Министерство образования Республики Беларусь

Учреждение образования
Белорусский государственный университет
информатики и радиоэлектроники

УДК 004.415

Лимонтов
Александр Сергеевич

Оптимизация приложения для сферы розничных продаж с использованием платформы LogicBlox

АВТОРЕФЕРАТ

на соискание академической степени
магистра информатики и вычислительной техники

по специальности 1-40 81 04 - Обработка больших объемов информации

Научный руководитель
Анисимов В.Я.
кандидат физико-математических наук,
доцент

Минск 2019

КРАТКОЕ ВВЕДЕНИЕ

Розничная торговля, или ритейл (от англ. retail – "розничный "в розницу") – продажа товаров или услуг небольшим количеством, поштучно. Осуществляется через предприятия розничной торговли. Объектом розничной торговли является покупатель, приобретающий товар, предназначенный для личного, семейного, домашнего или иного пользования, не связанного с предпринимательской деятельностью. Субъектом розничной торговли является продавец.

В нынешнее время продавец заинтересован в удовлетворении потребительских потребностей. Исходя из быстрого и занятого стиля жизни теперешних продавцов, они также предлагают некоторые услуги, в отличие от самих продуктов. Розничная торговля охватывает важные места в экономике страны. Это финальная стадия распределения продуктов либо услуг. Она не только повышает ВВП государства, но также и позволяет большему количеству людей приобрести работу.

Любая организация, которая продает товары для пользования в личных, семейных, бытовых целях потребителей завязана на сфере розничной торговли. При этом редко встречаются такие организации, которые ограничиваются продажей лишь одного продукта. Как правило, выход на рынок связан с целым ассортиментом товаров, в целом имеющие схожие параметры, но принадлежащие к разным категориям жизни деятельности.

В повседневной жизни крупным организациям становится сложнее следить за движением их продуктов. Им постоянно приходится отвечать на следующие вопросы:

- где найти товар;
- по какой цене купить товар;
- сколько товара купить;
- куда его доставить;
- как его доставить;
- по какой цене продать товар.

Вместе с высоким темпом роста объемов такой информации появилась необходимость создавать специализированные системы, которые будут не только учитывать все изменения в этих цепочках, но также и автоматически предпринимать действия (отправлять запросы на закупку, генерировать отчеты по продажам). Кроме того, возрастает необходимость анализа совершенных действий. В частности, требуется следить за ходом продаж, чтобы

научиться предсказывать их в ближайшем будущем. Также полученная система должна быть достаточно распределенной, для того чтобы управлять ею из разных точек. Дополнительным требованием будет и время обработки результатов – подсчет продаж, агрегация значений.

Чтобы глубже понять саму проблему и перенести ее на математический язык, стоит построить упрощенные модели. Типичными моделями здесь будут являться:

– финансовые модели сколько продавец хочет выручить в ближайшее время при определенных условиях;

– модели данных – это могут быть скидки, распродажи, акции;

– модели сети цепей поставок – процесс доставки до конечного пользователя.

Но, даже построив приведенные модели, сложно понять, как именно проходят процессы в этой среде. Многие продавцы создают информативные таблицы. Головную боль здесь доставить могут не столько большие таблицы, сколько много маленьких, у которых зачастую еще и разный формат содержания. Количество таких таблиц может достигать нескольких тысяч. Отсюда появляется сложность перевода этих данных в необходимые математические модели. Самые ценные модели – такие модели, которые могут быть построены и поддерживаемы экспертами, при этом их можно без особых трудностей изменять во времени.

Стоит сфокусироваться на тех моделях, которые зачастую используют продавцы. Пожалуй, самая простая, какую можно представить, будет следующая формула `Profit = revenue - cost`.

Основная задачи этой модели максимизация функции *profit*, которая зависит от двух параметров: *revenue* (выручка) и *cost* (затраты).

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Цель и задачи исследования

Целью диссертационной работы является оптимизация программного продукта на платформе LogicBlox, автоматизирующего операции с данными в сфере розничной торговли, для увеличения производительности обработки сложных запросов и быстродействия при пересчете планов.

Объектом исследования является готовый программный продукт, используемый крупными торговыми компаниями.

Предметом исследования является исходный код продукта, платформа и база данных LogicBlox.

Личный вклад соискателя

Результаты, приведенные в диссертации, получены соискателем лично при изучении и анализе работы платформы LogicBlox.

Опубликованность результатов диссертации

Опубликовано 2 тезиса в сборнике трудов и материалов конференций.

Структура и объем диссертации

Диссертация состоит из введения, общей характеристики работы, трех глав, заключения, списка использованных источников, списка публикаций автора. В первой главе произведен обзор предметной области - какие требования ставятся при построении приложений для розничной торговли, как разрабатываются такие приложения в общих случаях и какие частные практики используются при построении на платформе LogicBlox. Вторая глава посвящена используемым технологиям, в частности дается разбор составляющих платформы, используемого языка запросов, веб-сервисов. Третья глава содержит непосредственно детальный разбор проблем при работе с базой данных и их решений. В заключении подводится краткий итог всех примененных оптимизаций.

Общий объем работы составляет 73 с., 21 рис., 1 табл., 1 формула, 31 источник.

ОСНОВНОЕ СОДЕРЖАНИЕ

Во **введении** делается краткий обзор на область розничной торговли: что из себя представляет, чем занимается, какие основные задачи решает, что можно автоматизировать, какого рода приложения нужны.

Первая глава производит обзор предметной области, а именно уделяется внимание конкретно построению приложений с большими объемами информации. Исходя из самой области применения, бизнес-требования интерпретируются в технические требования, позволяя сформулировать базовые концепты всей системы. Также в этой главе приводится архитектурная модель (схема) стандартных приложений, а также выделяются ее недостатки. В частности, делается вывод о том, что такое приложение быстро превращается в *"клубок"*, который сложно поддерживать и расширять. Вместе с этим дается описание платформы LogicBlox, которую называют еще *smart database* за то, что она в себе содержит все необходимые фукнции, благодаря чему расширяемость приложения не становится больным местом. Среди основных ее идей - сочетание в себе OLAP и OLTP технологий, декларативный язык, единая модель модель для бизнес-логики и обработки данных, а также вычислительсные оптимизации, которые хорошо ложатся на данную область. Кроме того, приводится небольшая информация о самой компании LogicBlox, ее структуре, составе, достижениям и тп.

Вторая глава посвящена используемым технологиям. Первая ее часть более подробно раскрывает детали из предыдущей главы, в частности показывается схема самой платформы, делается сравнение OLTP и OLAP подходов: когда они применяются, как устроены, какими характеристиками или метриками обладают. Архитектура же базы данных LogicBlox выбрана именно как *общая* (вместо *специализированной*), несмотря на показательные преимущества последней. Успех такого выбора доказывается на примере смартфона, который содержит в себе сторонние функции, кроме мобильной связи.

Далее делается проверка LogicBlox БД на соответствие принципам ACID и как они там выражены. Следующий пункт поввящен языку LogiQL-центральному компоненту платформы. LogiQL является экспрессивным декларативным языком, то есть его легко читать и понимать, а порядок следования инструкций в исходном коде не означает такого же порядка их выполнения, что дарит такую скрытую возможность, как *автоматический параллелизм* выполнения запросов. Также в нем нет пустых значений (), исполь-

зуется 6-я нормальная форма, простота в изменении схемы БД. Одной из ключевой особенностью является и подход к выполнению операции JOIN: в случае с попыткой сделать объединение по нескольким предикатам (или таблицам в SQL) платформа LogicBlox пройдется сразу по многим связям, вместо попарных, тем самым позволяя обрабатывать меньший объем данных засчет "отсеивания"по нескольким параметрам сразу. Это достигается с помощью алгоритма *Leapfrom trie-join*, который сводит асимптотику обхода данных в требуемом классе до $O(n \log n)$.

Продолжается описание платформы определением предикатов и этапов выполнения транзакции и как можно с помощью постфиксов `@prev`, `@initial` и `@final` обратиться к данным на этих этапах. Еще приводится немного примеров синтаксиса языка LogiQL для агрегаций, правил, ограничений (и как ограничения помогают решать оптимизационные задачи), триггеры и события (они устроены немного по-другому, чем в обычном представлении SQL), запросы и обновления (*queries* и *spreads*). После этого делается переход к самому процессу оптимизации приложений, для чего рассматривается термин *Move computation to data, not vice versa* - дается его толкование, как применяется в платформе и какие выгоды из этого можно получить (к примеру, как на самом деле устроен *автоматический параллелизм*, упомянутый ранее). В завершении дается краткое описание пакетного менеджера Hydra и встроенных веб-сервисов (поскольку они попадают под оптимизацию приложения).

Третья глава рассказывает о цели всей работы - об оптимизации приложения. Из приведенных пояснений о работе платформы делаются заключения по конкретным направлениям, например, исходя из представления данных на диске, операций случайного добавления (размещение по страницам и итерирование по ним по разным ключам). В случае с операцией JOIN детально разбирается случай с поведением алгоритма в зависимости от порядка ключей, что делает это самым простым, но одним из лучших методов оптимизации. Рассматриваются создание индексов (они создаются автоматически и как при этом расходуются ресурсы по времени и памяти), приводятся рекомендации к пониманию правильных и неправильных (которые лишь усугубят метрики) методов оптимизации. С выбором правильного порядка ключей в предикатах тесно переплетается и другой метод оптимизации - переформулировка правил. Он заключается в том, чтобы содержать правила предельно маленькими по реализации, и разбивать сложные на несколько мелких, логически верных и понятных. Интуиция такого

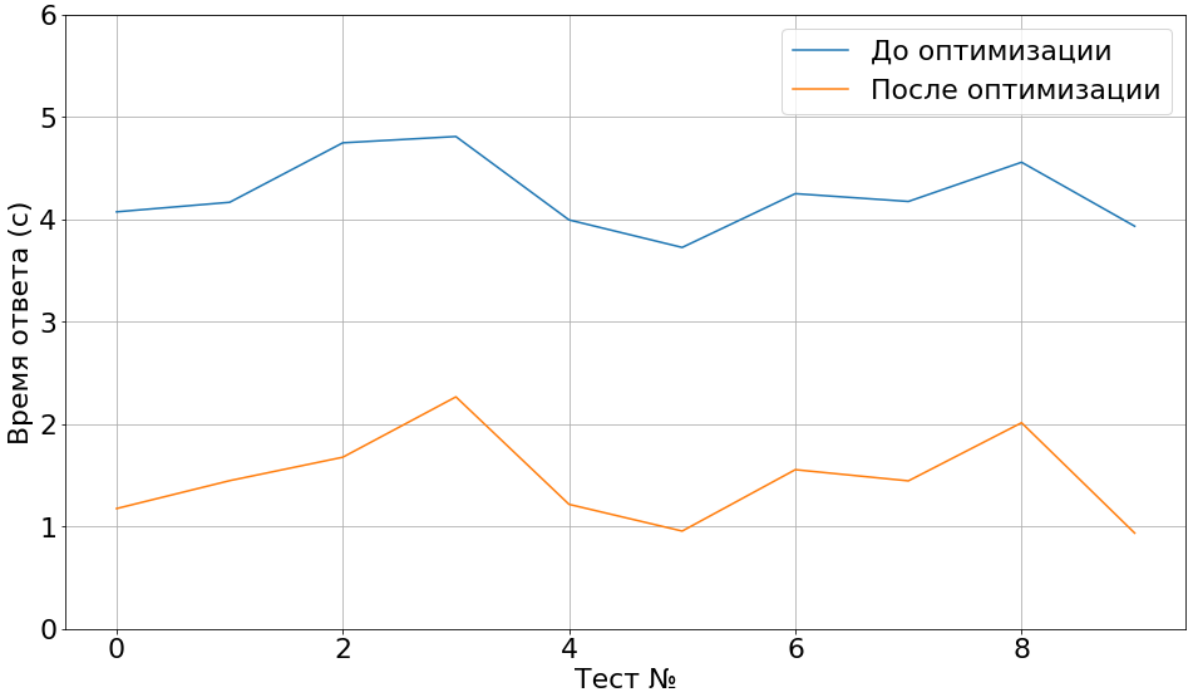

Рисунок 1 – Сравнительный график времени выполнения запроса `resultByProduct` до оптимизации и после

подхода заключается в том, что делать JOIN по многим ключам дорого, если при этом нужно обойти несколько придикатов (да еще и, возможно, с разными порядком ключей), поэтому такое "дробление"действительно хорошо сказывается на результатах.

В главе **Заключение** подытоживаются накопленные советы по оптимизации и приводятся простым перечислением. Также в качестве примера приводятся сравнительные графики работы классических типов запросов в приложении до оптимизации и после. Из них видно, что оптимизация почти не дает выигрыша на мелких запросах, но сравнимо улучшает производительность на сложных запросах и обновлениях. К примеру, приведены два графика с улучшением результатов и без.

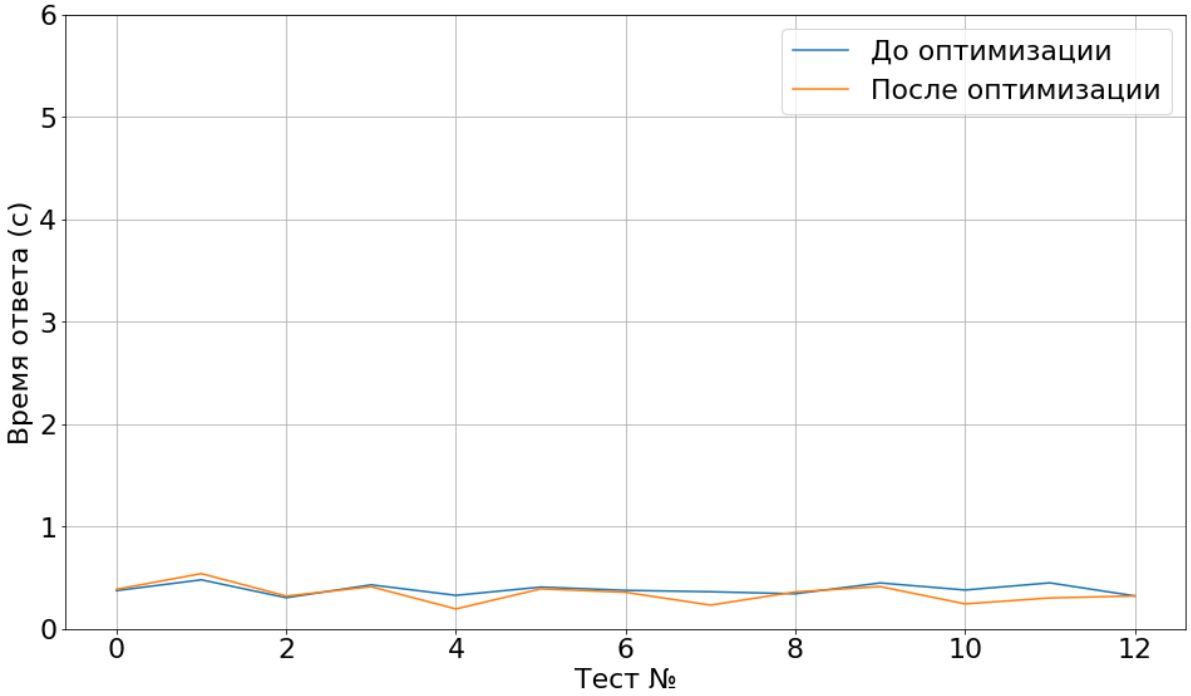

Рисунок 2 – Сравнительный график времени выполнения запроса упрощенного `resultByProduct` для одного товара до оптимизации и после

СПИСОК ОПУБЛИКОВАННЫХ РАБОТ

1-А. Тишковский М.А., Лимонтов А.С., Евжик Д.А. Эффективное встроенное ансамблирование нейронных сетей / Тишковский М.А., Лимонтов А.С., Евжик Д.А. // 55-я юбилейная научная конференция аспирантов, магистрантов и студентов. – 2019. – с. 220-221.

2-А. Тишковский М.А., Лимонтов А.С., Подвальников Д.С. Оценка алгоритмов машинного обучения. Дилемма смещения-дисперсии / Тишковский М.А., Лимонтов А.С., Подвальников Д.С. // 55-я юбилейная научная конференция аспирантов, магистрантов и студентов. - 2019. - с. 221-223.