



Московский Государственный Университет им. М.В. Ломоносова

Факультет Вычислительной Математики и Кибернетики

Кафедра Автоматизации Систем Вычислительных Комплексов

Шпилевой Владислав Дмитриевич

**Разработка и реализация алгоритма  
отложенного обновления вторичных индексов  
на LSM-деревьях**

Магистерская диссертация

**Научный руководитель:**

к.ф.-м.н. ассистент

Д.Ю.Волканов

**Научный консультант:**

К.А.Осипов

Москва, 2018

## Аннотация

В настоящее время растет популярность баз данных, хранящих данные на диске не в виде традиционных В-деревьев и их производных, а в виде LSM деревьев. Главное преимущество LSM деревьев в том, что их обновление всегда приводит только к последовательной записи на диск, в отличие от В-деревьев. Это возможно благодаря тому, что LSM дерево способно хранить множество версий одной и той же записи - за счет этого при обновлении дерева не нужно точно читать и удалять старые данные - это происходит позже во время слияния уровней LSM дерева. Это работает, когда в таблице только один индекс - первичный. При наличии вторичных индексов LSM деревья лишаются преимуществ версионности, так как при обновлении дерева нужно явно читать и удалять старые данные из всех вторичных индексов. В настоящей работе представлен обзор существующих способов решения этой проблемы, а также разработанная модификация LSM дерева, которая позволяет не делать явных чтений и удалений старых данных из вторичных индексов. Проведенное экспериментальное исследование нового LSM дерева показало прирост скорости на порядок при наличии нескольких вторичных индексов.

# Содержание

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Введение</b>   | <b>4</b>  |
| 1.1      | Цель работы . . . . .   | 4         |
| 1.2      | Актуальность . . . . .  | 4         |
| 1.3      | Постановка задачи . . . . .                                     | 5         |
| 1.4      | Структура работы . . . . .                                      | 5         |
| <b>2</b> | <b>Обзор существующих способов ускорения записи</b>             | <b>7</b>  |
| 2.1      | Способ 1 . . . . .  | 7         |
| 2.2      | Способ 2 . . . . .  | 7         |
| 2.3      | Способ 3 . . . . .  | 7         |
| <b>3</b> | <b>LSM дерево</b>   | <b>8</b>  |
| 3.1      | Слияния уровней . . . . .                                       | 8         |
| 3.2      | Устройство индекса . . . . .                                    | 8         |
| 3.3      | Проблема обновления индекса . . . . .                           | 8         |
| <b>4</b> | <b>Отложенное обновление вторичных индексов на LSM-деревьях</b> | <b>9</b>  |
| 4.1      | Вставка и удаление . . . . .                                    | 9         |
| 4.2      | Слияние уровней . . . . .                                       | 9         |
| 4.2.1    | Первичный индекс . . . . .                                      | 9         |
| 4.2.2    | Вторичный индекс . . . . .                                      | 9         |
| 4.3      | Чтения . . . . .  | 9         |
| 4.4      | Реализация . . . . .  | 9         |
| <b>5</b> | <b>Экспериментальное исследование</b>                           | <b>10</b> |
| <b>6</b> | <b>Заключение</b>   | <b>11</b> |
|          | <b>Список литературы</b>  | <b>12</b> |

# 1 Введение

## 1.1 Цель работы

Целью работы является увеличение скорости обновления базы данных, которая хранит индексы таблиц в LSM деревьях при помощи модификации процедур обновления и слияния уровней LSM дерева.

## 1.2 Актуальность

LSM (Log-Structured Merge) деревья были разработаны в 1990-х [1] годах для задач с интенсивной записью, и использовались в файловых системах и для резервного копирования. Но их применение в СУБД (Система Управления Базой Данных) было ограничено из-за *скрытых чтений*. Скрытыми называют чтения, которые выполняются СУБД при обновлении данных, чтобы либо найти старые, данные и удалить их, либо чтобы проверить ограничения уникальности при вставке новых данных. Значительная часть чтений в СУБД - скрытая, так как почти любое обновление данных требует проверки различных ограничений и удаления старых данных, и это почти ничего не стоит в В-деревьях, где для обновления данных в любом случае нужно читать [2]. Но на LSM деревьях скрытые чтения значительно снижают производительность, поскольку лишают их преимуществ версионности данных, когда можно сохранять новые данные не читая и не удаляя старые явно.

С появлением SSD (Solid-State Drive) дисков абсолютная скорость любых чтений и записи возросла на порядки по сравнению со старыми механическими дисками, но существенно увеличился разрыв между скоростями последовательной записи и последовательного чтения [3], буквально на порядки. Благодаря тому, что LSM дерево всегда выполняет запись на диск последовательно, а чтения из него на SSD по скорости мало отличаются от В-дерева, LSM дерево стало одной из стандартных структур данных для СУБД. Например, на момент написания работы LSM деревья уже используются в LevelDB [4], RocksDB [5], Cassandra [6], Tarantool [7], BigTable [6], HBase [6], Riak [8], MySQL [9].

Однако SSD хоть и делает LSM дерево более конкурентноспособным, но не решает проблему существования скрытых чтений на любое обновление при наличии вторичных индексов, что не позволяет использовать все возможности LSM деревьев, когда в таблице в БД (База Данных) больше одного индекса.

Когда в таблице только один индекс на LSM дереве, то любые изменения, не требующие знания старых данных (такие как *REPLACE*, *DELETE*), возможны без скрытых чтений. Например, в случае *REPLACE* запись просто попадает в дерево с новой версией. То же самое при *DELETE* - ключ (набор индексируемых колонок и их значений), по которому производится удаление, попадает в дерево с новой версией и пометкой, что это именно удаление, а не вставка. Скрытых чтений не выполняется. Но при появлении

вторичного индекса даже *REPLACE* и *DELETE* вынуждены читать старые данные из первичного индекса, чтобы узнать, какой у старой записи был вторичный ключ, и вставить его *DELETE* в LSM дерево вторичного индекса.

Это обычная процедура для индексов на B-деревьях, где нет версионности данных, и на классических LSM деревьях ее тоже нельзя избежать. Это происходит из-за того, что удаление старых версий данных в LSM дереве работает так, что записи считаются разными версиями одних и тех же данных, только если они равны по ключу, по которому сортируется дерево. И если некоторый запрос меняет этот ключ в уже существующей записи, не читая и не удаляя ее явно, то новая запись становится никак не связанной со старой, и старая не удалится никогда - LSM дерево видит их как разные ключи.

Таким образом, задача борьбы со скрытыми чтениями в таблицах с индексами на LSM деревьях становится актуальной.

### 1.3 Постановка задачи

Разработать и реализовать алгоритм отложенного обновления вторичных индексов на LSM-деревьях на основе базы данных Tarantool. Для достижения заданной цели необходимо решить следующие подзадачи:

1. исследовать существующие способы уменьшения влияния скрытых чтений и ускорения записи;
2. разработать алгоритм отложенного обновления вторичных индексов на LSM-деревьях;
3. реализовать алгоритм на основе архитектуры базы данных Tarantool;
4. провести экспериментальную апробацию реализации.

### 1.4 Структура работы

Во второй главе приведен обзор существующих способов уменьшения влияния скрытых чтений на запись в БД, рассказано о том, какие у них недостатки и преимущества, и почему их не достаточно для решения задачи.

Для более тонкого понимания актуальности проблемы скрытых чтений именно на LSM деревьях в третьей главе раскрываются основные детали устройства LSM дерева, кратко изложены алгоритмы его обновления (вставки, удаления, слияния уровней), устройство индекса, хранящего данные в LSM дереве. В той же главе на примере таблицы с множеством индексов на LSM деревьях разобрана проблема ее обновления.

В главе 4 описан предложенный алгоритм отложенного обновления индексов на LSM деревьях. В этой главе детально разобран каждый шаг алгоритма, и приведены математические доказательства ускорения записи. Здесь же разобрана реализация новых LSM деревьев на архитектуре БД Tarantool.

В пятой главе представлены результаты экспериментального измерения производительности новых таблиц на LSM деревьях в сравнении со старыми.

Последняя глава, шестая, содержит описание результатов работы, и рассуждения о том, в какую сторону можно развивать новый алгоритм, и какие еще есть способы ускорить запись в БД.

## 2 Обзор существующих способов ускорения записи

Цели обзора:

- показать, что проблема скрытых чтений появилась давно, действительно актуальна до сих пор, и не только на LSM деревьях;
- вычленить из существующих алгоритмов идеи, которые были бы применимы для поставленной задачи;
- учесть ошибки обзриваемых решений.

### 2.1 Способ 1

1

### 2.2 Способ 2

1

### 2.3 Способ 3

1

## 3 LSM дерево

LSM дерево [1] - это структура данных, оптимизированная для задач с интенсивной записью. Оно состоит из множества уровней: нулевой уровень хранится в оперативной памяти, а остальные - на диске. В нулевой уровень записываются все новые данные, обновления и удаления старых данных. Удаление в LSM дереве тоже реализовано через вставку особой записи в нулевой уровень.

Когда нулевой уровень становится слишком велик, то он записывается на диск, и удаляется из памяти. При этом запись нулевого уровня на диск выполняется последовательно независимо от того, какие ключи, какие операции и в каком порядке он содержит. Записанный уровень становится первым, и в памяти создается новый пустой нулевой уровень для последующих обновлений.

При достижении числом уровней определенного порога LSM дерево выполняет процедуру слияния уровней, во время которой некоторые соседние уровни объединяются в один. LSM дерево может содержать разные версии одних и тех же данных на разных уровнях, например, если запись с некоторым ключом сначала вставили в нулевой уровень, затем он был записан на диск, и в новый нулевой уровень было вставлено обновление или удаление того же ключа. Во время слияния уровней старые версии записей обнаруживаются и удаляются - они просто не попадают в новый уровень. То есть слияние в том числе выполняет функцию сборки мусора. После окончания объединения старые уровни удаляются и заменяются одним новым.

Способ организации нулевого уровня и уровней на диске не специфицируется явно, и зависит от реализации. Например, нулевой уровень может быть организован как B+ или красно-черное дерево. На диске уровни могут храниться в виде B деревьев, или как просто массивы записей, отсортированные по ключу.

### 3.1 Слияния уровней

1

### 3.2 Устройство индекса

1

### 3.3 Проблема обновления индекса

1



## 4 Отложенное обновление вторичных индексов на LSM-деревьях

1

### 4.1 Вставка и удаление

1

### 4.2 Слияние уровней

1

#### 4.2.1 Первичный индекс

1

#### 4.2.2 Вторичный индекс

1

### 4.3 Чтения

1

### 4.4 Реализация

1

## 5 Экспериментальное исследование

1

## 6 Заключение

1

## Список литературы

- [1] O’Neil P. et al. The log-structured merge-tree (LSM-tree) //Acta Informatica. – 1996. – Т. 33. – No. 4. – С. 351-385.
- [2] Comer D. Ubiquitous B-tree //ACM Computing Surveys (CSUR). – 1979. – Т. 11. – №. 2. – С. 121-137.
- [3] Agrawal N. et al. Design Tradeoffs for SSD Performance //USENIX Annual Technical Conference. – 2008. – Т. 8. – С. 57-70.
- [4] Wang P. et al. An efficient design and implementation of LSM-tree based key- value store on open-channel SSD //Proceedings of the Ninth European Conference on Computer Systems. – ACM, 2014. – С. 16.
- [5] Yang F. et al. Optimizing NoSQL DB on flash: A case study of RocksDB //Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC- ScalCom), 2015 IEEE 12th Intl Conf on. – IEEE, 2015. – С. 1062-1069.
- [6] Wang P. et al. An efficient design and implementation of LSM-tree based key- value store on open-channel SSD //Proceedings of the Ninth European Conference on Computer Systems. – ACM, 2014. – С. 16.й
- [7] Сайт базы данных Tarantool [Электронный ресурс] : сайт содержит документацию всех выпущенных версий Tarantool. — Режим доступа: <https://tarantool.io>. - Загл. с экрана.
- [8] Lersch L. et al. An analysis of LSM caching in NVRAM //Proceedings of the 13th International Workshop on Data Management on New Hardware. – ACM, 2017. – С. 9.
- [9] Dong S. et al. Optimizing Space Amplification in RocksDB //CIDR. – 2017.