ClickHouse для инженеров и архитекторов БД

Индексы в ClickHouse

otus.ru

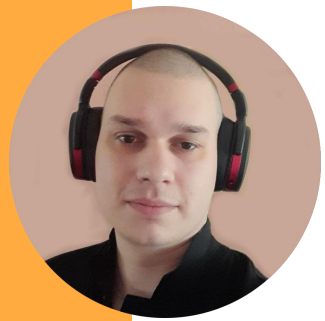
Проверить, идет ли запись

Меня хорошо видно && слышно?

Ставим “+”, если все хорошо
“-”, если есть проблемы

Тема вебинара

Индексы в ClickHouse



Константин Трофимов

Senior SRE / ClickHouse DBA в VK

Занимаюсь эксплуатацией ClickHouse с первых версий: 5 лет в VK, до этого в AdNow, до этого занимался Vertica. Сотни серверов, десятки кластеров, десятки петабайт данных.

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в учебной группе
#OTUS ClickHouse-2024-08

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

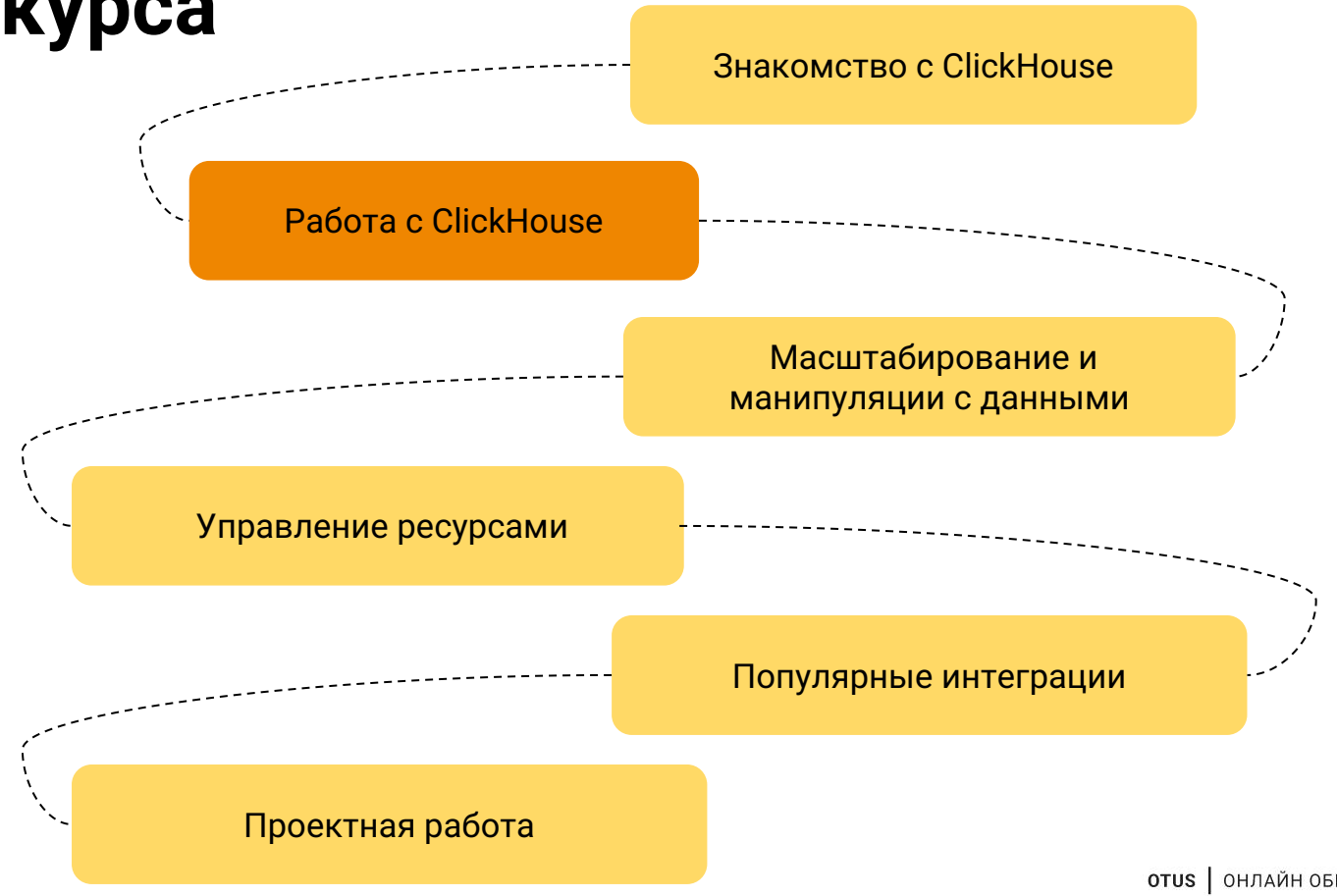
Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

Карта курса


Знакомство с ClickHouse
Работа с ClickHouse
Масштабирование и манипуляции с данными
Управление ресурсами
Популярные интеграции
Проектная работа

Темы модуля
Работа с ClickHouse

1 (5). Язык запросов SQL

2 (6). Функции для работы с типами данных, агрегатные функции и UDF

3 (7). Движки MergeTree Family

4 (8). ⚠ Индексы в ClickHouse

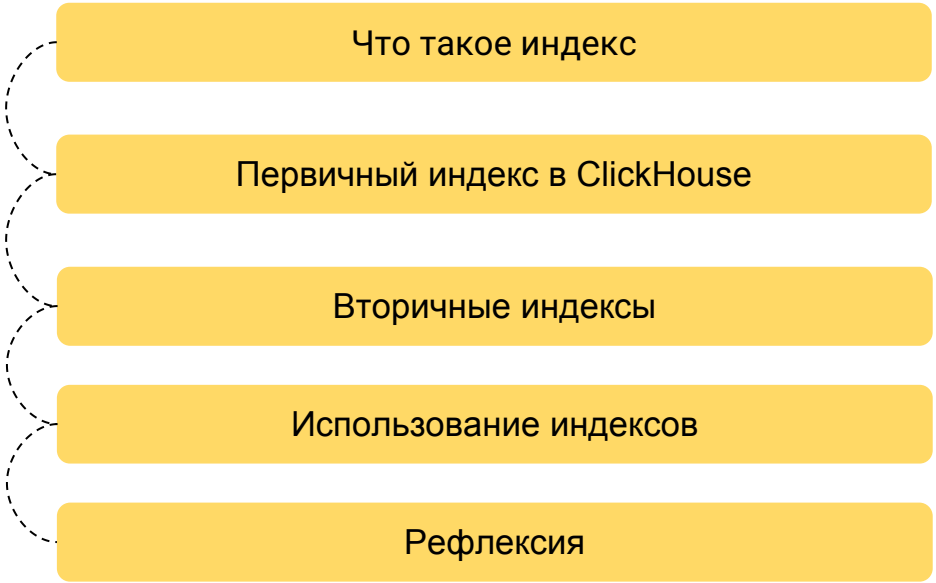
5 (9). Другие движки

6 (10). Джоины и агрегации

7 (11). Словари, оконные и табличные функции

8 (12). Сессия Q&A

Маршрут вебинара



Индексы в ClickHouse

Цели вебинара

После занятия вы сможете

1. Ответить на вопрос что такое индекс
2. Корректно задавать индексы при разработке и сопровождении схем БД
3. Выявлять потребность во вторичных индексах

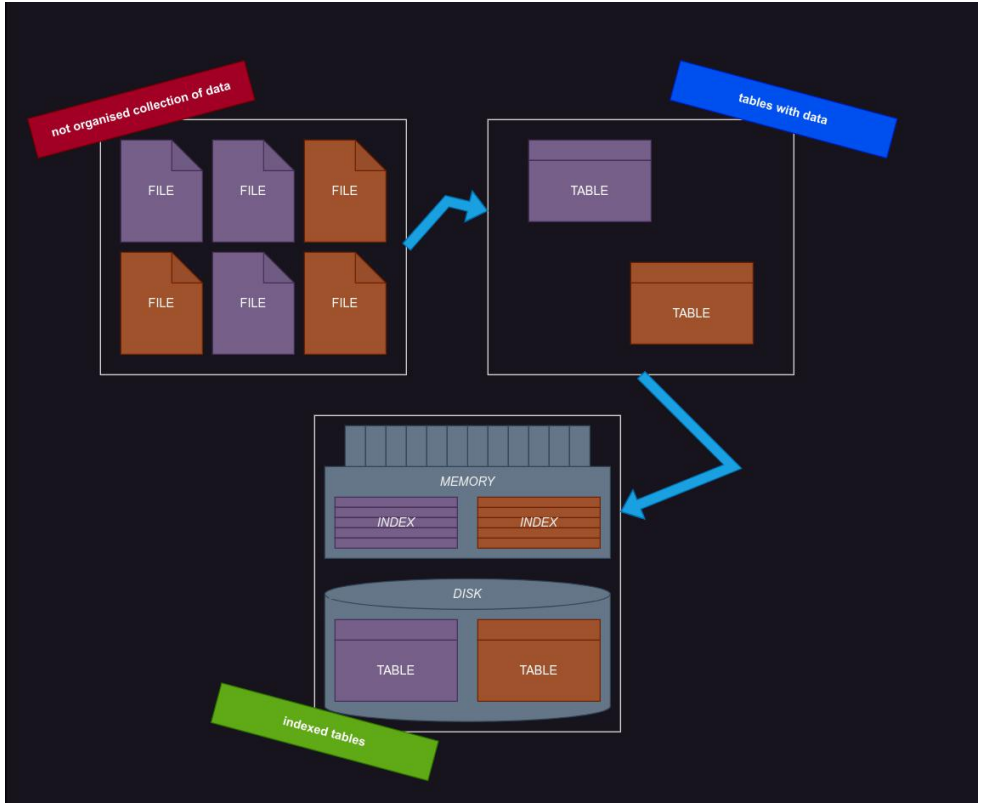
Что такое индекс

Зачем нужен индекс

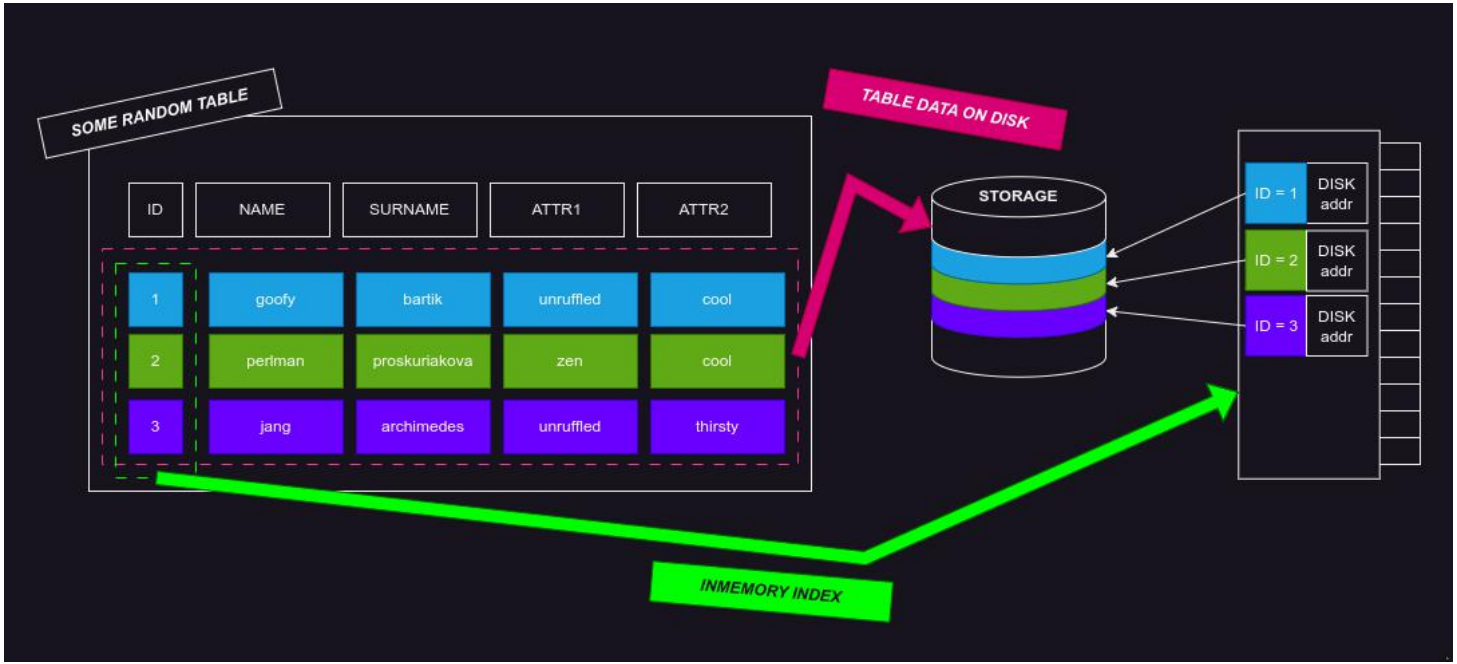
Когда хранить данные на диске файлами становится неоптимально и неудобно, и/или характер работы с данными требует структурированого хранения, данные огранизуются в таблицы, а таблицы в БД.

Индекс центральное понятие такой структуры хранения, обеспечивающее скорость доступа к данным.

Индекс располагается оперативной памяти, данные на диске. В зависимости от типа индекса, могут налагаться требования, такие как уникальность каждой «координаты», и наоборот, ограничения на кардинальность (количество уникальных значений) индекса, в зависимости от устройства и назначения индекса.



Что такое индекс



Индекс в базах данных - карта области, где хранятся данные таблицы, где в качестве координат используются значение произвольного выражения, чаще всего просто значения конкретного столбца, для ускорения получения данных по этим координатам.

Первичный и вторичный

Индексы бывают первичными (основными), задаваемыми при создании таблиц, жестко фиксируемые в структуре таблицы, и не подлежащие после этого изменению.

А так же индексы бывают вторичными, добавляемыми и удаляемыми уже на готовой таблице. Основной индекс чаще всего называют «первичный ключ».

Первичный индекс в ClickHouse

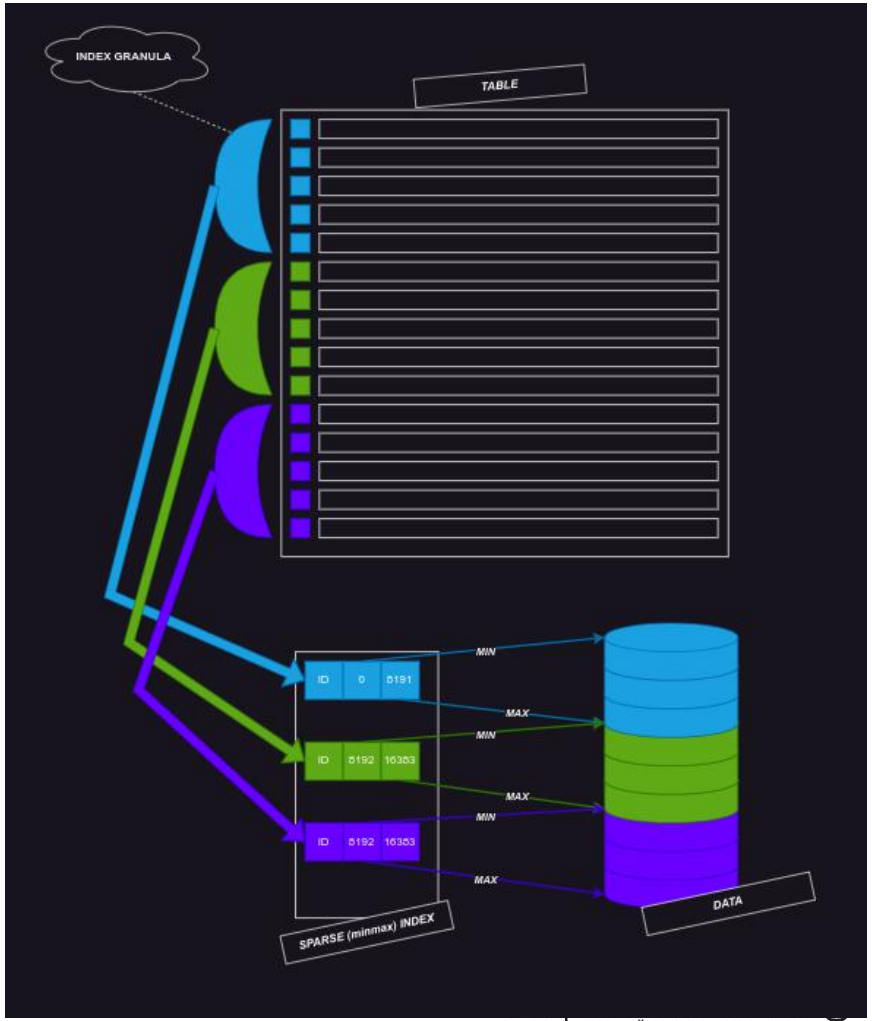
Разреженный minmax индекс

Хранит не единичные значения ключевого поля в единичной строке, а вместо этого минимальное и максимальное значения этого поля на N строк.

Последовательность N таких строк называется гранулой, размерность N - гранулярностью индекса. В ClickHouse по умолчанию размер такой гранулы 8192 строк.

Используется в качестве первичного ключа (основного индекса) для таблиц в ClickHouse.

Допускает неуникальные значения, допускает более 8192 строк (или гранулы) одинаковых значений.



Почему основной в ClickHouse?

ClickHouse - аналитическая база данных, это предполагает OLAP профиль нагрузки - не точечные запросы, обрабатывающие миллионы и миллиарды строк, для подсчета итоговых значений за долгий период или вывления на этом периоде закономерностей в данных. Объем данных, типичных для хранения в такой системе, измеряется как правило в терабайтах и петабайтах.

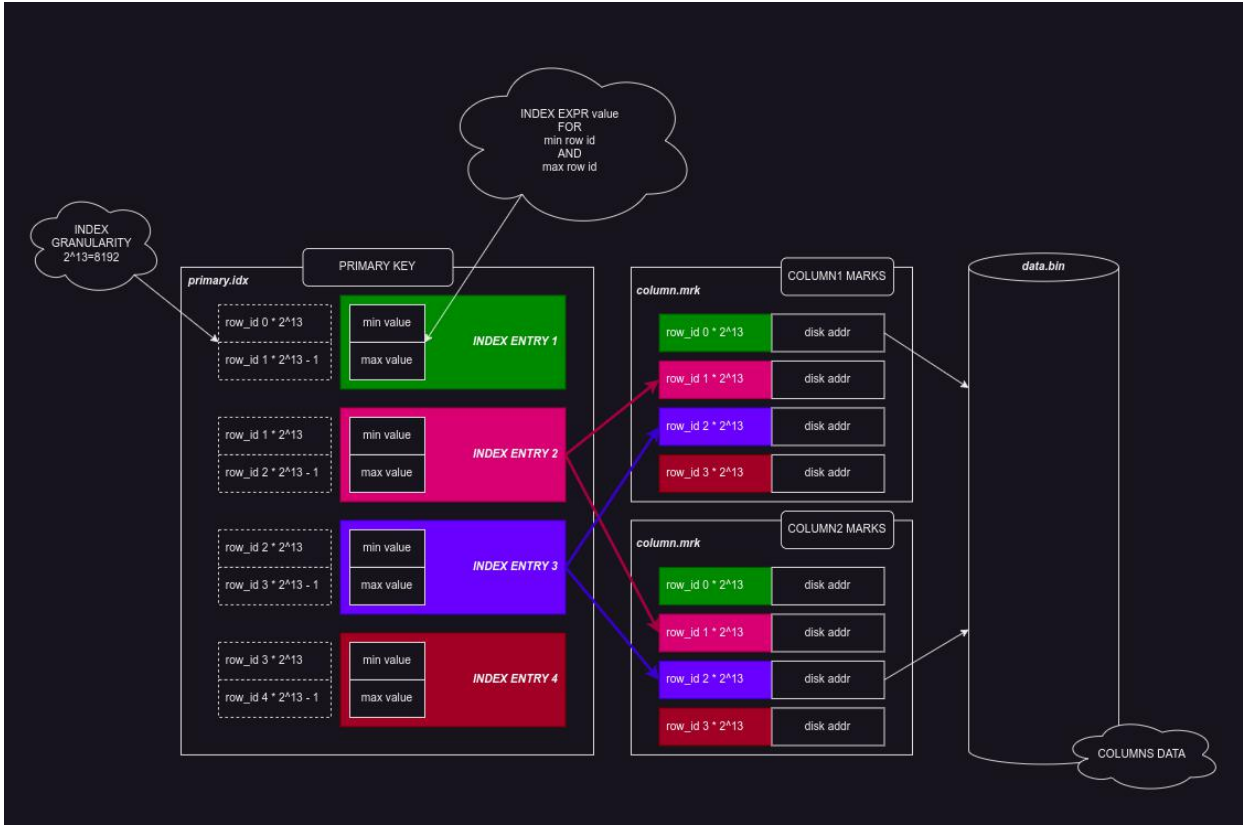
Классический индекс в таком сценарии, будет очень расточителен по оперативной памяти, и показывать низкую производительность из за необходимости пробегать весь ключ, и даже бинарное дерево не спасает.

Вместо этого, разреженный индекс позволяет ClickHouse быстро найти перечень гранул, в которых будут содержаться нужные значения. Не нужные значения будут отсеяны в дальнейших этапах выполнения запроса.

Реализация в ClickHouse

ClickHouse, что типично для аналитики - колоночная база данных. Ключом индекса по прежнему являются min/max, а в качестве значения индекса используется row_id, кроме того, строгий размер гранулы позволяет хранить только её номер, поскольку row_id вычисляются из этого номера.

Для каждой колонки существует набор «засечек», которые хранят уже непосредственно адреса на диске для каждой гранулы в колонке.



Для построения такого индекса, данные при вставке в ClickHouse сортируются по основному ключу. В качестве ключа допускается только последовательный список столбцов ключа, задается как “PRIMARY KEY (column1,column2, ... columnN)”, произвольное выражение не допускается.

На каждую вставку данных рожается собственный набор файлов с индексами и данными, такой набор набор называется part, и эти part в фоном режиме объединяются между собой для непрерывной оптимизации поступающих данных. В процессе объединения основной ключ пересчитывается заново. Алгоритм называется MergeTree, и подробно разобран в основном курсе.

Разряженный индекс позволяет эффективно работать с большими объемами данных, что и стало главной причиной его выбора в качестве основного в ClickHouse.

Однако, такой индекс не позволяет покрыть потребности в поиске регулярными выражениями, и не эффективен в проверках на вхождение, для этого есть вторичные индексы.

Пример

```
CREATE TABLE regular_table_with_primary_index
(
    `index_column1` Date MATERIALIZED toDate(index_column2),
    `index_column2` DateTime,
    `sampling_key` MATERIALIZED toUInt8(index_column2 % 60),
    `data1` String,
    `data2` UInt64
)
ENGINE = MergeTree
PARTITION BY index_column1
PRIMARY KEY (index_column1, index_column2, sampling_key)
ORDER BY (index_column1, index_column2, sampling_key)
SAMPLE BY sampling_key
```

PRIMARY KEY - первичный ключ, заданный колонками index_column1, index_column2, sampling_key

Вопросы?

Задаем
вопросы в чат

Ставим “–”,
если вопросов нет

Вторичные индексы

Вторичные индексы в ClickHouse

Вторичные индексы можно добавлять на существующую таблицу. Это позволяет повысить скорость выборки данных.

ClickHouse поддерживает вторичные индексы:

- MinMax, как вторичный индекс, индекс того же вида как и основной ключ
- set, простой и быстрый индекс на все уникальные значения, с ограничением на кардинальность
- bloom filter индексы, когда ограничения на кардинальность не приемлимы
  - bloom_filter - классический фильтр Блума
  - ngrambf_v1 - фильтр Блума по разбитым на N-частей ключам
  - tokenbf_v1 - тоже самое, но разбиение по не-букво-численным символам

Общий синтакис для добавления индекса:

ALTER TABLE [db.]table_name [ON CLUSTER cluster] ADD INDEX [IF NOT EXISTS] name expression TYPE type [GRANULARITY value] [FIRST|AFTER name]

Так же можно удалить индекс - ALTER TABLE ... DROP INDEX или очистить индекс ALTER TABLE ... CLEAR INDEX.

После добавления вторичный индекс начинает собираться для вновь поступающих данных. Для уже существующих необходимо сделать ALTER TABLE ... MATERIALIZE INDEX

MinMax как вторичный индекс

Наряду с использованием такого типа индекса в качестве основного ключа, можно создавать и вторичные индексы такого же типа.

От основного индекса отличается:

- можно использовать произвольное выражение

- не включает сортировку данных по ключу выражения.

Дополнить ключ сортировки можно, и рекомендуется, когда ключ аналогичного основному вида (column1,column2 ...).

Ключ сортировки меняется отдельной операцией изменения ключа сортировки «ALTER TABLE table MODIFY ORDER BY», можно выполнять только одновременно с добавлением самого столбца.

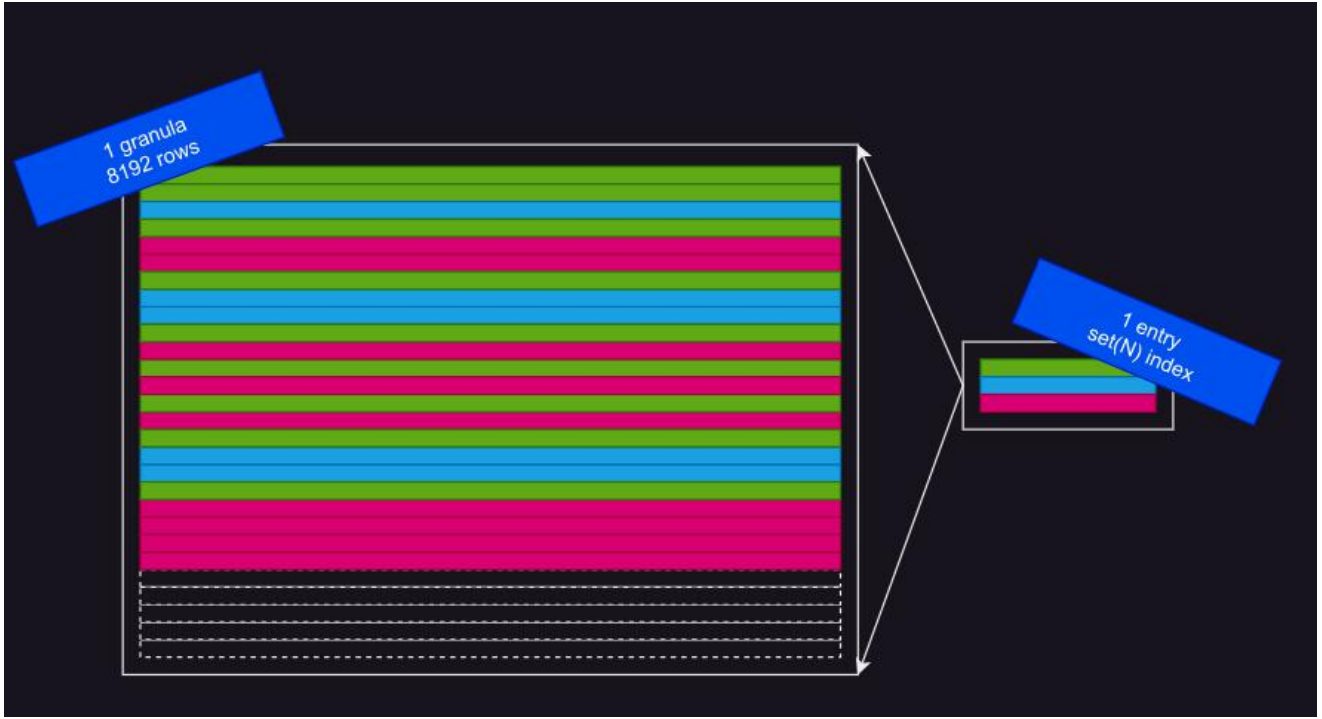
MinMax вторичный - пример

```
60ecb7566bef :) RENAME TABLE regular_table_with_primary_index TO regular_table_with_primary_index_and_secondary_index;

RENAME TABLE regular_table_with_primary_index TO regular_table_with_primary_index_and_secondary_index

Query id: 3513034f-78bb-41a7-bdd7-ce3b78094a31

Ok.

0 rows in set. Elapsed: 0.003 sec.

60ecb7566bef :) ALTER TABLE regular_table_with_primary_index_and_secondary_index ADD column project_id UInt16, MODIFY ORDER BY (index_column1, index_column2, sampling_key, project_id);

ALTER TABLE regular_table_with_primary_index_and_secondary_index
    (ADD COLUMN `project_id` UInt16),
    (MODIFY ORDER BY (index_column1, index_column2, sampling_key, project_id))

Query id: b6353628-8467-4a00-8373-fc827388423b

Ok.

0 rows in set. Elapsed: 0.017 sec.

60ecb7566bef :) ALTER TABLE regular_table_with_primary_index_and_secondary_index ADD INDEX minmax_project_id project_id TYPE minmax GRANULARITY 1;

ALTER TABLE regular_table_with_primary_index_and_secondary_index
    (ADD INDEX minmax_project_id project_id TYPE minmax GRANULARITY 1)

Query id: 0502f874-fbd8-42c3-8648-ed62f10274ab

Ok.

0 rows in set. Elapsed: 0.015 sec.

60ecb7566bef :) ALTER TABLE regular_table_with_primary_index_and_secondary_index MATERIALIZE INDEX minmax_project_id;

ALTER TABLE regular_table_with_primary_index_and_secondary_index
    (MATERIALIZE INDEX minmax_project_id)

Query id: fdf2b3fb-f79c-4de9-bd3c-fb7e736ee18b

Ok.
```

set(N)

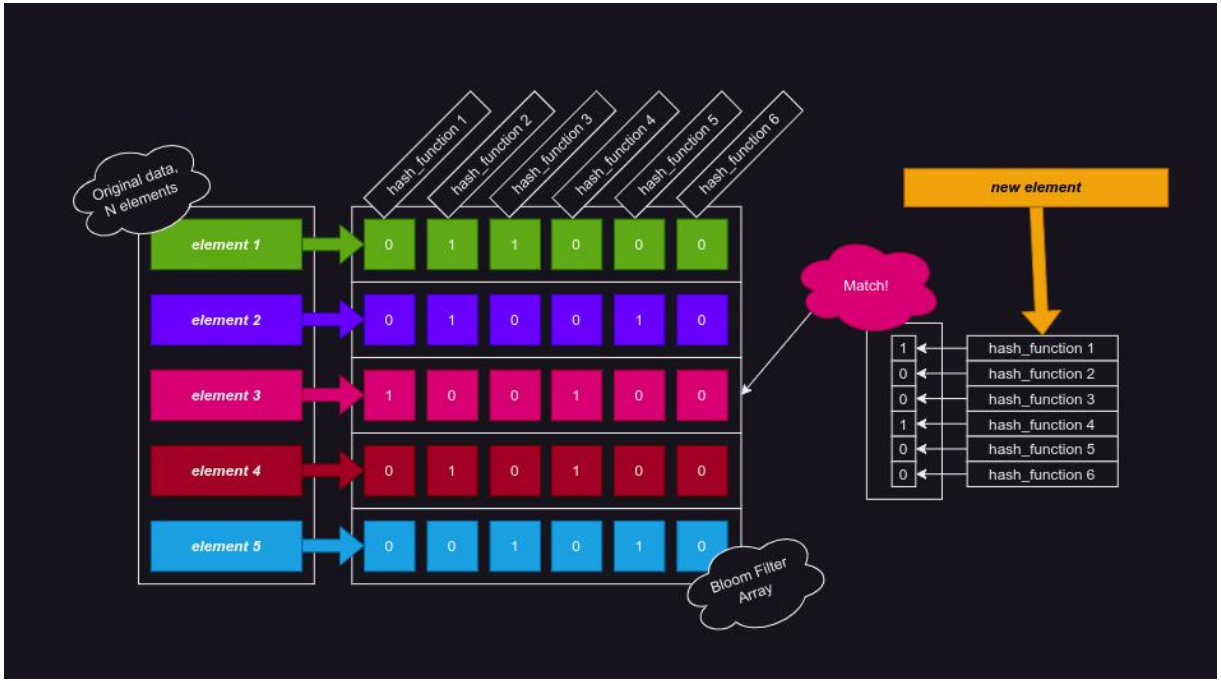
Простой индекс, хранит заданный N уникальных ключей на гранулу.

При превышении N количества уникальных ключей в индексе на гранулу, гранула остается неиндексированной.



Фильтр Блума

Метод получения битового массива из результатов хеширования множества ключей, и последующее использование этого массива для проверки ключей на вхождение в множество.

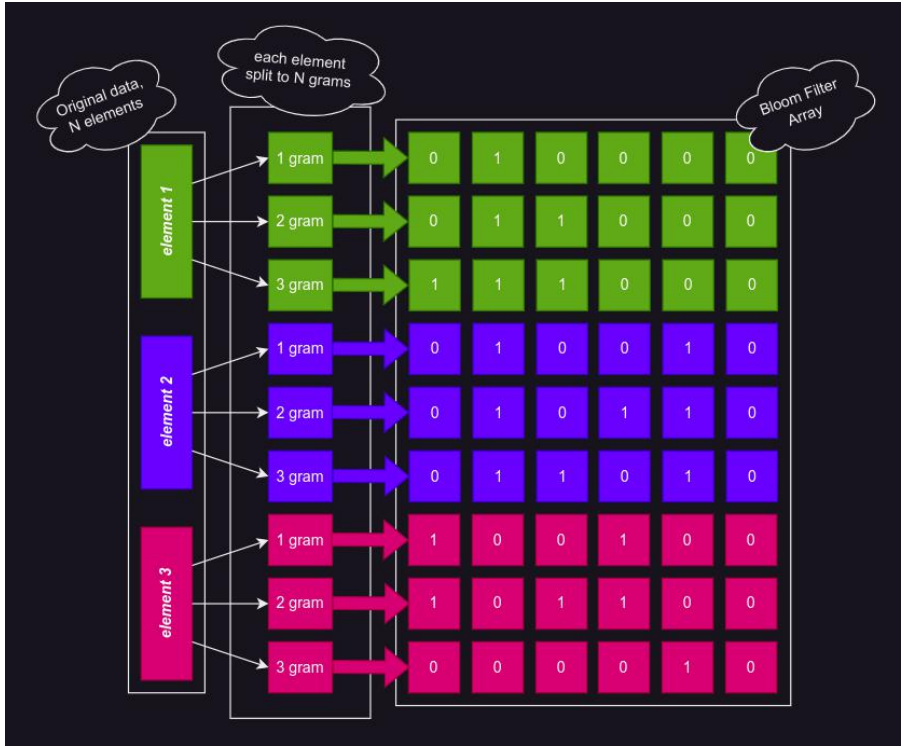


Хешируется множество ключей. Сохраняется битовый массив. При необходимости проверить ключ на вхождение, он подвергается аналогичному хешированию, и битовый массив накладывается на результат хеширования теми же функциями. Если наложение дает совпадение по всем хеширующим функциям хотя бы для одного элемента массива, элемент может принадлежать множеству. Возможны ложно-положительные срабатывания, ложно-отрицательные исключены.

ngrambf_v1

Каждый элемент делится на N частей, именуемых «gram», фиксированного размера.

Массив для фильтра Блума собирается из таких «gram», вместо оригинальных элементов.

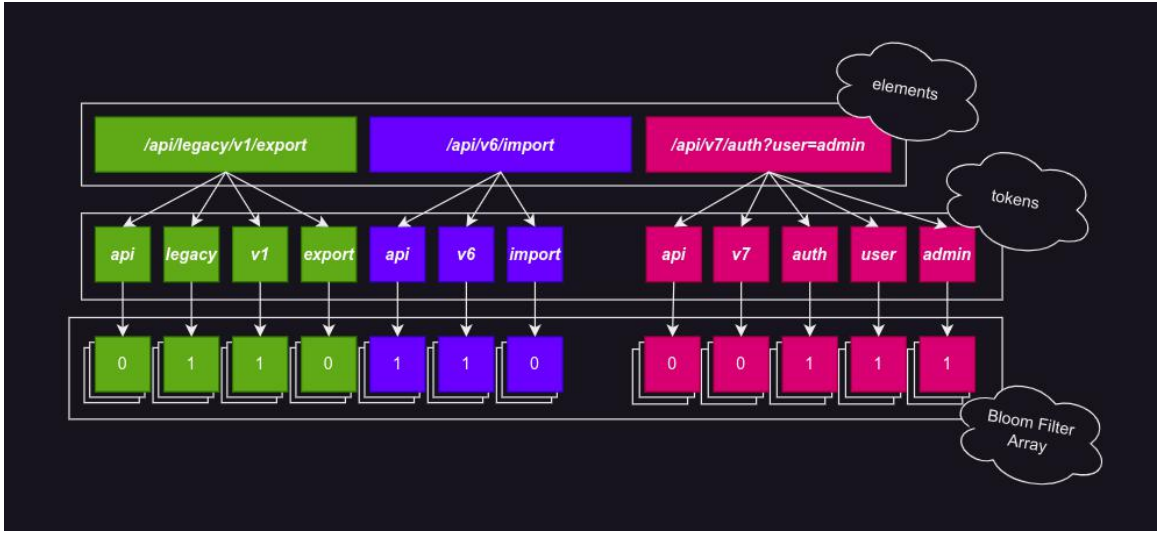


tokenbf_v1

Каждый элемент делится на N токенов.

Токеном является непрерывная цифро-буквенная последовательность, например строка «/api/v1/import» будет содержать токены:

- api
- v1
- import



Массив для фильтра Блума собирается из токенов. В отличие от ngrambf_v1, размер токена не константный

Вопросы?

Задаем
вопросы в чат

Ставим “–”,
если вопросов нет

Использование индексов

Для чего подходят индексы?

MinMax хорошо подходит для поиска данных по:

- прямому совпадению (column == ‘value’)
- совпадению по маске (column LIKE ‘%some%pattern%’)
- лексиграфическому и алгебраическим сравнениям (<, > , <=, >=, between)

Bloom Filter позволяет:

- эффективнее чем MinMax проверять на вхождение (column in (‘value1’,’value2’), has(array, column))

ngrambf_v1 расширяет возможности Bloom Filter полнотекстовым поиском

tokenbf_v1 тоже самое, более эффективен например для URL-ов и других данных со структурированным логическим разделением не-цифро-буквенными символами

важное отличие ngrambf_v1 и tokenbf_v1 от MinMax: позволяют использовать регулярные выражения

Set(N) сочетает возможности всех индексов, но только на низкокардинальных данных, с не более чем N уникальных ключей.

Получение информации о доступных индексах

Проверьте доступные вам индексы в таблице

```
/* основной ключ */
SELECT primary_key
FROM system.tables
WHERE table = 'my_table'
```

```
┌─primary_key──────────────────┐
│ event_date, event_time       │
└──────────────────────────────┘
```

```
/* вторичные индексы */
SELECT
    type_full,
    expr
FROM system.data_skipping_indices
WHERE table = 'my_table'
```

```
┌─type_full─────────────────────────┬─expr──────────────┐
│ set(100)                          │ client            │
│ tokenbf_v1(10240, 3, 0)           │ lowerUTF8(api_url)│
│ ngrambf_v1(1024, 5, 0, 0)         │ json_payload      │
└───────────────────────────────────┴───────────────────┘
```

Так же можно получить специальным запросом «SHOW INDEXES FROM table» или из «SHOW CREATE TABLE»

Выявление потребности

Изучаем секцию WHERE нашего запроса:

1. получаем информцию об индексах как на предыдущем слайде
2. выявляем поля, не присутсвующими в основном ключе, или обрабатываемые регулярными выражениям
3. проверяем покрытие полей вторичными индексами
4. добавляем вторичные индексы

Пример:

SELECT .... FROM table WHERE match(json_payload,’ ... validation_regex ... ’)

ngrambf_v1 индекс на поле json_payload отсутствует

требуется добавить такой индекс

Резюме

Специальные индексы в колоночных БД отличаются от классического индекса, направленностью на анализ большого количества данных наиболее эффективным способом, работая не с точными значениями ключевого столбца, а с диапазонами Min/Max на N строк.

Вторичные индексы покрывают дополнительную потребность в полнотекстовом поиске, и более дешевой проверки на вхождение, алгоритмом фильтра Блума.

Вопросы?

Задаем
вопросы в чат

Ставим “–”,
если вопросов нет

Рефлексия

Цели вебинара

Проверка достижения целей

1. Ответить на вопрос что такое индекс
2. Корректно задавать индексы при разработке и сопровождении схем БД
3. Выявлять потребность во вторичных индексах

Вопросы для проверки

Вопросы для проверки

1.	Что такое индекс
2.	Какой вторичный индекс выбрать для низкокардинальных данных
3.	Приведите пример WHERE ... секции запроса, для которой недостаточно PRIMARY INDEX

Рефлексия

С какими впечатлениями уходите с вебинара?

Как будете применять на практике то,
что узнали на вебинаре?

Следующий вебинар

28 октября 2024 (понедельник)

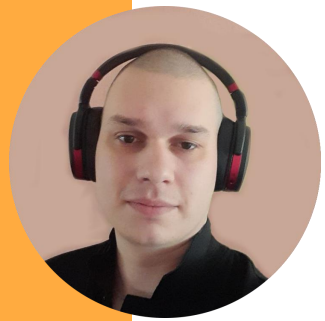
Другие движки

- Ссылка на вебинар будет в ЛК за 15 минут
- Материалы к занятию в ЛК — можно изучать
- Обязательный материал обозначен красной лентой

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Спасибо за внимание!

Приходите на следующие вебинары



Senior SRE / ClickHouse DBA в VK

Занимаюсь эксплуатацией ClickHouse с первых версий: 5 лет в VK, до этого в AdNow, до этого занимался Vertica. Сотни серверов, десятки кластеров, десятки петабайт данных.