

**PGConf.Russia 2023**



# Перепланирование запросов

**Алена Рыбакина**

[a.rybakina@postgrespro.ru](mailto:a.rybakina@postgrespro.ru)

**Участники проекта:**

**Дамир Белялов, Андрей Лепихов, Алена Рыбакина**

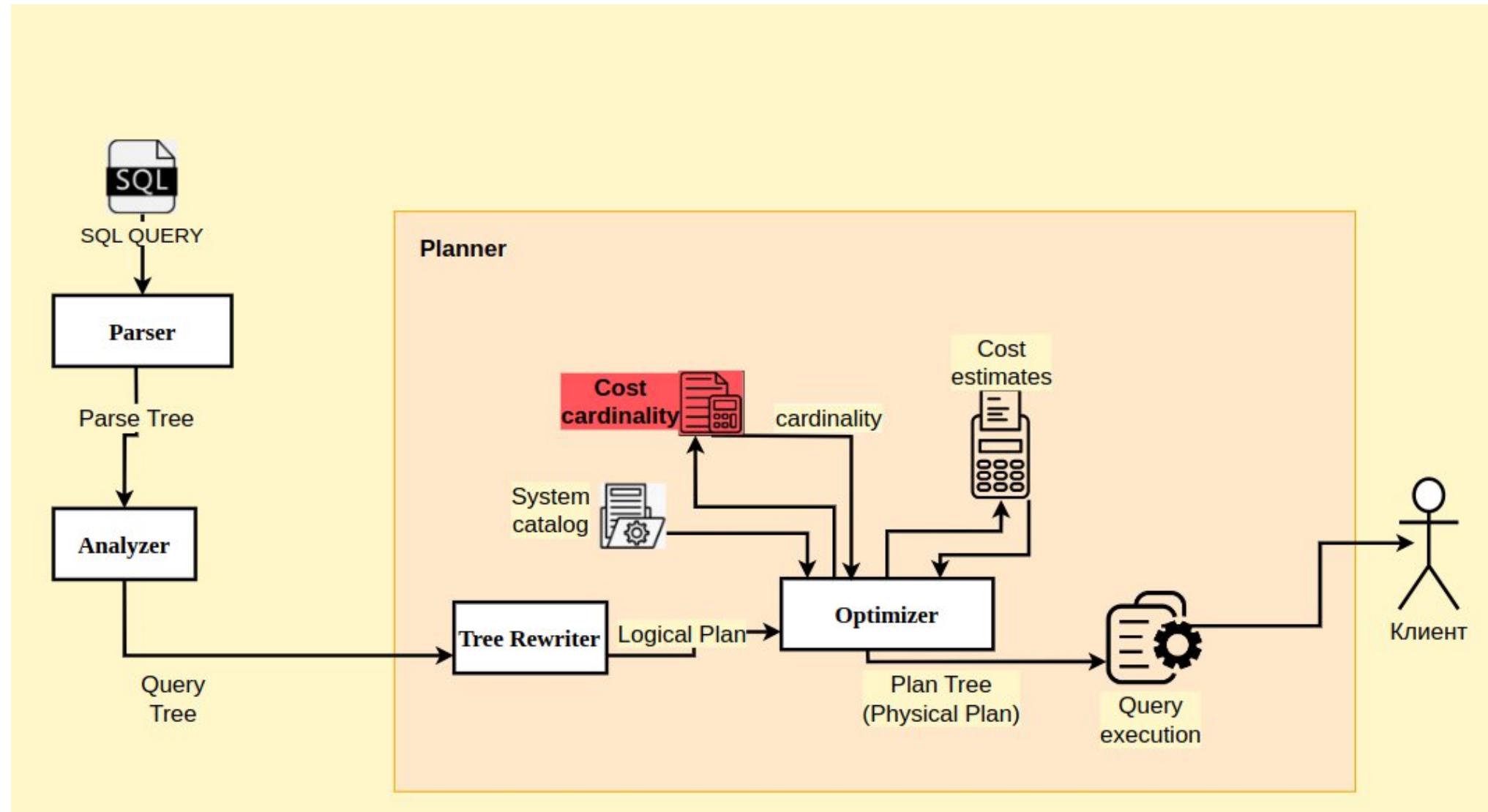
- Как работает оптимизатор
- Основная проблема оптимизатора
- Возможные решения
- Идея перепланирования запросов (Replan)
- Как работает Replan
- Результаты тестирования Replan
- Случай из моей практики

PGConf.CПб 2023



Об исходной проблеме

# Устройство оптимизатора



Как работает реплан

Результаты тестирования

# Как работает оптимизатор

- **Задача**
  - Найти лучший из всех возможных планов через расчет стоимости
  - Используется метод на основе стоимостной модели System R (1978 - 1979 гг.)
- **Проблема**
  - Много возможных планов  $\sim \exp(\text{количество таблиц})$
- **Приближенное решение**
  - Алгоритм динамического программирования
- **Поиск**
  - Решение гарантированно будет найдено, но может оказаться не лучшим



# Как работает оптимизатор

Стоимость плана



стоимость частей плана



# Как работает оптимизатор



Стоимость плана



стоимость частей плана



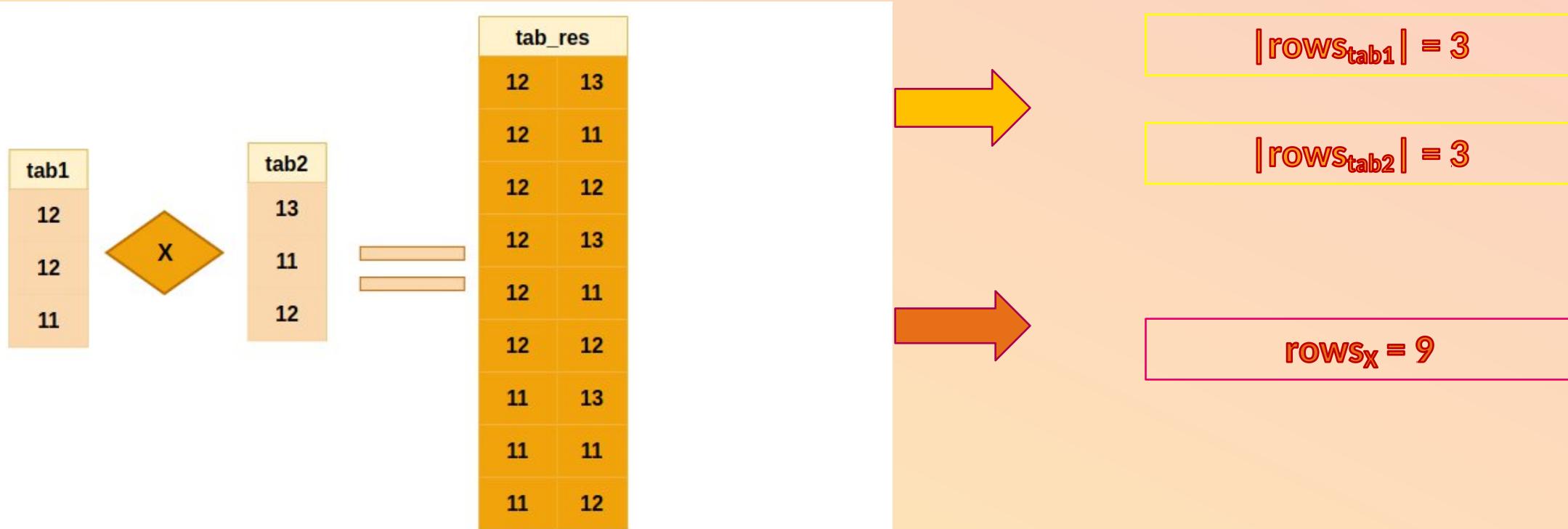
Кардинальность



Селективность

# Что такое кардинальность и селективность

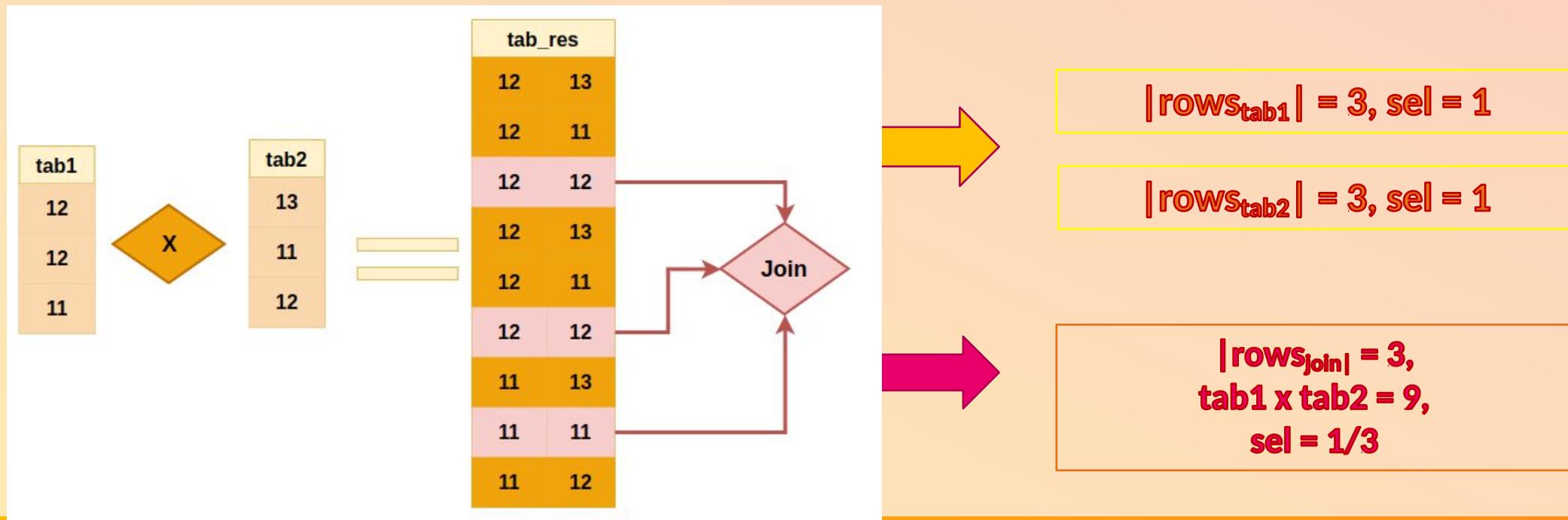
Кардинальность – количество туплов, полученных после выполнения операции



# Что такое кардинальность и селективность

**Кардинальность** – количество туплов, полученных после выполнения операции

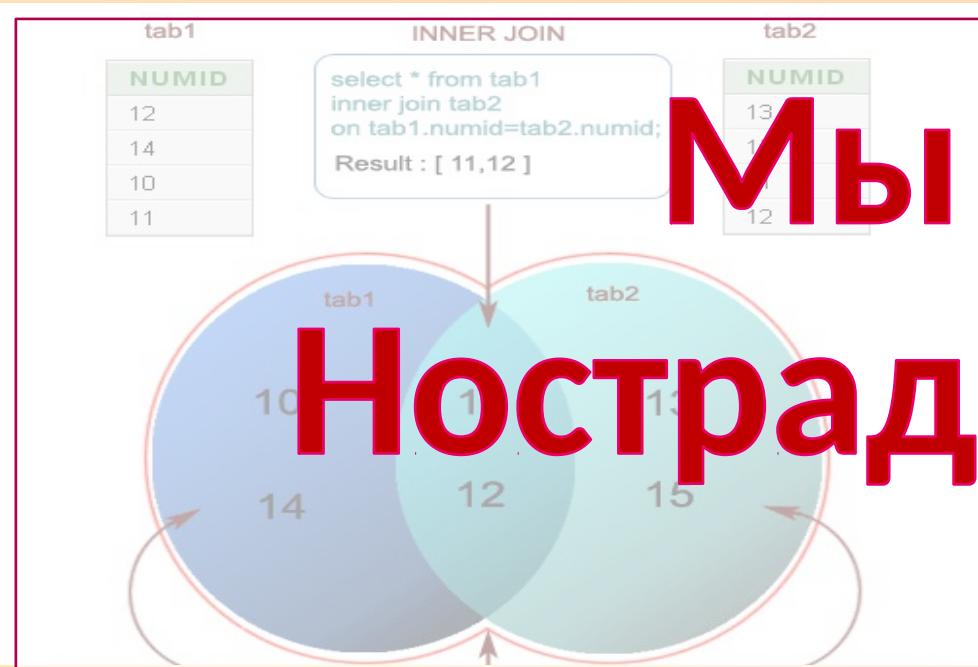
**Селективность** – доля строк от декартового произведения двух наборов, которая остается после соединения



# Что такое кардинальность и селективность

Кардинальность – количество туплов, полученных после выполнения операции

Селективность – доля строк от декартового произведения двух наборов, которая остается после соединения



Мы не  
Нострадамусы!

$\text{rows}_{\text{tab1}} = 4, \text{sel} = 1$

$\text{rows}_{\text{tab2}} = 4, \text{sel} = 1$

$\text{rows}_{\text{join}} = 2, \text{sel} = 0.$



most common values  
(MCV)

если количество  
различных значений не  
очень велико

## Статистика

Гистограмма

если количество  
различных значений  
большое

Доля неопределенных  
значений (Null\_frac)

# Статистика: Most Common Values

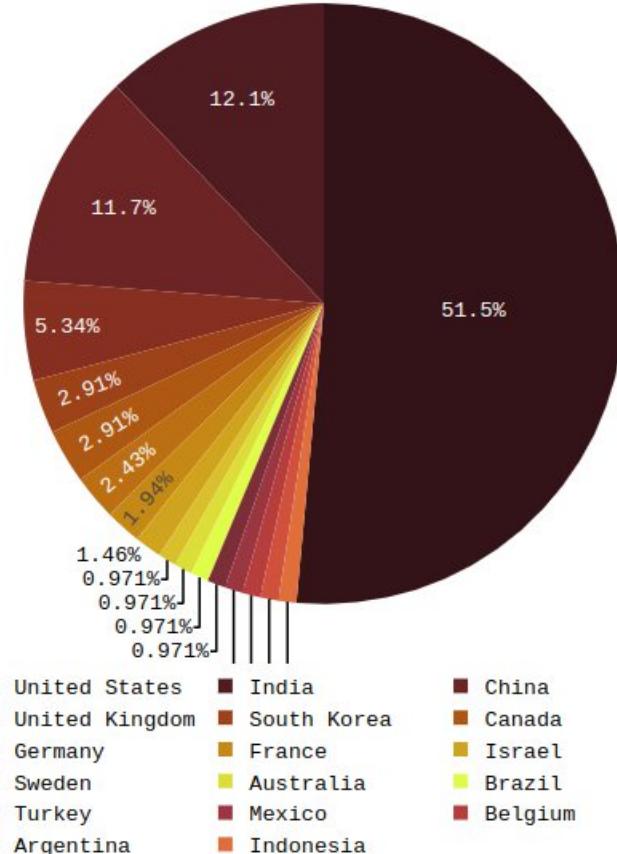
## MCV:

- `most_common_values`
- `most_common_frequencies`

```
SELECT most_common_vals, most_common_freqs
FROM pg_stats
WHERE tablename = 'startups'
AND attname   = 'country'
```

```
attname          | country
null_frac        | 0
most_common_vals | {"United
States", India, China, "United Kingdom", Canada, "South
Korea", Germany, France, Israel, Argentina, Australia, Bel
gium, Brazil, Indonesia, Mexico, Sweden, Turkey}
most_common_freqs |
{0.515, 0.121, 0.117, 0.053, 0.029, 0.029, 0.024, 0.019, 0.0
14, 0.009, 0.009, 0.009, 0.009, 0.009, 0.009, 0.009}
```

Number of startups by country



# Статистика: Гистограмма

```
postgres=# SELECT attname, histogram_bounds
  FROM pg_stats
 WHERE tablename = 'startups'
   OR attname    = 'country'
   OR attname    = 'company' \gx
-[ RECORD 1 ]-----+
attname          | country
histogram_bounds |
-[ RECORD 2 ]-----+
attname          | company
most_common_vals |
most_common_freqs |
null_frac        | 0
histogram_bounds | {Acorns,Addepar,Airtable,"AmWINS Group","Animoca Brands","Arctic
Wolf Networks",Articulate,"Automation Anywhere",Better.com,"Biosplice
Therapeutics",BitSight,Bitpanda,Blockchain.com,Bolt,BrowserStack,ByteDance,...}
```

# Статистка: гистограмма и MCV

```
postgres=# SELECT attname, histogram_bounds
  FROM pg_stats
 WHERE tablename = 'startups'
   OR attname   = 'valuation' \gx
-[ RECORD 1 ]-----+
attname           | valuation
most_common_vals  |
{4,3.7,2.75,3.1,3.4,3.55,2.25,3,3.15,3.35,3.5,3.6,3.65,3.75,2.2,2.45,2.6,2.8,...}
most_common_freqs |
{0.032,0.028,0.019,0.019,0.019,0.019,0.014,0.014,0.014,0.014,0.014,0.014,0.014,...}
histogram_bounds  |
{2.1,2.17,2.23,2.3,2.33,2.35,2.4,2.48,2.5,2.52,2.55,2.56,2.61,2.63,2.65,2.68,2.7,2.
78,2.81,2.82,2.88,3.05,3.27,3.3,3.31,3.33,3.95,4.05,4.2,4.25,4.4,4.6,4.65,4.75,...}
```

# Как считается кардинальность

attnname	valuation
n_distinct	-0.7196262
most_common_vals	{4,3,7,2.75,3.1,...} -35 значений, соответствующие условию
most_common_freqs	{0.032,0.028,0.019,0.019,0.019,0.019,0.014,...}
histogram_bounds	{...,18,20,30, <b>40,50</b> ,60,80,90,100,110,120,140}

$$\text{NDistinct\_tuples} = 217 * 0.7196 = 156$$

93

$$\text{Selectivity\_hyst} = \text{bound}[i]/(\text{N}_\text{BOUNDS}-1) - 1/(\text{N}_\text{DISTINCT}-\text{N}_\text{MCV}) = (93/(101-1) - 1/(156-35)) = 0.922$$

$$\begin{aligned}\text{Selectivity\_mcv} &= (1-\text{null\_frac}-\text{Sum}_\text{MCV}) * \text{Selectivity}_\text{HYST} + \text{Selectivity}_\text{MCV} = \\ &= (1-0.44393) * 0.922 + 0.444 = 0.956\end{aligned}$$

$$\text{Cardinality} = \text{NTuples} * \text{Selectivity} = 0.956 * \text{NTuples}(217) = 207.23$$

```
SELECT count(*) from startups WHERE valuation < 50;
```

---

Aggregate (cost=7.19..7.20 rows=1 width=8)

-> Seq Scan on best\_cities\_for\_startups (cost=0.00..6.68 rows=207 width=0)

Filter: (valuation < '50'::double precision)

# Расчет стоимости

Стоимость плана  стоимость частей плана

Стоимость одного узла зависит от типа этого узла и от объема обрабатываемых этим узлом данных

# Расчет стоимости

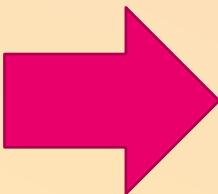
Стоимость плана:



стоимость частей плана

Стоимость одного узла зависит от типа этого узла и от объема обрабатываемых этим узлом данных

Npages: 45  
Ntuples: 10000



SeqScan tbl1:

- `cpu_run_cost:`
  - `cpu_tuple_cost * Ntuples`
  - `cpu_operator_cost * Ntuples`
- `disk_run_cost:`
  - `seq_page_cost * Npages`

Total Cost: 170.0

# Расчет стоимости

Selectivity: 0.024  
NIndexPage: 30  
NTuples: 10000  
NPage: 45



IndexScan tbl1\_idx:

- start-up cost:  $\text{cpu\_operator\_cost} * (\log_2(\text{NTuples}) + 100)$
- run\_cost:
  - index\_cpu\_cost:  $\text{Selectivity} * \text{NTuples} * (\text{cpu\_index\_tuple\_cost} + \text{qual\_op\_cost})$
  - table\_cpu\_cost:  $\text{Selectivity} * \text{NTuples} * \text{cpu\_tuple\_cost}$
  - index\_io\_cost:  $\text{Selectivity} * \text{NIndexPage} * \text{random\_page\_cost}$
  - table\_io\_cost:  $\text{NPage} * \text{random\_page\_cost} + \text{indexCorrelation}^2 * (\text{NPage} * \text{random\_page\_cost} - \text{Selectivity} * \text{NPage} * \text{seq\_page\_cost})$

TotalCost: 13.485

# Итак, проблема

Причины неточной оценки кардинальности:

- большое количество соединений в запросе
- наличие корреляций в данных
- наличие перекоса значений в данных
- несовершенство алгоритмов оценки затрат
- использование устаревшей статистики

Неточность может быть очень большой, а последствия – серьезными

Основная проблема оптимизатора

Возможные решения

# Итак, проблема

Причины неточной оценки кардинальности:

- большое количество соединений в запросе
- наличие корреляций в данных
- наличие перекоса значений в данных
- несовершенство алгоритмов оценки затрат
- использование устаревшей статистики

Неточность может быть очень большой, а последствия – серьезными

Основная проблема оптимизатора

Возможные решения

# Итак, проблема

Причины неточной оценки кардинальности:

- большое количество соединений в запросе
- наличие корреляций в данных
- наличие перекоса значений в данных
- несовершенство алгоритмов оценки затрат
- использование устаревшей статистики

Неточность может быть очень большой, а последствия – серьезными

Основная проблема оптимизатора

Возможные решения

# Итак, проблема

Причины неточной оценки кардинальности:

- большое количество соединений в запросе
- наличие корреляций в данных
- наличие перекоса значений в данных
- несовершенство алгоритмов оценки затрат
- использование устаревшей статистики

Неточность может быть очень большой, а последствия – серьезными

Основная проблема оптимизатора

Возможные решения

# Итак, проблема

Причины неточной оценки кардинальности:

- большое количество соединений в запросе
- наличие корреляций в данных
- наличие перекоса значений в данных
- несовершенство алгоритмов оценки затрат
- использование устаревшей статистики

Неточность может быть очень большой, а последствия – серьезными

Основная проблема оптимизатора

Возможные решения

**PGConf.CПб 2023**



**О решениях**

# Решения от PostgreSQL



Генетический алгоритм GEQO

# О GEQO

Познакомимся с GEQO за 20 минут

PostgresPro



ПАВЕЛ ТОЛМАЧЕВ

Postgres Professional

Специалист образовательного отдела



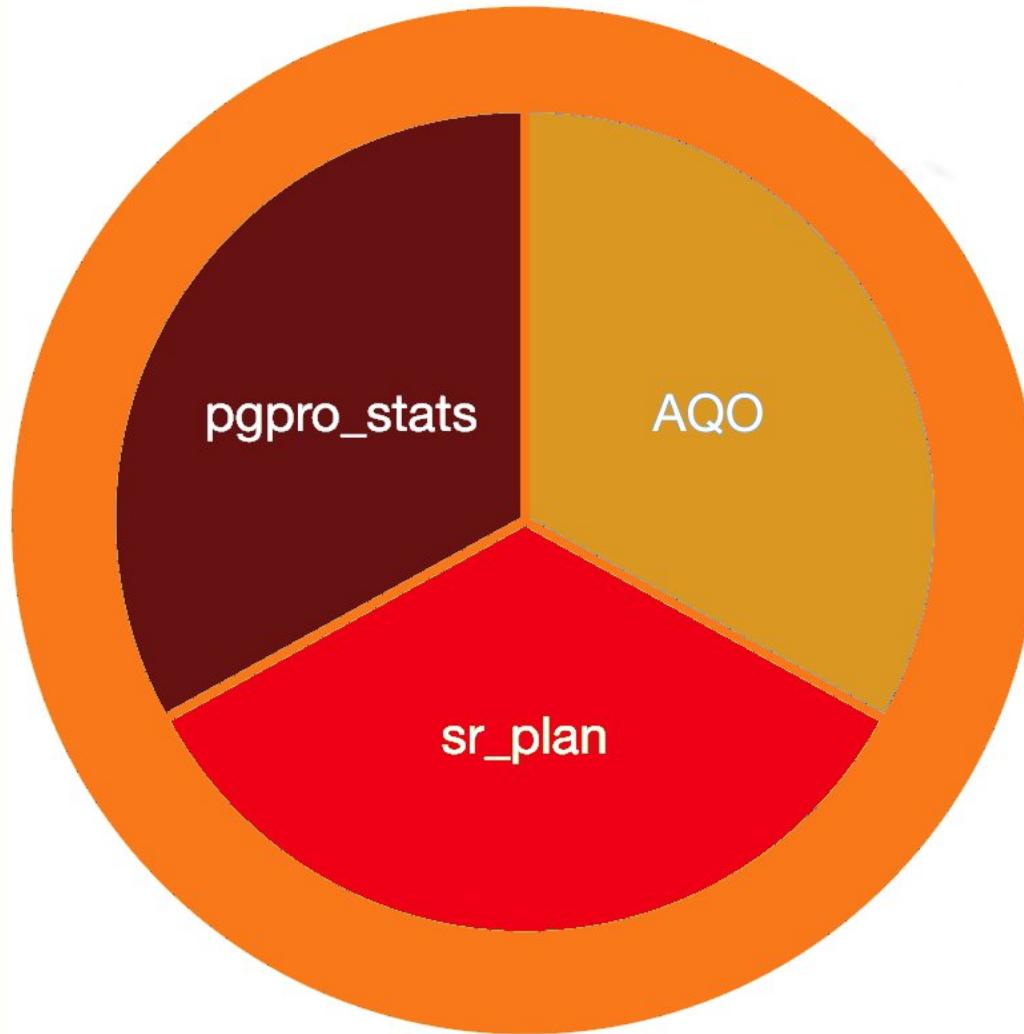
Генетический  
алгоритм  
GEQO

Основная проблема оптимизатора

Возможные решения

# Решения от PostgresPro Enterprise

PostgresPro

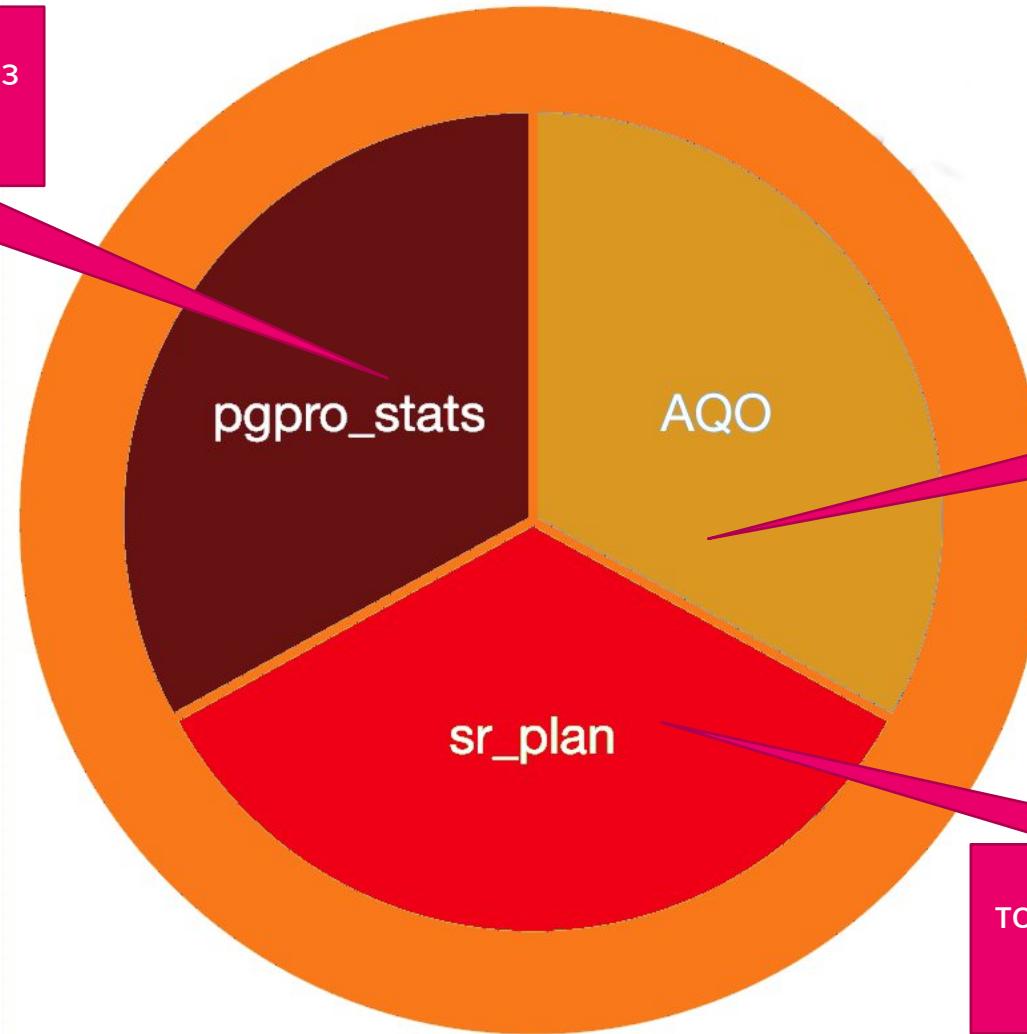


Возможные решения

Как работает Replan

# Решения от PostgresPro Enterprise

требуется ручной анализ  
и поиск лучшего плана



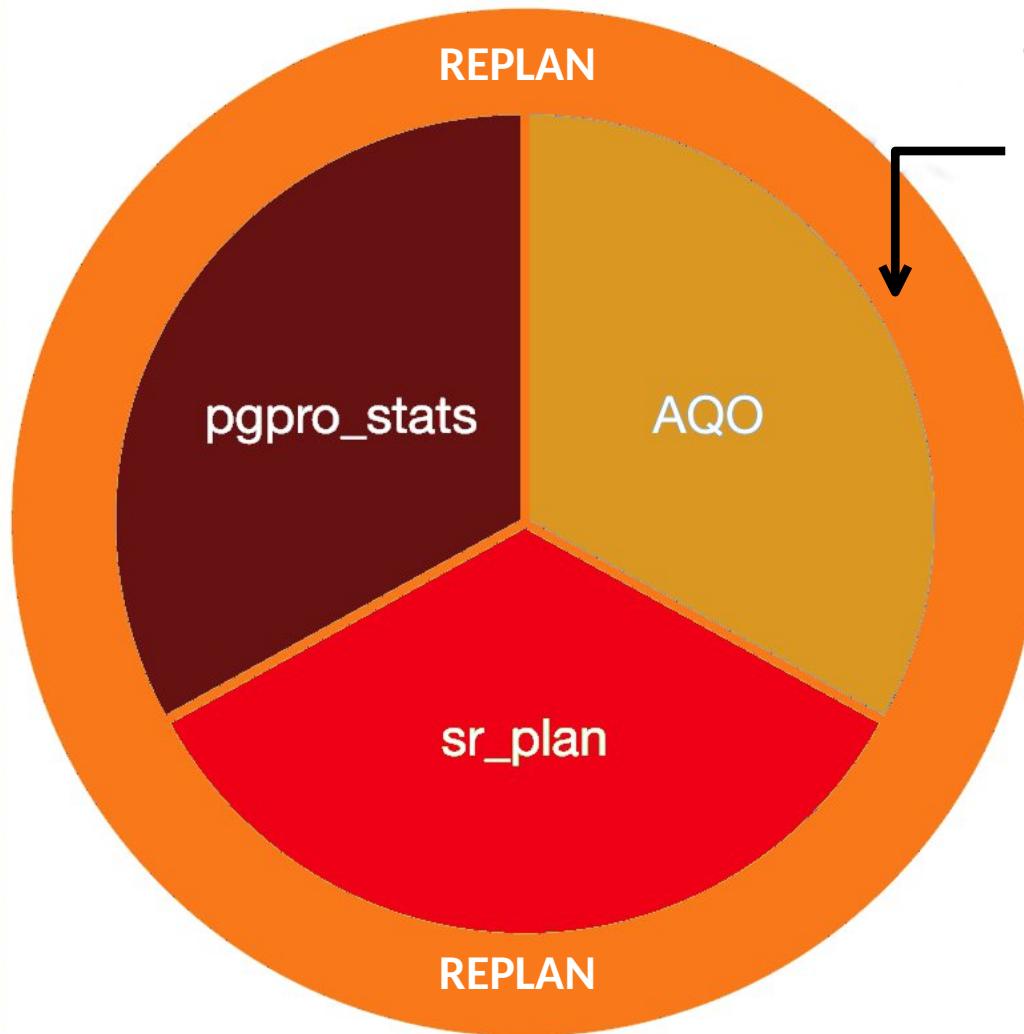
в основе него используется алгоритм  
Machine Learning, который может  
ошибаться

только фиксирует найденное  
решение

Возможные решения

Как работает Replan

# Решения от PostgresPro Enterprise

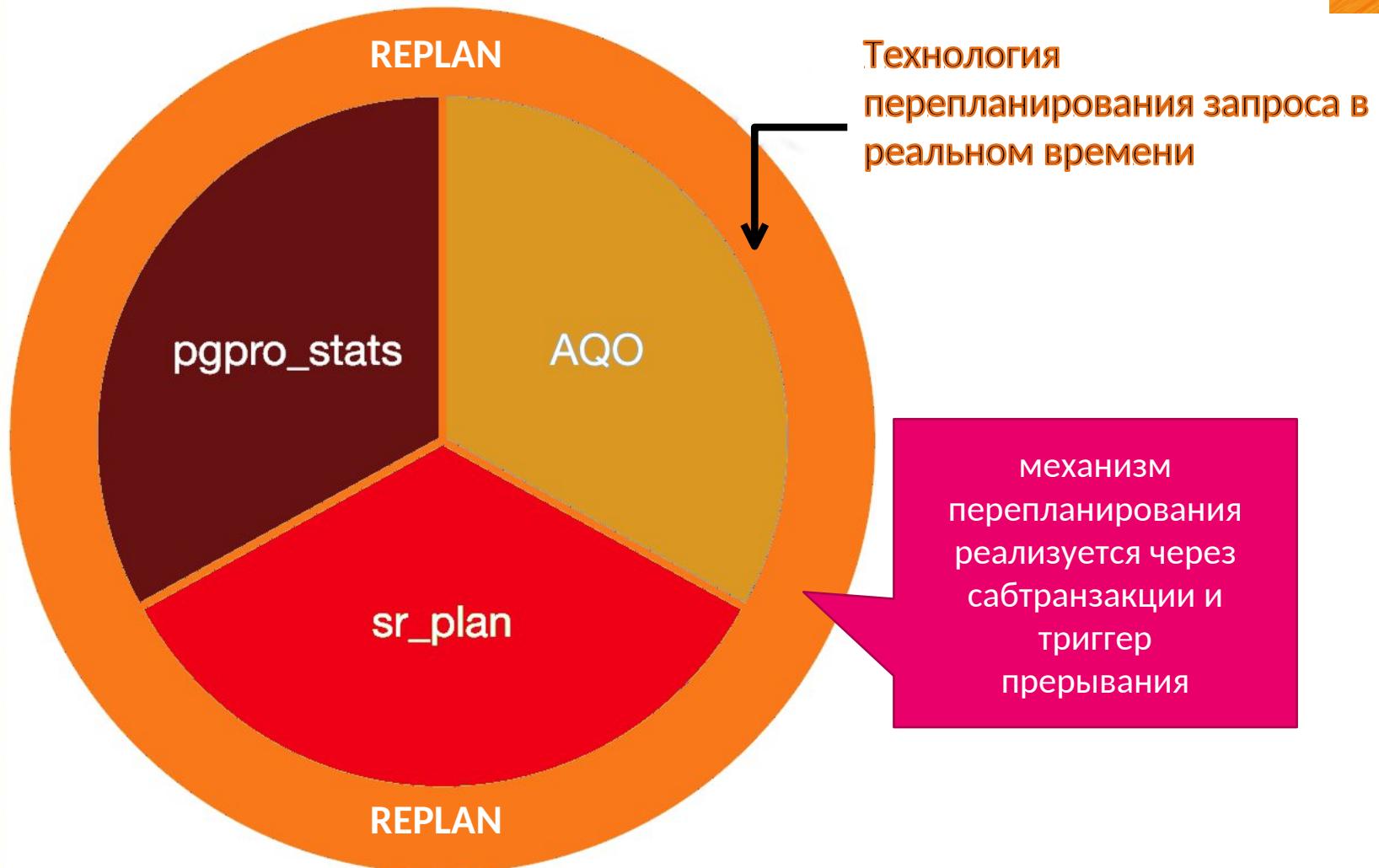


Технология  
перепланирования запроса в  
реальном времени

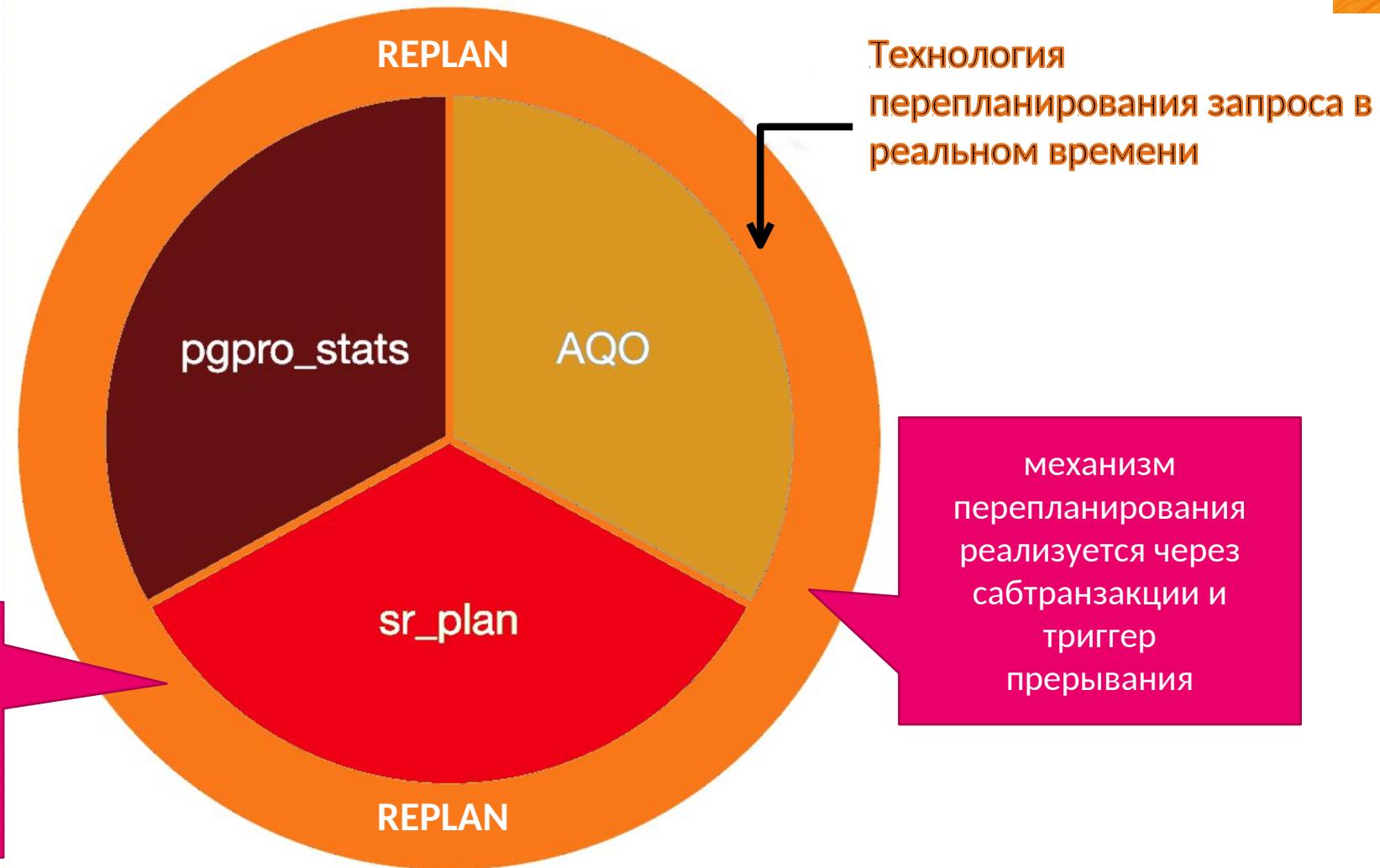
Возможные решения

Как работает Replan

# Решения от PostgresPro Enterprise



# Решения от PostgresPro Enterprise

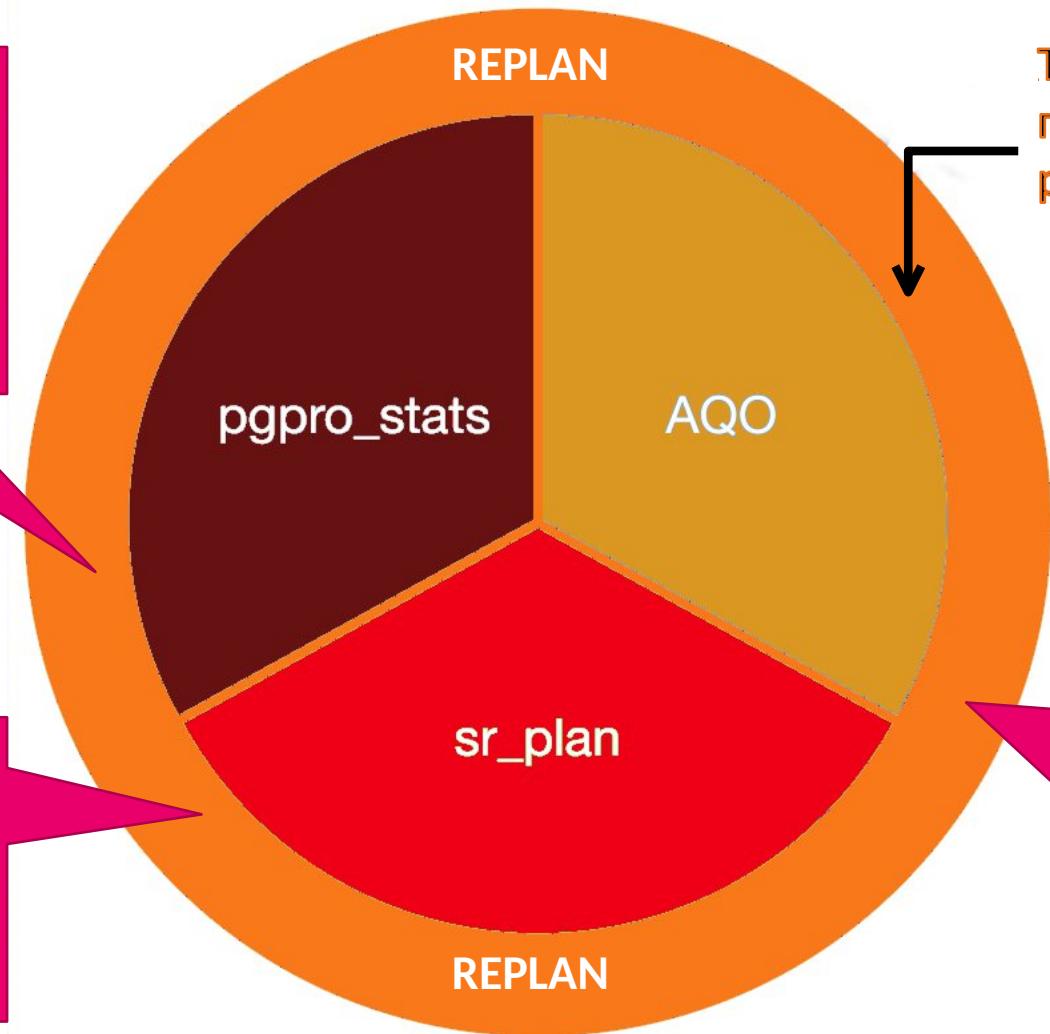


Возможные решения

Как работает Replan

# Решения от PostgresPro Enterprise

как и в AQO сохраняет кардинальности и использует их в новой попытке перепланирования



решение о возможном перепланировании принимает оптимизатор

Технология перепланирования запроса в реальном времени

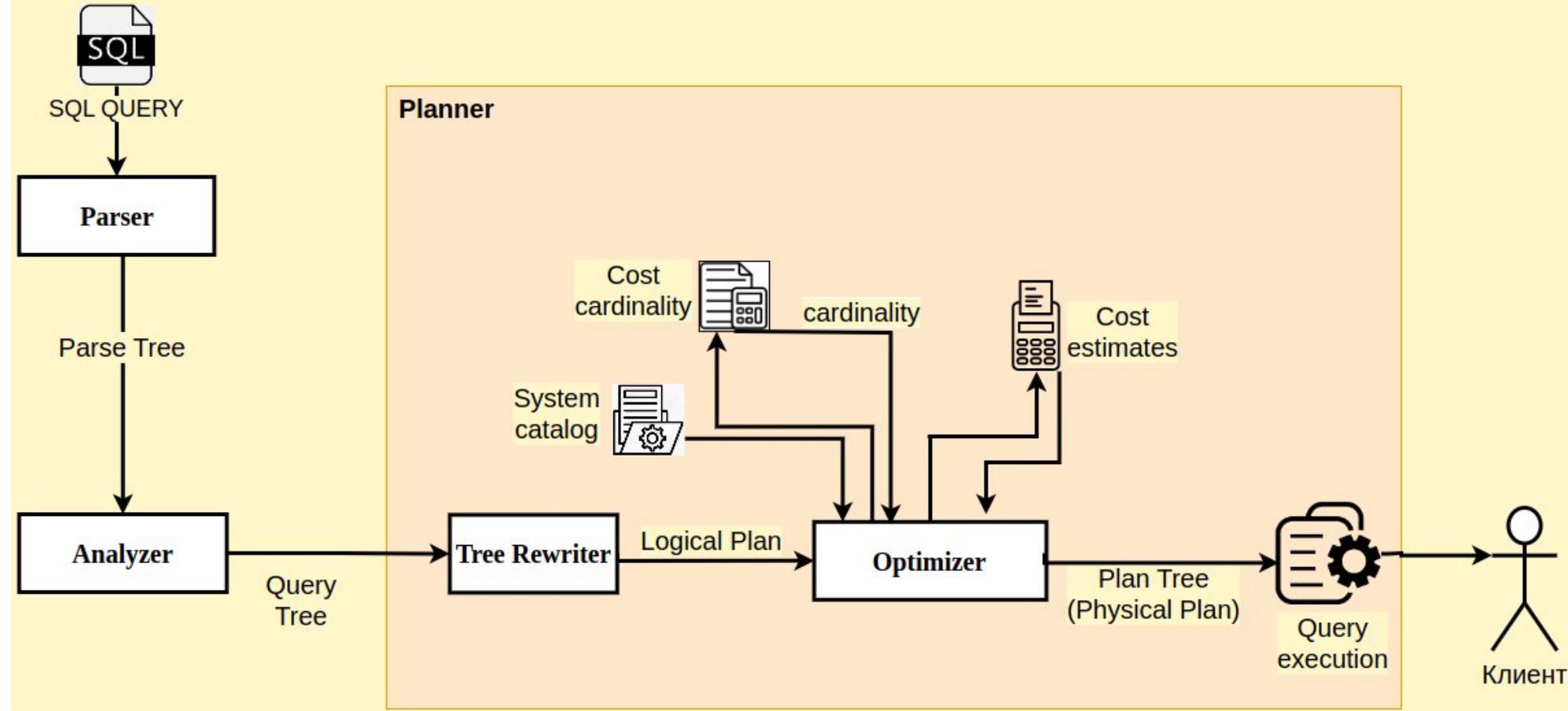
механизм перепланирования реализуется через сабтранзакции и триггер прерывания

**PGConf.CПб 2023**



# Об устройстве Replan

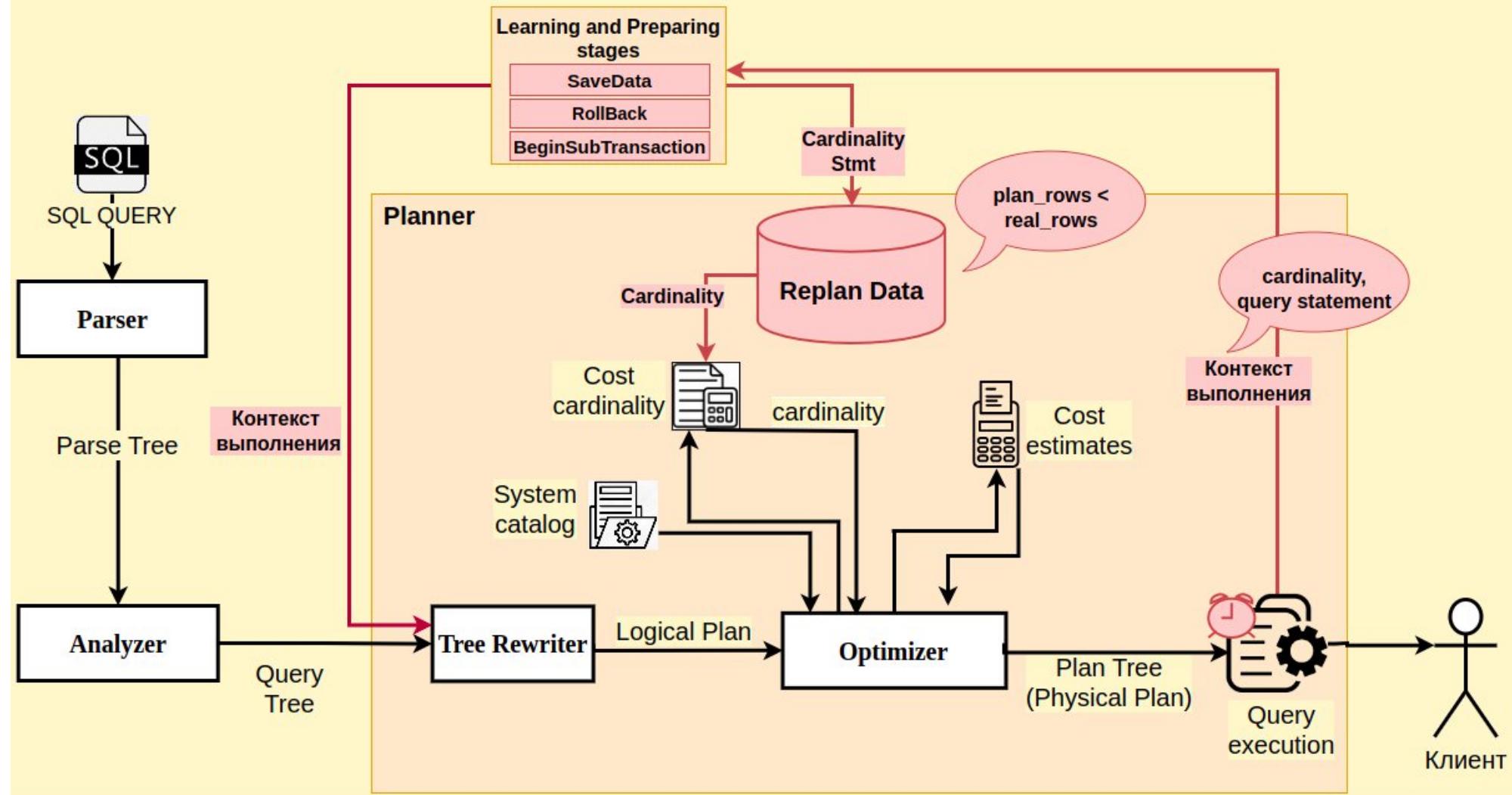
# Устройство оптимизатора



Как работает Replan

Результаты тестирования

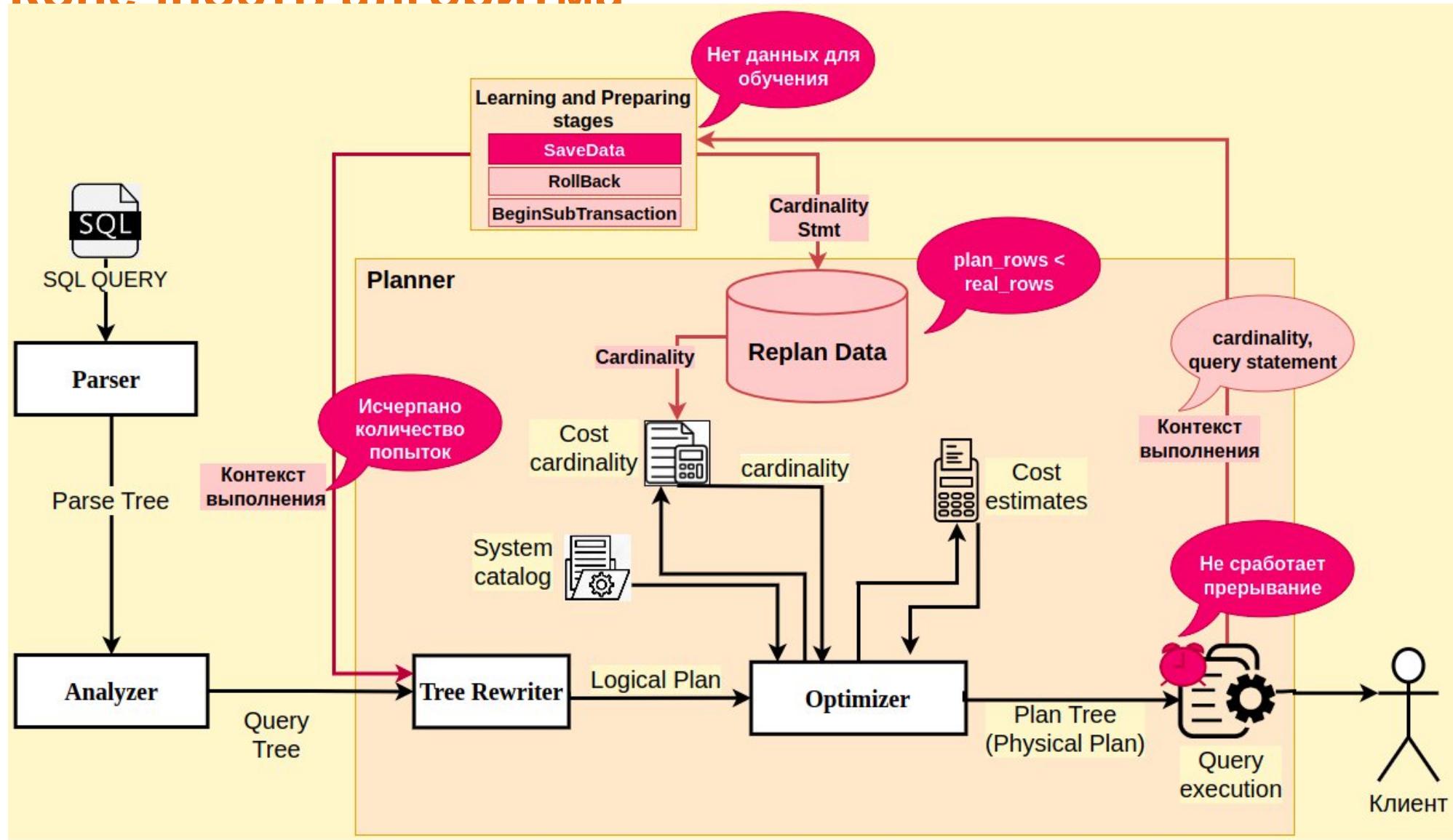
# Устройство реплана



Как работает Replan

Результаты тестирования

# Конечность алгоритма



Как работает Replan

Результаты тестирования

# Параметры Replan

`query_inadequate_execution_time`

максимальное время на  
перепланирование одной  
попытки

`replan_max_attempts`

максимальное количество  
попыток на перепланирование  
запроса

`replan_overrun_limit`

количество туплов, когда  
прерываем запрос во время  
попытки Replan

`show_node_sign`

вывод дополнительной  
информации в EXPLAIN

Как работает Replan

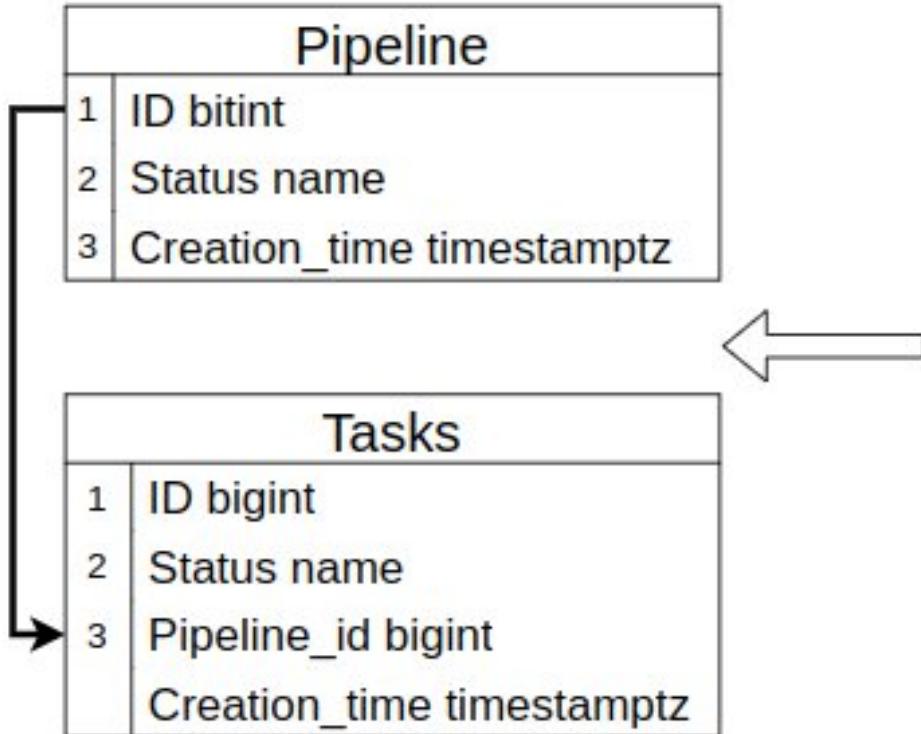
Результаты тестирования Replan

**PGConf.CПб 2023**



**Теперь к практике**

# Больше о Replan. Исходная база



- генерируется 50000 pipelines
- в каждом pipeline генерируется по 5 задач
- status pipeline может быть New или Started
- status задач может быть New или Completed

Как работает Replan

Результаты тестирования Replan

# Представление запроса

```
EXPLAIN ANALYZE SELECT count(1) FROM pipeline p,tasks t  
WHERE t.status = 'NEW' AND p.id = t.pipeline_id AND t.creation_time > now() - interval '1 hour';
```

```
Aggregate (rows=1) (rows=1)  
-> Nested Loop (rows=1) (rows=1889920)  
  -> Index Scan using tasks_status_creation_time_idx1 on tasks t (rows=1) (rows=1889920)  
      Index Cond: ((status = 'NEW'::text) AND (creation_time > (now() - '01:00:00'::interval)))  
  -> Index Only Scan using pipeline_pkey on pipeline p (rows=1) (rows=1)  
      Index Cond: (id = t.pipeline_id)  
      Heap Fetches: 1889920
```

Planning Time: 1.045 ms

Execution Time: 9595.701 ms

# Представление запроса

```
EXPLAIN ANALYZE SELECT count(1) FROM pipeline p,tasks t
WHERE t.status = 'NEW' AND p.id = t.pipeline_id AND t.creation_time > now() - interval '1 hour';
```

Aggregate (rows=1) (rows=1)

-> Nested Loop (rows=1) (rows=1889920)

-> Index Scan using tasks\_status\_creation\_time\_idx1 on tasks t (rows=1) (rows=1889920)

Index Cond: ((status = 'NEW'::text) AND (creation\_time > (now() - '01:00:00'::interval)))

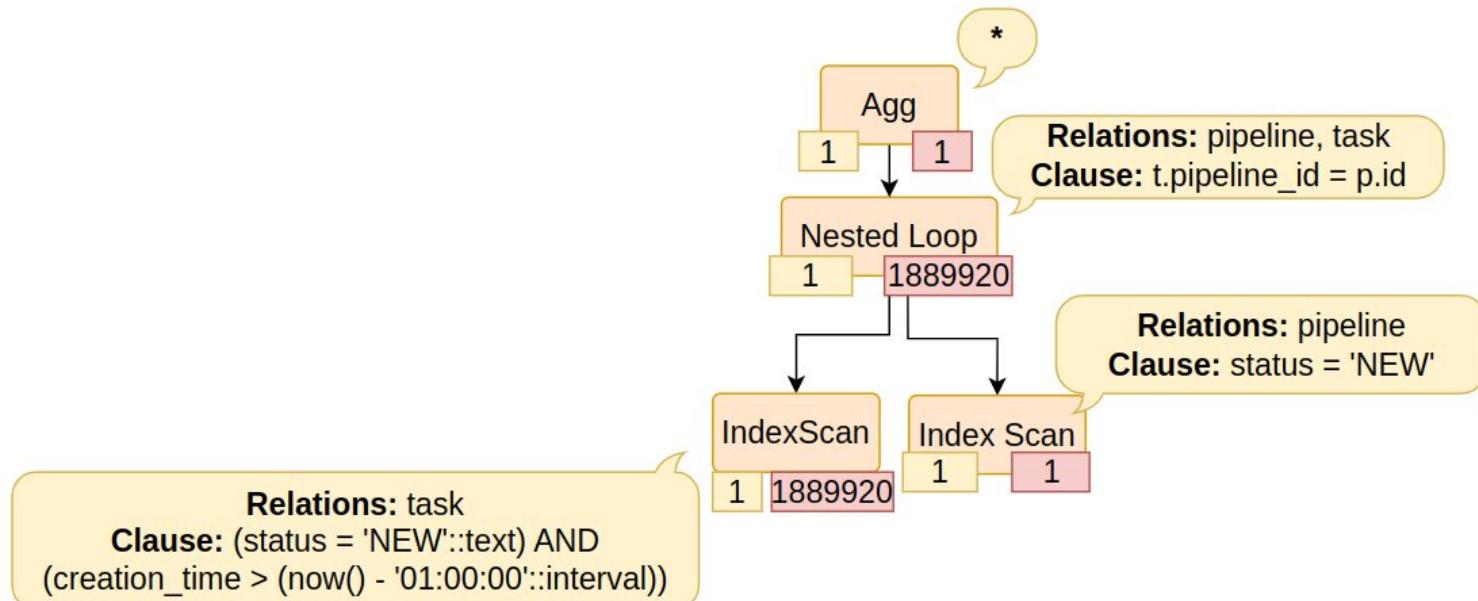
-> Index Only Scan using pipeline\_pkey on pipeline p (rows=1) (rows=1)

Index Cond: (id = t.pipeline\_id)

Heap Fetches: 1889920

Planning Time: 1.045 ms

Execution Time: 9595.701 ms



# Explain запроса с репланом в терминале

```
EXPLAIN ANALYZE SELECT count(1) FROM pipeline p,tasks t  
WHERE t.status = 'NEW' AND p.id = t.pipeline_id AND t.creation_time > now() - interval '1 hour';
```

```
Finalize Aggregate (rows=1) (rows=1)  
-> Gather (rows=2) (rows=3)  
  -> Partial Aggregate (rows=1) (rows=1)  
    -> Hash Join (rows=8355) (rows=629973)  
      Hash Cond: (p.id = t.pipeline_id)  
      -> Parallel Seq Scan on pipeline p (rows=764484) (rows=628661)  
      -> Hash (rows=240357) (rows=1889920)  
        -> Index Scan using tasks_status_creation_time_idx1 on tasks t (rows=240357) (rows=1889920)  
          Index Cond: ((status = 'NEW'::text) AND (creation_time > (now() - '01:00:00'::interval)))
```

Planning Time: 0.452 ms

Replanning Attempts: 5

[Execution Time: 2229.972 ms](#)

[Total Execution Time: 2964.835 ms](#)

# Как выглядит лог при рабочем реплане

2023-09-21 13:31:49.469 MSK [8320] LOG: Replanning triggered by timeout 100 (0-th shift) ms.

Attempt: 1.

Duration: 158.919477 ms plan:

Query Text: explain (analyze) select count(1) from pipeline p,tasks t...

Finalize Aggregate (cost=30098.31..30098.32 rows=1 width=8) (actual time=162.220..162.220 rows=0 loops=1)  
(early terminated)

NodeSign: 4304986849338732187

Cardinality: -1

Groups Number: -1

Output: count(1)...

2023-09-21 13:31:49.957 MSK [8320] LOG: Replanning triggered by timeout 100 (0-th shift) ms.

Attempt: 4.

Duration: 156.986469 ms plan:

Query Text: explain (analyze) select count(1) from pipeline p,tasks t...

Finalize Aggregate (cost=41139.66..41139.67 rows=1 width=8) (actual time=160.432..160.432 rows=0 loops=1)  
(early terminated)

NodeSign: 4304986849338732187

Cardinality: -1

Groups Number: -1

Output: count(1)...

PGConf.CПб 2023



**Replan может не решить  
проблему**

# Основная проблема в оверэстимации

```
EXPLAIN ANALYZE SELECT count(1) FROM pipeline p,tasks t  
WHERE p.status = 'STARTED' AND p.id = t.pipeline_id and t.status IN ('NEW', 'COMPLETED')  
AND t.creation_time > now() - interval '1 hour';
```

Aggregate (rows=1) (rows=1)

-> Nested Loop (rows=9) (rows=25000)

  -> Index Scan using pipeline\_status\_idx on pipeline p (rows=1) (rows=5000)

    Index Cond: (status = 'STARTED'::text)

  -> Index Scan using tasks\_pipeline\_id\_idx on tasks t (rows=5) (rows=5)

    Index Cond: (pipeline\_id = p.id)

    Filter: ((status = ANY ('{NEW,COMPLETED}'::text[])) AND (creation\_time > (now() - '01:00:00'::interval)))

Planning Time: 3.043 ms

Execution Time: 43.531 ms

(9 rows)

Как работает Replan

Результаты тестирования Replan

# Улучшать нечего

```
EXPLAIN ANALYZE SELECT count(1) FROM pipeline p,tasks t  
WHERE p.status = 'STARTED' AND p.id = t.pipeline_id and t.status IN ('NEW', 'COMPLETED')  
AND t.creation_time > now() - interval '1 hour';
```

```
Aggregate (rows=1) (rows=1)  
-> Hash Join (rows=491) (rows=25000)  
  Hash Cond: (t.pipeline_id = p.id)  
  -> Index Scan using tasks_status_creation_time_idx on tasks t (rows=24752) (rows=25000)  
    Index Cond: ((status = 'NEW'::text) AND (creation_time > (now() - '01:00:00'::interval)))  
  -> Hash (rows=99) (rows=5000)  
    -> Index Scan using pipeline_status_idx on pipeline p (rows=99) (rows=5000)  
      Index Cond: (status = 'STARTED'::text)
```

Planning Time: 0.414 ms

Execution Time: 22.380 ms

Replanning Attempts: 1

Total Execution Time: 1025.460 ms

Как работает Replan

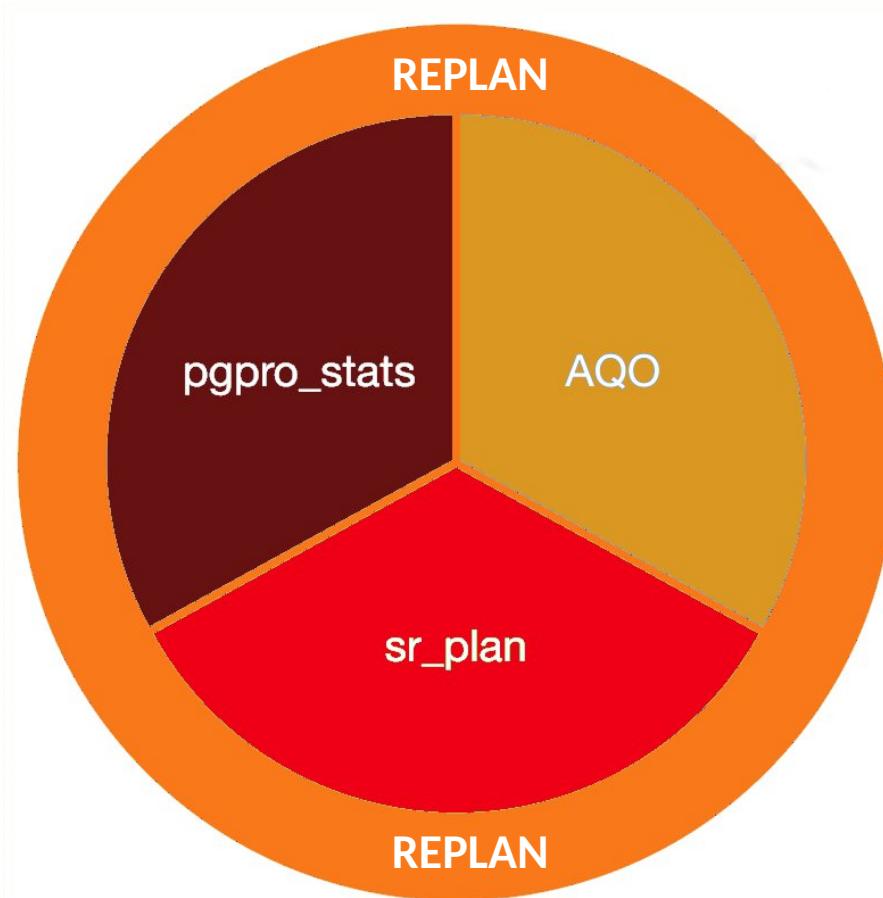
Результаты тестирования Replan

# Что Replan вылечить не сможет

1. Запрос просто может быть написан плохо

2. Replan потребуется большое количество итераций,  
чтобы найти лучший план

# Стоит попробовать другие решения



Как работает Replan

Результаты тестирования Replan

**PGConf.CПб 2023**



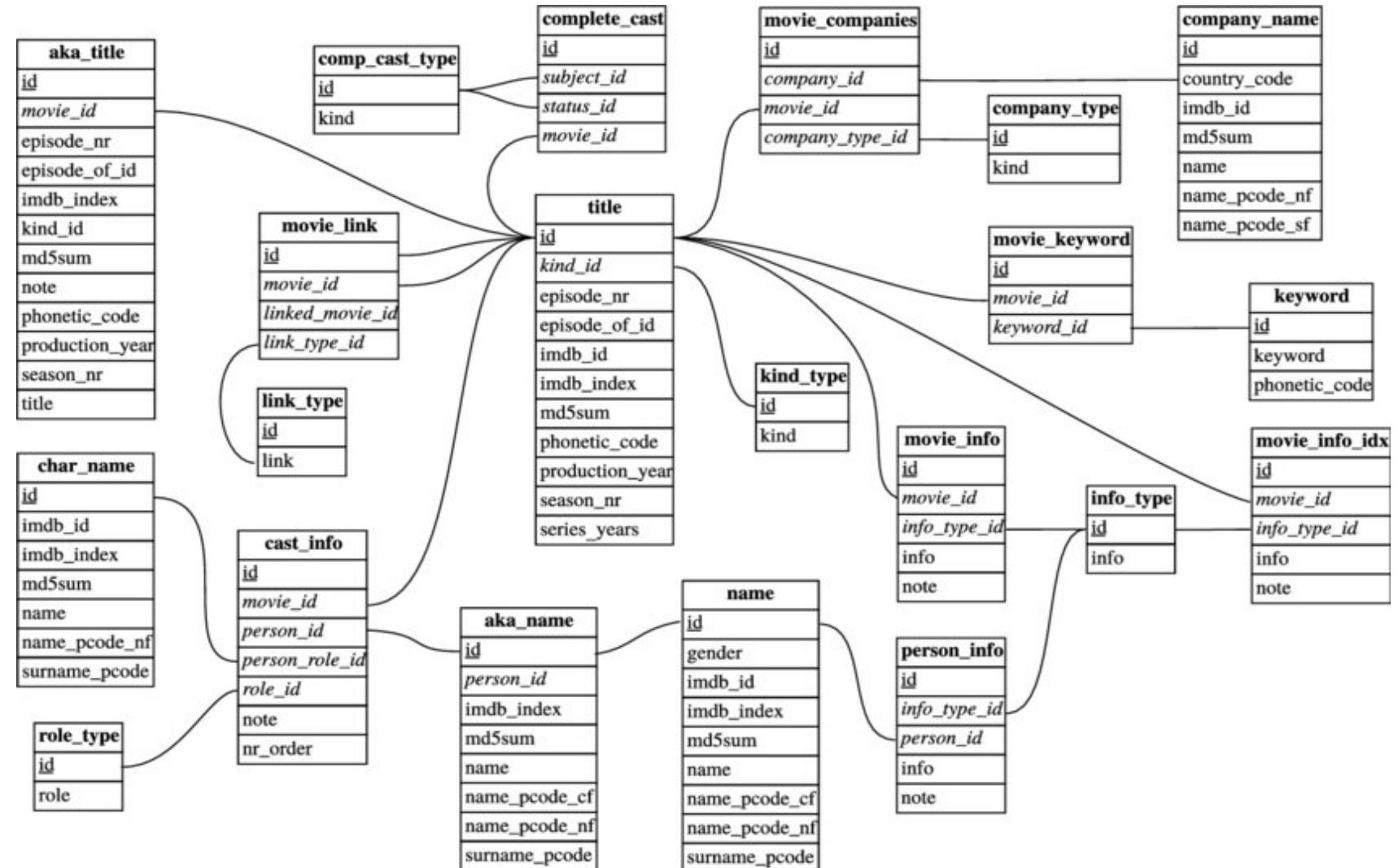
**Тестирование**

# Тестирование на Join Order Benchmark

**Интернет-база данных фильмов (IMDB)**

**Join Order Benchmark (JOB)** состоит из 113 запросов по IMDB, в каждом из которых может быть от 3 до 16 соединений. Все запросы отвечают на логичные вопросы любителя кино.

Для оптимизатора запросы сложные из-за большого количества объединений и корреляций в наборе данных.

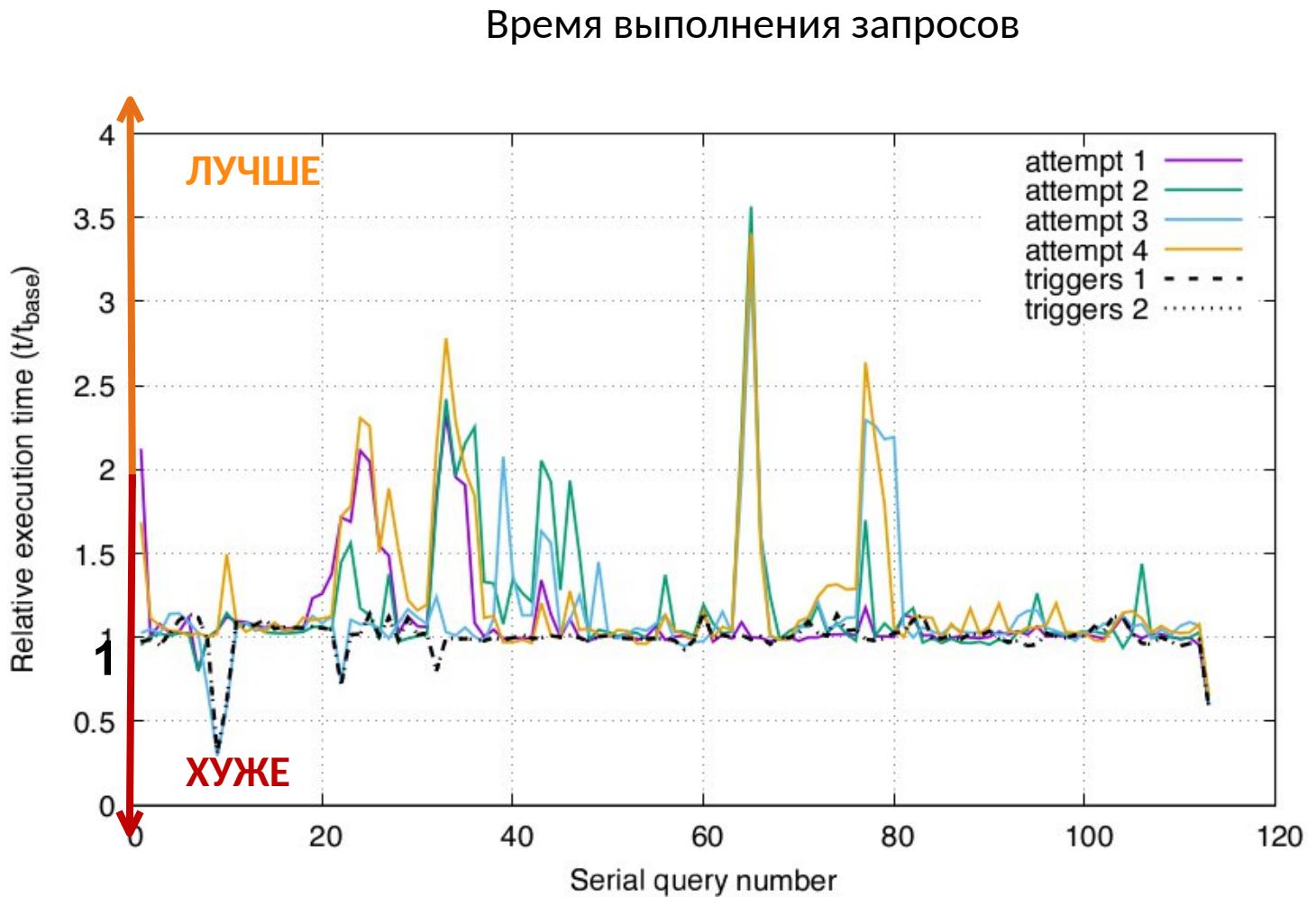


Результаты тестирования Replan

Случай из моей практики

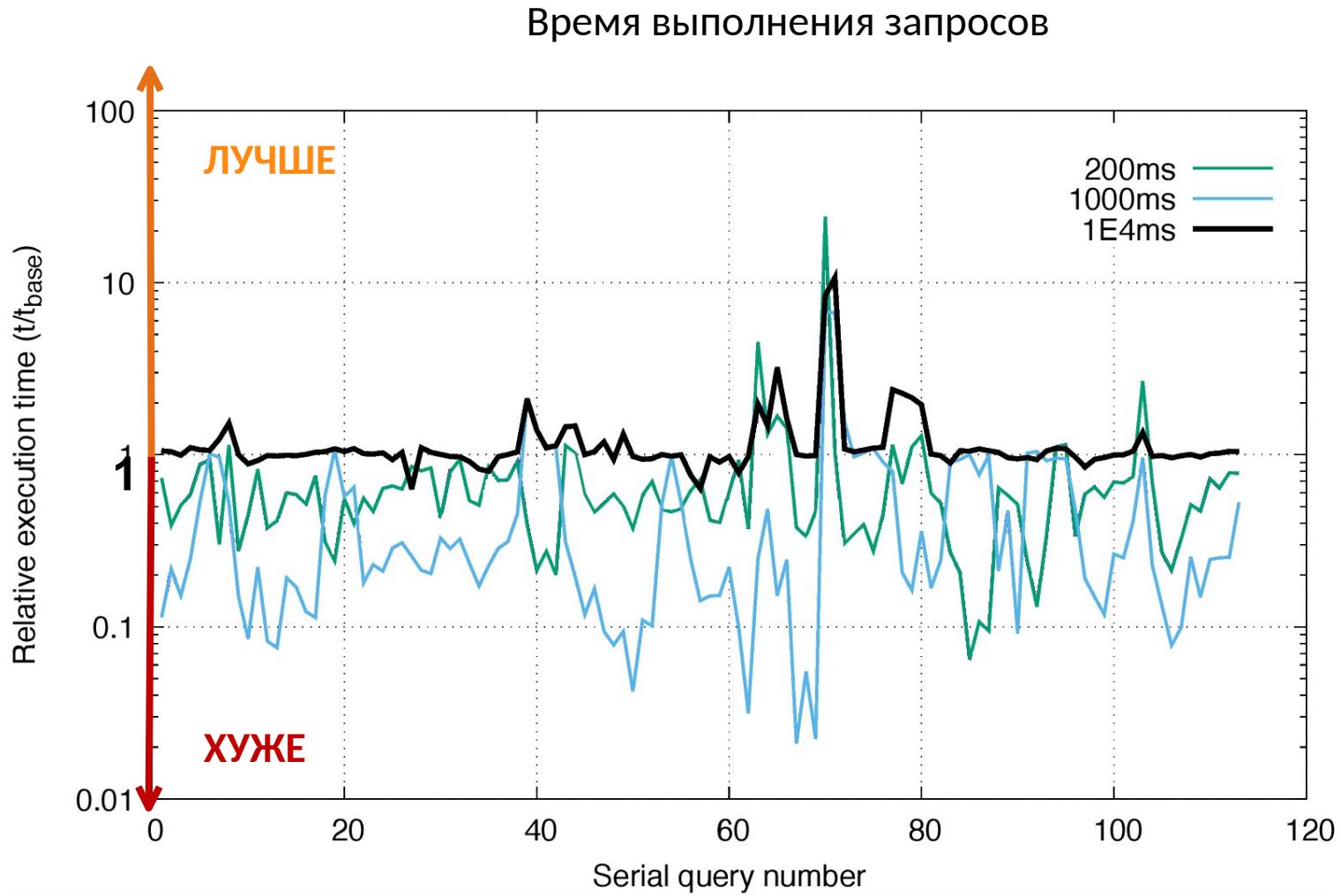
# Тестирование на Join Order Benchmark

Время выполнения запроса в JOB может сильно меняться от запуска к запуску



# Тестирование на Join Order Benchmark

Время выполнения запроса после перепланирования может ухудшиться



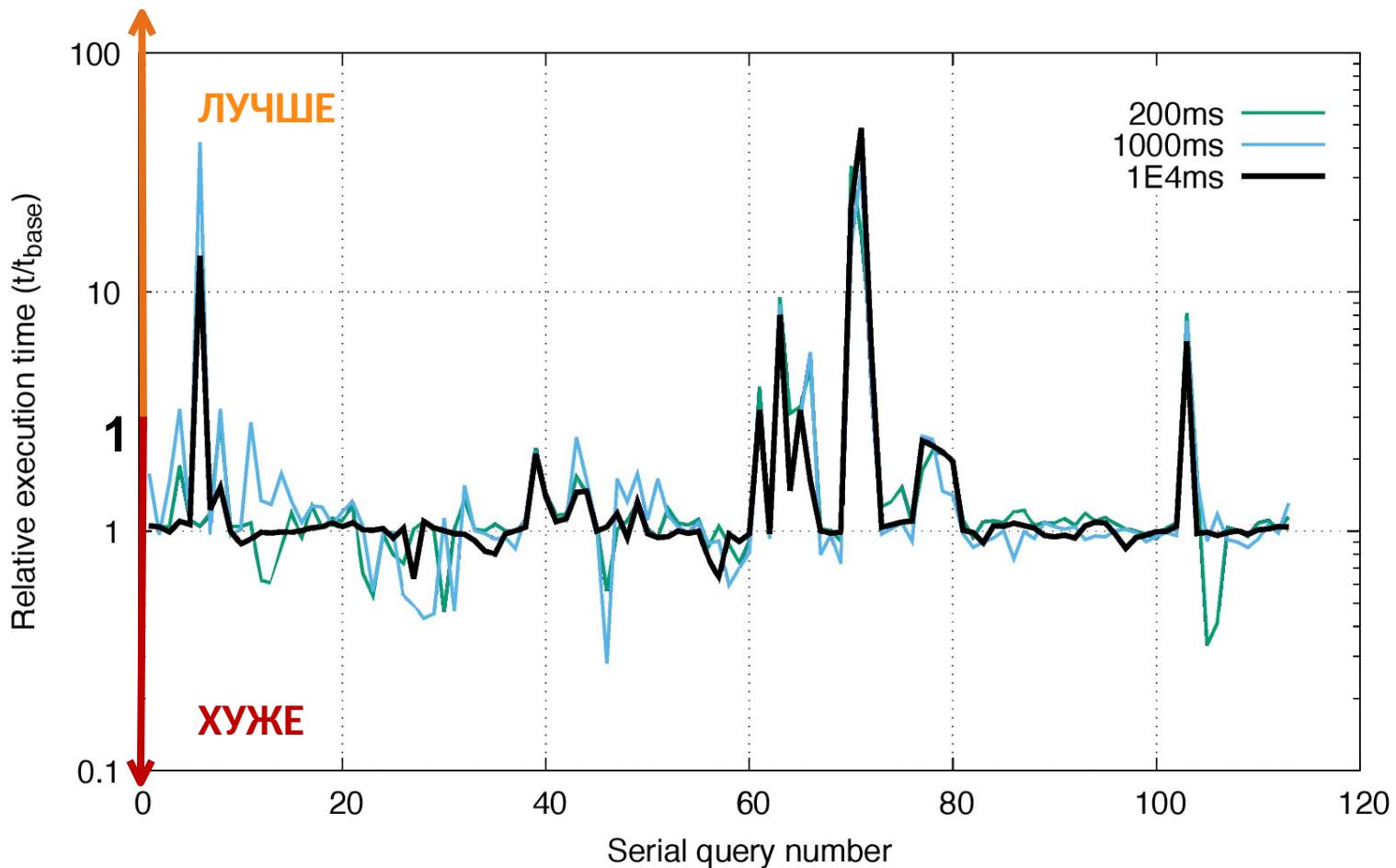
Результаты тестирования Replan

Случай из моей практики

# Тестирование на Join Order Benchmark

Итоговое время потраченное на перепланирование запросов

Перепланирование –  
крайняя мера



Результаты тестирования Replan

Случай из моей практики

PGConf.CПб 2023

PostgresPro

# Случай из моей практики



# Непонятный запрос из мира 1С

```

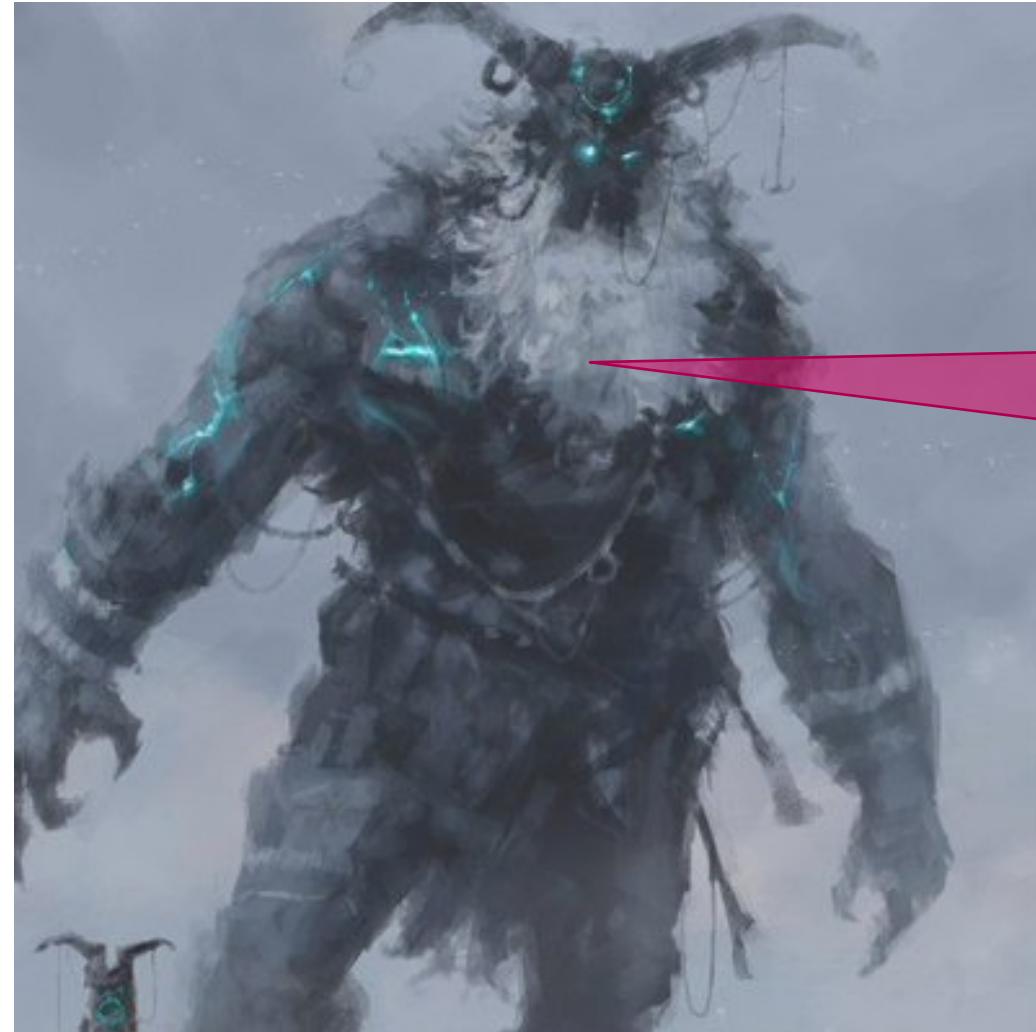
1 QUERY PLAN
2
3 Limit  (cost=9658602.66..9658602.66 rows=1 width=712) (actual time=0.156..0.184 rows=0 loops=1)
4   -> Sort  (cost=9658602.66..9658602.66 rows=1 width=712) (actual time=0.154..0.181 rows=0 loops=1)
5     Sort Key: (CASE WHEN (NOT t1._folder) THEN true ELSE false END) DESC, t1._description, t1._idrref
6     Sort Method: quicksort Memory: 25kB
7     -> Nested Loop Left Join  (cost=3878174.38..9658602.65 rows=1 width=712) (actual time=0.051..0.078 rows=0 loops=1)
8       Join Filter: ((t1._id = t117._fld6) AND (t117._fld6 = t2._q_000))
9       Filter: ((NOT t1._folder) OR ((hashed SubPlan 13) AND (hashed SubPlan 14) AND (hashed SubPlan 15) AND (hashed SubPlan 16) AND (hashed SubPlan 17)))
10      -> Nested Loop Left Join  (cost=1973142.48..7753544.73 rows=1 width=371) (actual time=0.050..0.073 rows=0 loops=1)
11        Join Filter: (t1._idrref = t130._fld62860rref)
12        -> Nested Loop Left Join  (cost=1912438.62..7684615.32 rows=1 width=351) (actual time=0.050..0.072 rows=0 loops=1)
13          Join Filter: ((t1._idrref = t50._fld6) AND (t50._fld6 = t2._q_000))
14          -> Nested Loop Left Join  (cost=1912435.33..5755830.28 rows=1 width=331) (actual time=0.050..0.066 rows=0 loops=1)
15            Join Filter: ((t1._idrref = t77._fld) AND (t77._fld = t2._q_000))
16            -> Nested Loop Left Join  (cost=1912432.85..3834506.35 rows=1 width=331) (actual time=0.049..0.060 rows=0 loops=1)
17              Join Filter: ((t1._idrref = t97._fld62752rref) AND (t97._fld62753rref = t2._q_000_f_001rref))
18              -> Merge Left Join  (cost=1912430.38..1912456.40 rows=1 width=331) (actual time=0.049..0.056 rows=0 loops=1)
19                Merge Cond: ((t1._id = t23._fld6) AND (t2._q_000 = t23._fld6))
20                -> Sort  (cost=14.72..14.73 rows=1 width=295) (actual time=0.048..0.049 rows=0 loops=1)
21                  Sort Key: t1._id, t2._q_000
22                  Sort Method: quicksort Memory: 25kB
23                  -> Nested Loop Left Join  (cost=0.90..14.71 rows=1 width=295) (actual time=0.042..0.043 rows=0 loops=1)
24                    -> Index Scan using _reference273_8 on _reference273 t1  (cost=0.48..6.27 rows=1 width=220) (actual time=0.042..0.042 rows=0 loops=1)
25                      Index Cond: (_fld1769 = '0':numeric) AND (_parentidrref = '\x5c3030...':bytea) AND (_folder = true))
26                      Filter: (((NOT _folder) OR ((hashed SubPlan 16) AND (hashed SubPlan 17))) AND (_description > 'Здание (копия 15)':mvarchar) OR (_description = 'Здание (копия 15)':mvarchar) AND (_idrref >= '\x5c32...':bytea)))
27                      SubPlan 16
28                        -> Values Scan on "*VALUES*_16"  (cost=0.00..0.03 rows=2 width=32) (never executed)
29                      SubPlan 17
30                        -> Values Scan on "*VALUES*_17"  (cost=0.00..0.03 rows=2 width=32) (never executed)
31                      -> Index Scan using tmpind_1 on tt11 t2  (cost=0.42..8.44 rows=1 width=92) (never executed)
32                        Index Cond: (_q_000_f_000rref = t1._idrref)
33                      -> Materialize  (cost=1912415.66..1912441.65 rows=1 width=76) (never executed)
34                        -> Nested Loop Semi Join  (cost=1912415.66..1912441.65 rows=1 width=76) (never executed)
35                          Join Filter: (CASE WHEN (hashed SubPlan 2) THEN true ELSE false END = CASE WHEN (hashed SubPlan 4) THEN true ELSE false END)
36                          -> Nested Loop  (cost=1912414.84..1912423.17 rows=1 width=76) (never executed)
37                            -> GroupAggregate  (cost=1912414.42..1912414.70 rows=1 width=112) (never executed)
38                              Group Key: t6._fld62799rref, t6._fld, (max(t6._period))
39                              -> Sort  (cost=1912414.42..1912414.42 rows=1 width=76) (never executed)
40                                Sort Key: t6._fld, t6._fld, (max(t6._period))
41                                -> Nested Loop Semi Join  (cost=1881151.57..1912414.41 rows=1 width=76) (never executed)
42                                  Join Filter: (CASE WHEN (hashed SubPlan 19) THEN true ELSE false END = CASE WHEN (hashed SubPlan 21) THEN true ELSE false END)
43                                  -> Hash Join  (cost=1881150.76..1912379.32 rows=3 width=96) (never executed)
44                                    Hash Cond: ((t15._fld = t6._fld) AND (t15._fld = t6._fld) AND (t15._period = (max(t6._period))))
45                                    -> Seq Scan on _info t15  (cost=0.00..22886.46 rows=221473 width=76) (never executed)
46                                      Filter: (_fld1769 = '0':numeric) AND (_fld98579 = '0001-01-01 00:00:00':timestamp without time zone) AND (_record <> '\x5c3...':bytea) OR (_record <> '\x5c32...':bytea))
47                                    -> Hash  (cost=1878238.88..1878238.88 rows=110736 width=48) (never executed)
48                                      -> GroupAggregate  (cost=1874916.80..1877131.52 rows=110736 width=48) (never executed)
49                                        Group Key: t6._fld62799rref, t6._fld62800rref
50                                        -> Sort  (cost=1874916.80..1875193.64 rows=110736 width=48) (never executed)

```

Запрос, который выполняется  
3 недели...

И так еще 450 строк плана...

# Как я вижу запрос



Какой-то страшный и  
непонятный запрос из  
мира 1С...

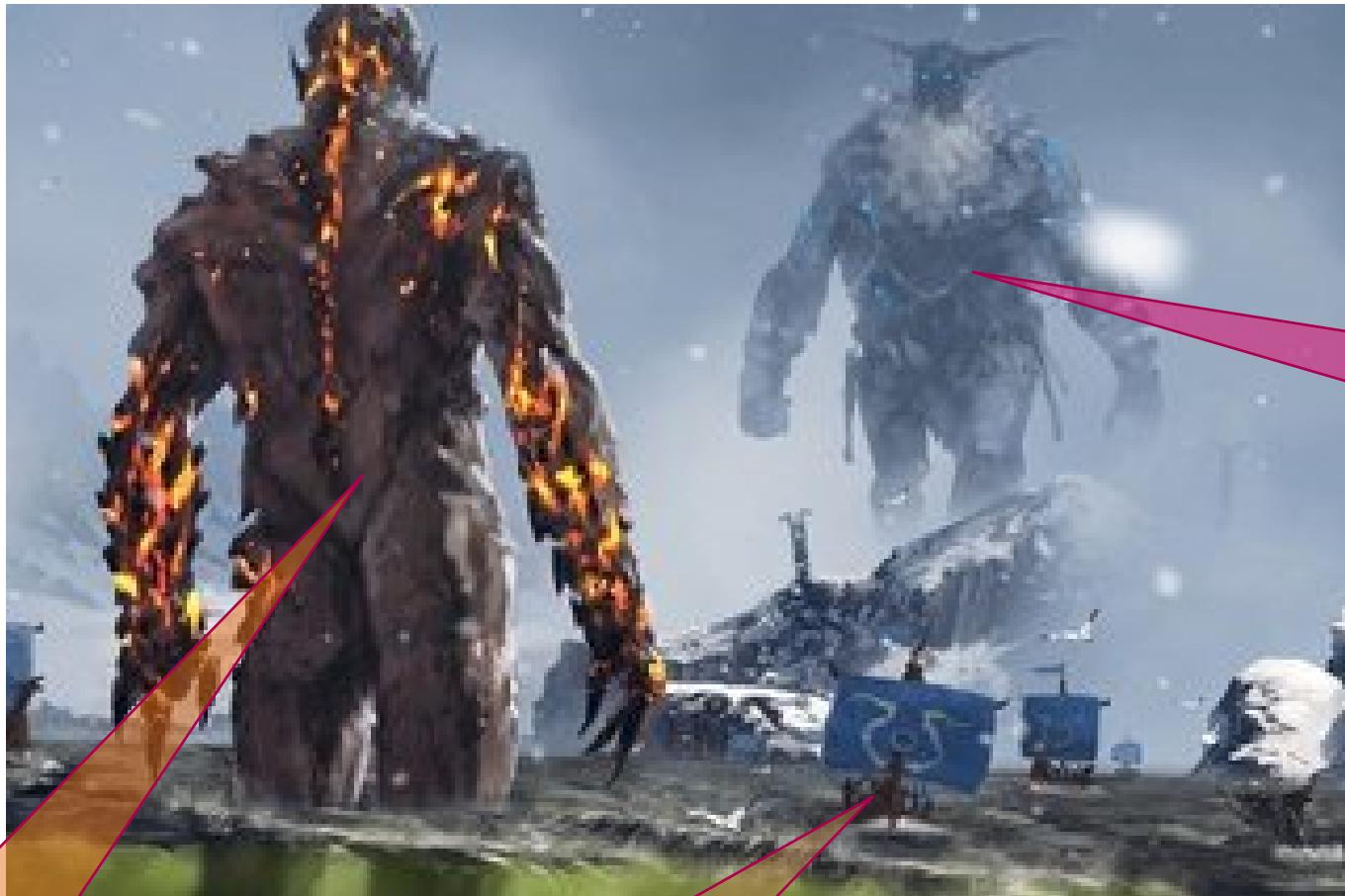
# Попытка выполнить запрос с помощью AQO



AQO

Запрос

# Пробуем выполнить Replan



Replan

Где-то здесь я  
запускаю Replan...

Запрос, который  
AQO не смог  
выполнить(

# Итог



Запрос

Replan выполнил запрос  
за 6 минут!

# Итог



PGConf.CПб 2023



А теперь ДЕМО

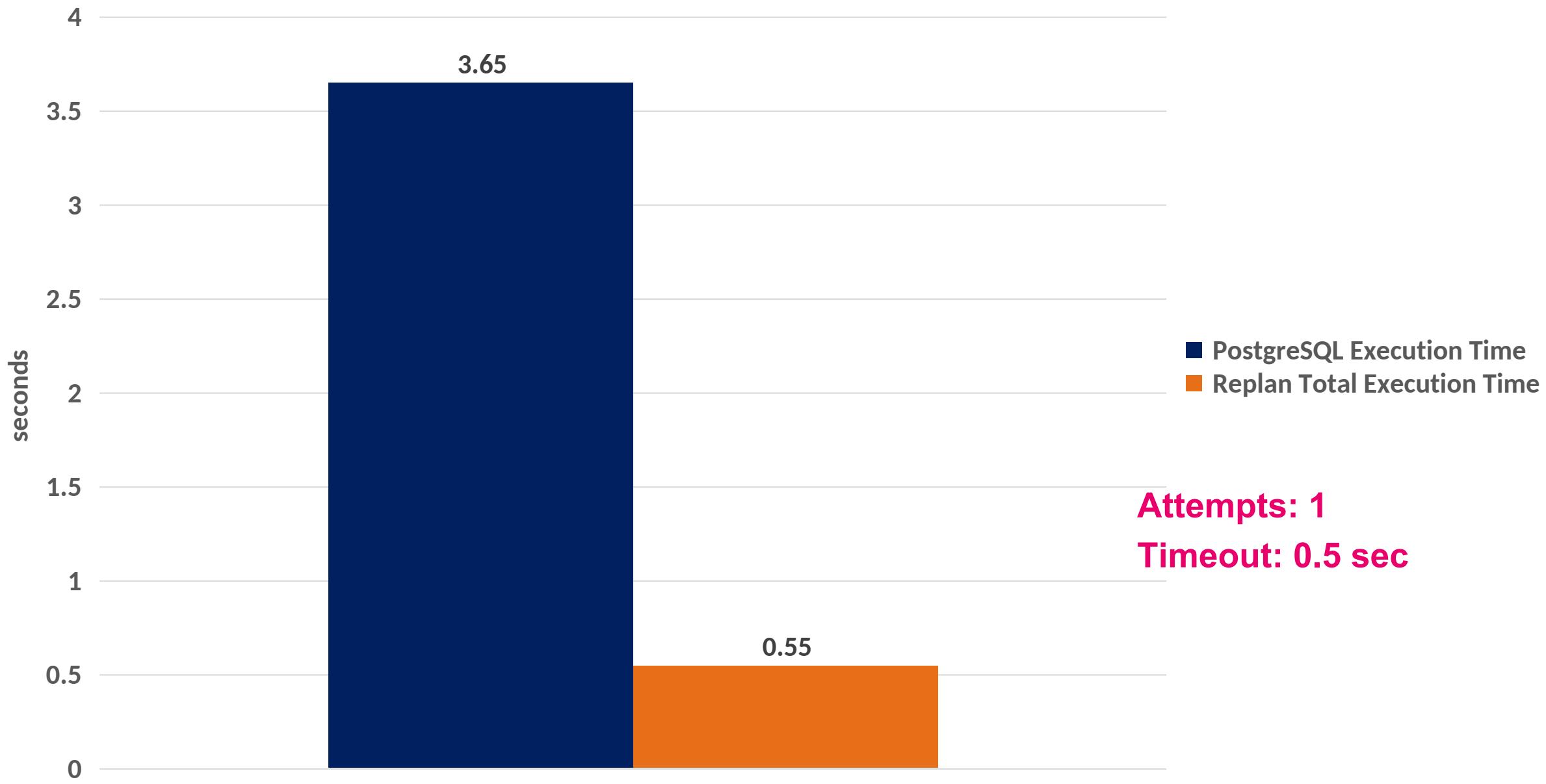


выпускайте кракена

```
(base) alena on alena-ThinkPad-L13-Gen-2 in ~
$ tmp_install/bin/pgsql -d alena
psql (12.15, server 17devel)
WARNING: psql major version 12, server major version 17.
          Some psql features might not work.
Type "help" for help.
```

```
alena=# 
```

## Query execution time

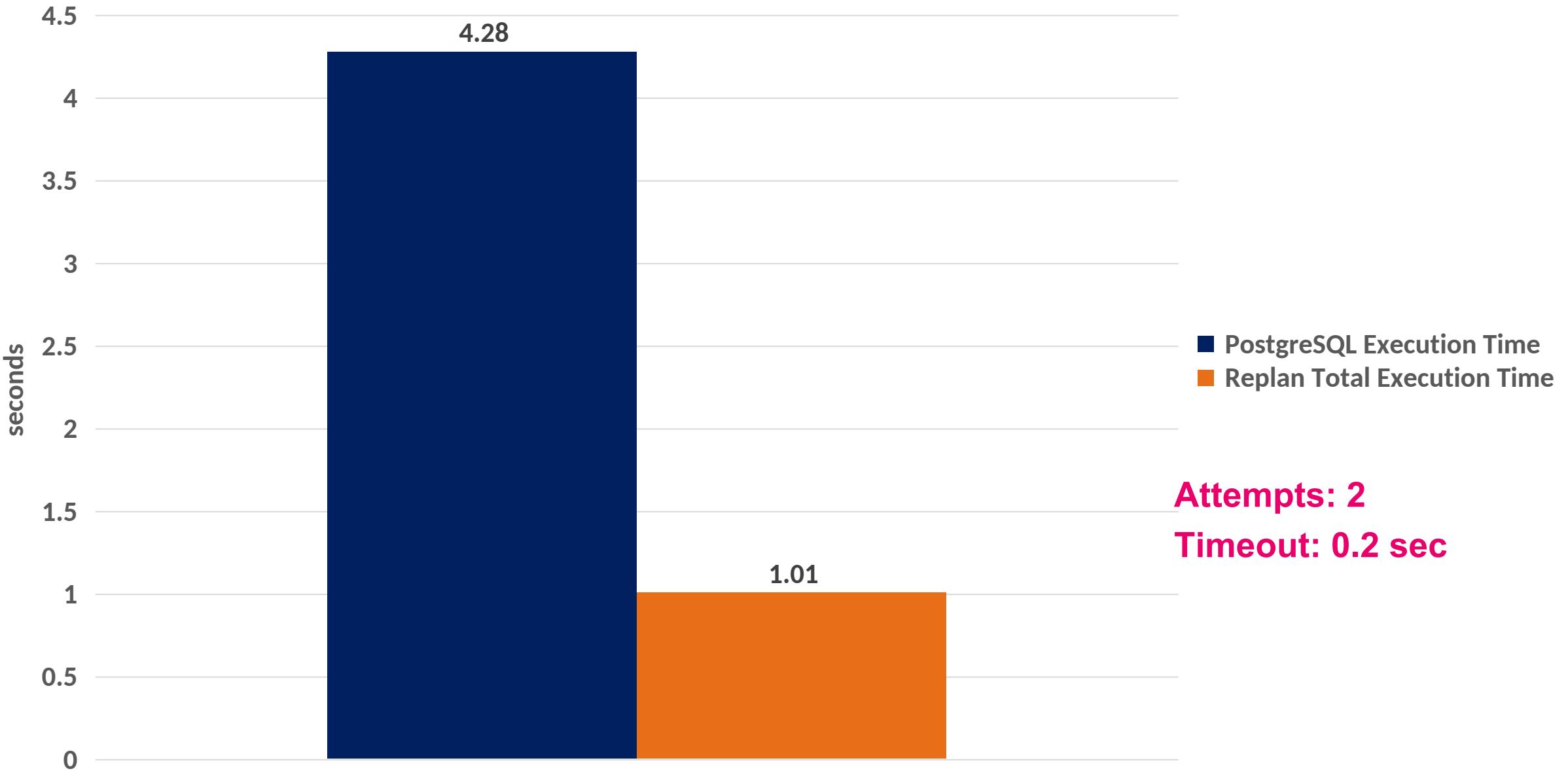


File Edit View Search Terminal Help

tmp\_install/bin/psql -d alena

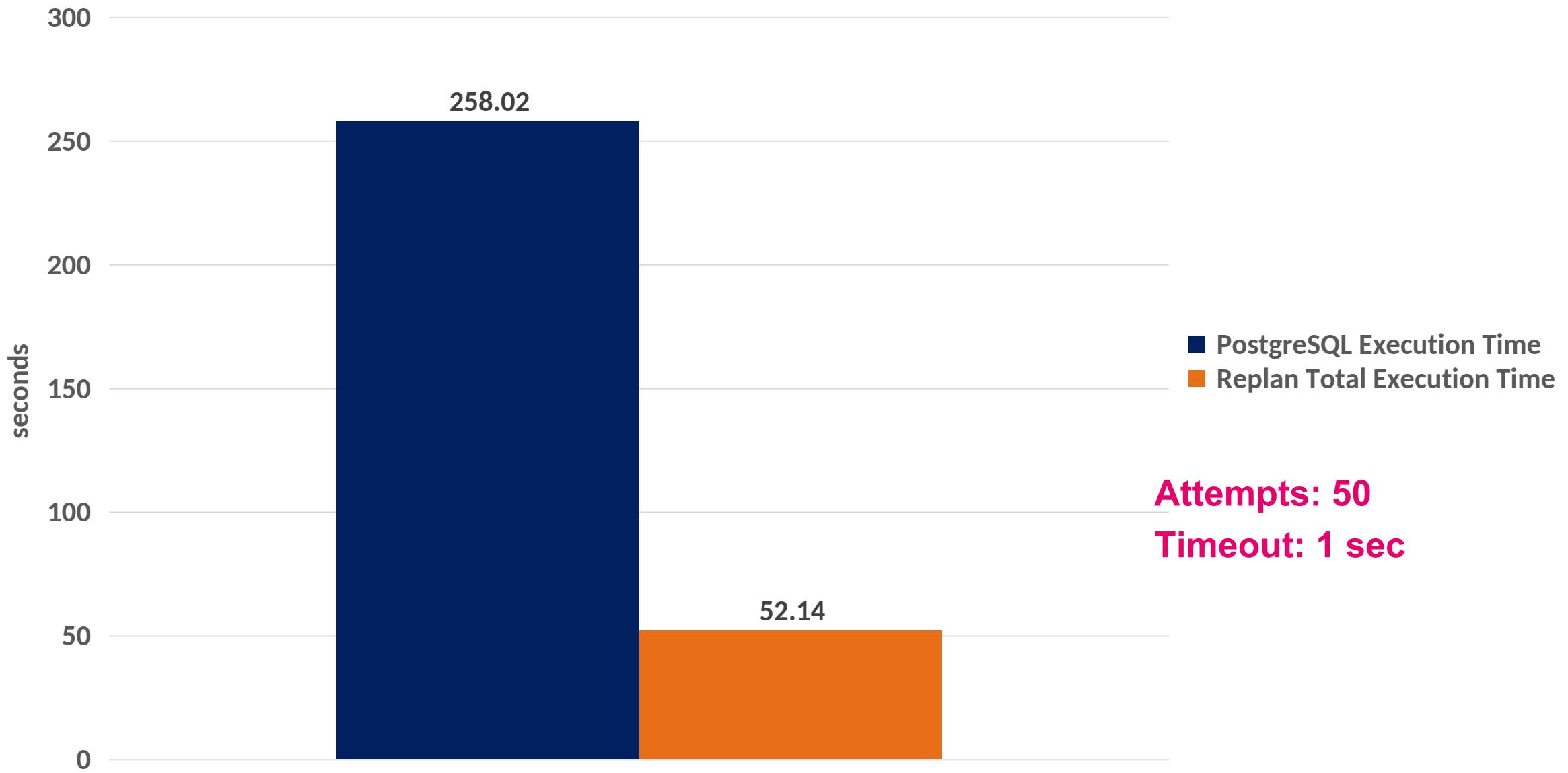
```
SET
SET
alena=# EXPLAIN (ANALYZE, COSTS OFF)
SELECT MIN(cn.name) AS from_company,
       MIN(mc.note) AS production_note,
       MIN(t.title) AS movie_based_on_book
FROM company_name AS cn,
     company_type AS ct,
     keyword AS k,
     link_type AS lt,
     movie_companies AS mc,
     movie_keyword AS mk,
     movie_link AS ml,
     title AS t
WHERE cn.country_code !='[pl]'
    AND (cn.name LIKE '20th Century Fox%'
          OR cn.name LIKE 'Twentieth Century Fox%')
    AND ct.kind != 'production companies'
    AND ct.kind IS NOT NULL
    AND k.keyword IN ('sequel',
                      'revenge',
                      'based-on-novel')
    AND mc.note IS NOT NULL
    AND t.production_year > 1950
    AND lt.id = ml.link_type_id
    AND ml.movie_id = t.id
    AND t.id = mk.movie_id
    AND mk.keyword_id = k.id
    AND t.id = mc.movie_id
    AND mc.company_type_id = ct.id
    AND mc.company_id = cn.id
    AND ml.movie_id = mk.movie_id
    AND ml.movie_id = mc.movie_id
    AND mk.movie_id = mc.movie_id;
alena=# 
```

## Query execution time 11c



```
MIN(n.name) AS voicing_actress,
MIN(t.title) AS voiced_animation
FROM aka_name AS an,
complete_cast AS cc,
comp_cast_type AS cct1,
comp_cast_type AS cct2,
char_name AS chn,
cast_info AS ci,
company_name AS cn,
info_type AS it,
info_type AS it3,
keyword AS k,
movie_companies AS mc,
movie_info AS mi,
movie_keyword AS mk,
name AS n,
person_info AS pi,
role_type AS rt,
title AS t
WHERE cct1.kind ='cast'
AND cct2.kind ='complete+verified'
AND ci.note IN ('(voice)',
'(voice: Japanese version)',
'(voice) (uncredited)',
AND cct2.id = cc.status_id;d_id000 AND 2010
alena=# 
```

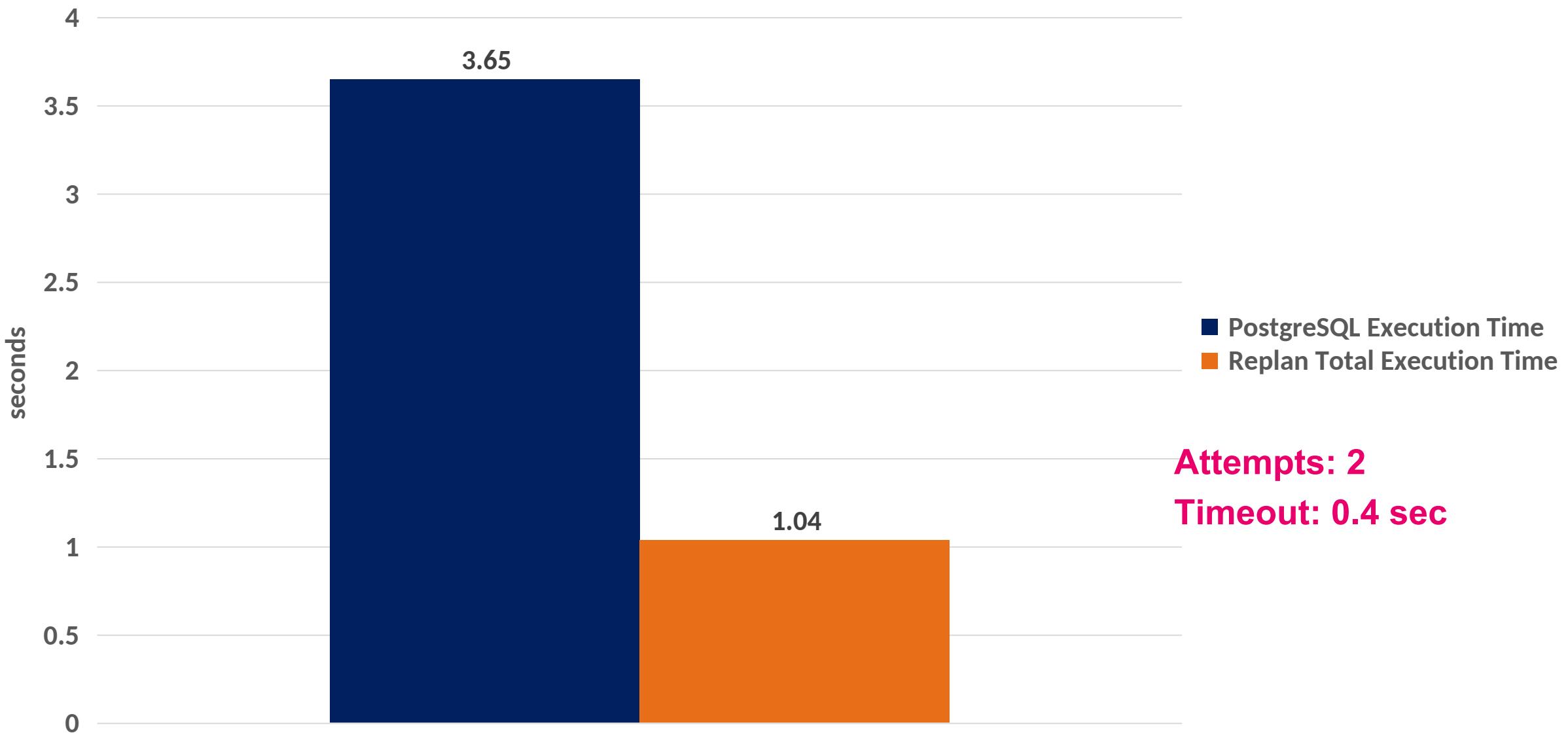
## Query execution time 26c





```
File Edit View Search Terminal Help
MIN(n.name) AS voicing_actress,
MIN(t.title) AS voiced_animation
FROM aka_name AS an,
complete_cast AS cc,
comp_cast_type AS cct1,
comp_cast_type AS cct2,
char_name AS chn,
cast_info AS ci,
company_name AS cn,
info_type AS it,
info_type AS it3,
keyword AS k,
movie_companies AS mc,
movie_info AS mi,
movie_keyword AS mk,
name AS n,
person_info AS pi,
role_type AS rt,
title AS t
WHERE cct1.kind ='cast'
AND cct2.kind ='complete+verified'
AND ci.note IN ('(voice)',
                 '(voice: Japanese version)',
                 '(voice) (uncredited)',
                 '(voice: English version)')
AND cn.country_code ='[us]'
AND it.info = 'release dates'
AND it3.info = 'trivia'
AND k.keyword = 'computer-animation'
AND mi.info IS NOT NULL
AND (mi.info LIKE 'Japan:%200%'
     OR mi.info LIKE 'USA:%200%')
AND n.gender ='f'
AND cct2.id = cc.status_id;d_id000 AND 2010
alena=# 
```

Query execution time: 29c





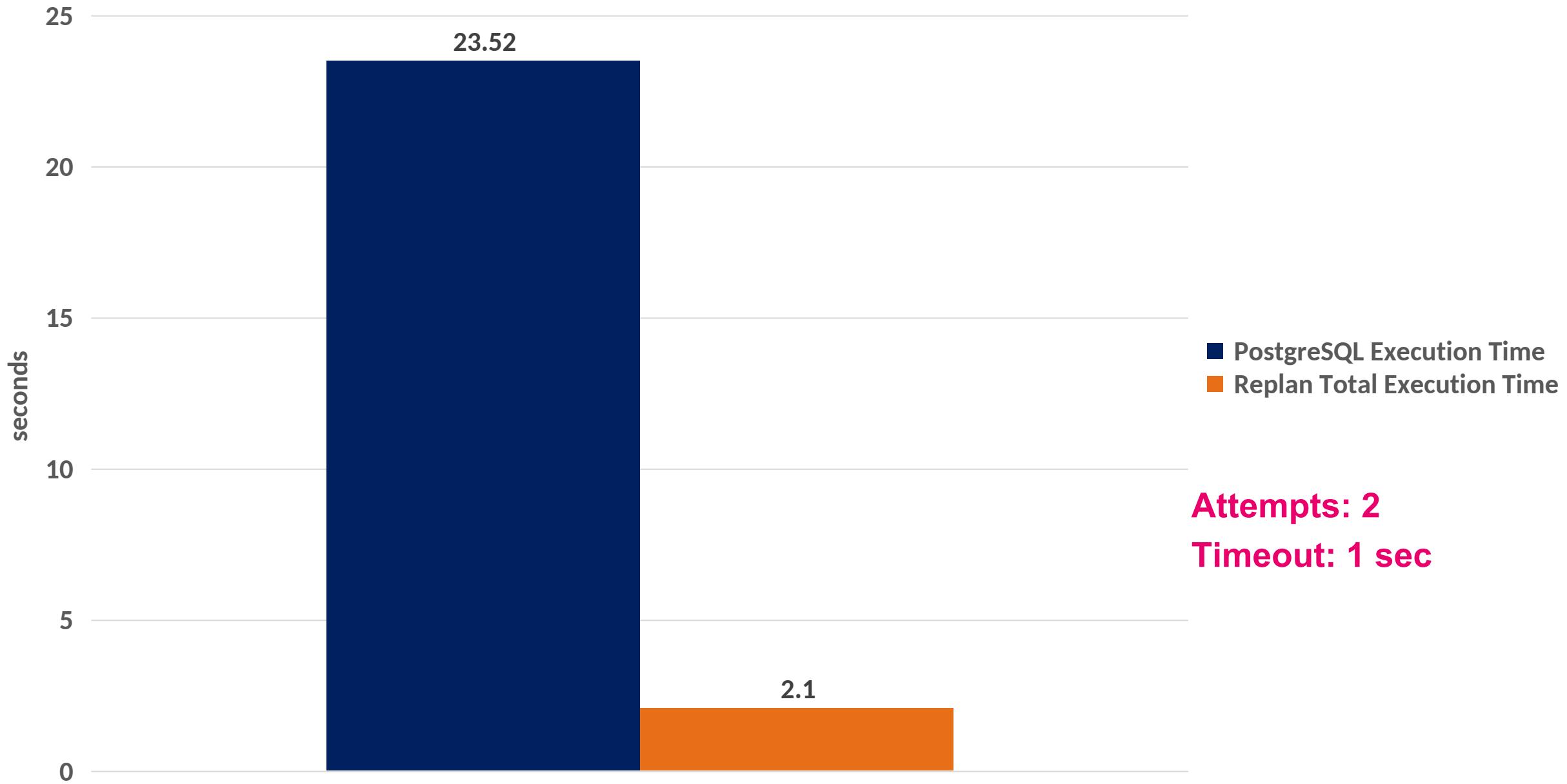
File Edit View Search Terminal Help

SET

SET

```
alena=# EXPLAIN (ANALYZE, COSTS OFF)
SELECT MIN(cn.name) AS from_company,
       MIN(mc.note) AS production_note,
       MIN(t.title) AS movie_based_on_book
FROM company_name AS cn,
     company_type AS ct,
     keyword AS k,
     link_type AS lt,
     movie_companies AS mc,
     movie_keyword AS mk,
     movie_link AS ml,
     title AS t
WHERE cn.country_code !='[pl]'
    AND (cn.name LIKE '20th Century Fox%'
        OR cn.name LIKE 'Twentieth Century Fox%')
    AND ct.kind != 'production companies'
    AND ct.kind IS NOT NULL
    AND k.keyword IN ('sequel',
                      'revenge',
                      'based-on-novel')
    AND mc.note IS NOT NULL
    AND t.production_year > 1950
    AND lt.id = ml.link_type_id
    AND ml.movie_id = t.id
    AND t.id = mk.movie_id
    AND mk.keyword_id = k.id
    AND t.id = mc.movie_id
    AND mc.company_type_id = ct.id
    AND mc.company_id = cn.id
    AND ml.movie_id = mk.movie_id
    AND ml.movie_id = mc.movie_id
    AND mk.movie_id = mc.movie_id;
alena=#
```

## Query execution time (31c)



# Где можно посмотреть Replan

**Бинарники с ванильной версией PostgreSQL**

**PostgresPro Enterprise 16**

# Q&A

Спасибо!

[a.rybakina@postgrespro.ru](mailto:a.rybakina@postgrespro.ru)