НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Факультет компьютерных наук
ООО "Яндекс"
Программный проект
Методы индексации данных на основе space filling curves.

Выполнил студент группы БПМИ-171
Чулков Андрей Сергеевич
Научный руководитель:
Руководитель команды
Миловидов А. Н.

ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

- Задача относится к методам поиска в базах данных, основывающихся на индексировании.

- В ClickHouse для ускорения поиска реализована возможность индексации по кортежу из столбцов таблицы.
- Часто при проектировании базы данных в ClickHouse, трудно выбрать порядок столбцов ключа в кортеже. Для примера, в базе данных рекламной системы, ключевыми столбцами является идентификатор рекламодателя (кто заказывал рекламу) и идентификатор рекламной площадки (на каком сайте размещена реклама). Отчёты надо строить иногда для рекламодателя, а иногда - для рекламной площадки. То есть, первым столбцом в ключе может быть или тот, или другой идентификатор.
- Хочется упорядочить данные по некоторому компромиссному индексу, чтобы запросы работали хорошо в обоих случаях.
- Для этого будут использованы **space filling curves.**

ОСНОВНЫЕ ПОНЯТИЯ, ОПРЕДЕЛЕНИЯ, ТЕРМИНЫ

ClickHouse — столбцовая система управления базами данных (СУБД) для онлайн обработки аналитических запросов (OLAP). (из https://clickhouse.yandex/docs/ru/)
MergeTree — вид таблиц в Clickhouse, поддерживающих индексирование и проверку выполнимости условий.
Space filling curve — некоторая функция, которая отображает многомерные данные в одномерные, сохраняя локальность точек данных.
Гранула —множество записей в таблице, записанных подряд. В таблицах MergeTree размер гранулы задается при создании таблицы и по умолчанию равен 8192.
Первичный ключ — выражение, по которому сортируются данные при разбиении на гранулы и вычисляются засечки.
Засечка — сохраняемое значение первичного ключа для первого элемента гранулы.
Запрос — набор условий на столбцы таблицы, по которым мы хотим тем или иным образом профильтровать таблицу — ((x >= 10) and (x <= 15)) or (x + y >= 25)

АКТУАЛЬНОСТЬ РАБОТЫ

- При равном числе запросов на с условиями на разные столбцы первичного ключа, или же условиями на несколько столбцов одновременно, запросы на столбцы, кроме первого, выполняются не очень эффективно.
- Порой даже простые запросы уже на второй столбец первичного ключа просматривают всю таблицу, имеющую очень большие размеры, а чтение данных с диска это очень медленная операция.
- Хоть это и работает довольно быстро, можно было бы уменьшить объем считываем данных, что должно существенно уменьшить время выполнения запроса.
- Производительность можно попробовать улучшить с помощью **space filling curves**, позволяющими смешать данные в один индекс, сохраняя при этом локальность, а значит запросы имеют шанс выполняться эффективно.

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Цель работы

Реализация индексирования на основе space filling curves, которое позволит уменьшить количество данных, читаемых с диска при SELECT запросе с условием на проиндексированные столбцы и, как следствие, ускорит выполнение таких запросов.

Задачи работы

1. Разработка алгоритма вычисления, индексируемой функции.
2. Разработка алгоритма проверки выполнимости запроса на грануле при помощи вычисленного индекса.
3. Реализация данных алгоритмов в существующем коде ClickHouse.
4. Проверка улучшения производительности.

АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ

- СУБД Amazon DynamoDB поддерживает возможность такого индексирования, в реализации со стороны клиента, запись в блоге: https://aws.amazon.com/blogs/database/z-order-indexing-for-multifaceted-queries-in-amazon-dynamodb-part-1/, Zack Slayton, May 2017

- Gaede, Volker; Guenther, Oliver, "Multidimensional Access Methods", ACM COMPUTING SURVEYS, 1997, Vol. 30, 170-231, http://www.cs.columbia.edu/~gravano/Qual/Papers/24%20-%20Multidimensional%20Access%20Methods.pdf
- Ramsak, Frank; Markl, Volker; Fenk, Robert; Zirkel, Martin; Elhardt, Klaus; Bayer, Rudolf (2000), "Integrating the UB-tree into a Database System Kernel", Int. Conf. on Very Large Databases (VLDB) (PDF), pp. 263–272, http://www.vldb.org/conf/2000/P263.pdf

Индекс в ClickHouse

- Таблица разбита на куски, которые в свою очередь разбиты на гранулы, которые и будут нас интересовать.
- Первичный ключ — набор столбцов или функций и выражений от них, по которым происходит сортировка таблицы. Сравнение происходит лексикографически.
- При заданном первичном ключе для каждой гранулы хранятся засечки — значение первичного ключа на первом элементе гранулы.
- Таким образом, для каждой гранулы имеет некоторый отрезок [L, R] на котором лежат значения в ней, где L значение первичного ключа на первом элементе текущей гранулы, а R на первом элементе следующей.
- L и R могут быть кортежами, поэтому при обработке запроса отрезок [L, R] преобразуется в набор параллелограммов, ограничивающих значения координат первичного ключа. Например [(1, 2), (+inf, +inf)] преобразуется в объединение прямоугольников (1, +inf) x [-inf, +inf] и [1] x [2, +inf).
- Условия запроса преобразуются в обратную польскую нотацию, после чего каждое атомарное условия (вида, например, x >= 1) проверяется на выполнимость на каждом построенном параллелограмме. Результаты этих проверок используются для пропуска заведомо неподходящих гранул.

Поддержка обратимых функций

- В запросах уже поддерживаются условия, содержащие монотонные функции, например negate(x) >= 1.
- Проверка происходила посредством применения цепочки монотонных функций, наложенных на элемент ключа в запросе, к проверяемому отрезку значений ключа. То есть, для того, чтобы проверить, условие f(x) ≥ t, где x — столбец ключа, лежащий на отрезке [L, R], делается проверка выполнимости условия y ≥ t на отрезке [f(L), f(R)]. Здесь проверяется монотонность функции на данном отрезке (и возможно меняются местами его концы).
- Эта проверка была модифицирована для поддержки обратимых функций, примененных в **индексе**. Например, если таблица проиндексирована по выражению g(x, y), то хотелось бы при проверке условия x ≥ t уметь делать какие-то пропуски, имея условие L ≤ g(x, y) ≤ R.
- Реализованный функционал обратимых функций позволяет задать у функции g метод invertRange, который по данному отрезку [L, R] и номеру аргумента i возвращает допустимые значения i-го аргумента функции g, при условии, что значение g лежит на отрезке [L, R].
- Такое построение может возвращать набор отрезков. Для этого реализован класс RangeSet, основывающийся на уже существующем классе для отрезка Range.

Поддержка множества отрезков

- Есть тонкости в связи с тем, что Range допускает различные комбинации включения / исключения концов отрезка из множества.
- Хранится в виде отсортированного массива **непересекающихся** отрезков.
- Для поддержания этого состояния отрезки сортируется, после чего используется алгоритм, называемый часто алгоритмом сканирующей прямой: идем слева направо поддерживая правую границу текущей области.
- Для дальнейшей корректной проверки условий в параллелограмме нужно уметь пересекать набор отрезков значений ключа с искомом отрезком значений в запросе.
- Проверка реализована в виде двух двоичных поисков по массиву.

Реализация шаблона space filling curves

- Так как предполагается возможность дальнейшего дополнения реализованного метода, **space filling curves** реализованы в виде шаблонного класса ZCurveBase
- Функция всегда вычисляется как чередование бит своих аргументов, возвращая 64-битный результат (см. картинку на следующем слайде).
- К каждому аргументу при этом применяется кодирующая функция encode, которая в зависимости от типа аргумента кодирует его требуемом образом в виде 64-битного числа. Эта функция передается в шаблоне.
- Реализация обращения ZCurveBase принимает диапазон значений функции и номер аргумента и возвращает в итоге множество диапазонов значений этого аргумента, при котором могут достигаться данные значения функции. Сначала функция обращения возвращает данный диапазон в сыром 64-битном виде. Для этого используется алгоритм, описанный на следующем слайде, который по диапазону значений функции ZCurve и номеру аргумента возвращает максимальное допустимое значение этого аргумента на данном диапазоне. Аналогично находится минимальное значение. Можно доказать, что искомый диапазон это весь отрезок между минимальным и максимальным значением.
- После этого построенный диапазон преобразуется к множеству отрезков исходного типа с помощью функции decodeRange, которая также передается в шаблоне.

Иллюстрация вычисления функции

y: \ x:	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 000	000000	000001	000100	000101	010000	010001	010100	010101
1 001	000010	000011	000110	000111	010010	010011	010110	010111
2 010	001000	001001	001100	001101	011000	011001	011100	011101
3 011	001010	001011	001110	001111	011010	011011	011110	011111
4 100	100000	100001	100100	100101	110000	110001	110100	110101
5 101	100010	100011	100110	100111	110010	110011	110110	110111
6 110	101000	101001	101100	101101	111000	111001	111100	111101
7 111	101010	101011	101110	101111	111010	111011	111110	111111

Алгоритм сырого обращения отрезка

- Сырой алгоритм обращения принимает два 64-битных числа L и R, а также индекс i аргумента по которому надо обратить отрезок [L, R] и количество аргументов k, от которых была изначально вызвана функция.
- Возвращает минимальное и максимальное допустимое значение данного аргумента на данном диапазоне.
- Разбиваем числа L и R на блоки по k бит таким образом, чтобы бит, отвечающий нашему аргументу, был всегда последним в блоке. Самый первый (старший) блок может иметь меньше чем k бит.
- Для каждого блока делается выбор между значениями L_b и L_b + 1, где L_b — значение соответствующего блока у числа L.
- Для получение максимального значения, мы хотим, чтобы последний бит этого числа (то есть значащий бит данного нам аргумента) был единичным. Иногда это нельзя сделать из-за значения R_b.
- Если у нас получилось выбрать значение, строго меньшее R_b, то после этого можно ставить любые биты, поэтому происходит дополнение остатка числа R единицами.
- Есть и другие случаи, которые опущены, но суть остается той же.
- Минимальное значение находится аналогично.
- Практически всегда все значения от минимального, до максимального достигаются и вернув отрезок [MIN, MAX] мы не сильно проигрываем.

Обычная функция zCurve

- Функция encode копирует аргументы в беззнаковый 64-битный тип. В данный момент есть поддержка числовых типов, времени и дат, а также вещественных чисел.
- После этого запись преобразуется в такую, что при сохраняется монотонность в беззнаковых 64-битных числах.
- Например, у знаковых чисел инвертируется бит знака — таким образом кодирования отрицательных чисел будут идти до кодировок положительных, как и хочется.
- Для вещественных чисел применяется аналогичная трансформация, в зависимости от знака (там иногда надо инвертировать биты мантиссы и/или экспоненты).
- После этого аргументы имеющие изначальный размер меньше, чем 4 байта, дополняются нулями до нужного размера, начиная с младших разрядов.
- Функция decodeRange к сырому обращенному диапазону применяет обратное преобразование к функции encode.
- Это возможно, так как кодирование было однозначным и все операции легко инвертируются.

Нормализованная функция zCurveN

- Удаляет ведущие старшие нули и кодирует их количество в младших битах числа, сначала применив преобразование encode из обычной функции zCurve.
- Таким образом, группы чисел существенно разного порядка будут действительно перемешаны этим индексированием.
- Функция decodeRange находит наибольший общий префикс k данных чисел L и R, после чего перебирает возможное окончание числа (количество удаленных бит), называемое суффикс, и находит минимальное и максимальное возможное число с таким окончанием в данном диапазоне.
- Чтобы найти минимальное, надо выделить в L часть между совпадающим префиксом L и R (первые k бит) и зафиксированным суффиксом (число удаленных бит) и, возможно прибавив 1 к этой части, приписать в конец требуемый суффикс.
- Прибавление единицы нужно, если аналогичный суффикс числа L меньше перебираемого. Аналогично находится правая граница.
- После этого к получившимся вариантам применяется функция decodeRange из обычной функции zCurve.

Использование и тестирование

Использование

- Реализованный функционал очень прост в использовании.
- Достаточно использовать функцию zCurve или zCurveN в индексе, после чего запросы с условиями на параметры этих функций будут оптимизироваться.
- Например,
  - CREATE TABLE ... ENGINE = MergeTree ORDER BY zCurveN(ClientID, UserID)
  - SELECT * FROM TABLE ... WHERE ClientID = 57

Тестирование

- Тестирование производилось на тестах доступных в ClickHouse, вручную, а также на небольшом куске данных сервиса “Яндекс.Метрика”, который можно скачать с официального сайта СУБД Clickhouse.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

Сравнение производится на данных hits из "Яндекс.Метрики" по столбцам ClientID и UserID

```
SELECT count()
FROM testu
WHERE CounterID = 57

┌─count()─┐
│     319 │
└─────────┘

1 rows in set. Elapsed: 0.049 sec. Processed 8.83 million rows, 35.33 MB (181.15 million rows/s., 724.61 MB/s.)

SELECT count()
FROM testz
WHERE CounterID = 57

┌─count()─┐
│     319 │
└─────────┘

1 rows in set. Elapsed: 0.056 sec. Processed 6.69 million rows, 26.77 MB (119.01 million rows/s., 476.03 MB/s.)

SELECT count()
FROM testzn
WHERE CounterID = 57

┌─count()─┐
│     319 │
└─────────┘

1 rows in set. Elapsed: 0.013 sec. Processed 696.32 thousand rows, 2.79 MB (52.91 million rows/s., 211.62 MB/s.)
```

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

Сравнение производится на данных hits из "Яндекс.Метрики" по столбцам ClientID и UserID

```
SELECT count()
FROM testu
WHERE intHash32(UserID) = 400000

┌─count()─┐
│       0 │
└─────────┘

1 rows in set. Elapsed: 0.004 sec. Processed 16.38 thousand rows, 131.07 KB (3.99 million rows/s., 31.88 MB/s.)

SELECT count()
FROM testz
WHERE intHash32(UserID) = 400000

┌─count()─┐
│       0 │
└─────────┘

1 rows in set. Elapsed: 0.005 sec. Processed 157.61 thousand rows, 1.26 MB (34.03 million rows/s., 272.22 MB/s.)

SELECT count()
FROM testzn
WHERE intHash32(UserID) = 400000

┌─count()─┐
│       0 │
└─────────┘

1 rows in set. Elapsed: 0.011 sec. Processed 811.01 thousand rows, 6.49 MB (75.39 million rows/s., 603.10 MB/s.)
```

ВЫВОДЫ ПО РАБОТЕ

- Разработанный функционал позволяет добавлять новые эвристики и дополнения к функциям на основе **space filling curves**, или же реализовывать другие обратимые функции, которые можно использовать в ключе сортировки или первичном ключе аналогичным образом.
- Добавленные функции позволяют ускорить время работы на некоторых типах запросов в несколько раз, что и являлось поставленной задачей.
- Хоть некоторые запросы ожидаемо замедляются (, цель получения примерно равного, но эффективного времени для запросов на разные координаты можно считать достигигнутой.

Направления дальнейшей работы

- Можно заметить, что скорость чтения у функций zCurve и zCurveN меньше, чем при обычном индексе. Особенно это заметно у нормализованной функции, так как данных считывается порой на порядок меньше, а ускорение выражается в сильно меньшей степени.
- Стоит провести детальный анализ написанного кода, чтобы выявить причины этого и выяснить, можно ли это исправить. Работа над этим уже ведется.
- При этом, основное замедление происходит из-за создания большого количества отрезков. Поэтому можно будет рассмотреть эвристики, в которых числа сдвигаются лишь на некоторое фиксированное число бит (например, на степени двойки, или на половину битности числа), а не перебираются все варианты.
- Также можно заметить, что в некоторых случаях обычная функция zCurve работает лучше, чем нормализованная. Поэтому можно подумать о том, чтобы применять разное кодирование к разным параметрам нашей функции, в зависимости от плотности данных.
- В дальнейшем требуется более детально протестировать реализованный алгоритм, чтобы экспериментально выявить наилучшие сценарии использования.
- Реализованный функционал можно использовать для имплементации других обратимых функций для индексирования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] ClickHouse Documentation [Электронный ресурс] / Yandex. — Режим доступа: https://clickhouse.yandex/docs/en/, свободный. (дата обращения: 21.03.18).

[2] Z-order curve [Электронный ресурс] — Режим доступа: https://en.wikipedia.org/wiki/Z-order_curve свободный. (дата обращения: 21.03.19).

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Спасибо за внимание!

Чулков Андрей Сергеевич
achulkov2@gmail.com

Москва - 2019