УНИВЕРСИТЕТ ИТМО

Автоматизация индексирования базы данных на основе истории запросов

Кульбако Артемий Юрьевич, P34112

Образовательная программа для поступления:

09.04.04 Программная инженерия: системное и прикладное ПО

Исходный код проекта:

https://github.com/testpassword/Database-auto-indexing-based-on-query-history

Проблема

Старые базы данных могут содержать очень сложные и непонятные запросы, скорость выполнения которых необходимо увеличить. Ручная оптимизации займёт много времени и напрямую зависит от опыта работы сотрудника с данной даталогической моделью.

Пример запроса к БД из рабочего проекта на 200+ строк, который тут даже не разглядеть.

Задачи и цели

УНИВЕРСИТЕТ ИТМО

Цели:

- Минимизировать время на оптимизацию БД пользователем
- Максимально увеличить скорость выполнения запросов к БД

Задачи:

- Проектирование и создание тестовой базы данных
- СОЗДАНИЕ ПРИЛОЖЕНИЯ АВТОИНДЕКСИРОВАНИЯ БАЗЫ ДАННЫХ
- Тестирование приложения
- Анализ полученных результатов

Обоснование оригинальности

УНИВЕРСИТЕТ ИТМО

8 из 10 самых популярных* СУБД подобной функциональности не имеют. Только Oracle Database и Microsoft SQLServer в составе подписки Azure имеют функцию автоиндексирования, но это дорогие коммерческие продукты, а сам процесс представляет из себя «черный ящик».

Нельзя официально приобрести в России.

Преимущества моего решения:

- Поддержка любой БД с JDBC-драйвером при его подключении
- Контролируемые процесс (формирование отчётов)
- Открытый исходный код

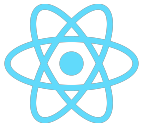
*по версии db-engines.com

Используемые технологии*

УНИВЕРСИТЕТ ИТМО

Клиент:

язык программирования
JavaScript



построение интерфейса
React

библиотека компонентов
Ant Design

отображение SQL-выражений
React CodeMirror

Сервер:

язык программирования
Kotlin

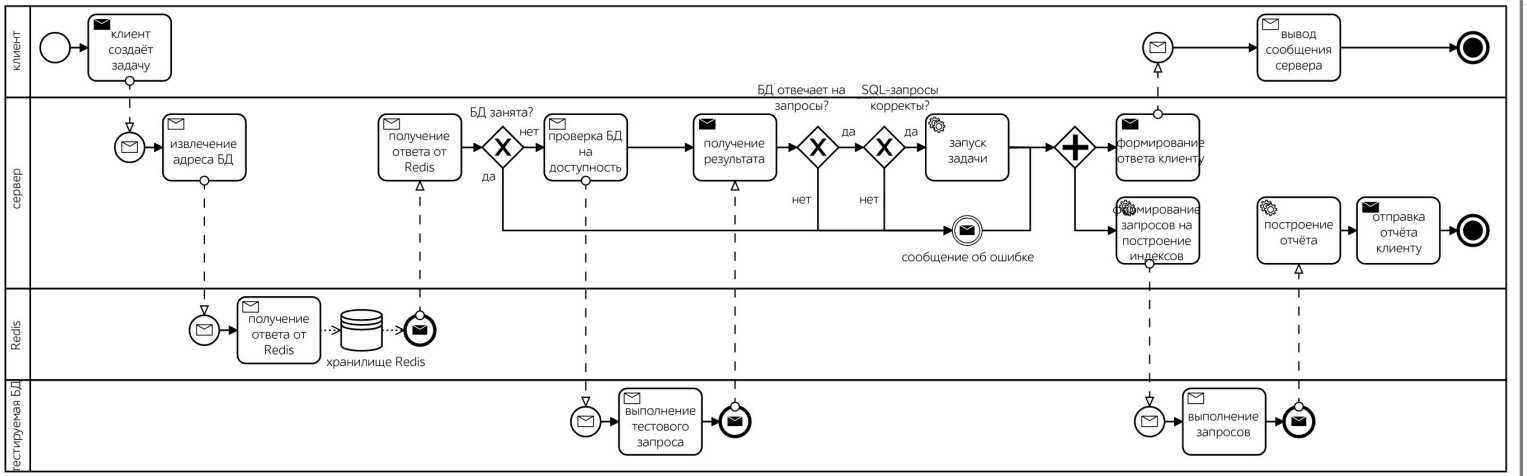
обработка HTTP-запросов
Ktor

взаимодействие с СУБД
JDBC

распределённый кэш
Redis

*подробное обоснование выбора описано в README.md репозитория

Описание проекта



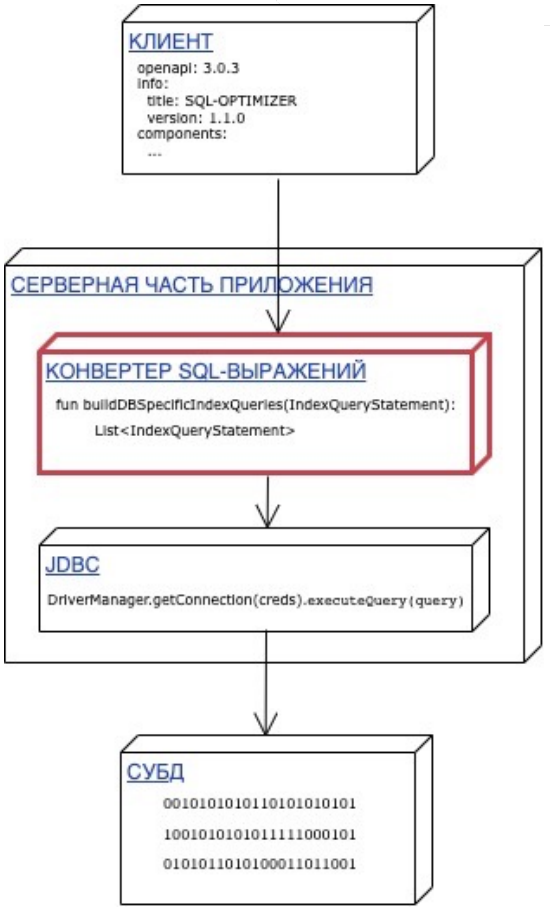
1. Пользователь формирует историю запросов SQL и запускает через клиент задачу на автоиндексирование этой истории.
2. Сервер подключение к БД пользователя, выбирает все уникальные запросы из истории и строит все возможные комбинации индексов.
3. Система замеряет время выполнения запроса с каждым из индексов, выбирает, и, по желанию пользователя, сохраняет наилучший.
4. После выполнения процесса автоиндексирования, пользователь получает подробный отчёт о работе программы.

Архитектура системы построена таким образом, что за трансформацию выражений из ANSI SQL в специфичный для СУБД SQL отвечает модуль конвертации.

Для добавления поддержки новой СУБД необходимо подключить соответствующий JDBC-драйвер и переопределить ОДИН метод:

```
private fun formIndexesQueries():
    List<IndexQueryStatement>
```

Система тестировалась для PostgreSQL и Microsoft SQLServer.



Оцениваемые результаты

УНИВЕРСИТЕТ ИТМО

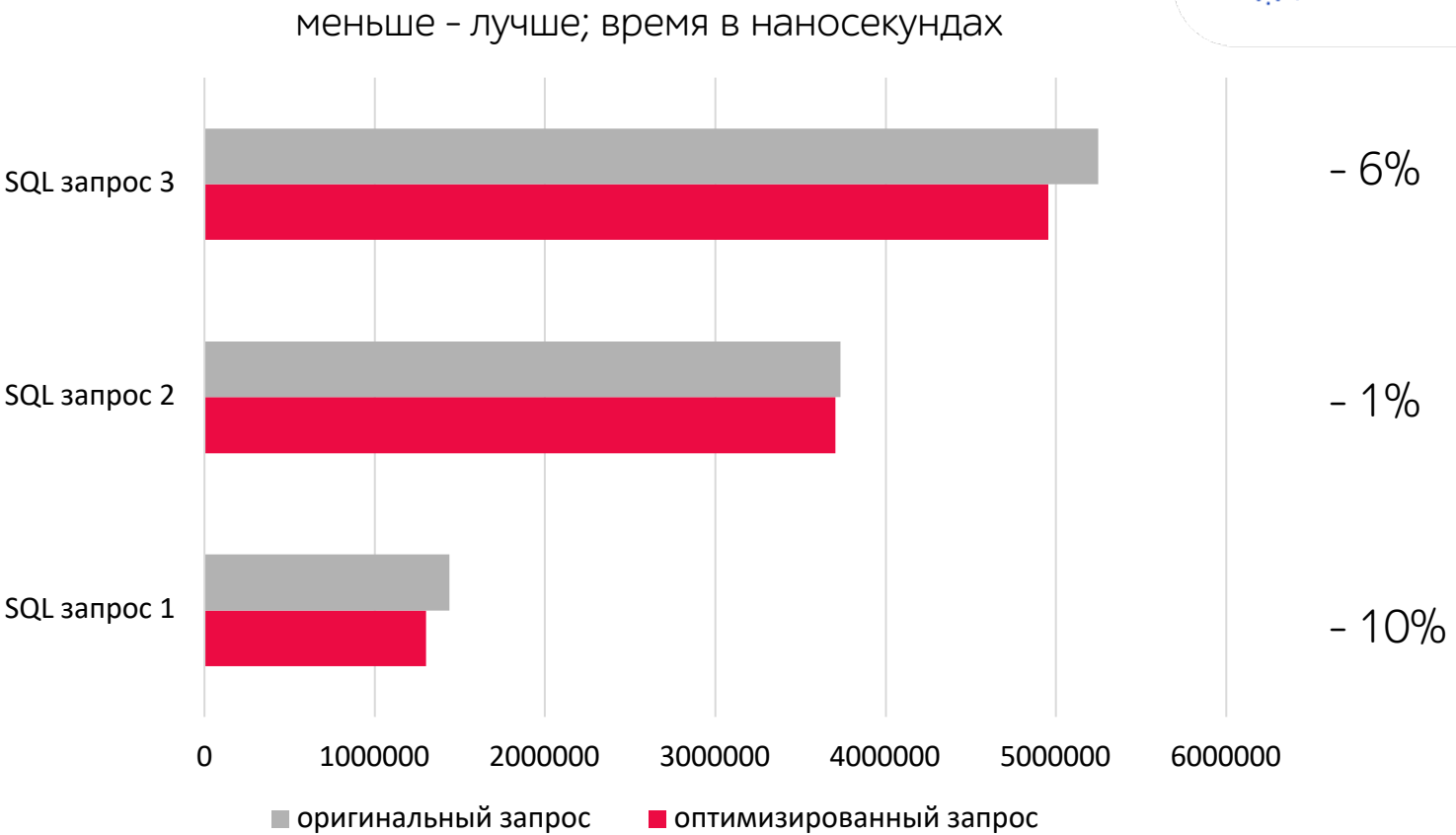
Запрос:

```
SELECT emp_id FROM employee JOIN position USING (pos_id) JOIN missions_emp USING
     (emp_id) JOIN mission USING (miss_id)
WHERE rank !~~ ''
ORDER BY is_married DESC, end_date_and_time DESC, hiring_date DESC
LIMIT 20;
```

Отрывок отчёта*:

	indexStatement	diff	timeTaken
1	CREATE INDEX Is_marriedEmployeeHASH ON employee USING HASH(is_married);	173166	4956042
2	CREATE INDEX Is_marriedHiring_dateEmployeeBTREE ON employee USING BTREE(is_married, hiring_date);	123791	5005417
3	CREATE INDEX End_date_and_timeMissionBTREE ON mission USING BTREE(end_date_and_time);	78250	5050958
4	CREATE INDEX Emp_idIs_marriedHiring_dateEmployeeBTREE ON employee USING BTREE(emp_id, is_married, hiring_date);	71375	5057833
5	CREATE INDEX RankPositionHASH ON position USING HASH(rank);	45833	5083375

*несколько полных отчётов можно найти в директории product/docs/reports репозитория


меньше - лучше; время в наносекундах
SQL запрос 3
SQL запрос 2
SQL запрос 1
- 6%
- 1%
- 10%
0
1000000
2000000
3000000
4000000
5000000
6000000
оригинальный запрос
оптимизированный запрос

Соответствие результатов оцениваемым показателям

УНИВЕРСИТЕТ ИТМО

Цель	Результат
Минимизировать время на оптимизацию БД пользователем	Пользователю нужно лишь отдать программе список запросов, которые необходимо оптимизировать, приложение подберёт наилучшие индексы и даже может сохранить их, тем самым полностью избавив его от необходимость оптимизировать БД самостоятельно.
Максимально увеличить скорость выполнения запросов к БД	Приложение полным перебором найдёт индекс, который сделает запрос наиболее быстрым.