Оценка результатов нагрузочного тестирования и отчет

Без индексов:

- **1 запрос**: Приемлемая задержка (166 мс), но крайне низкая пропускная способность (3.7 RPS)
- **10 запросов**: Задержка вырастает в 4 раза (685 мс), RPS лишь 6.9
- **100 запросов**: Катастрофическое падение производительности - задержка >13 сек, 10% отказов
- **1000 запросов**: Система не справляется - задержка ~4 мин, 81% отказов

С индексами:

- **1 запрос**: Задержка снижена в **55 раз** (3 мс), RPS вырос в **75 раз** (279 RPS)
- **1000 запросов**: Задержка 151 мс (в **1700 раз** лучше), RPS 924 (в **280 раз** выше), 0% отказов
- **Линейное масштабирование**: При росте нагрузки с 1 до 1000 запросов задержка увеличилась лишь с 3 до 151 мс

Полный отчет

Исследовано влияние индексов на производительность API поиска пользователей. База данных: PostgreSQL с >1 млн записей.

Методология

Уровни нагрузки: 1, 10, 100, 1000 одновременных запросов

Метрики:

- ✓ Latency (средняя, p90, p95, p99)
- ✓ Throughput (запросов/сек)
- ✓ Процент отказов

Индексы:

```
CREATE EXTENSION IF NOT EXISTS pg_trgm;
CREATE INDEX idx_users_firstname_trgm ON users USING gin ("FirstName" gin_trgm_ops);
CREATE INDEX idx_users_lastname_trgm ON users USING gin ("LastName" gin_trgm_ops);
```

Результаты до индексов

Параллелизм	Latency (avg)	p90	p95	p99	Throughput	Отказы
1	166 мс	183	202	211	3.7 RPS	0%
10	685 мс	1020	1074	1196	6.9 RPS	0%
100	13446 мс	13894	16069	18568	4.9 RPS	10%
1000	256830 мс	284094	284379	291418	3.3 RPS	81%

Результаты с триграммными индексами

Параллелизм	Latency (avg)	p90	p95	p99	Throughput	Отказы
1	3 мс	4	4	5	279.3 RPS	0%
10	5 мс	8	9	13	848.4 RPS	0%
100	19 мс	29	31	35	864.3 RPS	0%
1000	151 мс	246	270	302	924.2 RPS	0%

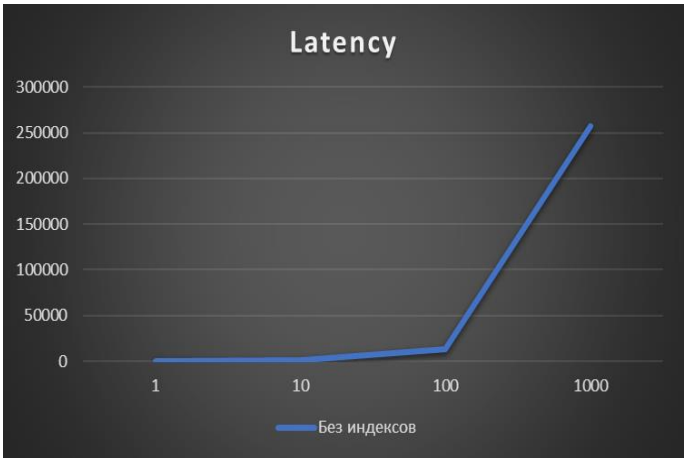
Сравнительный анализ

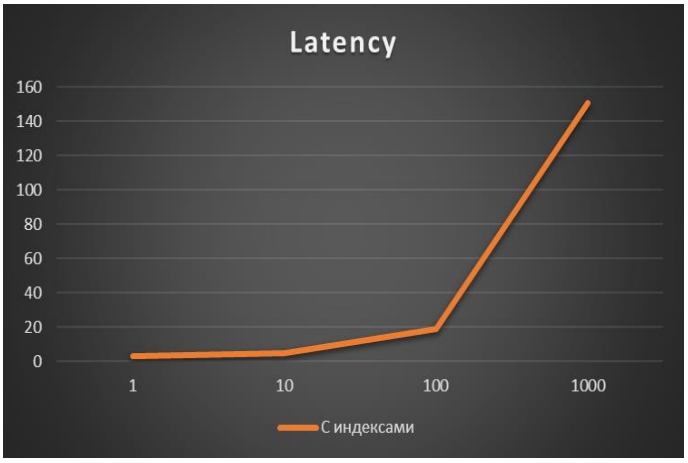
Метрика	**Улучшение (1000 запросов)**
Средняя задержка	В *1700 раз* (256 сек → 0.15 сек)
Пропускная способность	В *280 раз* (3.3 → 924 RPS)
Отказы	81% → 0%

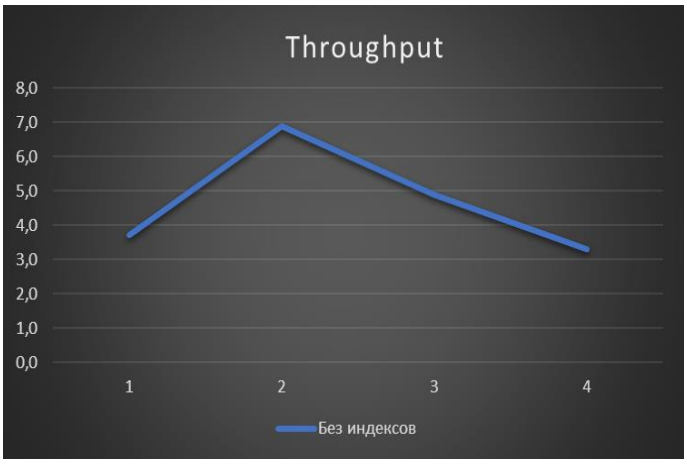
Тенденции**:**

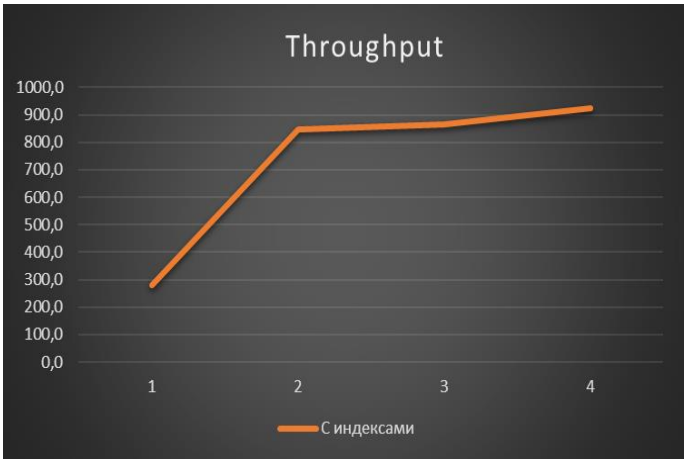
- Индекс устранил "узкое горлышко" при высокой нагрузке
- Система демонстрирует линейное масштабирование
- p99 задержка осталась предсказуемой (<302 мс)

Графики:









Обоснование выбора индекса

Преимущества триграммных индексов:

1. Поддержка сложных операторов:
   - LIKE/ILIKE с подстановкой в начале/середине (%value%)
   - Регулярные выражения (~, ~*)
   - Функции сходства (SIMILARITY(), word_similarity)
2. Эффективность для частичных совпадений - позволяет находить совпадения в любой позиции строки
3. Гибкость условий, работает для комбинаций:
   - Только FirstName
   - Только LastName
   - FirstName AND LastName
   - FirstName OR LastName

EXPLAIN анализ

Запрос:

```
EXPLAIN ANALYZE SELECT * FROM users
WHERE "FirstName" ILIKE '%Antloy%'
AND "LastName" ~ 'Abac';
```

Результат:

```
Bitmap Heap Scan on users  (cost=81.73..85.74 rows=1 width=85) (actual time=0.177..0.178 rows=0 loops=1)
  Recheck Cond: ((("LastName")::text ~ ''Abac''::text) AND (("FirstName")::text ~~* ''%Antloy%''::text))
  ->  BitmapAnd  (cost=81.73..81.73 rows=1 width=0) (actual time=0.176..0.177 rows=0 loops=1)
        ->  Bitmap Index Scan on idx_users_lastname_trgm  (cost=0.00..28.74 rows=99 width=0) (actual
time=0.096..0.096 rows=19 loops=1)
              Index Cond: (("LastName")::text ~ ''Abac''::text)
        ->  Bitmap Index Scan on idx_users_firstname_trgm  (cost=0.00..52.74 rows=98 width=0) (actual
time=0.080..0.080 rows=0 loops=1)
              Index Cond: (("FirstName")::text ~~* ''%Antloy%''::text)
Planning Time: 0.422 ms
Execution Time: 0.215 ms
```

Выводы

Приложение показывало катастрофическое падение без индексов при >100 запросах

Триграммные индексы решили ключевые проблемы:

- Уменьшили задержку в 1700 раз
- Увеличили пропускную способность в 280 раз
- Полностью устранили отказы

Итоговый эффект: система перешла из состояния "неработоспособна при высокой нагрузке" к стабильной обработке **1000+ RPS** с предсказуемой задержкой <300 мс.