ЗАПРОС ДЛЯ ОПТИМИЗАЦИИ

```
SELECT
   u.id,
   u.first_name,
   u.last_name,
   u.age,
   u.gender,
   u.biography,
   u.city,
   u.password
FROM users u
WHERE u.first_name LIKE {$1}% AND u.last_name LIKE {$2}%
ORDER BY u.id;
```

Поиск анкет по префиксу имени и фамилии (одновременно)

Сортировать вывод по id анкеты.

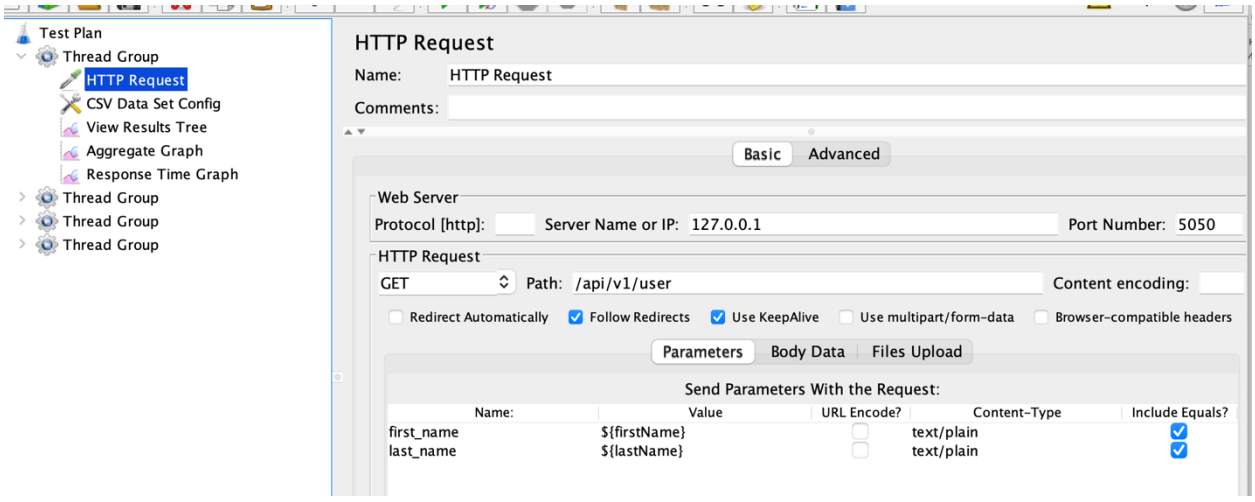
В качестве БД – ***Postgres 14.*** Нагрузочное тестирование - **Jmeter 5.5**

ГЕНЕРАЦИЯ ДАННЫХ

1ккк данных генерировались на python3.10 библиотекой Faker

Для query-параметров http запроса в нагрузочном тестировании были выбраны 10к префиксов от 1 до 4ех символов

НАСТРОЙКИ JMETER



4 Thread Group на 1/10/100/1000 тредов – запуск на 1 минуту

EXPLAIN ЗАПРОСА БЕЗ ИНДЕКСОВ

```
Gather Merge  (cost=62643.23..62758.97 rows=992 width=399)
(actual time=106.764..122.439 rows=528 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  ->  Sort  (cost=61643.21..61644.45 rows=496 width=399) (actual
time=85.354..88.245 rows=176 loops=3)
        Sort Key: id
        Sort Method: quicksort  Memory: 155kB
        Worker 0:  Sort Method: quicksort  Memory: 106kB
        Worker 1:  Sort Method: quicksort  Memory: 107kB
        ->  Parallel Seq Scan on users u  (cost=0.00..61621.00
rows=496 width=399) (actual time=0.440..81.834 rows=176 loops=3)
              Filter: ((first_name ~~ 'A%'::text) AND (last_name
~~ 'Ба%'::text))
              Rows Removed by Filter: 333157
Planning Time: 0.092 ms
Execution Time: 130.505 ms
```

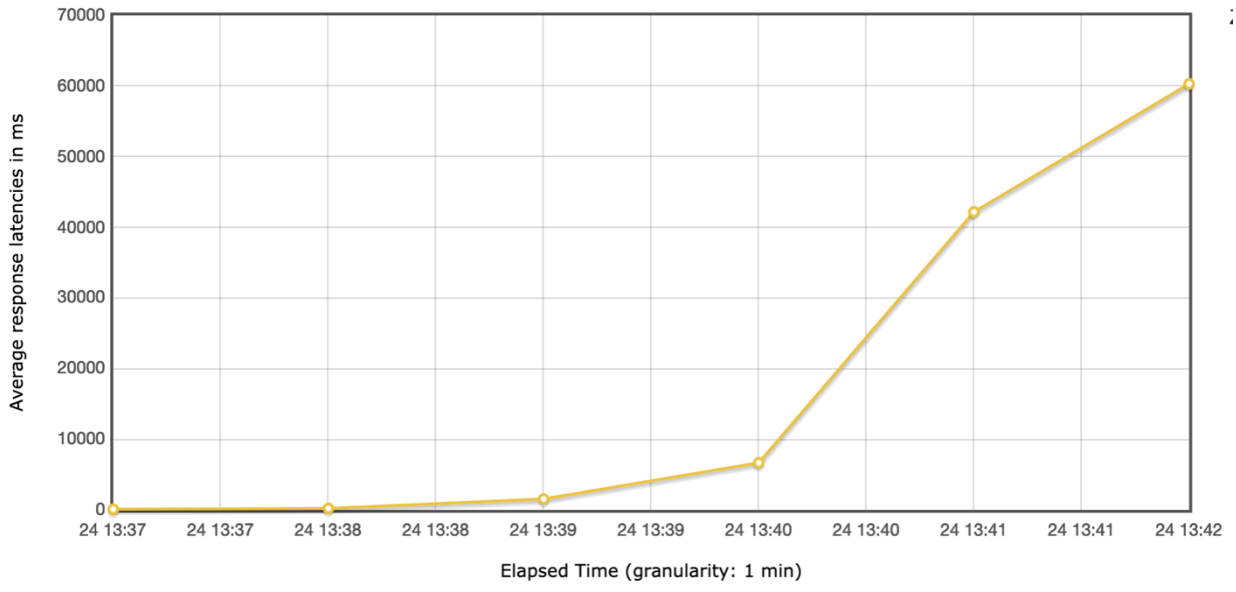
НАГРУЗОЧНОЕ ТЕСТИРОВАНИЕ БЕЗ ИНДЕКСОВ

```
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary +    141 in 00:00:14 =    9.8/s Avg:    98 Min:    65 Max:  2009 Err:     0 (0.00%) Active: 1 Started: 1 Finished: 0
summary +    386 in 00:00:30 =   12.9/s Avg:    77 Min:    66 Max:   160 Err:     0 (0.00%) Active: 1 Started: 1 Finished: 0
summary =    527 in 00:00:44 =   11.9/s Avg:    83 Min:    65 Max:  2009 Err:     0 (0.00%)
summary +    448 in 00:00:30 =   14.9/s Avg:   337 Min:    65 Max:  1311 Err:     0 (0.00%) Active: 10 Started: 11 Finished: 1
summary =    975 in 00:01:14 =   13.1/s Avg:   200 Min:    65 Max:  2009 Err:     0 (0.00%)
summary +    509 in 00:00:30 =   17.0/s Avg:   588 Min:   261 Max:  1132 Err:     0 (0.00%) Active: 10 Started: 11 Finished: 1
summary =   1484 in 00:01:44 =   14.2/s Avg:   333 Min:    65 Max:  2009 Err:     0 (0.00%)
summary +    486 in 00:00:30 =   16.2/s Avg:  2558 Min:   303 Max:  6896 Err:     0 (0.00%) Active: 100 Started: 111 Finished: 11
summary =   1970 in 00:02:14 =   14.7/s Avg:   882 Min:    65 Max:  6896 Err:     0 (0.00%)
summary +    492 in 00:00:30 =   16.4/s Avg:  6109 Min:  5512 Max:  7088 Err:     0 (0.00%) Active: 100 Started: 111 Finished: 11
summary =   2462 in 00:02:44 =   15.0/s Avg:  1927 Min:    65 Max:  7088 Err:     0 (0.00%)
summary +    489 in 00:00:30 =   16.3/s Avg:  7255 Min:  5471 Max: 14217 Err:     0 (0.00%) Active: 1000 Started: 1111 Finished: 111
summary =   2951 in 00:03:14 =   15.2/s Avg:  2810 Min:    65 Max: 14217 Err:     0 (0.00%)
summary +    504 in 00:00:30 =   16.9/s Avg: 28954 Min: 14264 Max: 43626 Err:     0 (0.00%) Active: 1000 Started: 1111 Finished: 111
summary =   3455 in 00:03:44 =   15.4/s Avg:  6623 Min:    65 Max: 43626 Err:     0 (0.00%)
summary +    495 in 00:00:30 =   16.5/s Avg: 55770 Min: 43627 Max: 65440 Err:     0 (0.00%) Active: 772 Started: 1111 Finished: 339
summary =   3950 in 00:04:14 =   15.5/s Avg: 12782 Min:    65 Max: 65440 Err:     0 (0.00%)
summary +    498 in 00:00:30 =   16.6/s Avg: 60231 Min: 59430 Max: 61084 Err:     0 (0.00%) Active: 274 Started: 1111 Finished: 837
summary =   4448 in 00:04:44 =   15.6/s Avg: 18095 Min:    65 Max: 65440 Err:     0 (0.00%)
summary +    273 in 00:00:16 =   17.1/s Avg: 60469 Min: 59929 Max: 61211 Err:     0 (0.00%) Active: 0 Started: 1111 Finished: 1111
summary =   4721 in 00:05:00 =   15.7/s Avg: 20545 Min:    65 Max: 65440 Err:     0 (0.00%)
```

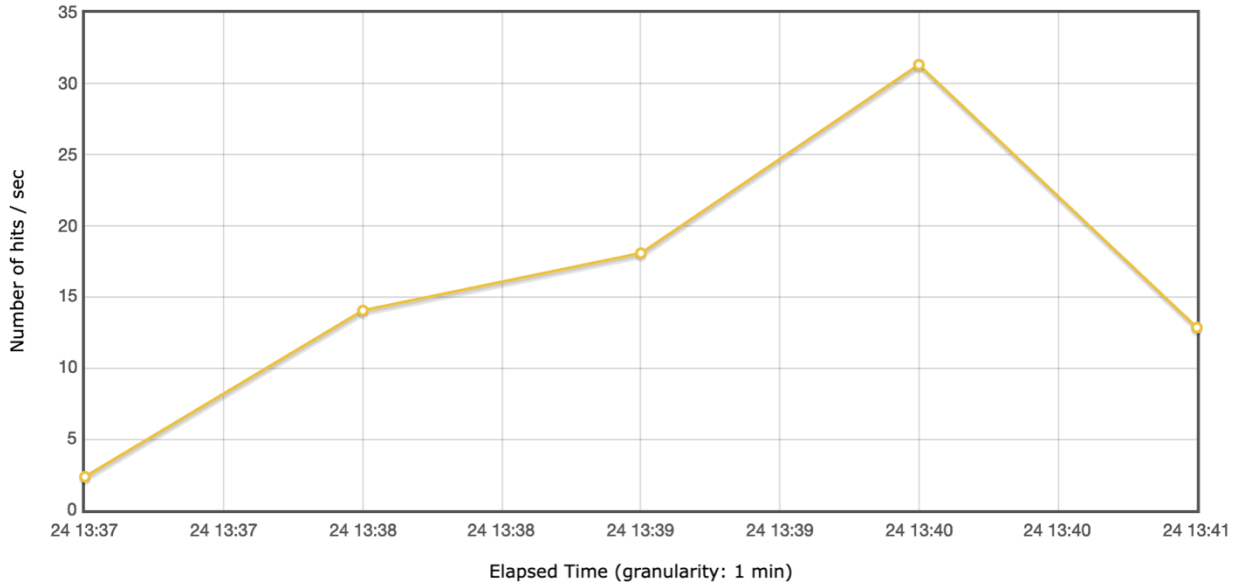
Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**4721**	**0**	**0.00%**	**20545.56**	**65**	**65440**	**6108.00**	**60297.00**	**60659.90**	**62329.50**	**15.74**	**1716.95**	**2.76**
HTTP Request	4721	0	0.00%	20545.56	65	65440	6108.00	60297.00	60659.90	62329.50	15.74	1716.95	2.76

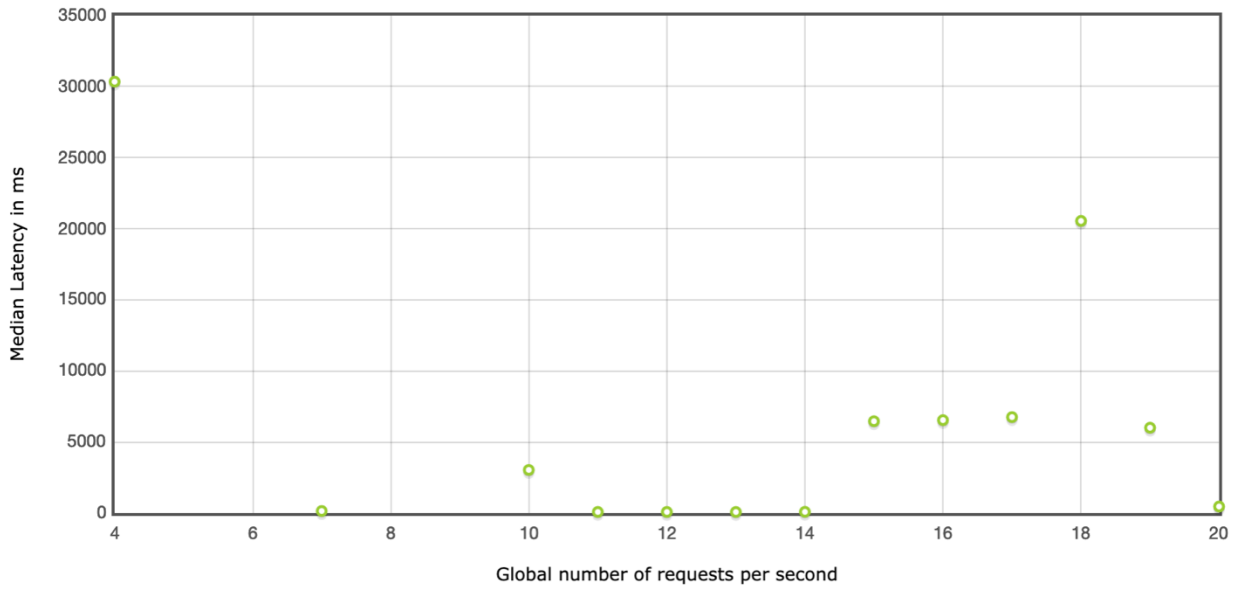
Latencies Over Time



Hits Per Second



Latency Vs Request



СОЗДАНИЕ ИНДЕКСОВ

Будем использовать b-tree индекс для first_name, last_name

По документации что бы эффективно матчить префикс, нужно создать индекс со специальным оператором либо использовать COLLATE “C”

```
CREATE INDEX users_first_name_idx ON users (first_name
text_pattern_ops);
CREATE INDEX users_last_name_idx ON users (last_name
text_pattern_ops);
```

EXPLAIN ЗАПРОСА С ИНДЕКСАМИ

```
Sort  (cost=6297.53..6300.51 rows=1190 width=399) (actual
time=23.231..30.848 rows=528 loops=1)
  Sort Key: id
  Sort Method: quicksort  Memory: 319kB
  ->  Bitmap Heap Scan on users u  (cost=1868.13..6236.74
rows=1190 width=399) (actual time=7.981..16.319 rows=528
loops=1)
        Filter: ((first_name ~~ 'A%'::text) AND (last_name ~~
'Ба%'::text))
        Heap Blocks: exact=527
        ->  BitmapAnd  (cost=1868.13..1868.13 rows=1237 width=0)
(actual time=7.888..8.013 rows=0 loops=1)
              ->  Bitmap Index Scan on users_last_name_idx
(cost=0.00..133.27 rows=9684 width=0) (actual time=0.307..0.319
rows=4088 loops=1)
                    Index Cond: ((last_name ~>=~ 'Ба'::text) AND
(last_name ~<~ 'Бб'::text))
              ->  Bitmap Index Scan on users_first_name_idx
(cost=0.00..1734.03 rows=127760 width=0) (actual
time=7.349..7.410 rows=129059 loops=1)
                    Index Cond: ((first_name ~>=~ 'А'::text) AND
(first_name ~<~ 'Б'::text))
Planning Time: 0.182 ms
Execution Time: 37.205 ms
```

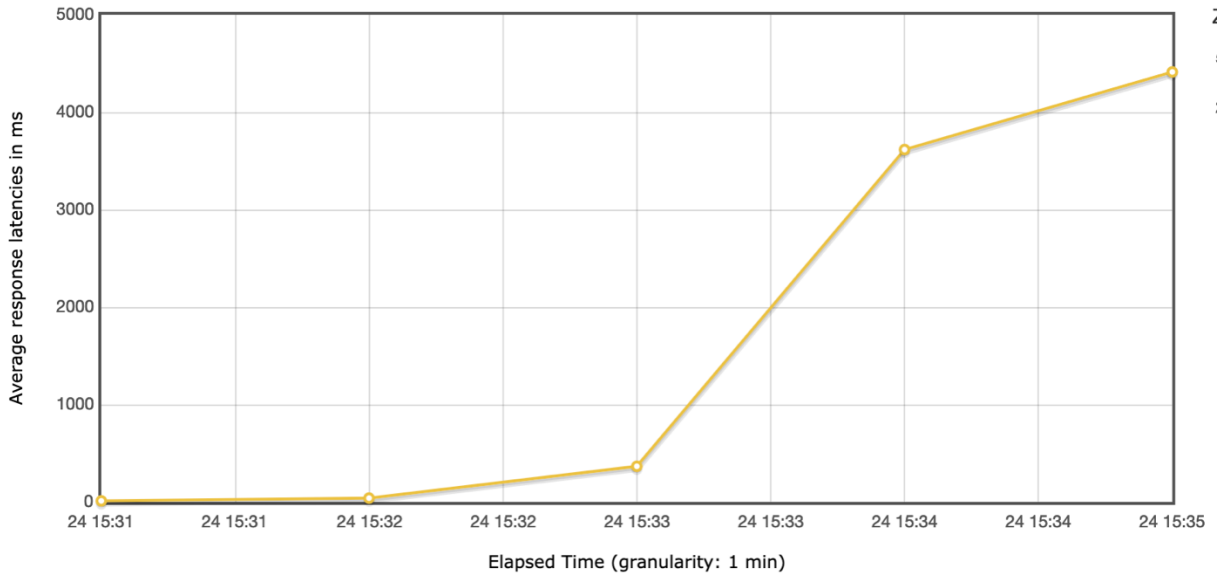
НАГРУЗОЧНОЕ ТЕСТИРОВАНИЕ С ИНДЕКСАМИ

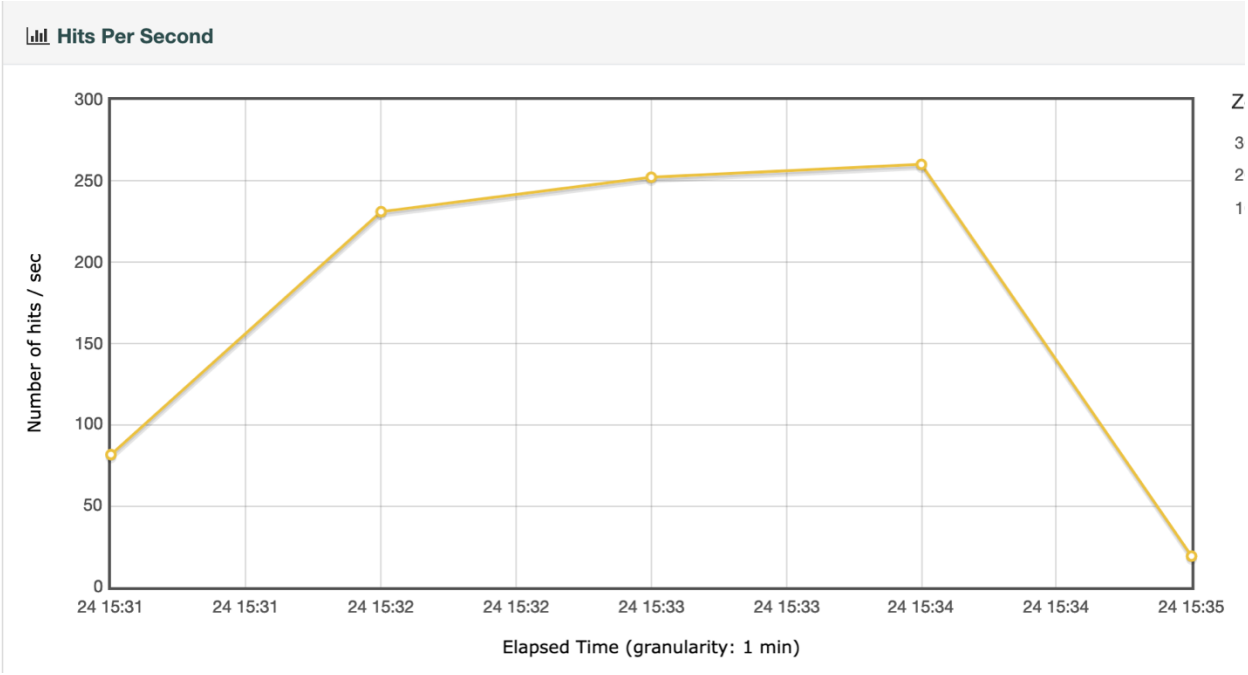
```
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary +      1 in 00:00:00 =    2.8/s Avg:    66 Min:    66 Max:    66 Err:     0 (0.00%) Active: 1 Started: 1 Finished: 0
summary +   2131 in 00:00:25 =   85.4/s Avg:    11 Min:     1 Max:   585 Err:     0 (0.00%) Active: 1 Started: 1 Finished: 0
summary =   2132 in 00:00:25 =   84.2/s Avg:    11 Min:     1 Max:   585 Err:     0 (0.00%)
summary +   2765 in 00:00:30 =   92.2/s Avg:    10 Min:     1 Max:   209 Err:     0 (0.00%) Active: 1 Started: 1 Finished: 0
summary =   4897 in 00:00:55 =   88.5/s Avg:    11 Min:     1 Max:   585 Err:     0 (0.00%)
summary +   6540 in 00:00:30 =  218.0/s Avg:    38 Min:     1 Max:   883 Err:     0 (0.00%) Active: 10 Started: 11 Finished: 1
summary =  11437 in 00:01:25 =  134.1/s Avg:    26 Min:     1 Max:   883 Err:     0 (0.00%)
summary +   7324 in 00:00:30 =  244.1/s Avg:    40 Min:     1 Max:   819 Err:     0 (0.00%) Active: 10 Started: 11 Finished: 1
summary =  18761 in 00:01:55 =  162.7/s Avg:    32 Min:     1 Max:   883 Err:     0 (0.00%)
summary +   7596 in 00:00:30 =  253.2/s Avg:   330 Min:     1 Max:  1465 Err:     0 (0.00%) Active: 100 Started: 111 Finished: 11
summary =  26357 in 00:02:25 =  181.4/s Avg:   118 Min:     1 Max:  1465 Err:     0 (0.00%)
summary +   7459 in 00:00:30 =  248.6/s Avg:   402 Min:   210 Max:  1443 Err:     0 (0.00%) Active: 100 Started: 111 Finished: 11
summary =  33816 in 00:02:55 =  192.9/s Avg:   180 Min:     1 Max:  1465 Err:     0 (0.00%)
summary +   7162 in 00:00:30 =  238.6/s Avg:  3263 Min:   209 Max:  5322 Err:     0 (0.00%) Active: 1000 Started: 1111 Finished: 111
summary =  40978 in 00:03:25 =  199.6/s Avg:   719 Min:     1 Max:  5322 Err:     0 (0.00%)
summary +   7558 in 00:00:30 =  252.0/s Avg:  3951 Min:  3540 Max:  5027 Err:     0 (0.00%) Active: 1000 Started: 1111 Finished: 111
summary =  48536 in 00:03:55 =  206.3/s Avg:  1223 Min:     1 Max:  5322 Err:     0 (0.00%)
summary +   2145 in 00:00:10 =  209.5/s Avg:  4413 Min:  3939 Max:  5955 Err:     0 (0.00%) Active: 0 Started: 1111 Finished: 1111
summary =  50681 in 00:04:06 =  206.4/s Avg:  1358 Min:     1 Max:  5955 Err:     0 (0.00%)
Tidying up ...    @ 2023 Jan 24 15:35:10 MSK (1674563710240)
```

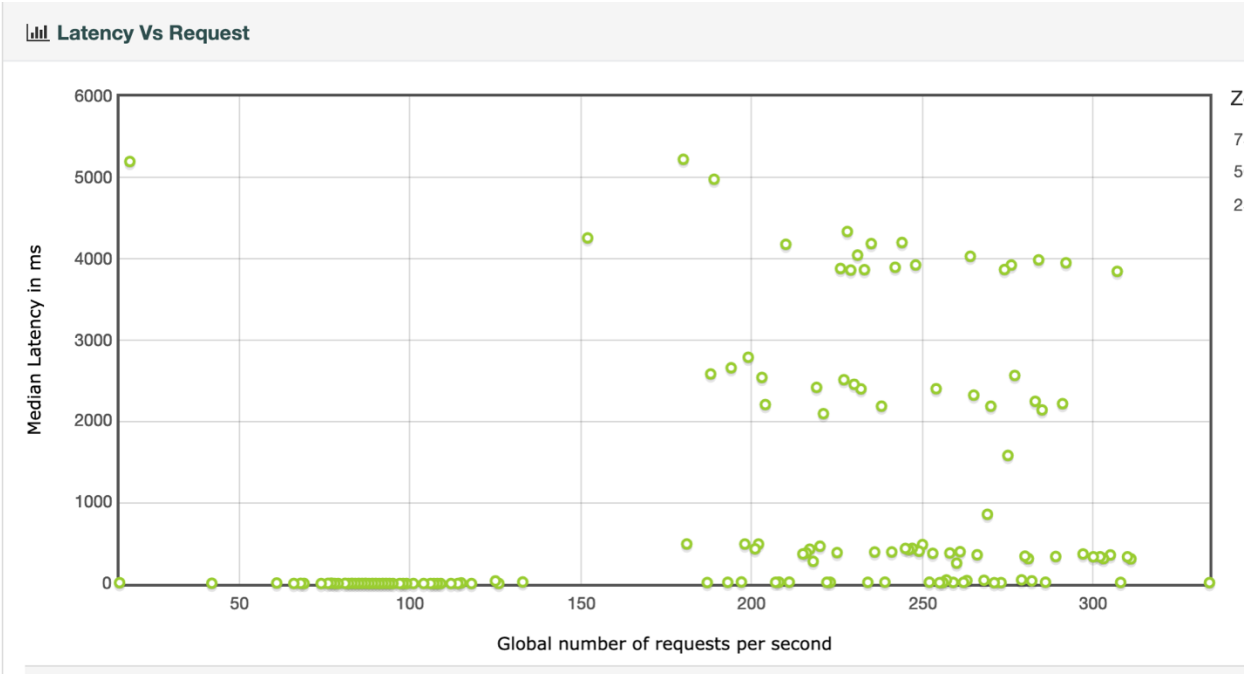
Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**50681**	**0**	**0.00%**	**1358.06**	**1**	**5955**	**3929.50**	**4392.00**	**4531.00**	**5136.97**	**206.64**	**23176.41**	**36.32**
HTTP Request	50681	0	0.00%	1358.06	1	5955	3929.50	4392.00	4531.00	5136.97	206.64	23176.41	36.32

Latencies Over Time







ВЫВОД

Благодаря индексу 99 перцентиль латенси уменьшился примерно в 12 раз

А пропускная способность увеличилась примерно в 12 раз

ДОПОЛНЕНИЕ

Дополнение требований: надо искать не только по префиксу, а в любой части слова => поиск регистронезависимый.

Запрос будет выглядеть так

```
SELECT
   u.id,
   u.first_name,
   u.last_name,
   u.age,
   u.gender,
   u.biography,
   u.city,
   u.password
FROM users u
WHERE u.first_name ILIKE %{$1}% AND u.last_name ILIKE %{$2}%
ORDER BY u.id;
```

Для оптимизации такого запроса в Psql можно использовать расширение **pg_trgm.**

Без индекса

```
Gather Merge  (cost=124635.60..126553.97 rows=16442 width=400)
(actual time=588.239..951.527 rows=16922 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  ->  Sort  (cost=123635.57..123656.13 rows=8221 width=400)
(actual time=562.722..630.967 rows=5641 loops=3)
        Sort Key: id
        Sort Method: quicksort  Memory: 3328kB
        Worker 0:  Sort Method: quicksort  Memory: 3313kB
        Worker 1:  Sort Method: quicksort  Memory: 3317kB
        ->  Parallel Seq Scan on users u  (cost=0.00..123101.00
rows=8221 width=400) (actual time=12.548..472.653 rows=5641
loops=3)
              Filter: ((first_name ~~* '%Ан%'::text) AND
(last_name ~~* '%Б%'::text))
              Rows Removed by Filter: 327693
```

```
Planning Time: 0.222 ms
JIT:
  Functions: 12
"  Options: Inlining false, Optimization false, Expressions
true, Deforming true"
"  Timing: Generation 1.705 ms, Inlining 0.000 ms, Optimization
0.988 ms, Emission 11.707 ms, Total 14.400 ms"
Execution Time: 1155.568 ms
```

Создадим индекс

```
CREATE EXTENSION IF NOT EXISTS pg_trgm;
CREATE INDEX users_trgm_idx ON users USING GIN (first_name
gin_trgm_ops, last_name gin_trgm_ops);
```

```
Gather Merge  (cost=86798.54..88716.91 rows=16442 width=400)
(actual time=1301.881..1665.122 rows=16922 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  ->  Sort  (cost=85798.52..85819.07 rows=8221 width=400)
(actual time=1274.179..1340.775 rows=5641 loops=3)
        Sort Key: id
        Sort Method: quicksort  Memory: 3431kB
        Worker 0:  Sort Method: quicksort  Memory: 3253kB
        Worker 1:  Sort Method: quicksort  Memory: 3274kB
        ->  Parallel Bitmap Heap Scan on users u
(cost=33902.23..85263.94 rows=8221 width=400) (actual
time=743.935..1189.288 rows=5641 loops=3)
              Recheck Cond: ((first_name ~~* '%Ан%'::text) AND
(last_name ~~* '%Б%'::text))
              Rows Removed by Index Recheck: 327693
              Heap Blocks: exact=21503
              ->  Bitmap Index Scan on users_trgm_idx
(cost=0.00..33897.30 rows=19730 width=0) (actual
time=757.267..757.278 rows=1000000 loops=1)
                    Index Cond: ((first_name ~~* '%Ан%'::text)
AND (last_name ~~* '%Б%'::text))
```

```
Planning Time: 0.348 ms
Execution Time: 1867.997 ms
```

Попробуем tsvector

Запрос меняем на аналогичный, через tsvector

```
ALTER TABLE users ADD COLUMN ts_first_name_index_col tsvector;
ALTER TABLE users ADD COLUMN ts_last_name_index_col tsvector;
```

Будем держать в этих колонках tsvector дробленный по символам.

```
UPDATE users SET ts_first_name_index_col = to_tsvector(

array_to_string(regexp_split_to_array(coalesce(first_name, ''),
'\s*'), ' ')
    );
UPDATE users SET ts_last_name_index_col = to_tsvector(

array_to_string(regexp_split_to_array(coalesce(last_name, ''),
'\s*'), ' ')
    );


SELECT
    u.id,
    u.first_name,
    u.last_name,
    u.age,
    u.gender,
    u.biography,
    u.city,
    u.password
FROM users u
WHERE u.ts_first_name_index_col @@ to_tsquery('а <-> н') AND
      u.ts_last_name_index_col @@ to_tsquery('б')
ORDER BY u.id;
```

Создаем индекс

```
CREATE INDEX ts_first_name_last_name_idx ON users USING GIN
(ts_first_name_index_col, ts_last_name_index_col);
```

```
Gather Merge  (cost=113466.66..117014.74 rows=30410 width=399)
(actual time=371.091..747.379 rows=16922 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  ->  Sort  (cost=112466.64..112504.65 rows=15205 width=399)
(actual time=346.382..414.771 rows=5641 loops=3)
        Sort Key: id
        Sort Method: quicksort  Memory: 3555kB
        Worker 0:  Sort Method: quicksort  Memory: 3191kB
        Worker 1:  Sort Method: quicksort  Memory: 3213kB
        ->  Parallel Bitmap Heap Scan on users u
(cost=822.53..108655.98 rows=15205 width=399) (actual
time=40.703..254.400 rows=5641 loops=3)
              Recheck Cond: ((ts_first_name_index_col @@
to_tsquery('а <-> н'::text)) AND (ts_last_name_index_col @@
to_tsquery('б'::text)))
              Rows Removed by Index Recheck: 7340
              Heap Blocks: exact=11817
              ->  Bitmap Index Scan on
ts_first_name_last_name_idx  (cost=0.00..813.41 rows=36491
width=0) (actual time=49.901..49.914 rows=38943 loops=1)
                    Index Cond: ((ts_first_name_index_col @@
to_tsquery('а <-> н'::text)) AND (ts_last_name_index_col @@
to_tsquery('б'::text)))
Planning Time: 0.242 ms
JIT:
  Functions: 18
"  Options: Inlining false, Optimization false, Expressions
true, Deforming true"
"  Timing: Generation 3.094 ms, Inlining 0.000 ms, Optimization
1.423 ms, Emission 23.702 ms, Total 28.219 ms"
Execution Time: 952.218 ms
```

Видно, что триграммы показали себя лучше

ВЫВОД V2

Для поиска по префиксу

```
WHERE u.first_name LIKE {$1}% AND u.last_name LIKE {$2}%
```

```
CREATE INDEX users_first_name_idx ON users (first_name
text_pattern_ops);
CREATE INDEX users_last_name_idx ON users (last_name
text_pattern_ops);
```

Для регистронезависимого поиска в словах

```
WHERE u.first_name ILIKE %{$1}% AND u.last_name ILIKE %{$2}%
```

```
CREATE INDEX users_trgm_idx ON users USING GIN (first_name
gin_trgm_ops, last_name gin_trgm_ops);
```

Для полнотекстового поиска, который не рассматривался в этом ДЗ, подойдет tsvector