Федеральное государственное автономное образовательное
учреждение высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук
Основная образовательная программа
Прикладная математика и информатика

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему

Управление прореживанием данных с помощью TTL выражений

Выполнил студент группы 165, 4 курса,
Сорокин Николай Анатольевич
Руководитель ВКР:
Руководитель группы разработки ClickHouse,
Миловидов Алексей Николаевич

Москва 2020

Содержание

1 Аннотация 3
2 Аннотация на английском 4
3 Обозначения и сокращения 5
4 Введение 6
4.1 Актуальность 6
4.2 Постановка задачи 7
5 Обзор аналогов в других СУБД 9
5.1 TTL-индексы в MongoDB 9
5.2 Команда EXPIRE и LRU/LFU режимы в Redis . . . 9
5.3 TTL атрибуты в Amazon DynamoDB 10
5.4 Retention policy и continuous query в InfluxDB 10
5.5 TTL выражения и движок GraphiteMergeTree в ClickHouse 11
6 Архитектура ClickHouse 12
7 Наш вклад 14
8 Заключение 15

1. Аннотация

Пользователи ClickHouse могут задать в таблице выражение, которое определяет, сколько времени хранятся данные. Обычно это выражение задаётся относительно значения столбца с датой - например: удалять данные через три месяца. Это может быть задано для всей таблицы (тогда строки целиком удаляются после указанного времени) или для отдельных столбцов (тогда данные столбца физически удаляются с диска, а строки в таблице остаются; при чтении значений столбца, они читаются как значения по-умолчанию).
Но пользователи также хотят более продвинутый вариант этой функциональности: не удалять строки или столбцы целиком, а прореживать их - оставлять меньшее количество строк.
В данной работе была реализована поддержка удаления части устаревших данных и возможность выполнять агрегацию данных, заменяя значения некоторых столбцов на значения агрегатных функций от множества значений в нескольких строках.

Ключевые слова: ClickHouse, большие данные, распределённые системы, OLAP, СУБД, TTL, прореживание данных

2. Аннотация на английском

In ClickHouse users can specify an expression for the table that determines how long the data is stored. Usually, this expression is relative to the value of the date column - for example: remove data older than three months. This can be specified for the entire table (delete rows after expiration) or just some columns of the table (data is removed from disk, but rows remain in the table; values of these columns in expired rows are read as default values).
Users also want a more advanced version of this functionality: rollup data - aggregate data to reduce the number of rows, rather than delete them entirely.
As a result of this work, we implemented an ability to delete only a subset of expired data or replace expired rows with aggregates of them.

Keywords: ClickHouse, big data, distrubted systems, OLAP, DBMS, TTL, data rollup

3. Обозначения и сокращения

База данных — структурированная совокупность данных, которые хранятся и доступны в электронном виде из компьютерной системы.

СУБД — система управления базами данных, программная система, позволяющая описывать, создавать, изменять базы данных, читать и модифицировать данные в них.

OLAP — online analytical processing, подход для вычисления многомерных аналитических запросов над большими массивами данных в режим онлайн (с большой скоростью).

TTL — time to live, момент времени, после которого данные считаются устаревшими.

JSON — формат сериализации данных для последующего обмена, который использует человеко-читаемые текстовые преставления объектов.

Key-value база данных — база данных в виде ассоциативного массива.

Индекс базы данных — объект, предназначенный для ускорения поиска строк в базе данных. Например, для ускорения поиска строк по значению столбца, индекс может иметь структуру сбалансированного дерева над значениями этого столбца.

4. Введение

4.1. Актуальность

В современном мире объем данных, сгенерированных различными программами, очень быстро растет. Исследования[1] показывают, что каждые два года количество данных увеличивается как минимум в два раза. Для того чтобы эффективно хранить и обрабатывать большие объемы данных, часто используют системы управления базами данных (СУБД). Существует множество различных типов архитектур этих систем. Их можно разделить на два класса: строковые и столбцовые. В строковых СУБД атрибуты одной строки хранятся на диске последовательно. Такое устройство хранения данных позволяет эффективно и удобно использовать базу данных в приложениях, которые в основном работают с единичными строками (например, банковские системы, в которых строка хранит данные об одном клиенте). Но такой способ хранить данные на диске плохо подходит для аналитических задач, в которых обычно надо обрабатывать большие объемы строк. Для таких задач используют столбцовые[2] СУБД, в которых на диске последовательно хранятся элементы столбцов. Такой подход позволяет применять различные оптимизации. Например, если для вычисления результата запроса требуется только подмножество столбцов, то можно считывать с диска только их. Так же, алгоритмы сжатия данных на диске работают более эффективно из-за того, что последовательные данные однородные. Кроме того, можно применять векторизованные функции для ускорения вычисления запроса.

ClickHouse – распределенная столбцовая система управления базами данных с открытым исходным кодом, разработанная в компании Яндекс для быстрого вычисления аналитических запросов в сервисе Яндекс.Метрика. Её ключевые особенности — это высокая производительность (в некоторых сравнениях с аналогами запросы выполняются быстрее более чем в 100 раз), линейное масштабирование, отказоустойчивость, возможность шардирования и распределенного

вычисления запросов на нескольких серверах.
Из-за большого роста объемов данных может возникнуть проблема нехватки памяти. Поэтому появляется большая необходимость поддержки различных механизмов для эффективного удаления или агрегации (прореживания) частей данных в системах хранения и обработки данных (таких как СУБД). Цель данной работы – реализовать поддержку дополнительных функций для прореживания данных в ClickHouse.

4.2. Постановка задачи

Сейчас в ClickHouse есть механизм TTL выражений. Для таблицы можно задать одно выражение, результат выполнения которого на строке таблицы будет являться моментом, когда данные можно удалить. Так же можно задать несколько выражений, результаты выполнения которых будут являться моментами, когда эти строки можно перемещать на соответствующие диски или тома. Пример синтаксиса в листинге 1.

```
CREATE TABLE example_table
(
    d DateTime, a Int
)
ENGINE = MergeTree
PARTITION BY toYYYYMM(d)
ORDER BY d
TTL d + INTERVAL 1 MONTH DELETE,
    d + INTERVAL 1 WEEK TO VOLUME 'aaa',
    d + INTERVAL 2 WEEK TO DISK 'bbb';
```

Листинг 1 — Пример TTL выражений

Первая задача этой работы – поддержать возможность задавать фильтрующее условие, для того чтобы удалять только те устаревшие строки, которые подходят под этот фильтр. Пример синтаксиса в листинге 2.

```
CREATE TABLE example_table
(
    d DateTime, a Int
)
ENGINE = MergeTree
PARTITION BY toYYYYMM(d)
ORDER BY d
TTL d + INTERVAL 1 MONTH DELETE WHERE cityHash(a) % 10 = 0;
```

Листинг 2 — Пример TTL выражений с WHERE

Вторая задача – поддержать возможность агрегации устаревших данных вместо удаления. Агрегация производится с помощью группировки по префиксу первичного ключа. Пример синтаксиса в листинге 3.

```
CREATE TABLE example_table
(
    d DateTime, k1 Int, k2 Int, x Int, y Int
)
ENGINE = MergeTree
ORDER BY k1, k2
TTL d + INTERVAL 1 MONTH GROUP BY k1, k2 SET x = sum(x), y = min(y);
```

Листинг 3 — Пример TTL выражений с GROUP BY

5. Обзор аналогов в других СУБД

В большинстве популярных систем управления базами данных нет поддержки прореживания устаревших данных, эту функциональность удалось найти только в InfluxDB и ClickHouse. Далее мы рассмотрим реализацию этой функциональности в этих системах. Так же, многие СУБД поддерживают возможность указания времени хранения данных (TTL). Мы так же рассмотрим устройство этих механизмов, так как реализация данной работы основана на TTL выражениях в ClickHouse.

5.1. TTL-индексы в MongoDB

MongoDB это распределенная строковая система управления базами данных с открытым исходным кодом. Записи в MongoDB называются документами и представлены в виде JSON объектов. Набор документов называется коллекцией и является аналогом таблицы в традиционных СУБД.

В MongoDB можно указать время, которое нужно хранить документы с помощью создания специального TTL-индекса[3], который связан со столбцом, обозначающим момент добавления документа в коллекцию. Момент времени, в который можно удалять данные вычисляется прибавлением к значению этого столбца какого-то числа (которое указывается в свойствах индекса). Это значение нельзя сделать разным для разных подмножеств документов в коллекции, но тем не менее можно добиться разного TTL, изменив нужным образом значения в столбце, связанном с индексом. Данные удаляются с диска с помощью фонового процесса, который запускается каждую минуту. Из-за этого документы могут оставаться в коллекции и после момента, когда истек их TTL.

5.2. Команда EXPIRE и LRU/LFU режимы в Redis

Redis это СУБД для управления key-value базами данных. В Redis есть команда EXPIRE[4], которая позволяет установить время,

которое должны храниться объекты с заданными ключами. В этой СУБД есть два способа поиска данных, которые надо удалять. Во-первых, когда пользователь пытается получить данные, система проверяет их TTL и если он истек, удаляет данные с диска. Во-вторых, специальный фоновый процесс запускается десять раз в секунду и проверяет TTL у небольшой случайной подвыборки ключей. Если TTL истек у более 25% ключей в этой выборке, то процесс автоматически запускается заново для поиска других устаревших ключей.
Кроме того в Redis есть специальные режимы[5] для более умного контроля потребляемой памяти и удаления ненужных данных. Пользователь может задать, сколько максимально памяти могут занимать все объекты в базе данных. Так же пользователь выбирает один из алгоритмов поиска ненужных строк. Например, считать ненужными те объекты, которые дольше всего не использовались, либо те, которые реже всего используются, либо же объекты со специальным атрибутом. Когда размер потребляемой памяти достигнет лимита, система автоматически удалит строки в соответствии с выбранным алгоритмом.

5.3. TTL атрибуты в Amazon DynamoDB

Amazon DynamoDB это проприетарная распределенная СУБД для управления key-value базами данных, которая предоставляется как часть Amazon Web Services. Для того чтобы установить время хранения данных [6], пользователь указывает столбец, хранящий момент, после которого строки можно удалять, в настройках таблицы. Данные удаляются с диска фоновым процессом. В зависимости от загруженности системы, задержка между устареванием строки и удалением данных с диска может достигать 48 часов.

5.4. Retention policy и continuous query в InfluxDB

InfluxDB это СУБД с открытым исходным кодом, предназначенная для хранения и обработки временных рядов. В InfluxDB поль-

зователь может задать для таблицы ”retention policy”[7] - правило, которое определяет как долго хранить данные в ней. Так же в InfluxDB можно создавать ”continuous query” - запрос, который будет автоматически запускаться через заданный период и агрегировать данные, записанные в таблицу в этот период. Значит, скомбинировав retention policy и continuous query, можно сделать умное прореживание данных. Например, можно создать две таблицы, в первой будут храниться только новые данные (это поддерживается с помощью retention policy), специальный continuous query будет вычислять агрегированный слепок этих данных и записывать их во вторую таблицу. Таким образом получаем хранение всех новых данных и агрегатные статистики старых данных.

5.5. TTL выражения и движок GraphiteMergeTree в ClickHouse

В ClickHouse можно задать выражение[8], результат вычисления которого будет будет являться моментом, когда строку можно удалять.

Так же в ClickHouse есть специальный режим хранения и обработки данных (называемый движком) GraphiteMergeTree[9], предназначенный для работы с данными системы мониторинга Graphite. Он позволяет указать настройки прореживания данных в таблицах, а именно, пользователь указывает время, которое данные должны храниться, а так же какой функцией агрегировать устаревшие данные (для каждого столбца) и точность агрегации (в итоге, после агрегации строка будет представлять группу за этот промежуток времени).

6. Архитектура ClickHouse

В Clickhouse существует несколько различных движков таблиц. Движок определяет:

— Как и где хранятся данные, куда их писать и откуда читать
— Какие запросы поддерживаются и каким образом
— Конкурентный доступ к данным
— Использование индексов, если есть
— Возможно ли многопоточное выполнение запроса
— Параметры репликации данных

Наиболее универсальный и функциональный движок таблиц для задач с высокой загрузкой - MergeTree. В этой работе новая функциональность будет реализована именно для этого движка.
Рассмотрим, как в движке MergeTree хранятся и обрабатываются данные. При вставке строк в таблицу, они сортируются по первичному ключу и сохраняются на диск. Такой отсортированный отрезок строк будем называть блоком. Во время выполнения SELECT запроса нужные строки ищутся во всех блоках на диске. Из-за этого очевидно, что количество блоков должно быть небольшим, иначе запросы будут выполняться не эффективно. Для этого есть специальный фоновый процесс (merge), который при достижении определенного количества блоков объединяет последовательные блоки в один и пересортировывает строки в нем в соответствии с первичным ключом. Заметим, что во время выполнения merge-процесса можно изменить данные перед записью на диск. Именно так работают TTL выражения: после объединения блоков для всех строк рассчитывается время их устаревания и устаревшие строки удаляются (или переносятся на заданный том или диск).

Запросы парсятся в древовидные структуры (интерфейс `IAST`). Запрос создания таблицы - описывается объектами класса `ASTCreateQuery`; часть, описывающая движок представляется объектами класса `ASTStorage`; информация о TTL выражениях -

объектами класса `ASTTTLElement`.

Вся логика работы с данными в движке MergeTree описана в классе MergeTreeData. При выполнении запросов CREATE TABLE или ALTER TABLE с указанием TTL выражений, вызывается метод `MergeTreeData::setTTLExpressions`, который преобразовывает объекты `ASTTTLElement` в структуру `TTLEntry`, конвертируя все выражения в цепочки действий и проверяя корректность.

Обработка данных в ClickHouse осуществляется с помощью потоков блоков (интерфейс `IBlockInputStream`). Для удаления устаревших данных используются объекты класса `TTLBlockInputStream`.

Логика фонового merge-процесса описана в классе `MergeTreeDataMergerMutator`. Основная работа происходит в методе `MergeTreeDataMergerMutator::mergePartsToTemporaryPart`: создается и запускается конвейер потоков блоков, в который в том числе входит `TTLBlockInputStream`.

7. Наш вклад

Для реализации удаления только части устаревших данных в `ASTTTLElement` и соответственно в `TTLEntry` добавляется выражение, результат вычисления которого будет фильтровать строки. Далее в `TTLBlockInputStream` для всех строк в блоке вычисляется столбец с результатом этого выражения и те строки, которые прошли фильтр и устарели удаляются.

Для реализации агрегации устаревших строк в `ASTTTLElement` и в `TTLEntry` сохраняются имена столбцов группировки (которые должны быть префиксом первичного ключа) и агрегатные выражения, на результаты вычисления которых надо заменять значения в столбцах устаревших строк. В `TTLBlockInputStream` добавляется объект класса `Aggregator`, который реализует логику агрегации строк. Во время итерации по строкам блока поддерживаем набор устаревших строк с текущим ключом группировки; строки, у которых еще не истек TTL сразу записываем в итоговый блок. Когда встречаем строку с новым ключом группировки или строки кончаются, обновляем состояния агрегатных функций с учетом текущего набора устаревших строк с помощью функции `Aggregator::executeOnBlock`; текущие состояния агрегатов хранятся в структуре `AggregatedDataVariants`. После этого, если мы встретили строку с новым ключом группировки или закончились блоки, финализируем текущие состояния агрегатов, т. е. из промежуточных состояний получаем готовые строки. Над ними считаем заданные выражения и итоговые строки записываем в результирующий блок. Так как на вход `TTLBlockInputStream` подается уже отсортированный по первичному ключу блок, заметим что описанный процесс не нарушает свойство сортировки. Так же, для сохранения сортировки, столбцы, входящие в первичный ключ, но не попавшие в ключ группировки, в результате будут автоматически равны максимальному значению в группе.

8. Заключение

В данной работе был проведен анализ существующих механизмов прореживания данных в различных популярных системах управления базами данных. Была изучена архитектура и внутренние особенности СУБД ClickHouse. В результате был подготовлен pull request, в котором полностью реализована поддержка возможности удаления только тех устаревших строк, которые проходят фильтрацию, и возможности агрегировать устаревшие строки, группируя их. Так же были добавлены функциональные тесты на новые внедрения, реализованные в этой работе.

Список литературы

1. Gantz, J. & Reinsel, D. *The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East* в (2012).

2. MATEI, G. Column-Oriented Databases, an Alternative for Analytical Environment. *Database Systems Journal* **1,** 3—16. https://ideas.repec.org/a/aes/dbjour/v1y2010i2p3-16.html (дек. 2010).

3. MongoDB Inc. *TTL Indexes* https://docs.mongodb.com/manual/core/index-ttl/.

4. Redis Labs. *Expire command* https://redis.io/commands/expire.

5. Redis Labs. *Using Redis as an LRU cache* https://redis.io/topics/lru-cache.

6. Amazon. *TTL* https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/TTL.html.

7. InfluxData. *Downsampling and data retention* https://docs.influxdata.com/influxdb/v1.7/guides/downsampling_and_retention/.

8. Yandex. *TTL for Columns and Tables* https://clickhouse.tech/docs/en/engines/table-engines/mergetree-family/mergetree/#table_engine-mergetree-ttl.

9. Yandex. *GraphiteMergeTree* https://clickhouse.tech/docs/en/operations/table_engines/graphitemergetree/.