Отчет по домашнему заданию на тему «Производительность индексов»

Четвериков В.В.

Цель и критерии оценки

Цель:
В результате выполнения ДЗ вы воспользуетесь/создадите набор тестовых данных для проведения нагрузочного тестирования, подберете наиболее подходящие индексы и проведете тесты производительности.

В данном задании тренируются навыки:
- (опционально) генерация тестовых данных;
- работа с индексами;
- нагрузочное тестирование;

Критерии оценки:
ДЗ принимается в виде отчета по выполненной работе. Оценка происходит по принципу зачет/незачет.

Требования:
- Правильно выбраны индексы.
- Нагрузочное тестирование проведено и результаты адекватны.
- В случае генерации тестовых данных самостоятельно, убедиться что эти данные "похожи" на настоящие

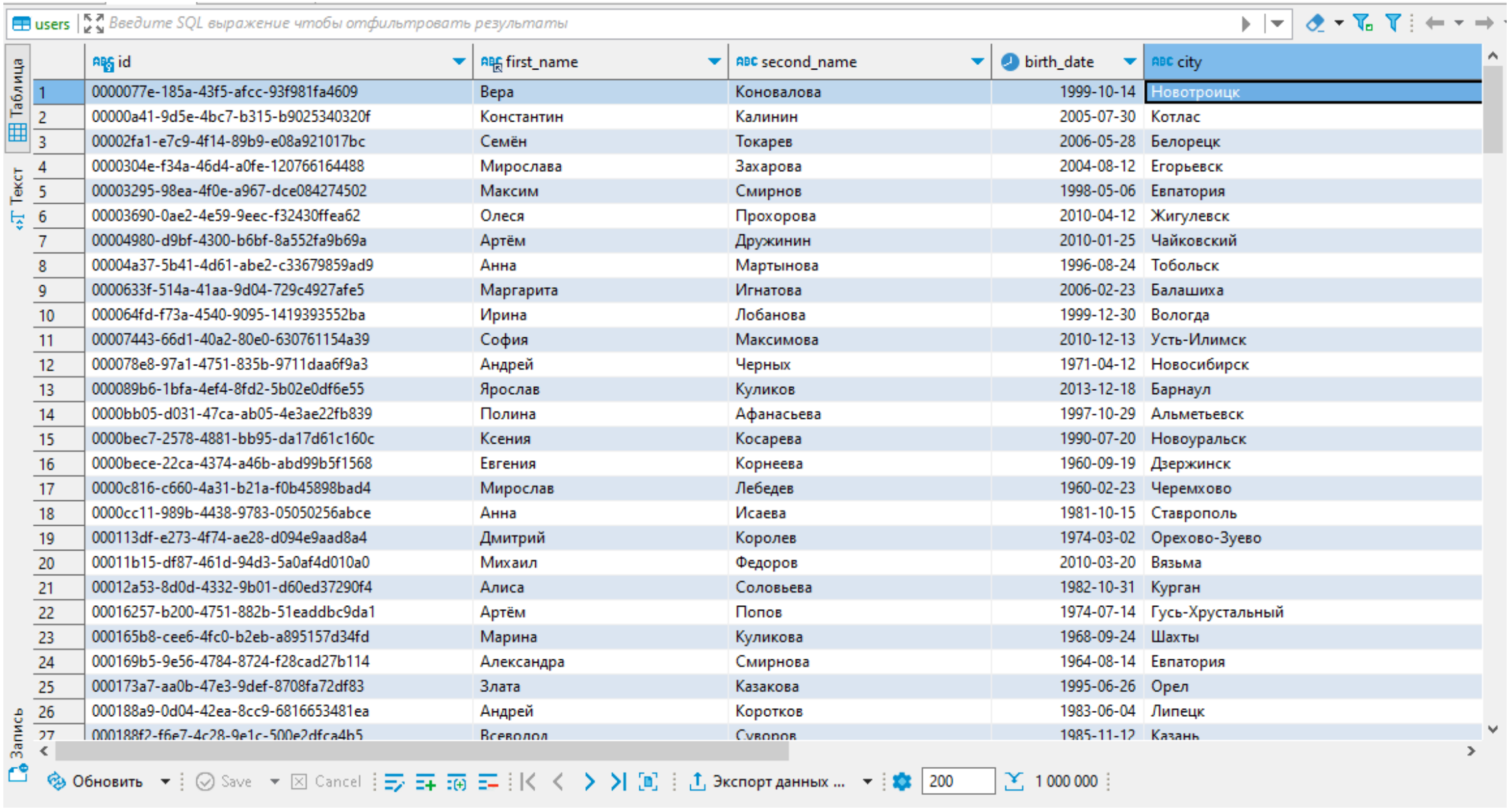
Описание выполнения домашнего задания

Описание/Пошаговая инструкция выполнения домашнего задания:

- Сгенерировать любым способ 1,000,000 анкет. Имена и Фамилии должны быть реальными (чтобы учитывать селективность индекса) (или воспользовать уже готовым списком)
- Реализовать функционал поиска анкет по префиксу имени и фамилии (одновременно) в вашей социальной сети (реализовать метод /user/search из спецификации) (запрос в форме firstName LIKE ? and secondName LIKE ?). Сортировать вывод по id анкеты. Использовать InnoDB движок.
- Провести нагрузочные тесты этого метода. Поиграть с количеством одновременных запросов. 1/10/100/1000. Построить графики и сохранить их в отчет
- Сделать подходящий индекс. Повторить пункт 3 и 4.
- В качестве результата предоставить отчет в котором должны быть:
- графики latency до индекса;
- графики throughput до индекса;
- графики latency после индекса;
- графики throughput после индекса;
- запрос добавления индекса;
- explain запросов после индекса;
- объяснение почему индекс именно такой;

Подготовка тестовых данных

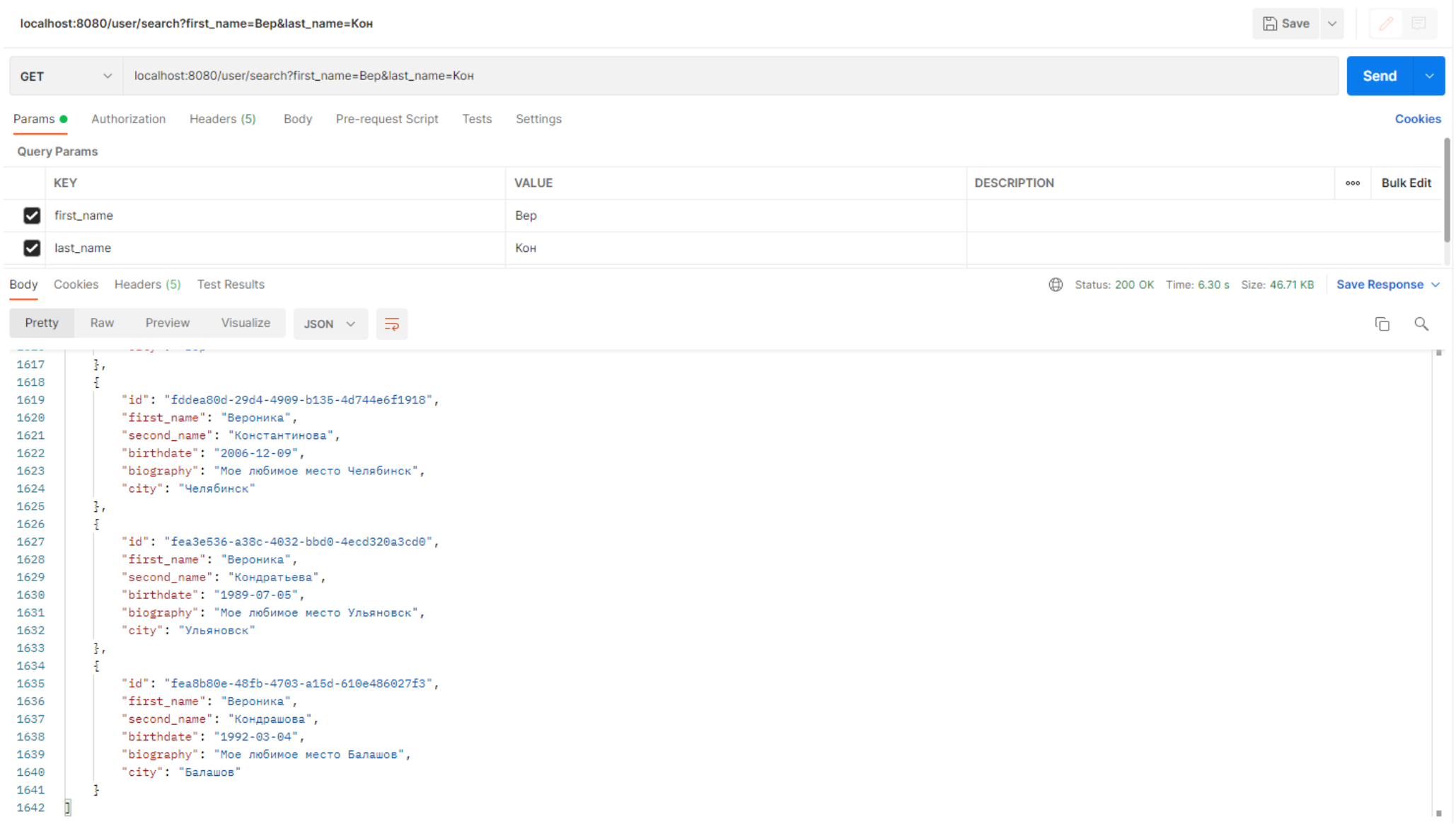
На основе предоставленного списка пользователей были сгенерированы и загружены в таблицу **users** анкеты для 1 млн. пользователей.



	id	first_name	second_name	birth_date	city
1	0000077e-185a-43f5-afcc-93f981fa4609	Вера	Коновалова	1999-10-14	Новотроицк
2	00000a41-9d5e-4bc7-b315-b9025340320f	Константин	Калинин	2005-07-30	Котлас
3	00002fa1-e7c9-4f14-89b9-e08a921017bc	Семён	Токарев	2006-05-28	Белорецк
4	0000304e-f34a-46d4-a0fe-120766164488	Мирослава	Захарова	2004-08-12	Егорьевск
5	00003295-98ea-4f0e-a967-dce084274502	Максим	Смирнов	1998-05-06	Евпатория
6	00003690-0ae2-4e59-9eec-f32430ffea62	Олеся	Прохорова	2010-04-12	Жигулевск
7	00004980-d9bf-4300-b6bf-8a552fa9b69a	Артём	Дружинин	2010-01-25	Чайковский
8	00004a37-5b41-4d61-abe2-c33679859ad9	Анна	Мартынова	1996-08-24	Тобольск
9	0000633f-514a-41aa-9d04-729c4927afe5	Маргарита	Игнатова	2006-02-23	Балашиха
10	000064fd-f73a-4540-9095-1419393552ba	Ирина	Лобанова	1999-12-30	Вологда
11	00007443-66d1-40a2-80e0-630761154a39	София	Максимова	2010-12-13	Усть-Илимск
12	000078e8-97a1-4751-835b-9711daa6f9a3	Андрей	Черных	1971-04-12	Новосибирск
13	000089b6-1bfa-4ef4-8fd2-5b02e0df6e55	Ярослав	Куликов	2013-12-18	Барнаул
14	0000bb05-d031-47ca-ab05-4e3ae22fb839	Полина	Афанасьева	1997-10-29	Альметьевск
15	0000bec7-2578-4881-bb95-da17d61c160c	Ксения	Косарева	1990-07-20	Новоуральск
16	0000bece-22ca-4374-a46b-abd99b5f1568	Евгения	Корнеева	1960-09-19	Дзержинск
17	0000c816-c660-4a31-b21a-f0b45898bad4	Мирослав	Лебедев	1960-02-23	Черемхово
18	0000cc11-989b-4438-9783-05050256abce	Анна	Исаева	1981-10-15	Ставрополь
19	000113df-e273-4f74-ae28-d094e9aad8a4	Дмитрий	Королев	1974-03-02	Орехово-Зуево
20	00011b15-df87-461d-94d3-5a0af4d010a0	Михаил	Федоров	2010-03-20	Вязьма
21	00012a53-8d0d-4332-9b01-d60ed37290f4	Алиса	Соловьева	1982-10-31	Курган
22	00016257-b200-4751-882b-51eaddbc9da1	Артём	Попов	1974-07-14	Гусь-Хрустальный
23	000165b8-cee6-4fc0-b2eb-a895157d34fd	Марина	Куликова	1968-09-24	Шахты
24	000169b5-9e56-4784-8724-f28cad27b114	Александра	Смирнова	1964-08-14	Евпатория
25	000173a7-aa0b-47e3-9def-8708fa72df83	Злата	Казакова	1995-06-26	Орел
26	000188a9-0d04-42ea-8cc9-6816653481ea	Андрей	Коротков	1983-06-04	Липецк
27	000188f2-f6e7-4c28-9e1c-500e2dfca4b5	Всеволод	Суворов	1985-11-12	Казань

Реализация метода поиска анкет

Был реализован метод /user/search из спецификации openapi для поиска анкет по префиксу имени и фамилии. Поиск производится с использованием запроса с условием поиска по префиксам с сортировкой по id:
where first_name like ‘?%’ and second_name like ‘?%’ order by id



Проведение нагрузочного тестирования

При проведении нагрузочного тестирования используется поиск анкет по первым 3 символам имени и фамилии пользователей, добавленных в таблицу при подготовке тестовых данных.

Для НТ использовался Gatling.

Пример запроса:

```
select id, first_name, second_name, birth_date, biography, city
from test.users
where first_name like 'Дмит%’ and second_name like 'Афан%’
order by id;
```

Проведение нагрузочного тестирования (1/6)

Проведено нагрузочное тестирование метода со следующими показателями:

- До добавления дополнительных индексов
- Количество одновременных пользователей – 1
- Длительность – 100 секунд.

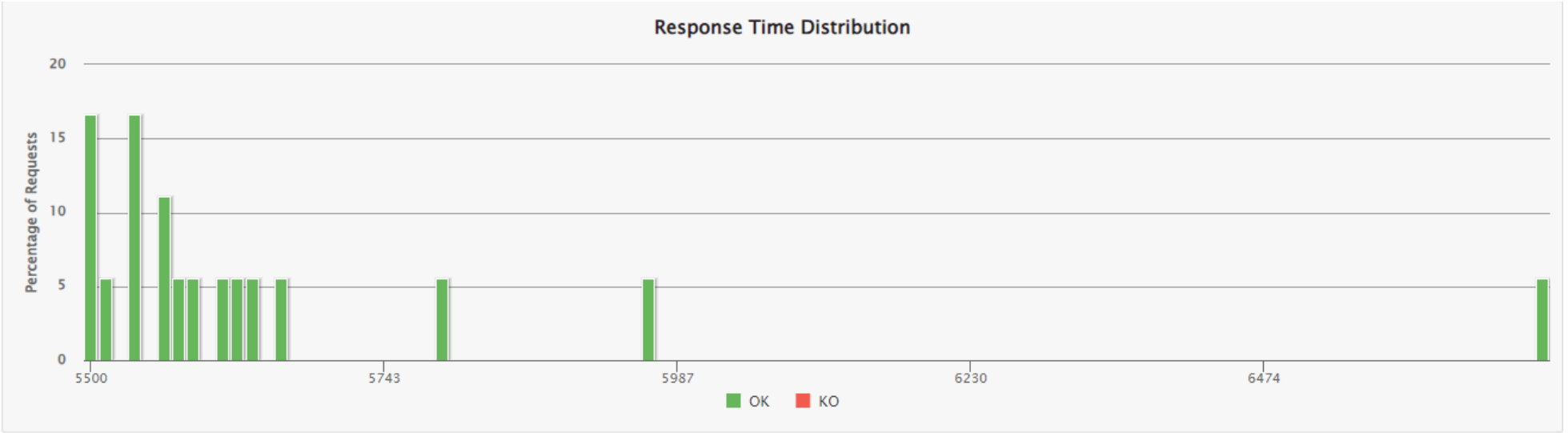
Stats

Executions

	Total	OK	KO
Total count	18	18	0
Mean count/s	0.175	0.175	-

Response Time (ms)

	Total	OK	KO
Min	5494	5494	-
50th percentile	5570	5570	-
75th percentile	5631	5631	-
95th percentile	6075	6075	-
99th percentile	6584	6584	-
Max	6711	6711	-
Mean	5660	5660	-
Standard Deviation	279	279	-



Результаты:

Все запросы выполнены успешно.

Пропускная способность: 0,175 RPS

Время ответа: 6075 мс. (95 перцентиль).

Проведение нагрузочного тестирования (2/6)

Проведено нагрузочное тестирование метода со следующими показателями:

- До добавления дополнительных индексов
- Количество одновременных пользователей – 10
- Длительность – 100 секунд.

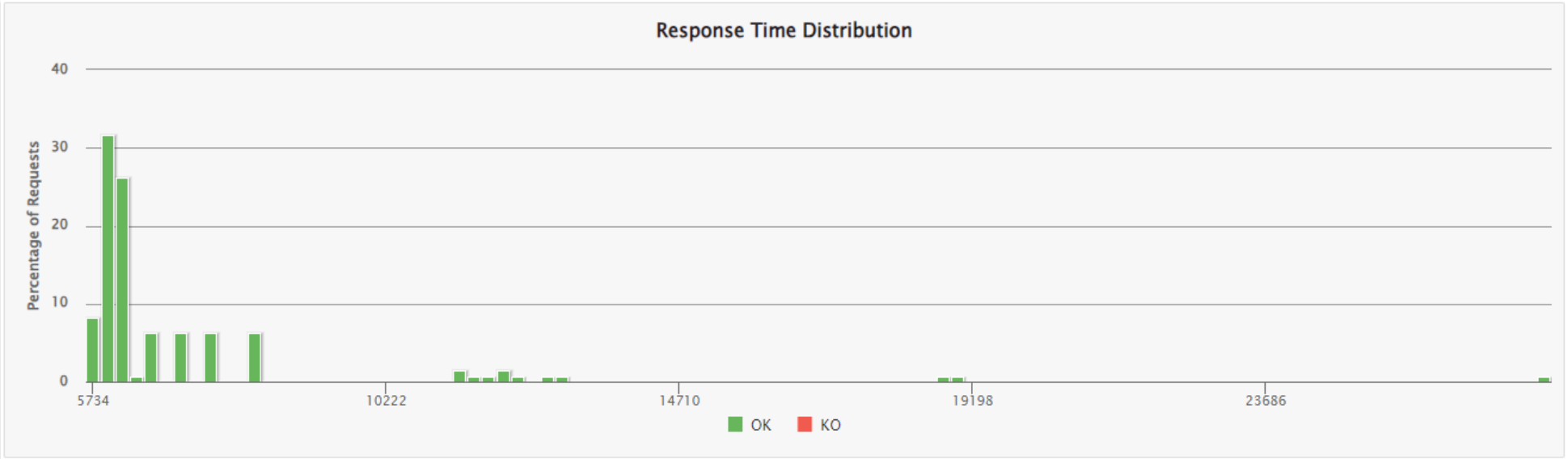
Stats

Executions

	Total	OK	KO
Total count	145	145	0
Mean count/s	1.343	1.343	-

Response Time (ms)

	Total	OK	KO
Min	5622	5622	-
50th percentile	6201	6201	-
75th percentile	7039	7039	-
95th percentile	11908	11908	-
99th percentile	18887	18887	-
Max	28062	28062	-
Mean	7053	7053	-
Standard Deviation	2704	2704	-



Результаты:

Все запросы выполнены успешно

Пропускная способность: 1.343 RPS

Время ответа: 11908 мс. (95 перцентиль).

Проведение нагрузочного тестирования (3/6)

Проведено нагрузочное тестирование метода со следующими показателями:

- До добавления дополнительных индексов
- Количество одновременных пользователей – 100
- Длительность – 100 секунд.

Stats

Executions

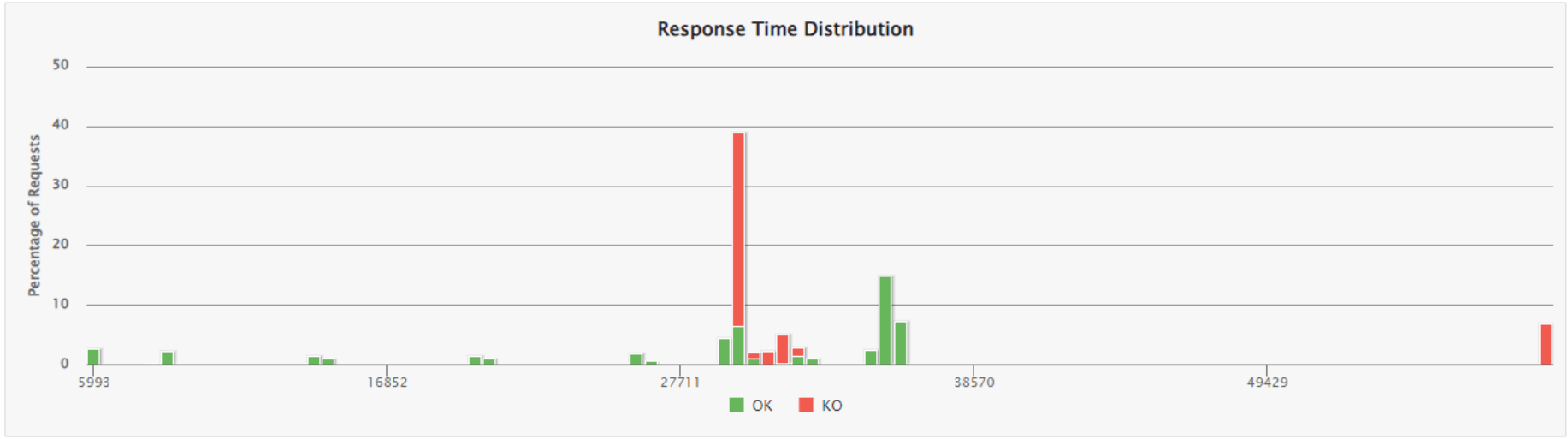
	Total	OK	KO
Total count	378	193	185
Mean count/s	2.589	1.322	1.267

Response Time (ms)

	Total	OK	KO
Min	5722	5722	30009
50th percentile	30043	32320	30035
75th percentile	35202	35369	31531
95th percentile	60003	35707	60011
99th percentile	60015	36008	60016
Max	60016	36017	60016
Mean	31704	29020	34503
Standard Deviation	10053	8997	10329

Errors

Error	Count	Percentage
status.find.in([200, 209], 304), found 500	159	85.946 %
Request timeout to localhost/127.0.0.1:8080 after 60000 ms	26	14.054 %



Результаты:

Почти половина запросов выполнена с ошибками

Пропускная способность: 2.589 RPS (по всем) и 1.322 RPS (по успешным)

Время ответа: 35707 мс. (95 перцентиль).

Добавление индекса

В запросе поиск производится по префиксам двух полей - first_name и second_name и сортировкой по id по возрастанию.

Для ускорения поиска добавляется составной индекс на два поля типа BTREE

create index users_first_name_second_name_IDX using btree on test.users (first_name,second_name);

Такой индекс подходит для поиска по префиксам (like abc%), причем порядок полей в индексе и запросе важно соблюдать.

Добавление в индекс поля id не даст преимуществ при сортировке, так как сортировка производится для всего полученного массива записей, а не внутри групп first_name + second_name

Проведение нагрузочного тестирования (4/6)

Проведено нагрузочное тестирование метода со следующими показателями:

- После добавления дополнительных индексов
- Количество одновременных пользователей – 1
- Длительность – 100 секунд.

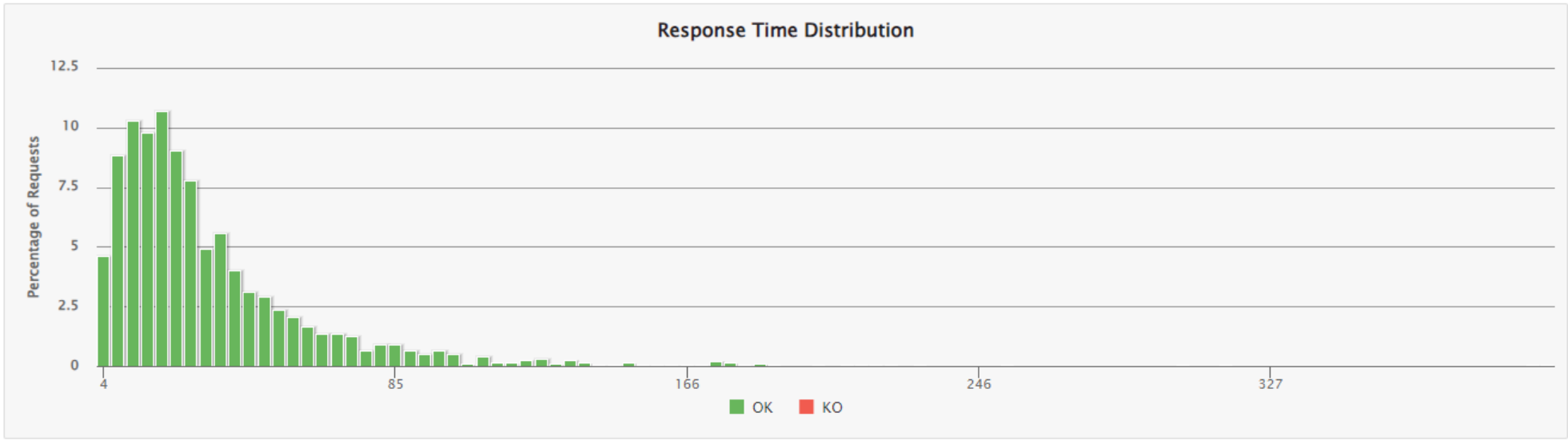
Stats

Executions

	Total	OK	KO
Total count	2847	2847	0
Mean count/s	28.188	28.188	-

Response Time (ms)

	Total	OK	KO
Min	2	2	-
50th percentile	25	25	-
75th percentile	42	42	-
95th percentile	91	91	-
99th percentile	171	171	-
Max	406	406	-
Mean	34	34	-
Standard Deviation	32	32	-



Результаты:

Все запросы выполнены успешно.

Пропускная способность: 28.188 RPS (было 0,175 RPS до добавления индекса)

Время ответа: 91 мс. (было 6075 мс. до добавления индекса) (95 перцентиль).

Проведение нагрузочного тестирования (5/6)

Проведено нагрузочное тестирование метода со следующими показателями:

- После добавления дополнительных индексов
- Количество одновременных пользователей – 10
- Длительность – 100 секунд.

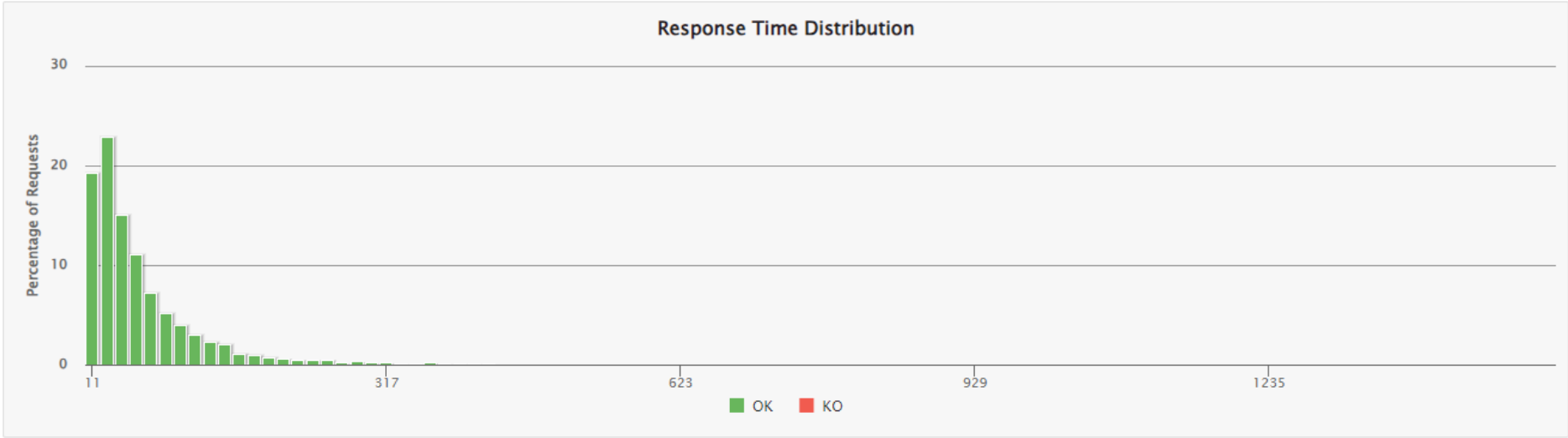
Stats

Executions

	Total	OK	KO
Total count	14944	14944	0
Mean count/s	147.96	147.96	-

Response Time (ms)

	Total	OK	KO
Min	3	3	-
50th percentile	41	41	-
75th percentile	78	78	-
95th percentile	202	202	-
99th percentile	388	388	-
Max	1533	1533	-
Mean	65	65	-
Standard Deviation	80	80	-



Результаты:

Все запросы выполнены успешно.

Пропускная способность: 147.96 RPS (было 1.343 RPS до добавления индекса)

Время ответа: 202 мс. (было 11908 мс. до добавления индекса) (95 перцентиль).

Проведение нагрузочного тестирования (6/6)

Проведено нагрузочное тестирование метода со следующими показателями:

- После добавления дополнительных индексов
- Количество одновременных пользователей – 100
- Длительность – 100 секунд.

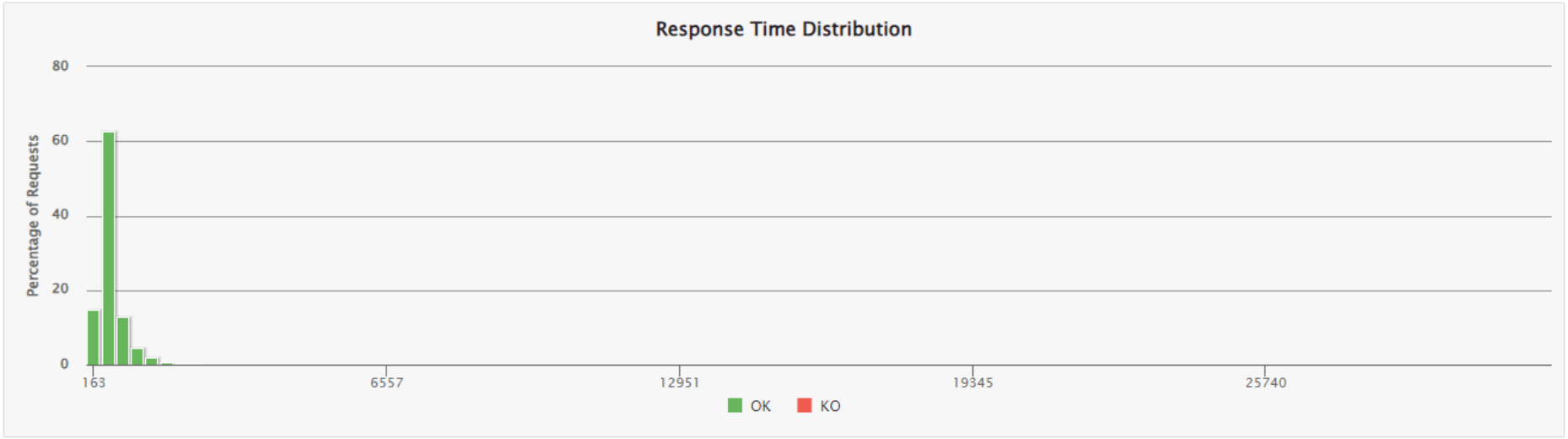
Stats

Executions

	Total	OK	KO
Total count	15255	15255	0
Mean count/s	151.04	151.04	-

Response Time (ms)

	Total	OK	KO
Min	3	3	-
50th percentile	482	482	-
75th percentile	607	607	-
95th percentile	1302	1302	-
99th percentile	5159	5159	-
Max	31974	31974	-
Mean	652	652	-
Standard Deviation	1187	1187	-



Результаты:

Все запросы выполнены успешно (до добавления индекса было выполнено успешно около половины запросов)

Пропускная способность: 151.04 RPS (было 1.322 RPS по успешным до добавления индекса)

Время ответа: 1302 мс. (было 35707 мс. до добавления индекса) (95 перцентиль).

Explain запроса до и после добавления индекса

Результаты выполнения запроса MySQL

explain analyze select id, first_name, second_name, birth_date, biography, city from test.users where first_name like 'Дмит%' and second_name like 'Афан%'order by id;

До добавления индекса по двум полям (Full scan):

```
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Sort: test.users.id  (cost=115744 rows=979639) (actual time=3243..3243 rows=60 loops=1)
    -> Filter: ((test.users.first_name like 'Дмит%') and (test.users.second_name like 'Афан%'))  (cost=115744 rows=979639) (actual time=41.2..3243 rows=60 loops=1)
        -> Table scan on users  (cost=115744 rows=979639) (actual time=0.352..3059 rows=1e+6 loops=1)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (3.24 sec)
```

После добавления индекса по двум полям (поиск с использованием индекса для условия where):

```
| -> Sort: test.users.id  (cost=39502 rows=37830) (actual time=10.8..10.8 rows=60 loops=1)
    -> Index range scan on users using users_first_name_second_name_IDX over ('Дмит' <= first_name <= 'Дмит???????????????????????????????????????????????????????????????????????????????????????????????????
????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
????????????????????????????????????????????????????????????????????????????????????????' AND 'Афан' <= second_name <= 'Афан????????????????????????????????????????????????????????????????????????????????????
????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
???????????????????????????????????????????????????????????????????????????????????????????????????????'), with index condition: ((test.users.first_name like 'Дмит%') and (test.users.second_name like 'Афан%'
))  (cost=39502 rows=37830) (actual time=0.593..10.8 rows=60 loops=1)
 |
```

Итоги

Разумное добавление индексов позволяет ускорить операции поиска.

При выборе типа индекса и полей, добавляемых в индекс, их порядка, следует ориентироваться на запросы, которые необходимо ускорить.

Большое количество индексов замедляет вставку, требует дополнительного места для хранения.