totetransp0_.complete_status as complete6_1_, totetransp0_.current_location_id as
 current15_1_, totetransp0_.destination_location_id as destina16_1_,
 totetransp0_.error_reason as error_re7_1_, totetransp0_.finished_date as finished8_1_,
 totetransp0_.forcedfinish as forcedfi9_1_, totetransp0_.number as number10_1_,
 totetransp0_.original_destination_location_id as origina17_1_,
 totetransp0_.source_location_id as source_18_1_, totetransp0_.status as status11_1_,
 totetransp0_.tote_number as tote_nu12_1_, totetransp0_.transport_batch_number as
 transpo13_1_, totetransp0_.type as type14_1_ from tote_transport_order totetransp0_ where
 totetransp0_.tote_number=$1 and (totetransp0_.status in ($2 , $3 , $4 , $5 , $6)) order
 by totetransp0_.modified desc limit $7
2024-09-29 00:00:00.686 MSK 105770 user@db_name from 10.13.19.81 [vxid:53/174763983
 txid:0] [SELECT] DETAIL:  parameters: $1 = 'T00000005212', $2 = 'CREATED', $3 = 'ACTIVE',
 $4 = 'FINISHED', $5 = 'CANCELLED', $6 = 'ERROR', $7 = '1'
```

Инструменты

```
2024-09-29 00:00:00.686 MSK 105770 user@db_name from 10.13.19.81 [vxid:53/174763983
 txid:0] [SELECT] LOG:  duration: 100.494 ms  execute S_17: select totetransp0_.id as
 id1_1_, totetransp0_.created as created2_1_, totetransp0_.modified as modified3_1_,
 totetransp0_.version as version4_1_, totetransp0_.active_date as active_d5_1_,
 totetransp0_.complete_status as complete6_1_, totetransp0_.current_location_id as
 current15_1_, totetransp0_.destination_location_id as destina16_1_,
 totetransp0_.error_reason as error_re7_1_, totetransp0_.finished_date as finished8_1_,
 totetransp0_.forcedfinish as forcedfi9_1_, totetransp0_.number as number10_1_,
 totetransp0_.original_destination_location_id as origina17_1_,
 totetransp0_.source_location_id as source_18_1_, totetransp0_.status as status11_1_,
 totetransp0_.tote_number as tote_nu12_1_, totetransp0_.transport_batch_number as
 transpo13_1_, totetransp0_.type as type14_1_ from tote_transport_order totetransp0_ where
 totetransp0_.tote_number=$1 and (totetransp0_.status in ($2 , $3 , $4 , $5 , $6)) order
 by totetransp0_.modified desc limit $7
2024-09-29 00:00:00.686 MSK 105770 user@db_name from 10.13.19.81 [vxid:53/174763983
 txid:0] [SELECT] DETAIL:  parameters: $1 = 'T00000005212', $2 = 'CREATED', $3 = 'ACTIVE',
 $4 = 'FINISHED', $5 = 'CANCELLED', $6 = 'ERROR', $7 = '1'
```

Инструменты

https://github.com/Nikitin-Alexandr/utils/blob/main/replace_parameters

Инструменты

```
CREATE OR REPLACE FUNCTION replace_parameters (query text, parameters
    text) RETURNS TEXT
    AS $body$
DECLARE
     rec
     record;
     SQL
     text;
BEGIN
  SQL := '';
  FOR rec IN
    ( SELECT
        array_to_string(REGEXP_MATCHES(parameters, '(?<=\$)(\d+)(?=\ =\ )', 'g'), ';')
        AS arg_no, array_to_string(REGEXP_MATCHES(parameters, '(?<=\= )(.*?)(?=(,\
        \$\d+|$))', 'g'), ';') AS arg_val)
     LOOP
                SQL:=regexp_replace(query, '\$'||rec.arg_no||'(?!\d)',
                rec.arg_val,'g'); query := sql;
     END LOOP;
  return
sql; END;
$body$
LANGUAGE 'plpgsql';

/*Example: select replace_parameters($$query$$,$$parameters$$);*/
```

Примеры

- pg_stat_statements - расширение
  (https://www.postgresql.org/docs/17/pgstatstatements.html)
- https://github.com/dataegret/pg-utils - набор утилит DBA
- https://github.com/Nikitin-Alexandr/utils/blob/main/replace_parameters
  замена параметров

Примеры

Создание индекса

Примеры

```
select "id", "external_id", "party_id", "shop_id", "invoice_id", "payment_id"
from "payment"
where "status_created_at" >= cast('2018-09-30 21:00:00' as timestamp) and
      "status_created_at" < cast('2018-10-31 21:00:00' as timestamp) and
      "party_id" = '58cad298' and
      "shop_id" = 'de788f93' and
      "status" = 'captured'
order by "status_created_at" desc;
```

Примеры

```
select "id", "external_id", "party_id", "shop_id", "invoice_id", "payment_id"
from "payment"
where "status_created_at" >= cast('2018-09-30 21:00:00' as timestamp) and
      "status_created_at" < cast('2018-10-31 21:00:00' as timestamp) and
      "party_id" = '58cad298' and
      "shop_id" = 'de788f93' and
      "status" = 'captured'
order by "status_created_at" desc;
```

Примеры

```
select "id", "external_id", "party_id", "shop_id", "invoice_id", "payment_id"
from "payment"
where "status_created_at" >= cast('2018-09-30 21:00:00' as timestamp) and
      "status_created_at" < cast('2018-10-31 21:00:00' as timestamp) and
      "party_id" = '58cad298' and
      "shop_id" = 'de788f93' and
      "status" = 'captured'
order by "status_created_at" desc;
```

Какие ваши ~~доказательства~~ предложения?

Примеры

```
SELECT
  count(*) FILTER (WHERE party_id = '58cad298') cnt_party,
  count(*) FILTER (WHERE shop_id = 'de788f93') cnt_shop,
  count(*) FILTER (WHERE party_id = '58cad298' AND shop_id = 'de788f93') cnt_party_shop,
  count(*) FILTER (WHERE status_created_at >= '2020-12-31 21:00:00' AND status_created_at < '2021-04-09 21:00:00')
cnt_st_created,
  count(*) FILTER (WHERE status_created_at >= '2020-12-31 21:00:00' AND status_created_at < '2021-04-09 21:00:00' and
      "status" = 'captured') cnt_st_created_status,
  count(*) FILTER (WHERE shop_id = 'de788f93' AND status_created_at >= '2020-12-31 21:00:00' AND
      status_created_at < '2021-04-09 21:00:00') cnt_shop_created,
  count(*) FILTER (WHERE party_id = '58cad298' AND status_created_at >= '2020-12-31 21:00:00' AND
      status_created_at < '2021-04-09 21:00:00') cnt_party_created,
  count(*) FILTER (WHERE party_id = '58cad298' AND shop_id = 'de788f93' AND status_created_at >= '2020-12-31 21:00:00’
      AND status_created_at < '2021-04-09 21:00:00') all_filter
FROM payment \gx
```

Примеры

```
cnt_party                | 329 810 (rows)
cnt_shop                 | 326 743
cnt_party_shop           | 326 743
cnt_st_created           | 1 112 292
cnt_st_created_status    | 910 527
cnt_shop_created         | 10 635
cnt_party_created        | 10 783
all_filter               | 9 212
```

Примеры

Статистика

Представление: pg_stats

Параметр: default_statistics_target

Поля: null_frac, n_distinct, correlation

300*dst блоков и
из них 300*dst строк

null_frac – доля null значений

Если n_distinct > 0 – это количество уникальных значений

Если n_distinct < 0 – это доля уникальных значений

Correlation – корреляция между расположением данных на диске и однородностью их изменения. Если последующее всегда больше, предыдущего, то будет близко к 1. Если всегда меньше ближе к -1.

Примеры

```
select null_frac, n_distinct, correlation from pg_stats where tablename =
'payment'and attname = 'party_id' \gx
null_frac   | 0.000246667
n_distinct  | 498
correlation | 0.0966761

select null_frac, n_distinct, correlation from pg_stats where tablename =
'payment' and attname = 'shop_id' \gx
null_frac   | 0.000246667
n_distinct  | 1289
correlation | 0.236482
```

Примеры

- count(*) FILTER (WHERE column_name = column_value) — пример подсчёта
- pg_stats (null_frac, n_distinct > 0, n_distinct < 0, correlation) - представление со статистикой
- default_statistics_target - параметр (детализация статистики)

Примеры

OR в условии запроса

Примеры

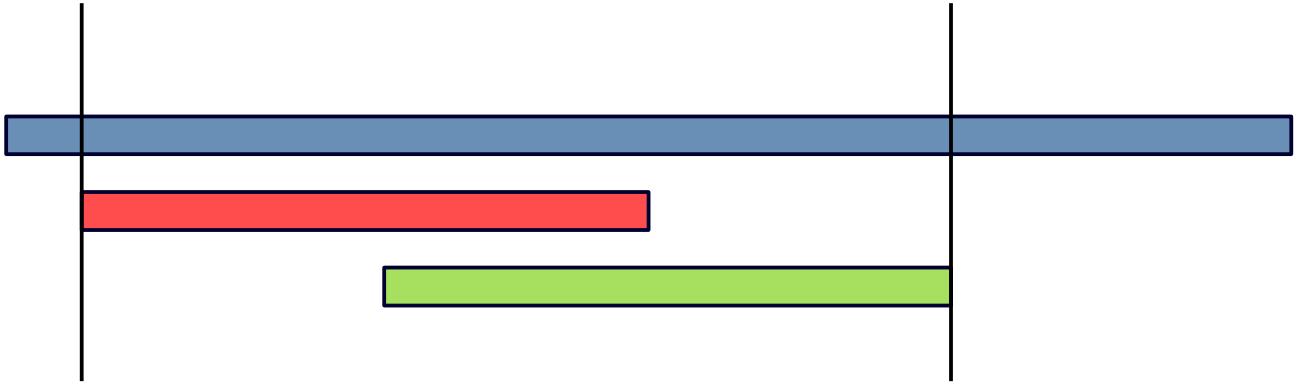
```
select * from tickets_ticket, auth_user
where (auth_user.id = tickets_ticket.buyer_id) and
((upper((username)::text) = '406240'::text) OR (tickets_ticket.buyer_id =
 406240));
```

Примеры

```
select * from tickets_ticket, auth_user
where (auth_user.id = tickets_ticket.buyer_id) and
((upper((username)::text) = '406240'::text) OR (tickets_ticket.buyer_id =
 406240));
```

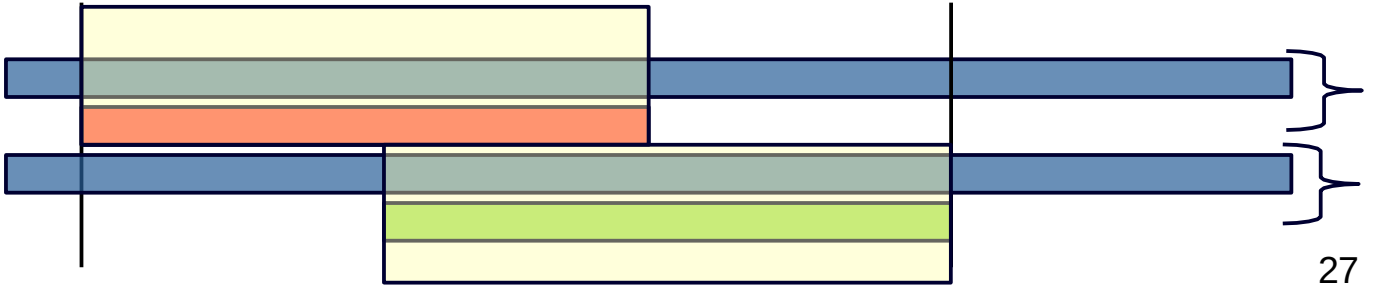
Примеры

```
select * from tickets_ticket, auth_user
where (auth_user.id = tickets_ticket.buyer_id) and
((upper((username)::text) = '406240'::text) OR (tickets_ticket.buyer_id =
 406240));
Execution Time: 814.906 ms
```



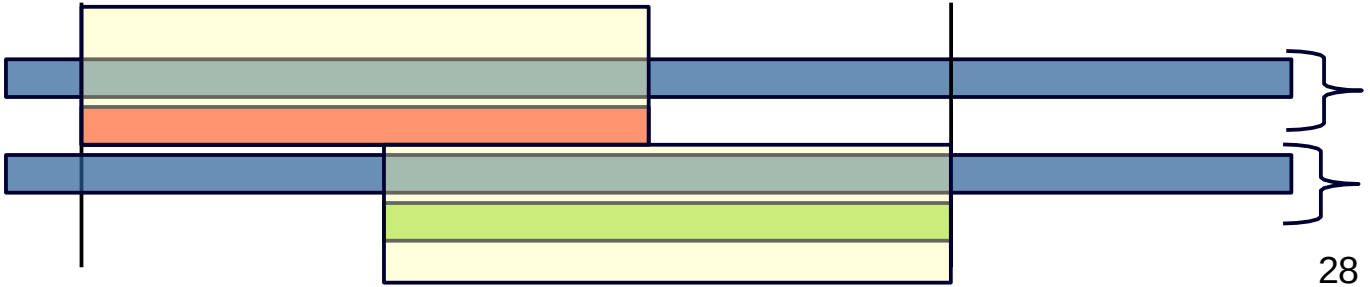
Примеры

```
select * from tickets_ticket, auth_user
where (auth_user.id = tickets_ticket.buyer_id) and
((upper((username)::text) = '406240'::text) OR (tickets_ticket.buyer_id =
 406240));
Execution Time: 814.906 ms

select * from tickets_ticket, auth_user
where auth_user.id = tickets_ticket.buyer_id  and upper((username)::text) =
 '406240'::text
union
select * from tickets_ticket, auth_user
where auth_user.id = tickets_ticket.buyer_id  and tickets_ticket.buyer_id =
 406240;
```

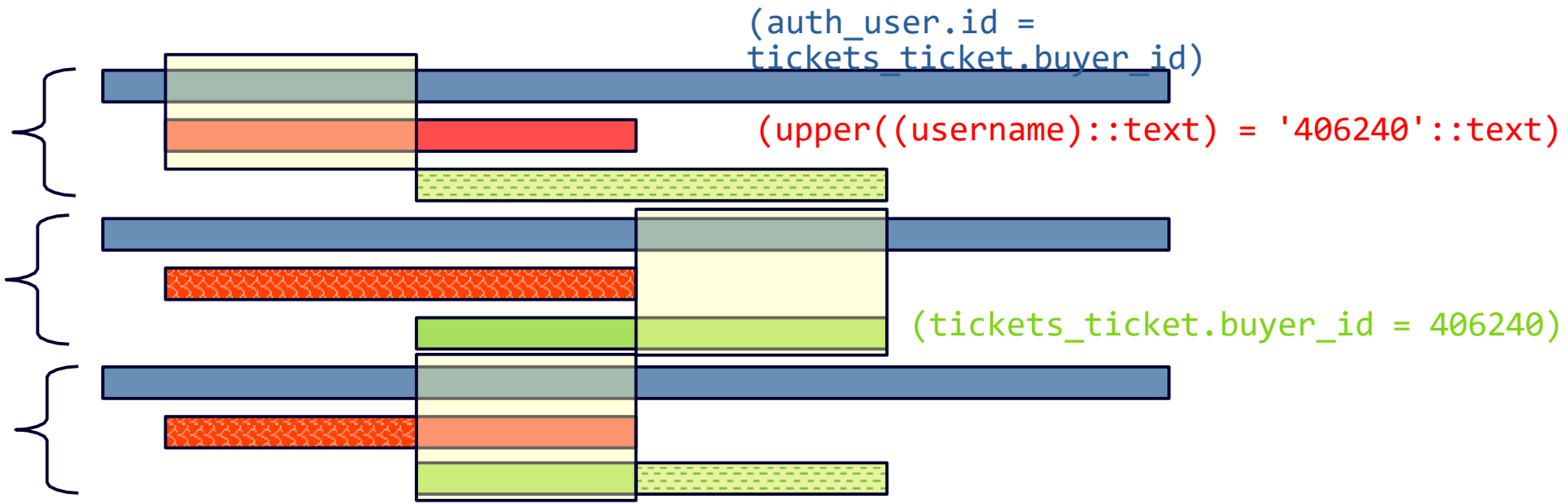


Примеры

```
select * from tickets_ticket, auth_user
where (auth_user.id = tickets_ticket.buyer_id) and
((upper((username)::text) = '406240'::text) OR (tickets_ticket.buyer_id =
 406240));
Execution Time: 814.906 ms

select * from tickets_ticket, auth_user
where auth_user.id = tickets_ticket.buyer_id  and upper((username)::text) =
 '406240'::text
union
select * from tickets_ticket, auth_user
where auth_user.id = tickets_ticket.buyer_id  and tickets_ticket.buyer_id =
 406240;
Execution Time: 76.741 ms
```



Примеры


(auth_user.id =
tickets_ticket.buyer_id)
(upper((username)::text) = '406240'::text)
(tickets_ticket.buyer_id = 406240)

Примеры

```
select * from tickets_ticket,auth_user
where auth_user.id = tickets_ticket.buyer_id and upper((username)::text) =
 '406240'::text and tickets_ticket.buyer_id != 406240
UNION ALL
select * from tickets_ticket,auth_user
where auth_user.id = tickets_ticket.buyer_id and tickets_ticket.buyer_id =
 406240 and upper((username)::text) != '406240'::text
UNION ALL
select * from tickets_ticket,auth_user
where auth_user.id = tickets_ticket.buyer_id and tickets_ticket.buyer_id =
 406240 and upper((username)::text) = '406240'::text;
Execution Time: 28.147 ms
```

Примеры

```
select * from tickets_ticket,auth_user
where auth_user.id = tickets_ticket.buyer_id and upper((username)::text) =
 '406240'::text and tickets_ticket.buyer_id != 406240
UNION ALL
select * from tickets_ticket,auth_user
where auth_user.id = tickets_ticket.buyer_id and tickets_ticket.buyer_id =
 406240 and upper((username)::text) != '406240'::text
UNION ALL
select * from tickets_ticket,auth_user
where auth_user.id = tickets_ticket.buyer_id and tickets_ticket.buyer_id =
 406240 and upper((username)::text) = '406240'::text;
Execution Time: 28.147 ms

До модификации:
Execution Time: 814.906 ms – ускорение ~ 30 раз.
```

Примеры

- Запросы, содержащие OR, можно переписать на ряд запросов с использованием UNION ALL

Примеры

Distinct + join

Примеры

```
select distinct tbl1.id
from tbl1
  inner join tbl2 on tbl1.id_pp = tbl2.id_pp;
```

Примеры

```
select distinct tbl1.id
from tbl1
  inner join tbl2 on tbl1.id_pp = tbl2.id_pp;

select id
from tbl1
where exists (
  select 1
  from tbl2
  where tbl1.id_pp = tbl2.id_pp);
```

https://habr.com/ru/company/tensor/blog/513324/

Примеры

```
create table t1 (id int, t text);
insert into t1 values(1,'a'), (1,'a'), (2,'b'), (3,'c');
create table t2 (id int, val text);
insert into t2 values (1,'true'),(2,'true'),(3,'false');
select distinct t1.* from t1 inner join t2 on t1.id = t2.id and t2.val = 'true’;
  1 | a
  2 | b
select t1.* from t1 where exists (select 1 from t2 where t1.id = t2.id and t2.val =
    'true’);
  1 | a
  1 | a
  2 | b
```

Примеры

Проверка эквивалентности запросов:

```
select * from (original_query) a
except
select * from (rewritten_query) b;
```

Примеры

Проверка эквивалентности запросов:

```
select * from (original_query) a
except
select * from (rewritten_query) b;


select * from (rewritten_query) b
except
select * from (original_query) a;
```

Примеры

- distinct+join — за это сочетание должен цепляться взгляд при анализе.

Примеры

Работа со статистикой

Примеры

```
Index Scan using personal_grades_reference_id_key on personal_grades g  (cost=0.42..2882542.35 rows=1 width=32) (actual time=572.884..572.898 rows=1 loops=1)
  Index Cond: ((student_id = '623611ef-3c03-42da-80c5-91925cddd720'::uuid) AND (reference_id = 52))
  Buffers: shared hit=183 read=24 dirtied=1
  I/O Timings: read=0.452
  SubPlan 4
    ->  Aggregate  (cost=2882541.67..2882541.68 rows=1 width=32) (actual time=4.523..4.533 rows=1 loops=1)
          Buffers: shared hit=176 read=24
          I/O Timings: read=0.452
          ->  Subquery Scan on t  (cost=2882541.50..2882541.62 rows=10 width=848) (actual time=3.730..3.742 rows=1 loops=1)
                Buffers: shared hit=176 read=24
                I/O Timings: read=0.452
                ->  Sort  (cost=2882541.50..2882541.52 rows=10 width=856) (actual time=3.571..3.580 rows=1 loops=1)
                      Sort Key: t_1."position"
                      Sort Method: quicksort  Memory: 201kB
                      Buffers: shared hit=176 read=24
                      I/O Timings: read=0.452
                      ->  Index Scan using personal_topics_grade_id_key on personal_topics t_1  (cost=0.43..2882541.33 rows=10 width=856) (actual time=3.386..3.398 rows=1 loops=1)
                            Index Cond: (grade_id = g.id)
                            Filter: (deleted IS FALSE)
                            Buffers: shared hit=173 read=24
                            I/O Timings: read=0.452
                            SubPlan 3
                              ->  Aggregate  (cost=288254.02..288254.03 rows=1 width=32) (actual time=2.937..2.945 rows=1 loops=1)
                                    Buffers: shared hit=170 read=23
                                    I/O Timings: read=0.219
                                    ->  Subquery Scan on ch  (cost=288253.60..288253.90 rows=24 width=335) (actual time=2.610..2.638 rows=5 loops=1)
                                          Buffers: shared hit=170 read=23
                                          I/O Timings: read=0.219
                                          ->  Sort  (cost=288253.60..288253.66 rows=24 width=343) (actual time=2.590..2.598 rows=5 loops=1)
                                                Sort Key: ch_1."position"
                                                Sort Method: quicksort  Memory: 199kB
                                                Buffers: shared hit=170 read=23
                                                I/O Timings: read=0.219
                                                ->  Index Scan using personal_chapters_topic_id_key on personal_chapters ch_1  (cost=0.43..288253.05 rows=24 width=343) (actual time=0.658..2.525 rows=5 loops=1)
                                                      Index Cond: (topic_id = t_1.id)
                                                      Filter: (deleted IS FALSE)
                                                      Buffers: shared hit=170 read=23
                                                      I/O Timings: read=0.219
                                                      SubPlan 2
                                                        ->  Aggregate  (cost=12010.48..12010.49 rows=1 width=32) (actual time=0.483..0.484 rows=1 loops=5)
                                                              Buffers: shared hit=166 read=22
                                                              I/O Timings: read=0.208
                                                              ->  Subquery Scan on l  (cost=12005.74..12009.12 rows=271 width=1579) (actual time=0.386..0.395 rows=5 loops=5)
                                                                    Buffers: shared hit=166 read=22
                                                                    I/O Timings: read=0.208
                                                                    ->  Sort  (cost=12005.74..12006.41 rows=271 width=1587) (actual time=0.380..0.381 rows=5 loops=5)
                                                                          Sort Key: l_1."position"
                                                                          Sort Method: quicksort  Memory: 57kB
                                                                          Buffers: shared hit=166 read=22
                                                                          I/O Timings: read=0.208
                                                                          ->  Index Scan using personal_lessons_chapter_id_key on personal_lessons l_1  (cost=0.56..11994.79 rows=271 width=1587) (actual time=0.110..0.366 rows=5 loops=5)
                                                                                Index Cond: (chapter_id = ch_1.id)
                                                                                Filter: (deleted IS FALSE)
                                                                                Buffers: shared hit=166 read=22
                                                                                I/O Timings: read=0.208
                                                                                SubPlan 1
                                                                                  ->  Aggregate  (cost=44.20..44.21 rows=1 width=32) (actual time=0.061..0.061 rows=1 loops=27)
                                                                                        Buffers: shared hit=134 read=21
                                                                                        I/O Timings: read=0.192
                                                                                        ->  Subquery Scan on c  (cost=36.90..42.11 rows=417 width=715) (actual time=0.018..0.022 rows=4 loops=27)
                                                                                              Buffers: shared hit=134 read=21
                                                                                              I/O Timings: read=0.192
                                                                                              ->  Sort  (cost=36.90..37.94 rows=417 width=691) (actual time=0.016..0.017 rows=4 loops=27)
                                                                                                    Sort Key: c_1."position"
                                                                                                    Sort Method: quicksort  Memory: 29kB
                                                                                                    Buffers: shared hit=134 read=21
                                                                                                    I/O Timings: read=0.192
                                                                                                    ->  Index Scan using personal_cards_lesson_id_key on personal_cards c_1  (cost=0.56..18.75 rows=417 width=691) (actual time=0.008..0.014 rows=4 loops=27)
                                                                                                          Index Cond: (lesson_id = l_1.id)
                                                                                                          Filter: (deleted IS FALSE)
                                                                                                          Buffers: shared hit=134 read=21
                                                                                                          I/O Timings: read=0.192
Planning Time: 3.397 ms
JIT:
  Functions: 50
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 6.664 ms, Inlining 105.226 ms, Optimization 290.538 ms, Emission 170.609 ms, Total 573.036 ms
Execution Time: 616.528 ms
(78 rows)
```

Примеры

```
Index Scan using personal_grades_reference_id_key on personal_grades g  (cost=0.42..2882542.35 rows=1 width=32) (actual time=572.884..572.898 rows=1 loops=1)
  Index Cond: ((student_id = '623611ef-3c03-42da-80c5-91925cddd720'::uuid) AND (reference_id = 52))
  Buffers: shared hit=183 read=24 dirtied=1
  I/O Timings: read=0.452
  SubPlan 4
    ->  Aggregate  (cost=2882541.67..2882541.68 rows=1 width=32) (actual time=4.523..4.533 rows=1 loops=1)
          Buffers: shared hit=176 read=24
          I/O Timings: read=0.452
          ->  Subquery Scan on t  (cost=2882541.50..2882541.62 rows=10 width=848) (actual time=3.730..3.742 rows=1 loops=1)
                Buffers: shared hit=176 read=24
                I/O Timings: read=0.452
                ->  Sort  (cost=2882541.50..2882541.52 rows=10 width=856) (actual time=3.571..3.580 rows=1 loops=1)
                      Sort Key: t_1."position"
                      Sort Method: quicksort  Memory: 201kB
                      Buffers: shared hit=176 read=24
                      I/O Timings: read=0.452
                      ->  Index Scan using personal_topics_grade_id_key on personal_topics t_1  (cost=0.43..2882541.33 rows=10 width=856) (actual time=3.386..3.398 rows=1 loops=1)
                            Index Cond: (grade_id = g.id)
                            Filter: (deleted IS FALSE)
                            Buffers: shared hit=173 read=24
                            I/O Timings: read=0.452
                            SubPlan 3
                              ->  Aggregate  (cost=288254.02..288254.03 rows=1 width=32) (actual time=2.937..2.945 rows=1 loops=1)
                                    Buffers: shared hit=170 read=23
                                    I/O Timings: read=0.219
                                    ->  Subquery Scan on ch  (cost=288253.60..288253.90 rows=24 width=335) (actual time=2.610..2.638 rows=5 loops=1)
                                          Buffers: shared hit=170 read=23
                                          I/O Timings: read=0.219
                                          ->  Sort  (cost=288253.60..288253.66 rows=24 width=343) (actual time=2.590..2.598 rows=5 loops=1)
                                                Sort Key: ch_1."position"
                                                Sort Method: quicksort  Memory: 199kB
                                                Buffers: shared hit=170 read=23
                                                I/O Timings: read=0.219
                                                ->  Index Scan using personal_chapters_topic_id_key on personal_chapters ch_1  (cost=0.43..288253.05 rows=24 width=343) (actual time=0.658..2.525 rows=5 loops=1)
                                                      Index Cond: (topic_id = t_1.id)
                                                      Filter: (deleted IS FALSE)
                                                      Buffers: shared hit=170 read=23
                                                      I/O Timings: read=0.219
                                                      SubPlan 2
                                                        ->  Aggregate  (cost=12010.48..12010.49 rows=1 width=32) (actual time=0.483..0.484 rows=1 loops=5)
                                                              Buffers: shared hit=166 read=22
                                                              I/O Timings: read=0.208
                                                              ->  Subquery Scan on l  (cost=12005.74..12009.12 rows=271 width=1579) (actual time=0.386..0.395 rows=5 loops=5)
                                                                    Buffers: shared hit=166 read=22
                                                                    I/O Timings: read=0.208
                                                                    ->  Sort  (cost=12005.74..12006.41 rows=271 width=1587) (actual time=0.380..0.381 rows=5 loops=5)
                                                                          Sort Key: l_1."position"
                                                                          Sort Method: quicksort  Memory: 57kB
                                                                          Buffers: shared hit=166 read=22
                                                                          I/O Timings: read=0.208
                                                                          ->  Index Scan using personal_lessons_chapter_id_key on personal_lessons l_1  (cost=0.56..11994.79 rows=271 width=1587) (actual time=0.110..0.366 rows=5 loops=5)
                                                                                Index Cond: (chapter_id = ch_1.id)
                                                                                Filter: (deleted IS FALSE)
                                                                                Buffers: shared hit=166 read=22
                                                                                I/O Timings: read=0.208
                                                                                SubPlan 1
                                                                                  ->  Aggregate  (cost=44.20..44.21 rows=1 width=32) (actual time=0.061..0.061 rows=1 loops=27)
                                                                                        Buffers: shared hit=134 read=21
                                                                                        I/O Timings: read=0.192
                                                                                        ->  Subquery Scan on c  (cost=36.90..42.11 rows=417 width=715) (actual time=0.018..0.022 rows=4 loops=27)
                                                                                              Buffers: shared hit=134 read=21
                                                                                              I/O Timings: read=0.192
                                                                                              ->  Sort  (cost=36.90..37.94 rows=417 width=691) (actual time=0.016..0.017 rows=4 loops=27)
                                                                                                    Sort Key: c_1."position"
                                                                                                    Sort Method: quicksort  Memory: 29kB
                                                                                                    Buffers: shared hit=134 read=21
                                                                                                    I/O Timings: read=0.192
                                                                                                    ->  Index Scan using personal_cards_lesson_id_key on personal_cards c_1  (cost=0.56..18.75 rows=417 width=691) (actual time=0.008..0.014 rows=4 loops=27)
                                                                                                          Index Cond: (lesson_id = l_1.id)
                                                                                                          Filter: (deleted IS FALSE)
                                                                                                          Buffers: shared hit=134 read=21
                                                                                                          I/O Timings: read=0.192
Planning Time: 3.397 ms
JIT:
  Functions: 50
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 6.664 ms, Inlining 105.226 ms, Optimization 290.538 ms, Emission 170.609 ms, Total 573.036 ms
Execution Time: 616.528 ms
(78 rows)
```

Примеры

```
->       Index Scan using personal_cards_lesson_id_key on personal_cards c_1
  (cost=0.56..18.75 rows=417 width=691) (actual time=0.008..0.014 rows=4
  loops=27)
Index Cond: (lesson_id = l_1.id) Filter:
(deleted IS FALSE) Buffers: shared
hit=134 read=21 I/O Timings: read=0.192

 Planning Time: 3.397 ms JIT:
   Functions: 50
   Options: Inlining true, Optimization true, Expressions true, Deforming true
   Timing: Generation 6.664 ms, Inlining 105.226 ms, Optimization 290.538 ms, Emission
   170.609 ms,
   Total 573.036 ms Execution Time:
 616.528 ms
```

Примеры

```
->         Index Scan using personal_cards_lesson_id_key on personal_cards c_1
  (cost=0.56..18.75 rows=417 width=691) (actual time=0.008..0.014 rows=4 loops=27)
Index Cond: (lesson_id = l_1.id)
Filter: (deleted IS FALSE) Buffers:
shared hit=134 read=21 I/O Timings:
read=0.192

 Planning Time: 3.397 ms
 JIT:
   Functions: 50
   Options: Inlining true, Optimization true, Expressions true, Deforming true
   Timing: Generation 6.664 ms, Inlining 105.226 ms, Optimization 290.538 ms, Emission 170.609 ms,
   Total 573.036 ms Execution
 Time: 616.528 ms
```

В PostgreSQL тоже есть JIT :)

Примеры

```
analyze personal_cards;
explain select distinct lesson_id from personal_cards;
                                QUERY PLAN

------------------------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=0.56..1257422.12 rows=151734 width=8)
   ->  Index Only Scan using personal_cards_lesson_id_key on personal_cards
 (cost=0.56..1092318.52 rows=66041439 width=8)

select count(*), count(distinct lesson_id) from personal_cards;
  count   |  count
----------+----------
 65844906 | 22326223
```

Примеры

```
analyze personal_cards;
explain select distinct lesson_id from personal_cards;
                                QUERY PLAN

------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=0.56..1257422.12 rows=151734 width=8)
   ->  Index Only Scan using personal_cards_lesson_id_key on personal_cards
 (cost=0.56..1092318.52 rows=66041439 width=8)

select count(*), count(distinct lesson_id) from personal_cards;
  count   |  count
----------+----------
 65844906 | 22326223
```

select count(*), count(distinct lesson_id) from personal_cards tablesample system (N);

N – проценты от всей таблицы

Примеры

```
select count(*), count(distinct lesson_id) from personal_cards;
  count   |  count
----------+----------
 65844906 | 22326223

select 22326223::float/65844906;
      ?column?
-------------------
 0.3390728965426726

alter table personal_cards alter lesson_id set (n_distinct=-0.4);

analyze personal_cards;
```

Примеры

```
select count(*), count(distinct lesson_id) from personal_cards;
  count   |  count
----------+----------
 65844906 | 22326223

select 22326223::float/65844906;
      ?column?
-------------------
 0.3390728965426726

alter table personal_cards alter lesson_id set (n_distinct=-0.4);

analyze personal_cards;
```

616 мс -> 4 мс

Примеры

- Мы можем переписать значение n_distinct исходя из наших представлений.
- Помним о том, что представляют из себя положительные и отрицательные значения в n_distinct.

Примеры

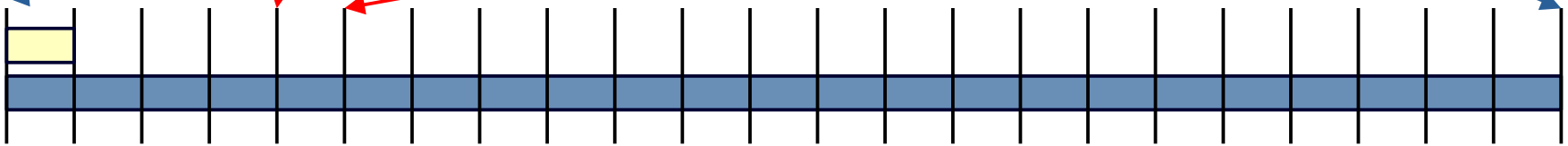
Пытаемся понять, что имел в виду автор запроса

Примеры

```
SELECT "id", "jdbc_id", ... FROM (
 SELECT
 t.id as jdbc_id,
 ...
 FROM
 tbl_1 as t
 WHERE created_at BETWEEN $5::date - interval $6 day AND $7::date) as x
WHERE "jdbc_id" >= $8 AND "jdbc_id" < $9;
```

Примеры

```
SELECT "id", "jdbc_id", ... FROM (
 SELECT
 t.id as jdbc_id,
 ...
 FROM
 tbl_1 as t
 WHERE created_at BETWEEN $5::date - interval $6 day AND $7::date) as x
WHERE "jdbc_id" >= $8 AND "jdbc_id" < $9;
```



Примеры

- Поймите, чего именно хотел добиться автор запроса, когда писал его.
- Подумайте, может быть вы можете решить эту задачу лучше.

Примеры

Сложный запрос

Примеры

```
SELECT COUNT(*)
FROM "_1_"
  INNER JOIN "_2_" ON "_2_"."id" = "_1_"."clb_id"
  INNER JOIN "_3_" ON "_3_"."id" = "_2_"."t_id"
WHERE "_3_"."s_id" IN
  (SELECT "_3_"."s_id"
  FROM "_1_"
       INNER JOIN "_2_" ON "_2_"."id" = "_1_"."clb_id"
       INNER JOIN "_3_" ON "_3_"."id" = "_2_"."t_id"
       INNER JOIN "_4_" ON "_1_"."id" = "_4_"."tmp_id"
  WHERE "_4_"."st_id" = 6251961 AND "_4_"."deleted_at" IS NULL)
   AND "_1_"."m_id" = 192 AND "_1_"."clb_id" IN
  (SELECT "_2_"."id"
  FROM "_2_"
  WHERE "_2_"."t_id" IN
       (SELECT "_3_"."id" FROM "_3_" WHERE "_3_"."s_id" = 262718)) AND ("_1_"."points" > 0)
```

Примеры

from_collapse_limit - Задаёт максимальное число элементов в списке FROM, до которого планировщик будет объединять вложенные запросы с внешним запросом.
При меньших значениях сокращается время планирования, но план запроса может стать менее эффективным.

join_collapse_limit - Задаёт максимальное количество элементов в списке FROM, до достижения которого планировщик будет сносить в него явные конструкции JOIN (за исключением FULL JOIN).
При меньших значениях сокращается время планирования, но план запроса может стать менее эффективным.

Примеры

```
from_collapse_limit/join_collapse_limit   = 8
    Execution Time: 640.367 ms
```

Примеры

```
from_collapse_limit/join_collapse_limit   = 8
    Execution Time: 640.367 ms

from_collapse_limit/join_collapse_limit   = 10
    Execution Time: 2.675 ms
```

https://habr.com/ru/company/postgrespro/blog/574702/

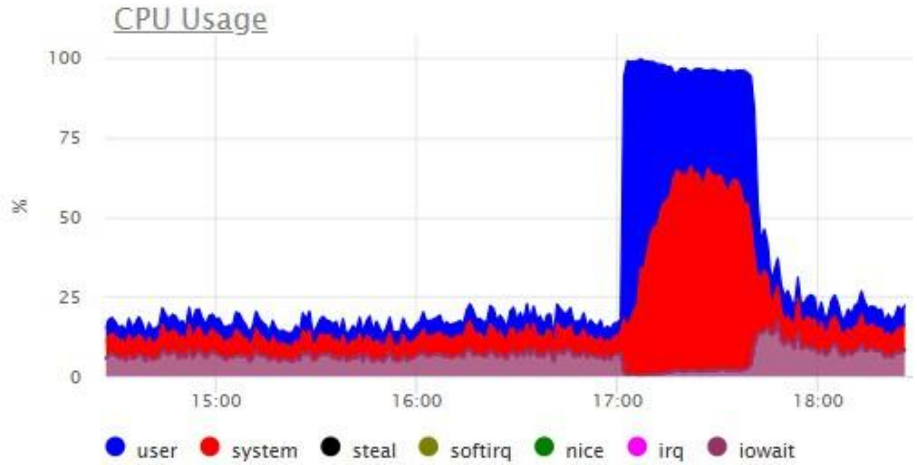
Примеры

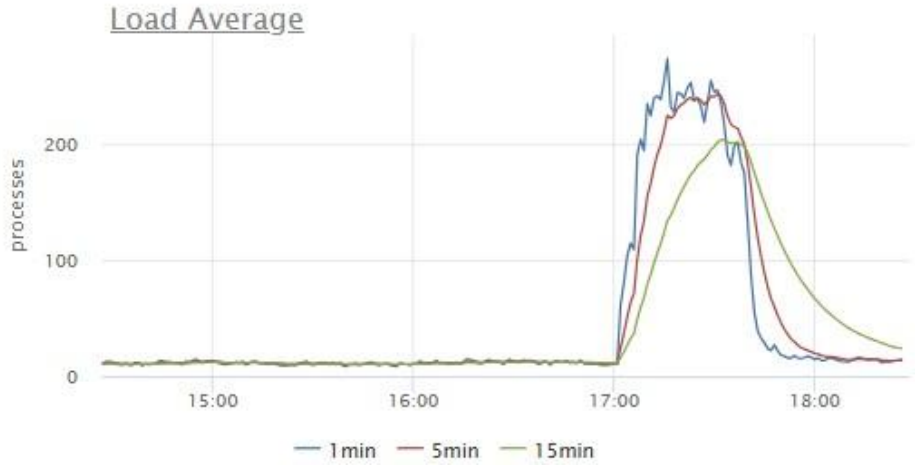
- from_collapse_limit и join_collapse_limit - опасные параметры.

Примеры

Американские горки

Примеры


CPU Usage
100
75
50
25
0
%
15:00
16:00
17:00
18:00
user
system
steal
softirq
nice
irq
iowait


Load Average
200
100
0
processes
15:00
16:00
17:00
18:00
1min
5min
15min

Примеры

- Бывают ситуации, когда ДБА может вам только указать на то, что есть проблема, но найти и устранить причину иногда можете только вы.

Выводы

1) Ваши друзья не только индексы.

Выводы

1) Ваши друзья не только индексы.

2) Не стесняйтесь смотреть в статистику.

Выводы

1) Ваши друзья не только индексы.

2) Не стесняйтесь смотреть в статистику.

3) Любите ваших ДБА :)

DBA Team

Ваши данные - наша забота!

- **Поддержка PostgreSQL 24/7**
- **Настройка кластеров конкретно под ваш профиль нагрузки;**
- **Помощь в оптимизации SQL-запросов;**
- **Выполнение миграций данных без простоя системы;**
- **Миграция БД на новые сервера или в другой ДЦ с минимальным простоем;**
- **Своевременное обновление PostgreSQL (как минорное, так и мажорное);**
- **Настройка резервного копирования и восстановления;**
- **Консультации по архитектуре и производительности БД;**
- **Проведение регулярных проверок ваших кластеров;**

PG BootCamp Russia 2025 Ekaterinburg

PGBootCamp.ru

Вопросы?

Александр Никитин

E-mail: contact@dba.team

tg: @anikitindba

Бонус: знаем команду DevOps.