
Яндекс

Яндекс

Адаптивная гранулярность индекса MergeTree

Александр Сапин, Разработчик

MergeTree

Семейство движков MergeTree

Преимущества:

- Вставки и чтения не блокируют друг друга
- Атомарная вставка
- Первичные и вторичные индексы
- Партиционирование
- `ALTER`, репликация, семплирование, `TTL` и т.д.

Семейство движков MergeTree

Преимущества:

- Вставки и чтения не блокируют друг друга
- Атомарная вставка
- Первичные и вторичные индексы
- Партиционирование
- `ALTER`, репликация, семплирование, `TTL` и т.д.

Особенности:

- Необходимы редкие вставки (батчинг)
- Фоновые операции над записями с одинаковыми ключами
- Нет уникальности первичного ключа

Создание и заполнение таблицы

Создадим таблицу:

```
CREATE TABLE mt (
    EventDate Date,
    OrderID Int32,
    BannerID UInt64,
    GoalNum Int8
) ENGINE = MergeTree()
PARTITION BY toYYYYMM(EventDate) ORDER BY (OrderID, BannerID)
```

Создание и заполнение таблицы

Создадим таблицу:

```
CREATE TABLE mt (
    EventDate Date,
    OrderID Int32,
    BannerID UInt64,
    GoalNum Int8
) ENGINE = MergeTree()
PARTITION BY toYYYYMM(EventDate) ORDER BY (OrderID, BannerID)
```

Заполним данными (дважды):

```
INSERT INTO mt SELECT toDate('2018-09-26'),
    number, number + 10000, number % 128 from numbers(1000000);
INSERT INTO mt SELECT toDate('2018-10-15'),
    number, number + 10000, number % 128 from numbers(1000000, 1000000);
```

Таблица на диске

metadata:

```
$ ls /var/lib/clickhouse/metadata/default/
mt.sql
```

Таблица на диске

metadata:

```
$ ls /var/lib/clickhouse/metadata/default/
mt.sql
```

data:

```
$ ls /var/lib/clickhouse/data/default/mt
201809_2_2_0  201809_3_3_0  201810_1_1_0  201810_4_4_0 201810_1_4_1
detached  format_version.txt
```

Таблица на диске

metadata:

```
$ ls /var/lib/clickhouse/metadata/default/
mt.sql
```

data:

```
$ ls /var/lib/clickhouse/data/default/mt
201809_2_2_0  201809_3_3_0  201810_1_1_0  201810_4_4_0 201810_1_4_1
detached  format_version.txt
```

Содержимое:

- Файл с форматом `format_version.txt`
- Директории с кусками (parts)
- Директории для отложенных (detached) кусков

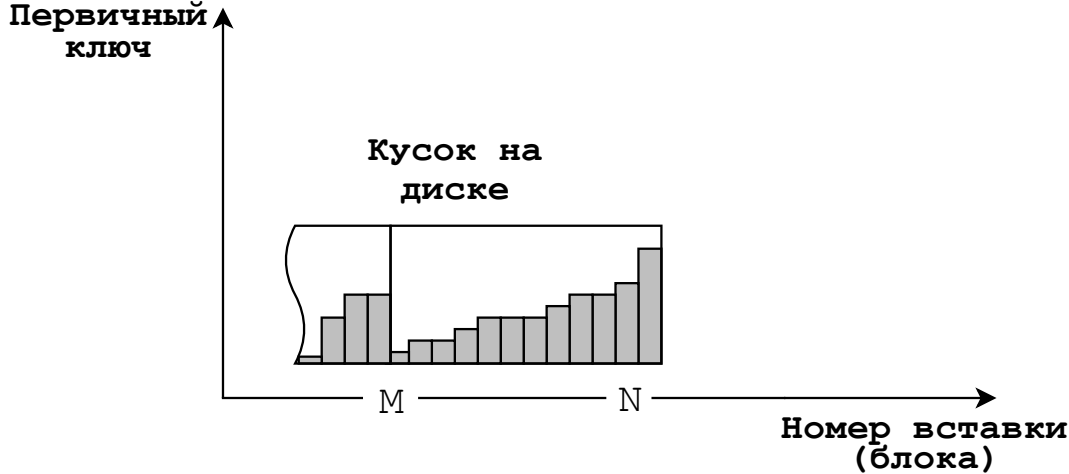
Куски: общее

- Часть данных, упорядоченных в порядке первичного ключа
- Содержат интервал блоков
- Создаются на каждую вставку и слияние
- Содержат метаинформацию в имени директории

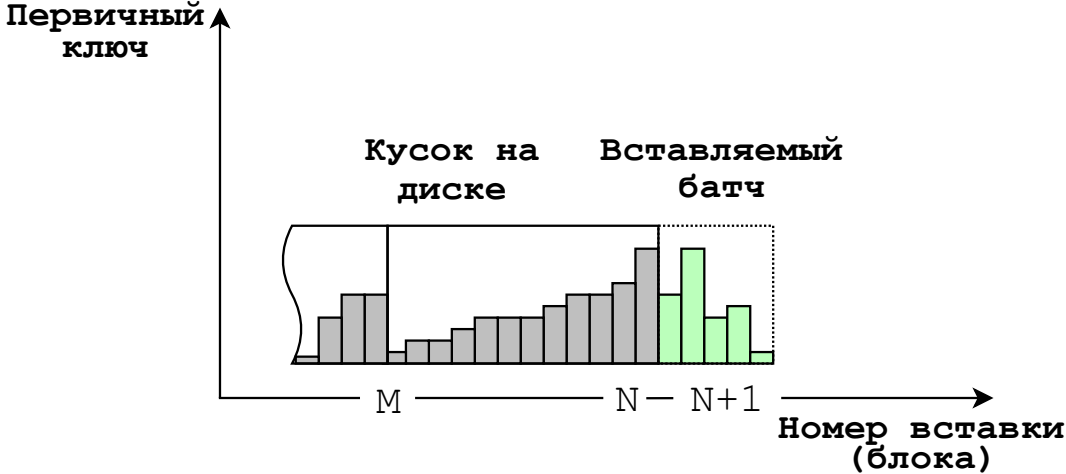
Куски: зачем

- Для эффективного выполнения range-запросов нужна упорядоченность по первичному ключу
  - В нашем случае по `(OrderID, BannerID)`
- Но данные поступают упорядоченными по времени
  - По `EventDate`
- Храним данные в виде небольшого набора упорядоченных кусков

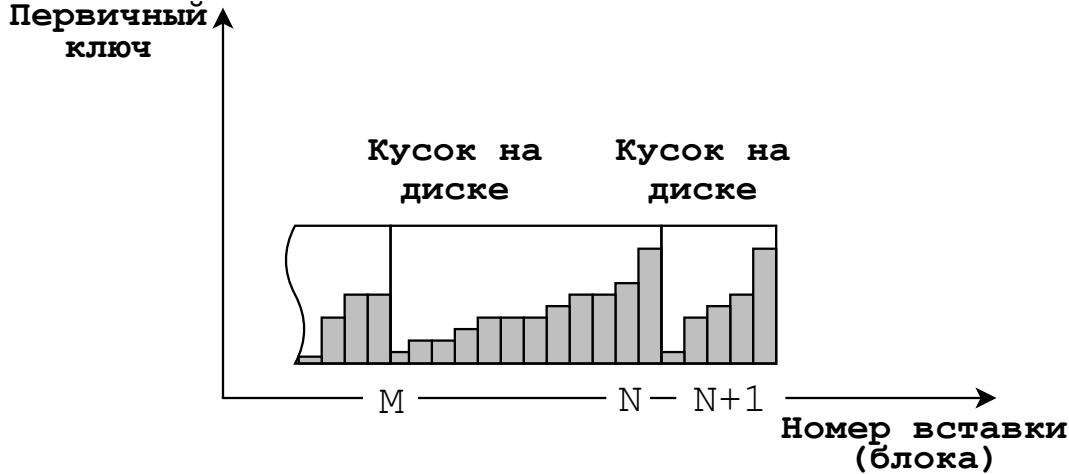
Куски: основная идея



Куски: основная идея



Куски: основная идея



Куски: метаданные

ID куска: `201810_1_4_1`

```
struct MergeTreePartInfo
{
    String partition_id = "201810";
    Int64 min_block = 1;
    Int64 max_block = 4;
    UInt32 level = 1;

    ...
};
```

Куски: данные

```
$ ls /var/lib/clickhouse/data/default/mt/201810_1_4_1
GoalNum.bin GoalNum.mrk BannerID.bin ...
primary.idx checksums.txt count.txt columns.txt
partition.dat minmax_EventDate.idx
```

Куски: данные

```
$ ls /var/lib/clickhouse/data/default/mt/201810_1_4_1
GoalNum.bin GoalNum.mrk BannerID.bin ...
primary.idx checksums.txt count.txt columns.txt
partition.dat minmax_EventDate.idx
```

Содержимое:

- `GoalNum.bin` – сжатые файлы с данными колонок
- `GoalNum.mrk` – засечки для сжатых данных

Куски: данные

```
$ ls /var/lib/clickhouse/data/default/mt/201810_1_4_1
GoalNum.bin GoalNum.mrk BannerID.bin ...
primary.idx checksums.txt count.txt columns.txt
partition.dat minmax_EventDate.idx
```

Содержимое:

- `GoalNum.bin` – сжатые файлы с данными колонок
- `GoalNum.mrk` – засечки для сжатых данных
- `primary.idx` – первичный ключ

Куски: данные

```
$ ls /var/lib/clickhouse/data/default/mt/201810_1_4_1
GoalNum.bin GoalNum.mrk BannerID.bin ...
primary.idx checksums.txt count.txt columns.txt
partition.dat minmax_EventDate.idx
```

Содержимое:

- `GoalNum.bin` – сжатые файлы с данными колонок
- `GoalNum.mrk` – засечки для сжатых данных
- `primary.idx` – первичный ключ
- `checksums.txt` – контрольные суммы всех файлов

Куски: данные

```
$ ls /var/lib/clickhouse/data/default/mt/201810_1_4_1
GoalNum.bin GoalNum.mrk BannerID.bin ...
primary.idx checksums.txt count.txt columns.txt
partition.dat minmax_EventDate.idx
```

Содержимое:

- `GoalNum.bin` – сжатые файлы с данными колонок
- `GoalNum.mrk` – засечки для сжатых данных
- `primary.idx` – первичный ключ
- `checksums.txt` – контрольные суммы всех файлов
- `partition.dat` – id партиции
- `minmax_EventDate.idx` – минимакс колонок ключа партиционирования

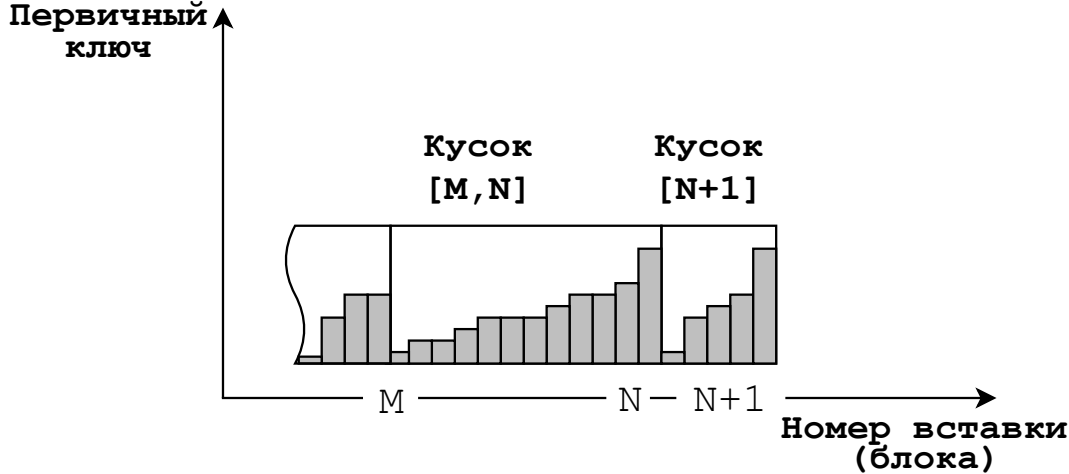
Куски: данные

```
$ ls /var/lib/clickhouse/data/default/mt/201810_1_4_1
GoalNum.bin GoalNum.mrk BannerID.bin ...
primary.idx checksums.txt count.txt columns.txt
partition.dat minmax_EventDate.idx
```

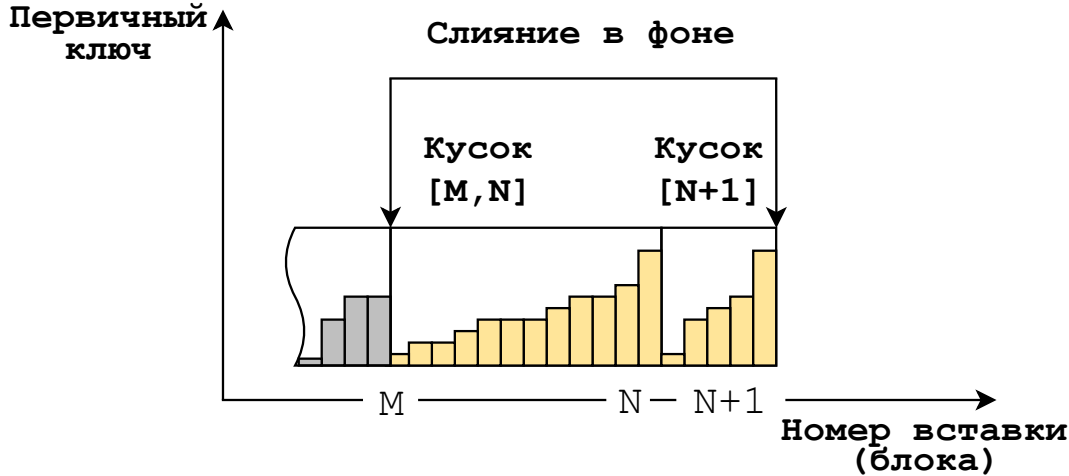
Содержимое:

- `GoalNum.bin` – сжатые файлы с данными колонок
- `GoalNum.mrk` – засечки для сжатых данных
- `primary.idx` – первичный ключ
- `checksums.txt` – контрольные суммы всех файлов
- `partition.dat` – id партиции
- `minmax_EventDate.idx` – минимакс колонок ключа партиционирования
- `columns.txt` – информация о колонках
- `count.txt` – число строк в куске

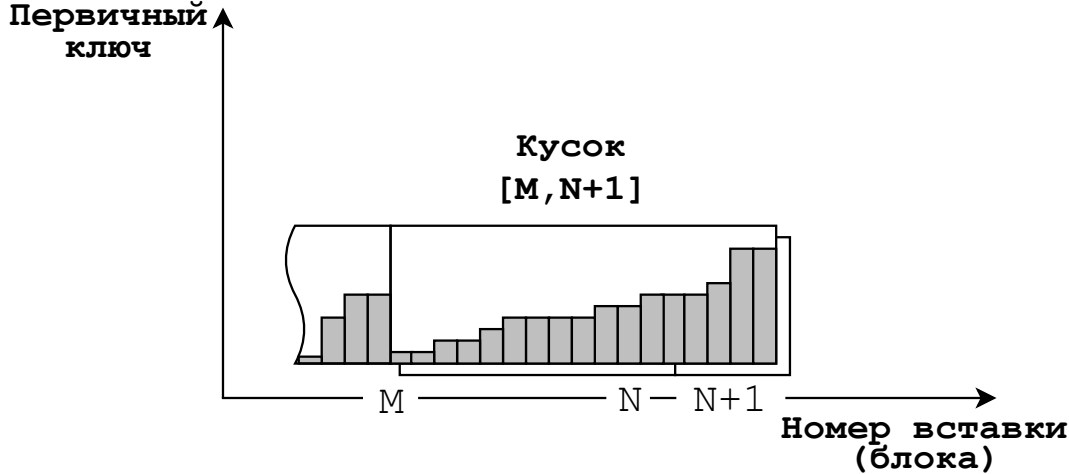
Слияние



Слияние


Первичный ключ
Слияние в фоне
Кусок [M,N]
Кусок [N+1]
M
N
N+1
Номер вставки (блока)

Слияние



Слияние: алгоритм

Горизонтальное:

- Читаем строки в порядке сортировки из множества кусков
- Поколоночно пишем на диск

Слияние: алгоритм

Горизонтальное:

- Читаем строки в порядке сортировки из множества кусков
- Поколоночно пишем на диск

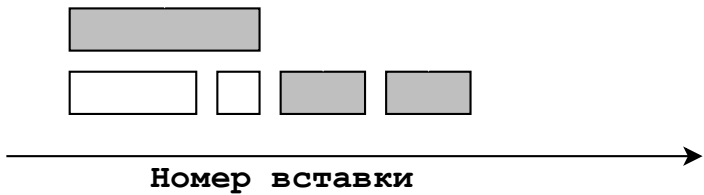
Вертикальное:

- Читаем только колонки первичного ключа
- Сохраняем порядок записи строк
- Поколоночно пишем первичный ключ на диск
- Пишем остальные колонки в соответствии с порядком

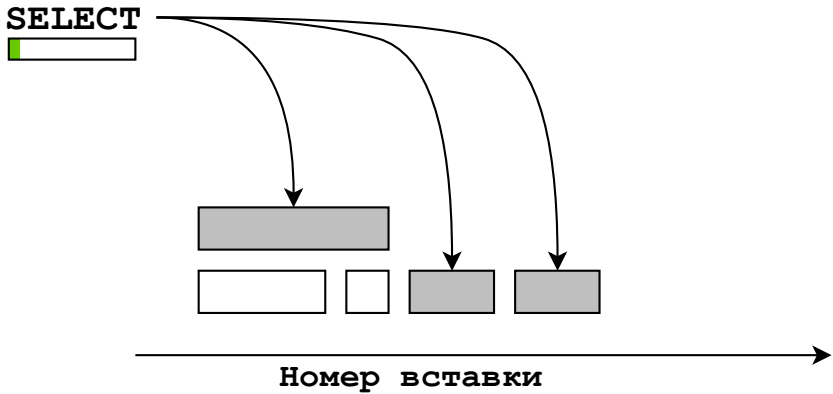
Слияние: свойства

- Один кусок участвует только в 1 успешном слиянии
- Куски из разных партиций никогда не сливаются
- Исходные куски становятся **неактивными**

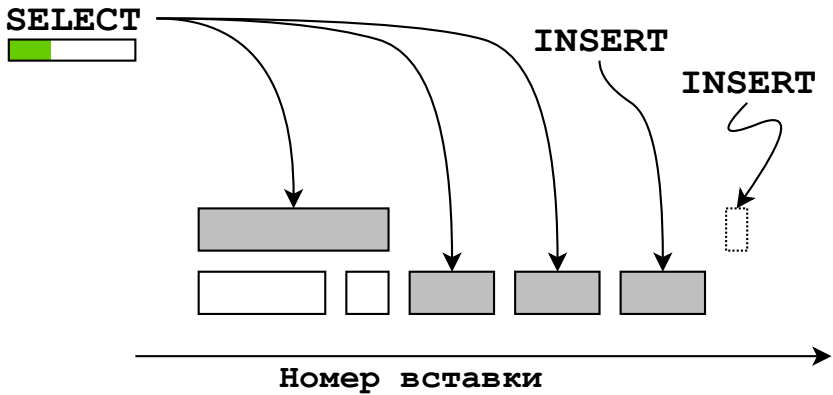
Жизненный цикл кусков



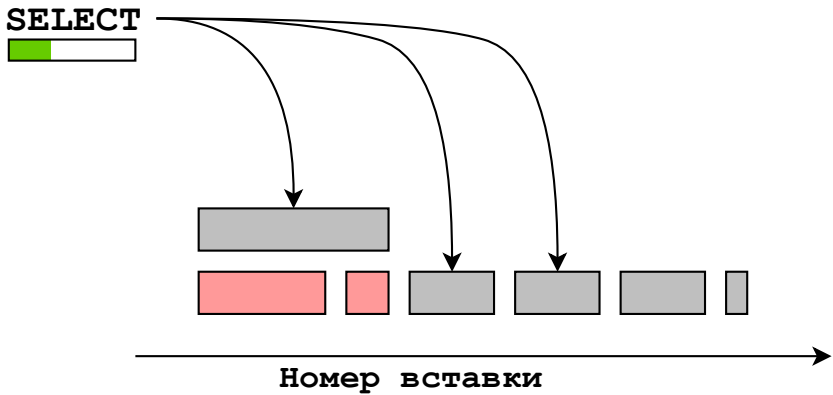
Жизненный цикл кусков



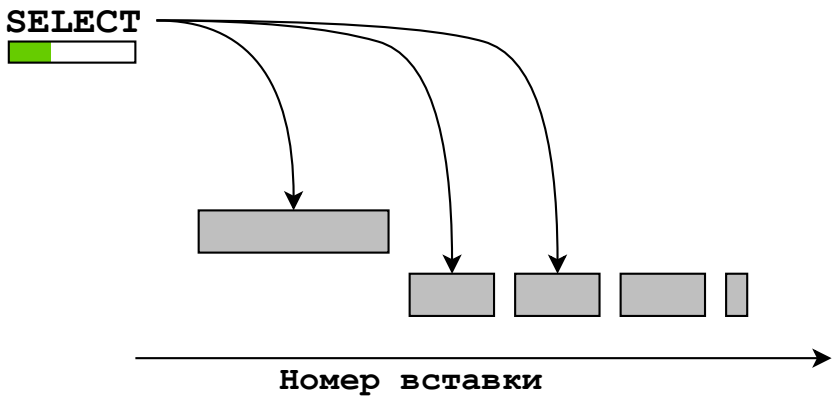
Жизненный цикл кусков



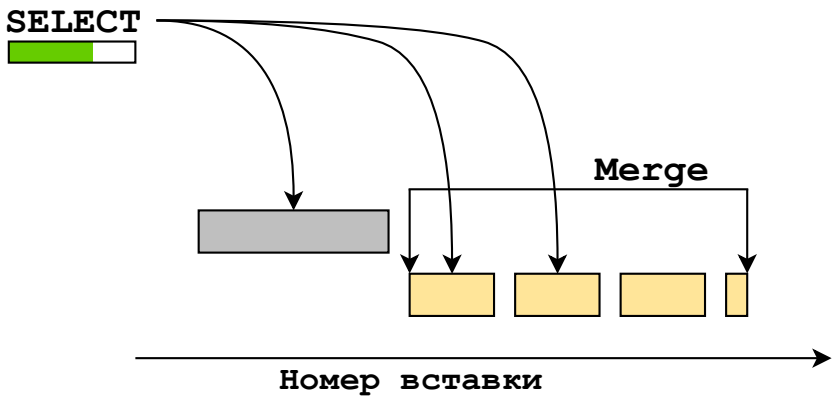
Жизненный цикл кусков



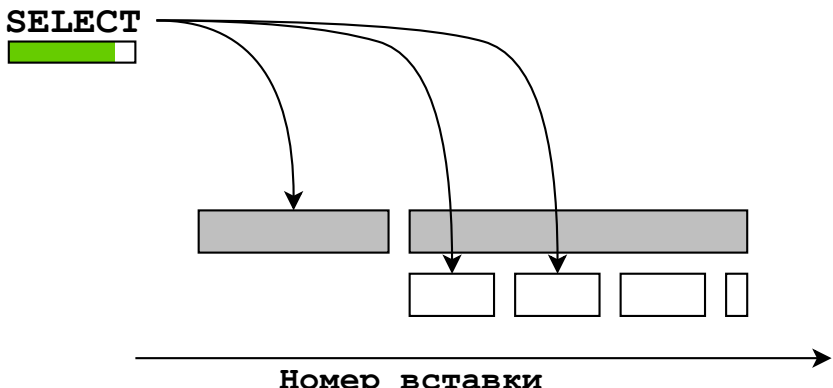
Жизненный цикл кусков



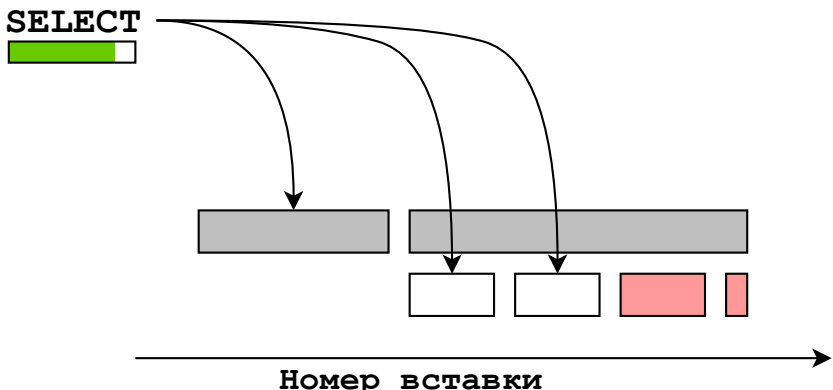
Жизненный цикл кусков



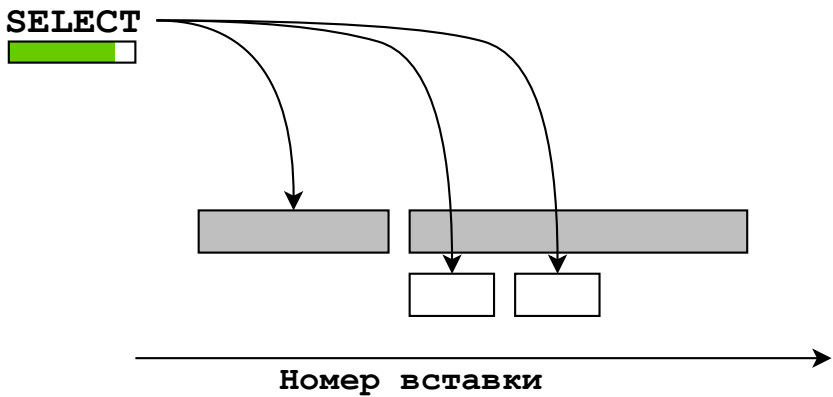
Жизненный цикл кусков



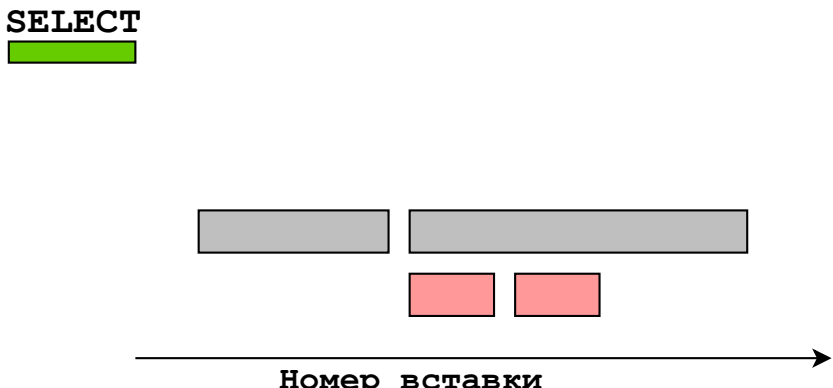
Жизненный цикл кусков



Жизненный цикл кусков



Жизненный цикл кусков



Жизненный цикл кусков

Чтение и индекс

Индекс в MergeTree

Ключ сортировки:

- Задается с помощью `ORDER BY`
- По умолчанию является первичным ключом
- Определяет порядок данных на диске
- Выступает ключом для слияний

Индекс в MergeTree

Ключ сортировки:

- Задается с помощью `ORDER BY`
- По умолчанию является первичным ключом
- Определяет порядок данных на диске
- Выступает ключом для слияний

Первичный ключ:

- Задается с помощью `PRIMARY KEY`
- Хранится в файле `primary.idx`
- **Разреженный** (`index_granularity=8192`)
- Всегда находится в памяти

Индекс на диске

Читаем индекс:

```
$ od -i -j 0 -N 4 primary.idx
  0 # OrderID (0)
$ od -l -j 4 -N 8 primary.idx
  10000 # BannerID (0)
$ od -i -j 12 -N 4 primary.idx
  8192 # OrderID (1)
$ od -l -j 16 -N 8 primary.idx
  18192 # BannerID (1)
```

Индекс на диске

Читаем индекс:

```
$ od -i -j 0 -N 4 primary.idx
  0 # OrderID (0)
$ od -l -j 4 -N 8 primary.idx
  10000 # BannerID (0)
$ od -i -j 12 -N 4 primary.idx
  8192 # OrderID (1)
$ od -l -j 16 -N 8 primary.idx
  18192 # BannerID (1)
```

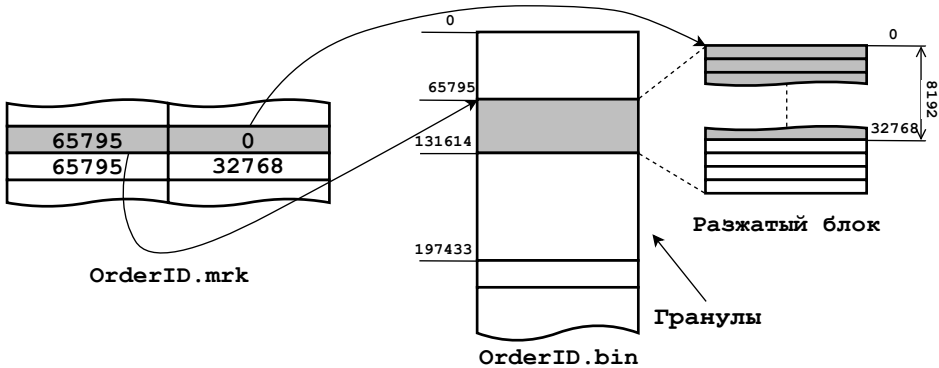
Как читать сжатые колонки?

Засечки

- Засечка – смещение в сжатом файле и в разжатом блоке (16 байт)
- По одной на каждую строку индекса
- Хранятся в файлах `ColumnName.mrk` и в кэше засечек

Засечки

- Засечка – смещение в сжатом файле и в разжатом блоке (16 байт)
- По одной на каждую строку индекса
- Хранятся в файлах `ColumnName.mrk` и в кэше засечек



Засечки на диске

Читаем засечку:

```
$ od -l -j 32 -N 16 OrderID.mrk
  65795 # offset in compressed file
  0 # offset in decompressed block
```

Засечки на диске

Читаем засечку:

```
$ od -l -j 32 -N 16 OrderID.mrk
  65795 # offset in compressed file
  0 # offset in decompressed block
```

Читаем колонку по засечке:

```
$ tail -c +65796 OrderID.bin | clickhouse-compressor -d | od -i -N 20
  16384
  16385
  16386
  16387
  16388
```

Чтение данных с диска

Алгоритм:

- Определяем нужные строки индекса (самое сложное)
- Находим соответствующие засечки
- Распределяем гранулы между потоками
- Читаем выбранные гранулы

Особенности:

- Не можем прочитать меньше гранулы (почти)
- Читаем гранулы параллельно
- Потоки могут воровать задачи

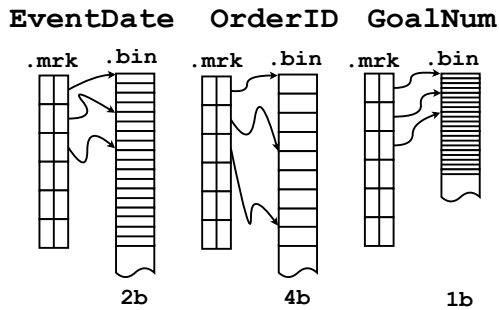
Чтение данных с диска

primary.idx

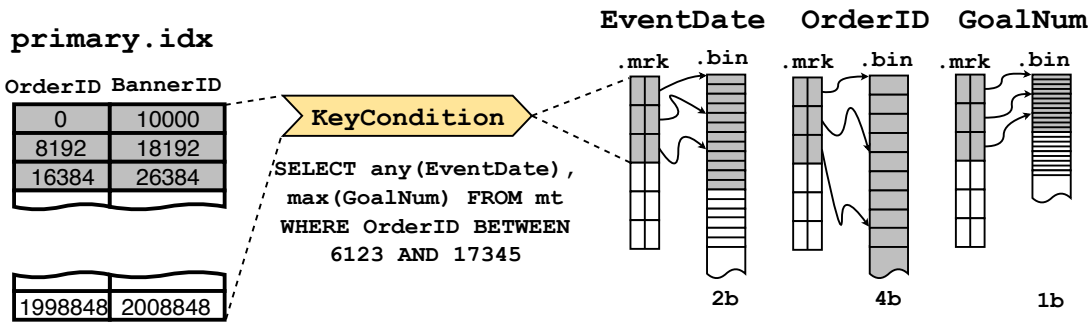
OrderID	BannerID
0	10000
8192	18192
16384	26384
1998848	2008848

KeyCondition

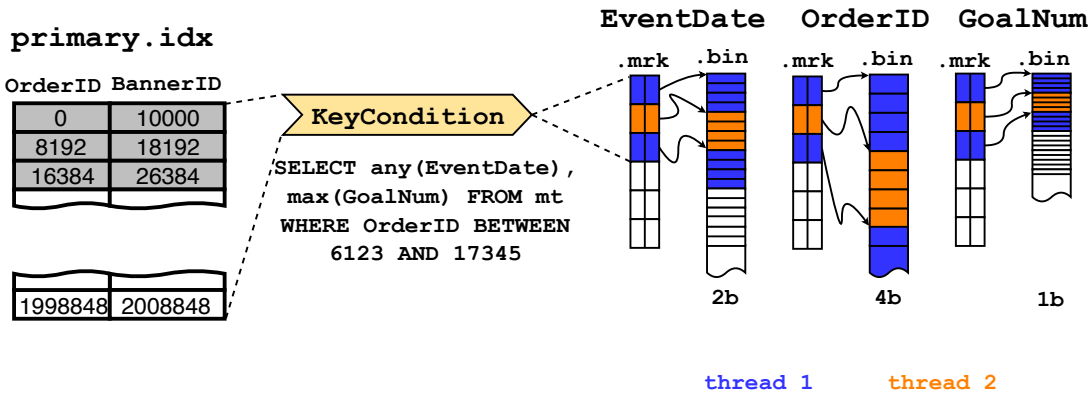
```
SELECT any(EventDate),
max(GoalNum) FROM mt
WHERE OrderID BETWEEN
6123 AND 17345
```



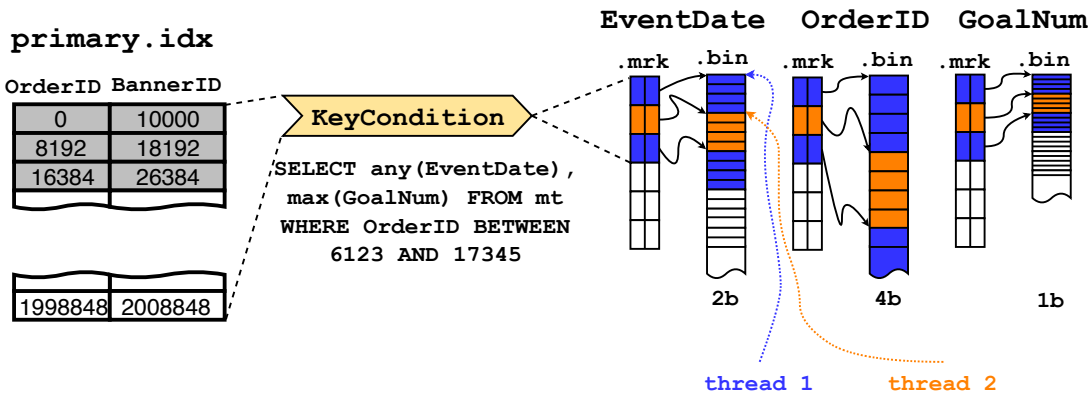
Чтение данных с диска



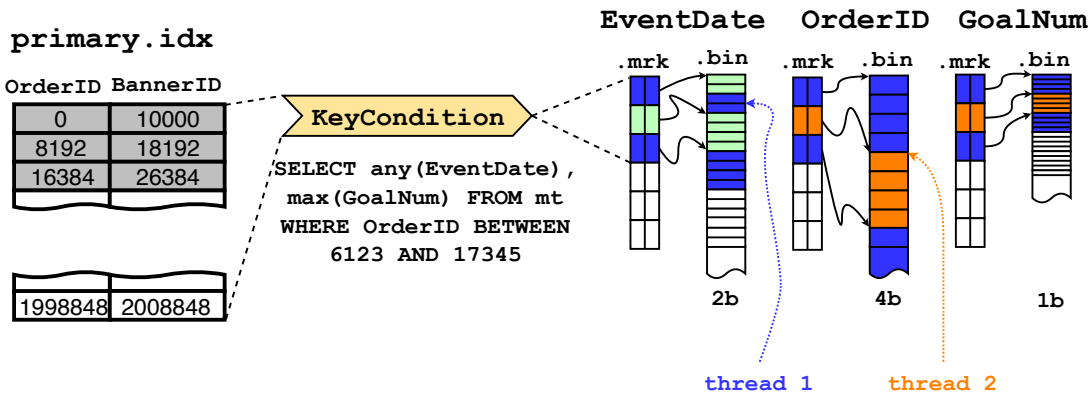
Чтение данных с диска



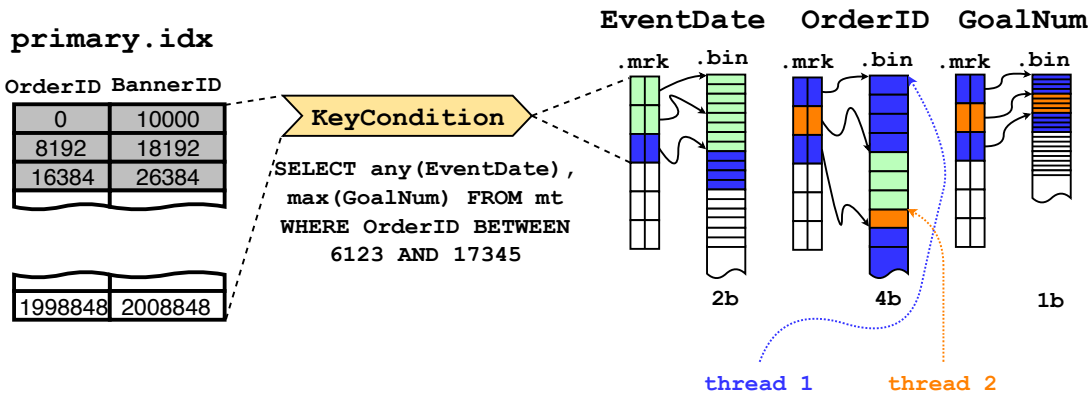
Чтение данных с диска



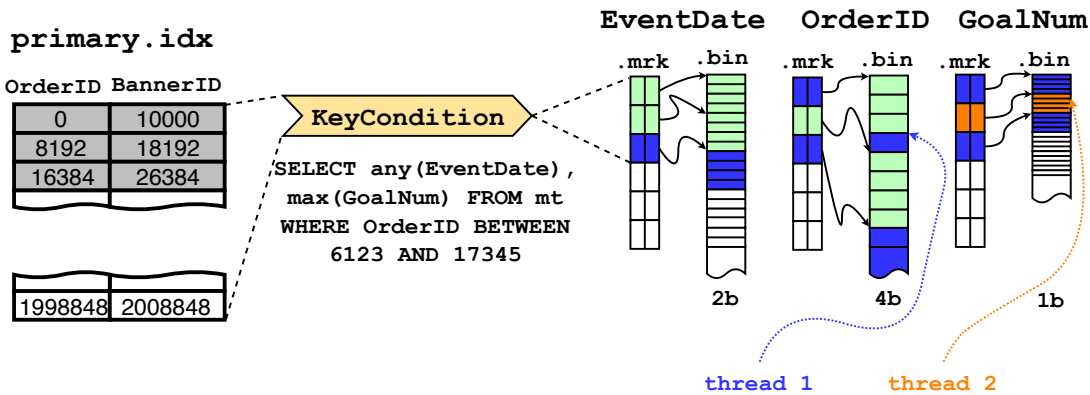
Чтение данных с диска



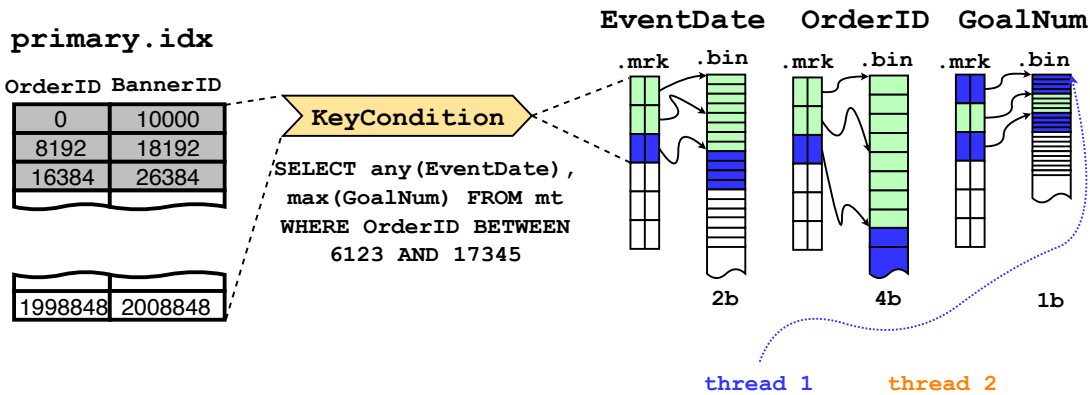
Чтение данных с диска



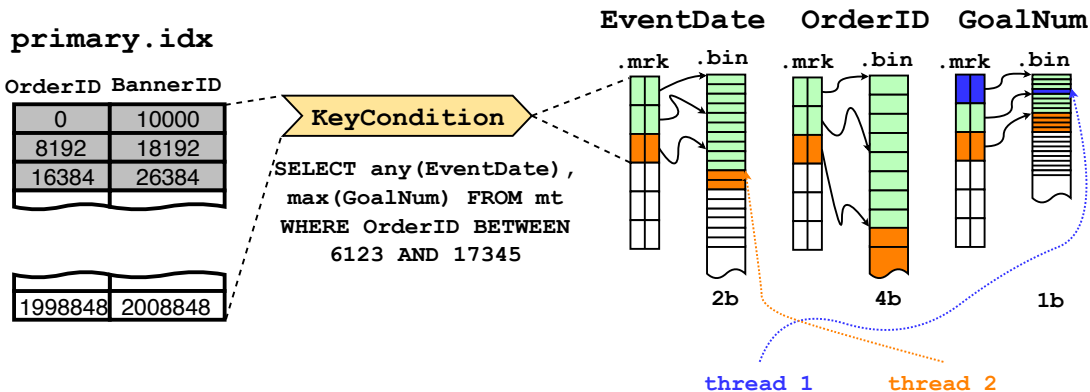
Чтение данных с диска



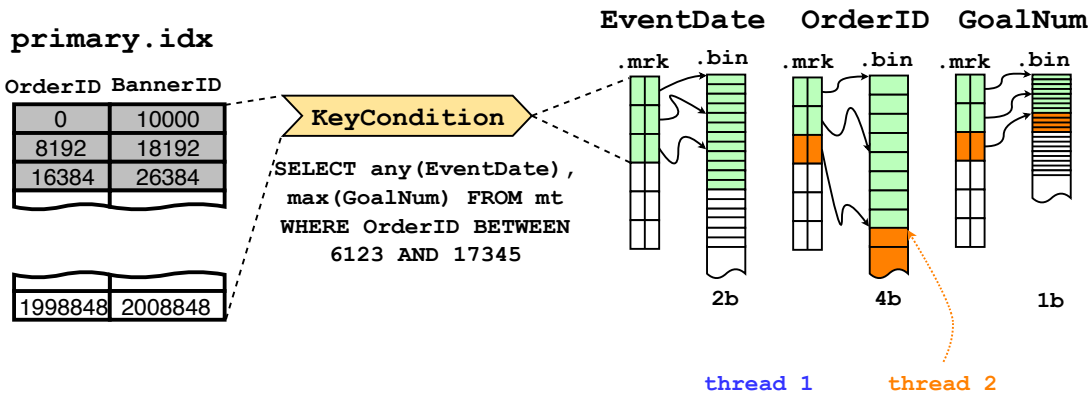
Чтение данных с диска



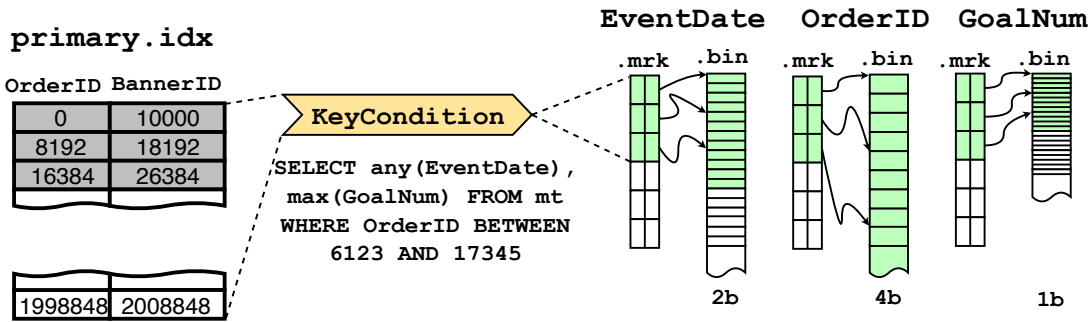
Чтение данных с диска



Чтение данных с диска



Чтение данных с диска



My Little Pony: Friendship is Magic (2010)

Wikipedia в ClickHouse

Создадим таблицу:

```
CREATE TABLE wikidata (
    domain String,
    path String,
    HTML String
) ENGINE=MergeTree() ORDER BY (domain, path);
```

Wikipedia в ClickHouse

Создадим таблицу:

```
CREATE TABLE wikidata (
    domain String,
    path String,
    HTML String
) ENGINE=MergeTree() ORDER BY (domain, path);
```

Заполним данными:

```
INSERT INTO wikidata SELECT *
    FROM file(
        '/var/lib/clickhouse/user_files/wiki100k.tsv',
        'TSV',
        'domain String, path String, HTML String')
```

Wikipedia в ClickHouse

Длина статьи про ClickHouse

```
SELECT max(length(HTML)) FROM wikidata
  WHERE domain='https://en.wikipedia.org' AND path='/wiki/ClickHouse'
58253

1 rows in set. Elapsed: 0.154 sec. (11.08 thousand rows/s., 3.05 MB/s.)
-- MemoryTracker: Peak memory usage (for query): 259.74 MiB.
```

Wikipedia в ClickHouse

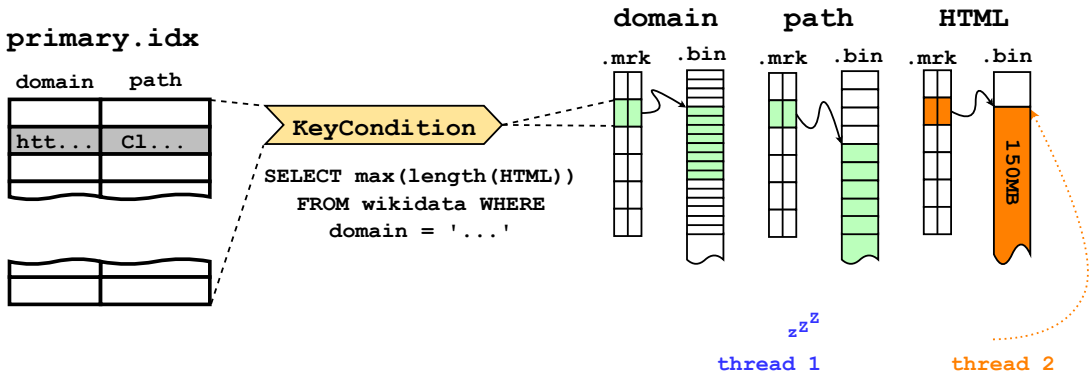
Длина статьи про ClickHouse

```
SELECT max(length(HTML)) FROM wikidata
  WHERE domain='https://en.wikipedia.org' AND path='/wiki/ClickHouse'
58253

1 rows in set. Elapsed: 0.154 sec. (11.08 thousand rows/s., 3.05 MB/s.)
-- MemoryTracker: Peak memory usage (for query): 259.74 MiB.
```

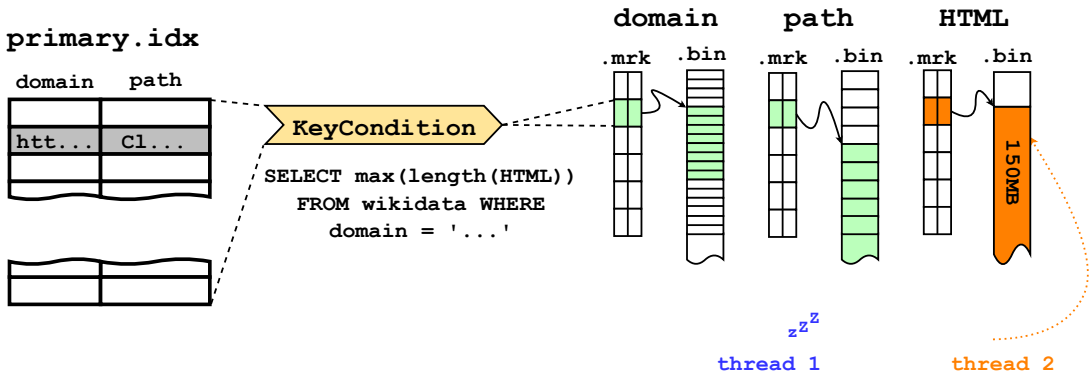
Самая длинная статья

```
SELECT max(length(HTML)) FROM wikidata
2134709

1 rows in set. Elapsed: 9.843 sec. (10.16 thousand rows/s., 1.12 GB/s.)
-- MemoryTracker: Peak memory usage (for query): 2.00 GiB.
```

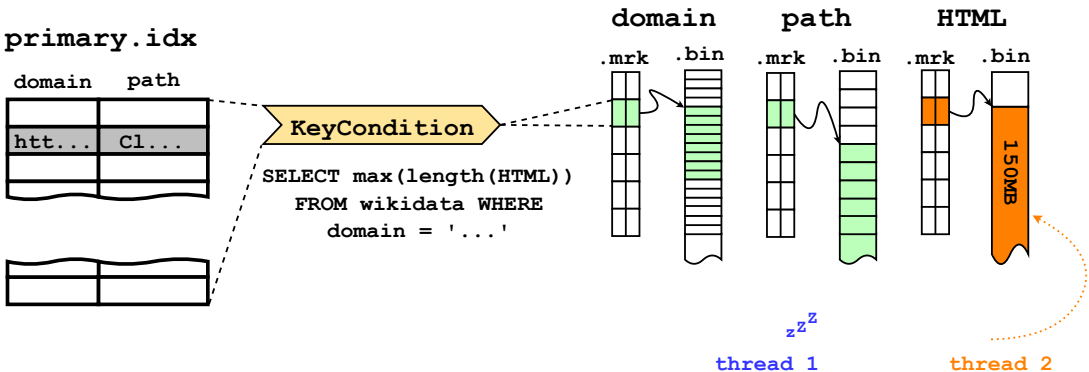
Проблема больших гранул



Проблема больших гранул



Проблема больших гранул



Что делать?

Уменьшить `index_granularity`

- Как оценивать?
- Нельзя изменять без пересоздания таблицы

Порезать строки на чанки

- Ручной подбор длины
- Неудобно при запросах

Адаптивная гранулярность

Идея

- При записи, ограничить размер гранулы количеством байт
- Отдельная настройка таблицы `index_granularity_bytes`
- Делить блок на несколько гранул по объему байт
- Разное количество строк в гранулах
- Новый формат засечек `.mrk2`
- Переписать кучу арифметики в коде...

Количество строк в грануле

Гранулярность на блок:

$$adaptive\_rows(b) = \begin{cases} \dfrac{rows(b)}{size(b) \cdot max\_bytes}, & \text{if } size(b) \geq max\_bytes \\ \dfrac{max\_bytes}{size(b) \cdot rows(b)}, & \text{otherwise} \end{cases}$$

Разумные ограничения:

$$granule\_rows(b) = min(max(adaptive\_rows(b), 1), max\_rows)$$

Адаптивная гранулярность

Свойства:

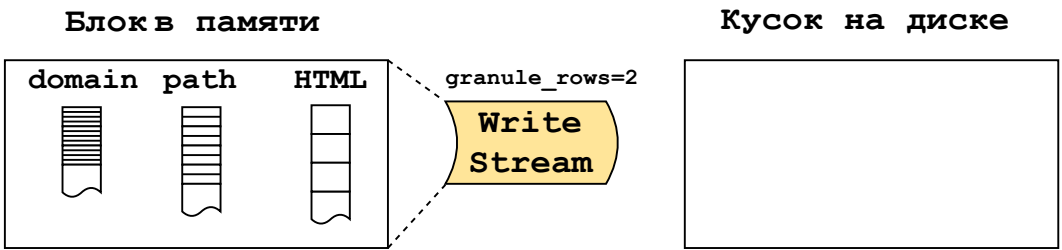
- Фиксированная гранулярность на блок
- Пересчет при каждом слиянии

Размер гранулы:

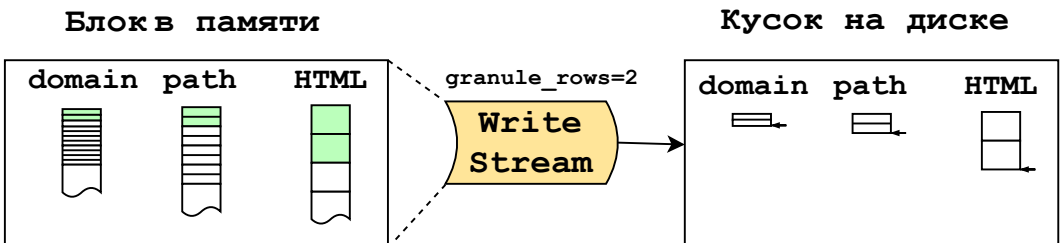
- Не меньше одной строки
- Ограничен снизу количеством байт
- Ограничен сверху количеством строк

Запись с адаптивной гранулярностью

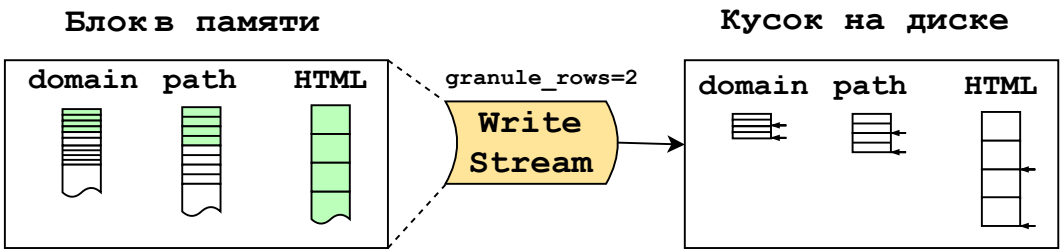
Запись с адаптивной гранулярностью



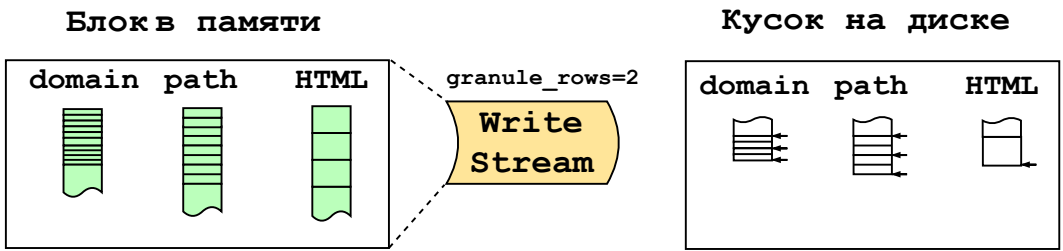
Запись с адаптивной гранулярностью



Запись с адаптивной гранулярностью

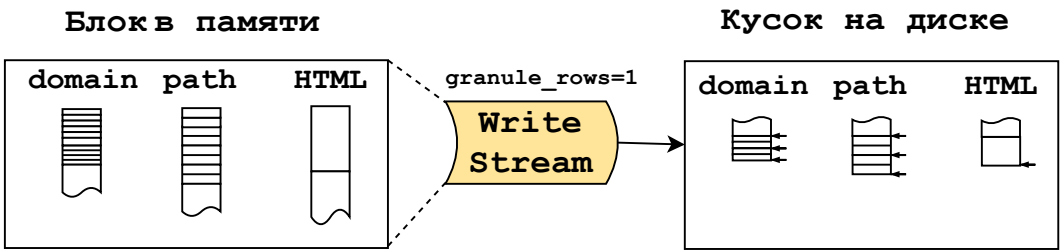


Запись с адаптивной гранулярностью

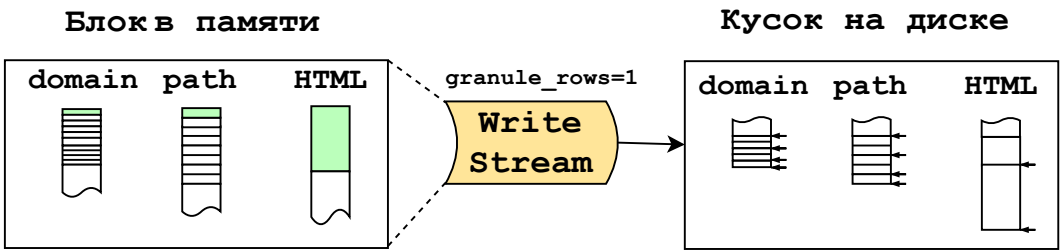


Запись с адаптивной гранулярностью

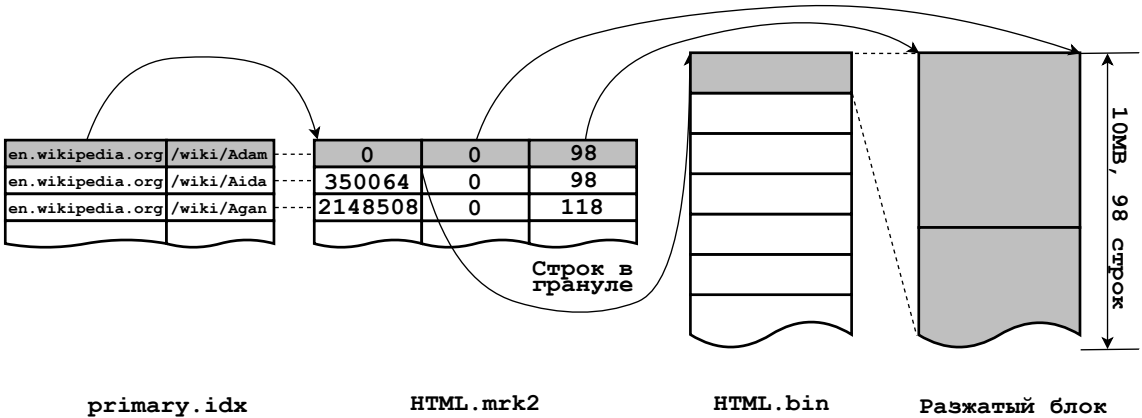
Запись с адаптивной гранулярностью



Запись с адаптивной гранулярностью



Новый формат засечек



Wikipedia в ClickHouse

```
$ sudo apt install clickhouse-server=19.11.* clickhouse-client=19.11.*
```

```
CREATE TABLE wikidata_adaptive (
    domain String,
    path String,
    HTML String
) ENGINE=MergeTree() ORDER BY (domain, path)
SETTINGS index_granularity_bytes = 10485760 -- 10MB;

INSERT INTO wikidata_adaptive SELECT *
    FROM file(
        '/var/lib/clickhouse/user_files/wiki100k.tsv',
        'TSV', 'domain String, path String, HTML String')
```

Wikipedia в ClickHouse

Длина статьи про ClickHouse раньше

```
SELECT max(length(HTML)) FROM wikidata
  WHERE domain='https://en.wikipedia.org' AND path='/wiki/ClickHouse'
58253
1 rows in set. Elapsed: 0.154 sec. (11.08 thousand rows/s., 3.05 MB/s.)
-- MemoryTracker: Peak memory usage (for query): 259.74 MiB.
```

Wikipedia в ClickHouse

Длина статьи про ClickHouse раньше

```
SELECT max(length(HTML)) FROM wikidata
  WHERE domain='https://en.wikipedia.org' AND path='/wiki/ClickHouse'
58253
1 rows in set. Elapsed: 0.154 sec. (11.08 thousand rows/s., 3.05 MB/s.)
-- MemoryTracker: Peak memory usage (for query): 259.74 MiB.
```

Длина статьи про ClickHouse сейчас

```
SELECT max(length(HTML)) FROM wikidata_adaptive
  WHERE domain='https://en.wikipedia.org' AND path='/wiki/ClickHouse'
58253
1 rows in set. Elapsed: 0.007 sec.
-- MemoryTracker: Peak memory usage (for query): 19.63 MiB.
```

Wikipedia в ClickHouse

Самая длинная статья раньше

```
SELECT max(length(HTML)) FROM wikidata
2134709

1 rows in set. Elapsed: 9.843 sec. (10.16 thousand rows/s, 1.12 GB/s.)
-- MemoryTracker: Peak memory usage (for query): 2.00 GiB.
```

Wikipedia в ClickHouse

Самая длинная статья раньше

```
SELECT max(length(HTML)) FROM wikidata
2134709

1 rows in set. Elapsed: 9.843 sec. (10.16 thousand rows/s, 1.12 GB/s.)
-- MemoryTracker: Peak memory usage (for query): 2.00 GiB.
```

Самая длинная статья сейчас

```
SELECT max(length(HTML)) FROM wikidata_adaptive
2134709

1 rows in set. Elapsed: 0.840 sec. (119.08 thousand rows/s, 13.09 GB/s.)
-- MemoryTracker: Peak memory usage (for query): 594.19 MiB.
```

Читаем конец таблицы

Запрос:

```
SELECT max(length(HTML)) FROM wikidata_adaptive
  WHERE domain='https://en.wikipedia.org' AND path > 'zzzzzz'
FORMAT JSON
```

Читаем конец таблицы

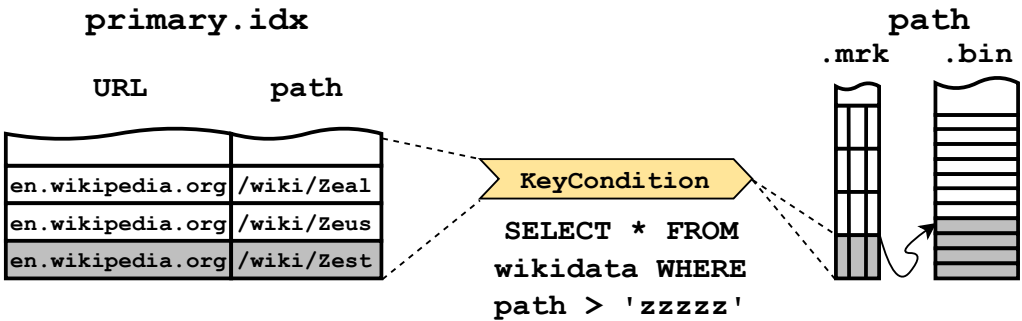
Запрос:

```
SELECT max(length(HTML)) FROM wikidata_adaptive
  WHERE domain='https://en.wikipedia.org' AND path > 'zzzzzz'
FORMAT JSON
```

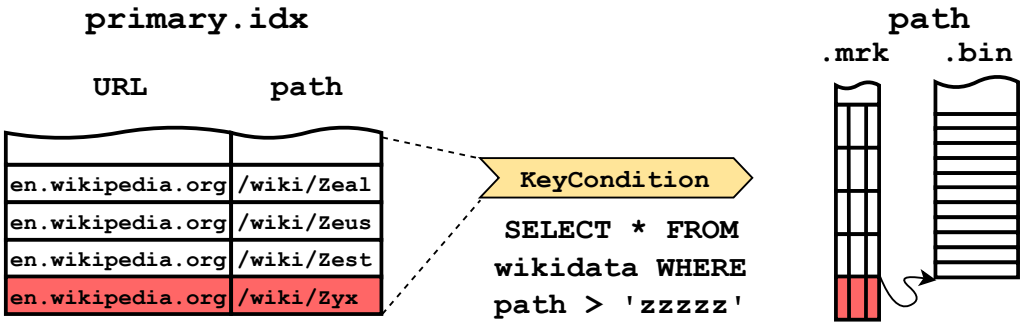
Результат:


```
{
    "statistics":
    {
            "elapsed": 0.00074179,
            "rows_read": 75,
            "bytes_read": 2284
    }
}
```

Последняя гранула



Замыкающая засечка



Замыкающая засечка

```
CREATE TABLE wikidata_adaptive_final_mark (
    domain String,
    path String,
    HTML String
) ENGINE=MergeTree() ORDER BY (key, path)
SETTINGS index_granularity_bytes = 10485760,
         write_final_mark = 1;

INSERT INTO wikidata_adaptive_final_mark SELECT *
    FROM file(
        '/var/lib/clickhouse/user_files/wiki100k.tsv',
        'TSV',
        'domain String, path String, HTML String')
```

Читаем конец таблицы

Запрос:

```
SELECT max(length(HTML)) FROM wikidata_adaptive_final_mark
  WHERE domain='https://en.wikipedia.org' AND path > 'zzzzzz'
  FORMAT JSON
```

Читаем конец таблицы

Запрос:

```
SELECT max(length(HTML)) FROM wikidata_adaptive_final_mark
  WHERE domain='https://en.wikipedia.org' AND path > 'zzzzzz'
  FORMAT JSON
```

Результат:

```
{
    "statistics":
    {
            "elapsed": 0.000350532,
            "rows_read": 0,
            "bytes_read": 0
    }
}
```

Как включить (19.11+)

- `index_granularity_bytes`
  - По умолчанию 10МБ
  - Включено для всех новых таблиц
- `write_final_mark`
  - Включено при адаптивной гранулярности
- `enable_mixed_granularity`
  - Адаптивные и неадаптивные куски в таблице
  - Актуально для существующих таблиц

Как поменять настройки MergeTree

Детачим таблицу

```
:) DETACH TABLE wikidata
```

Как поменять настройки MergeTree

Детачим таблицу

```
:) DETACH TABLE wikidata
```

Правим метаданные:

```
$ vim /var/lib/clickhouse/metadata/default/wikidata.sql
ATTACH TABLE wikidata
(
    `domain` String,
    `path` String,
    `HTML` String
)
ENGINE = MergeTree()
ORDER BY (domain, path)
SETTINGS index_granularity = 8192
```

Как поменять настройки MergeTree

Правим метаданные:

```
ATTACH TABLE wikidata
(
    `domain` String,
    `path` String,
    `HTML` String
)
ENGINE = MergeTree()
ORDER BY (domain, path)
SETTINGS index_granularity = 8192, enable_mixed_granularity = 1
```

Как поменять настройки MergeTree

Правим метаданные:

```
ATTACH TABLE wikidata
(
    `domain` String,
    `path` String,
    `HTML` String
)
ENGINE = MergeTree()
ORDER BY (domain, path)
SETTINGS index_granularity = 8192, enable_mixed_granularity = 1
```

Аттачим таблицу:

```
:) ATTACH TABLE wikidata
```

AggregatingMergeTree

Позволяет хранить состояния агрегатных функций

AggregatingMergeTree

Позволяет хранить состояния агрегатных функций

Агрегатные функции с толстыми состояниями:

- `uniqExact`
- `groupArray`
- `groupUniqArray`
- ...

AggregatingMergeTree

Позволяет хранить состояния агрегатных функций

Агрегатные функции с толстыми состояниями:

- `uniqExact`
- `groupArray`
- `groupUniqArray`
- ...

Неправильно оценивается размер состояния в памяти :(

AggregatingMergeTree

Позволяет хранить состояния агрегатных функций

Агрегатные функции с толстыми состояниями:

- `uniqExact`
- `groupArray`
- `groupUniqArray`
- ...

Неправильно оценивается размер состояния в памяти :(

Адаптивная гранулярность не помогает :(

В результате

Добавились:

- Ограничение размера гранулы байтами
- Замыкающая засечка
- 3 настройки для управления

Для обычных таблиц:

- Все будет работать как раньше
- Возможно немного быстрее

Для таблиц с толстыми строками:

- Запросы ускорятся
- Потребление памяти уменьшится

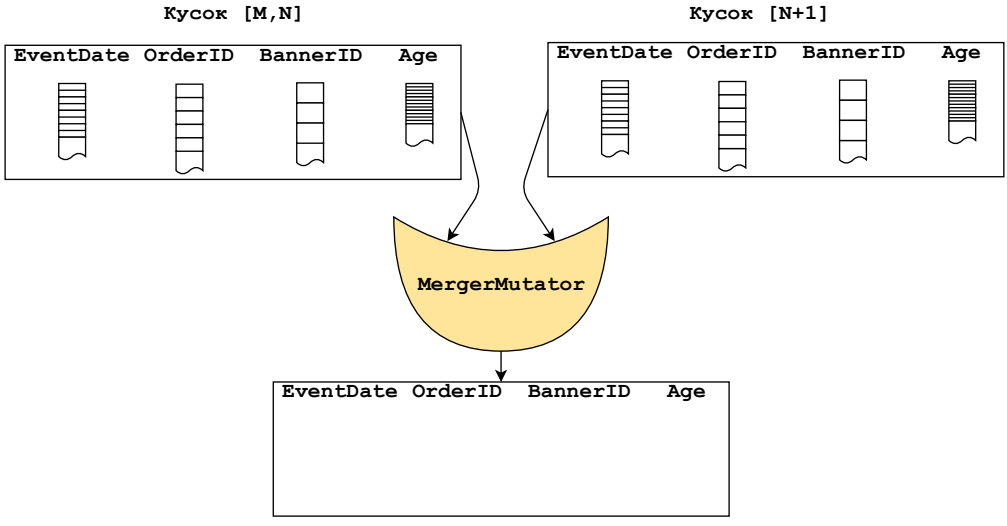
Для AggregatingMergeTree:

- Исправим

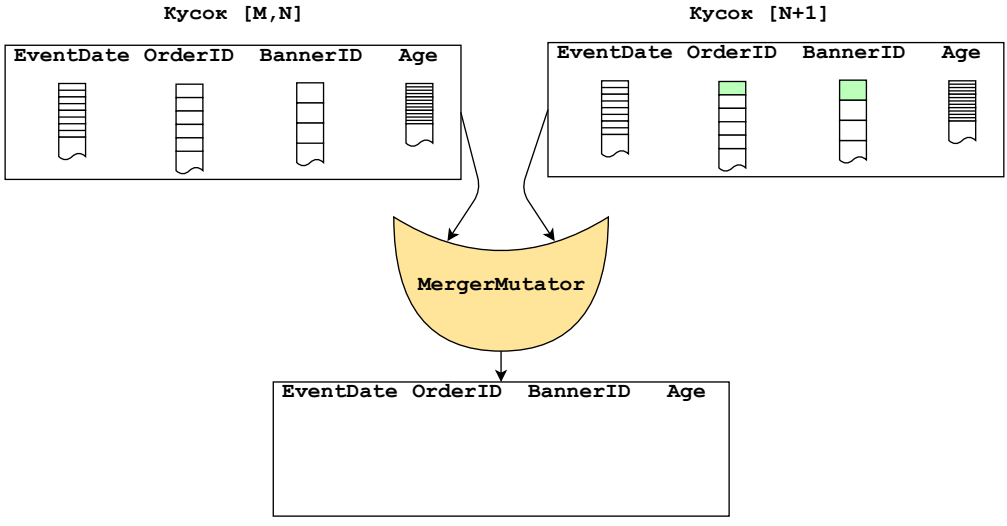
Спасибо

QA

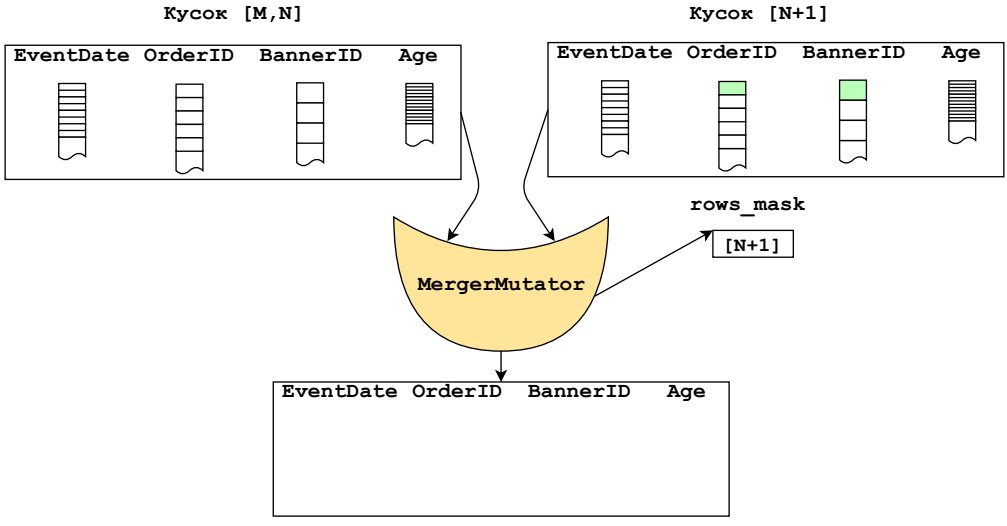
Вертикальное слияние



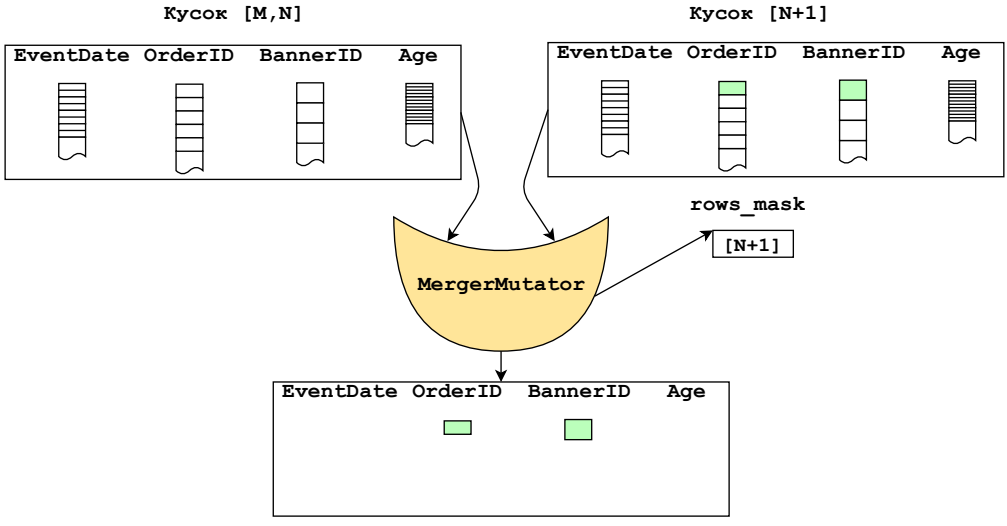
Вертикальное слияние



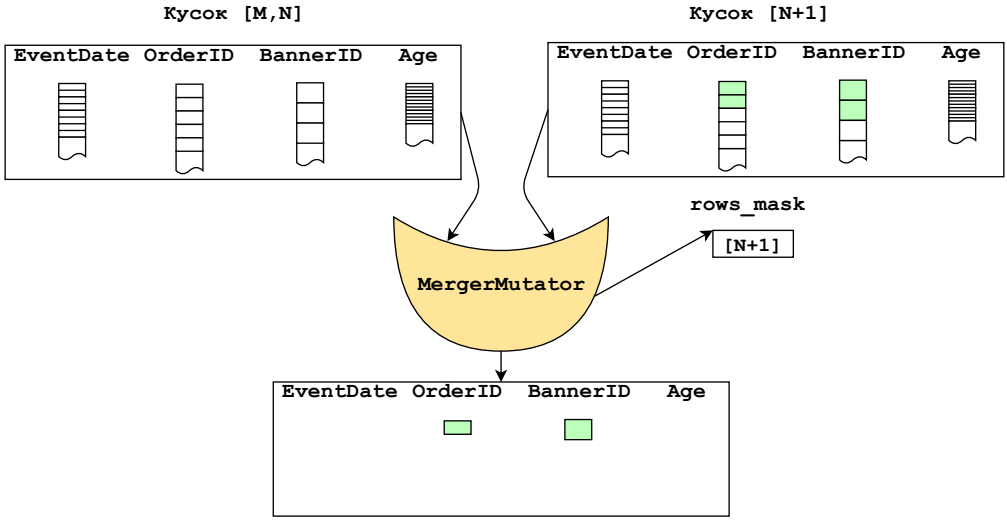
Вертикальное слияние



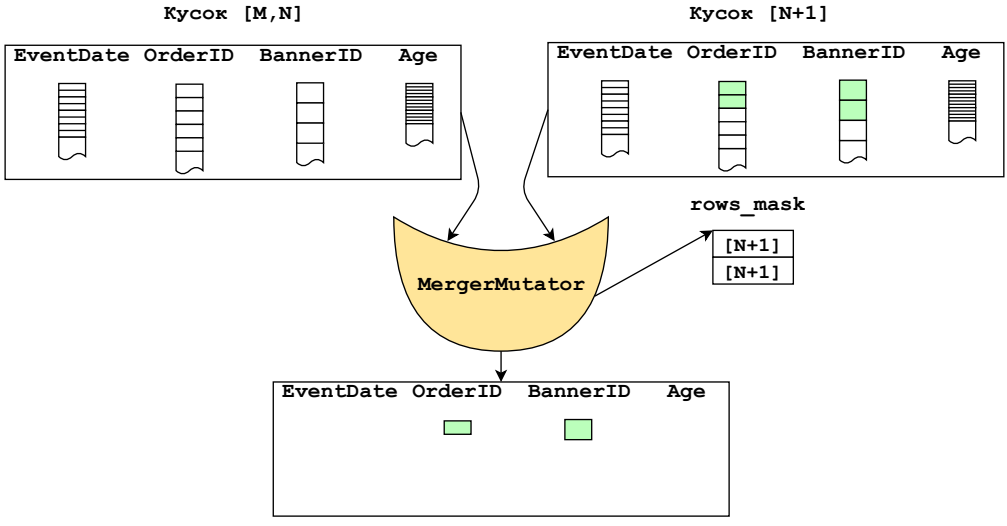
Вертикальное слияние



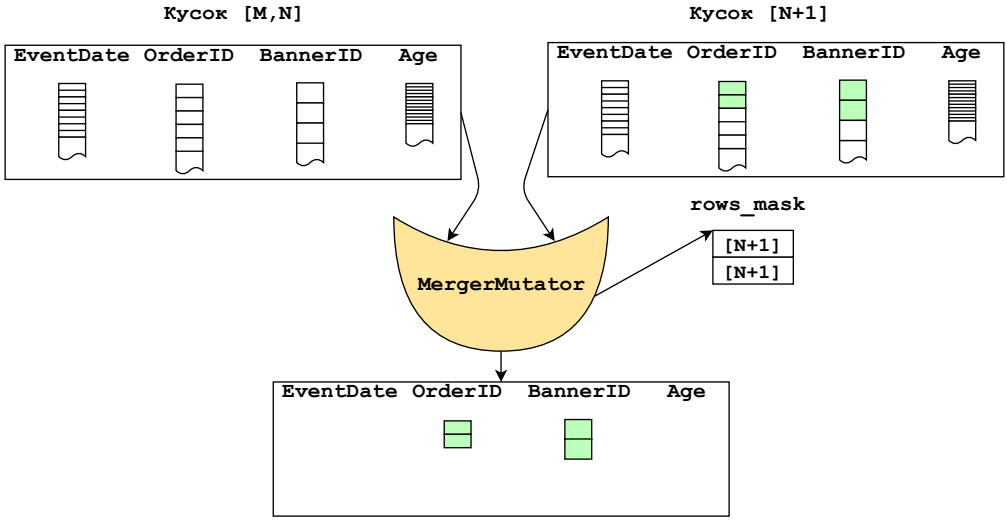
Вертикальное слияние



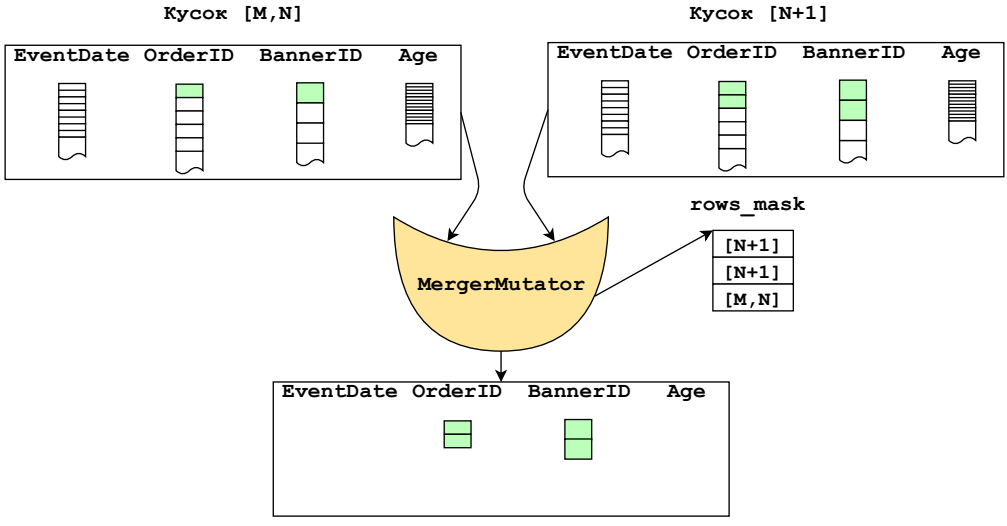
Вертикальное слияние



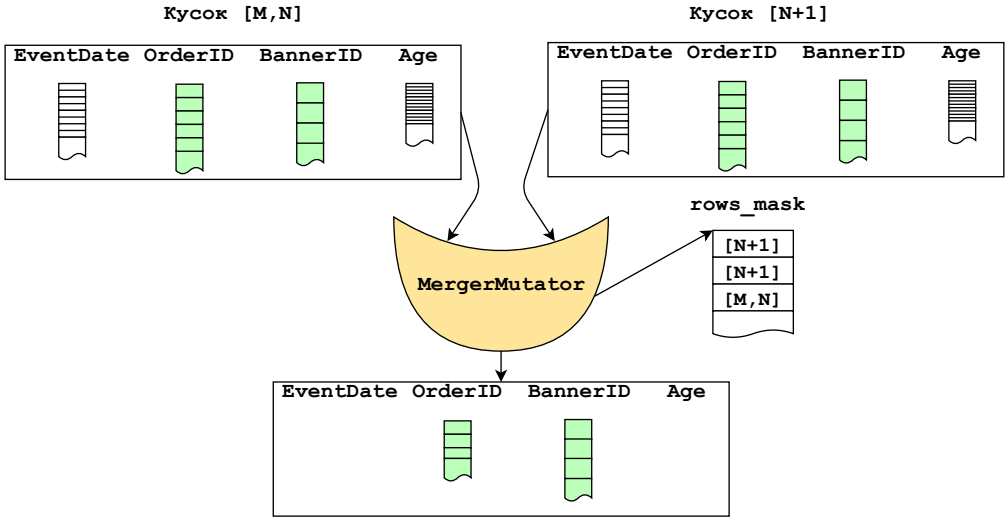
Вертикальное слияние



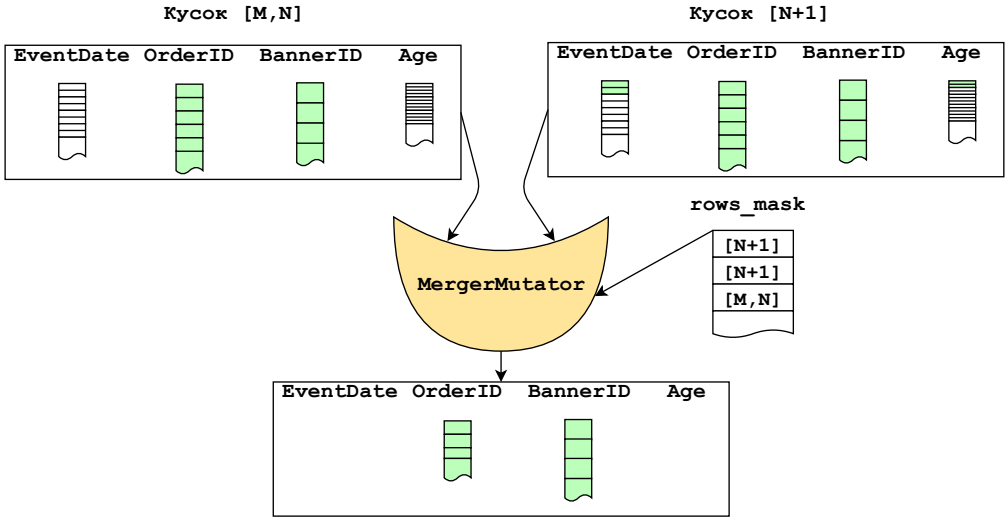
Вертикальное слияние



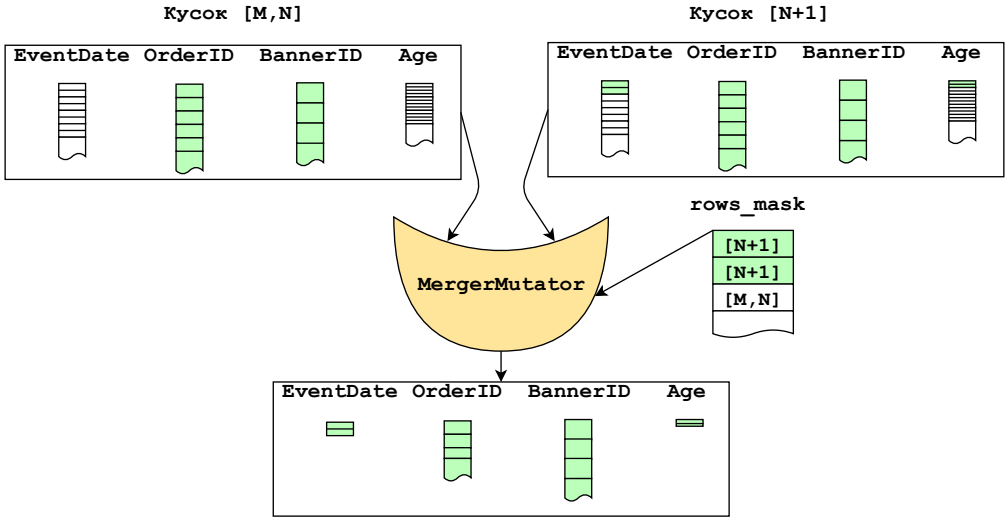
Вертикальное слияние



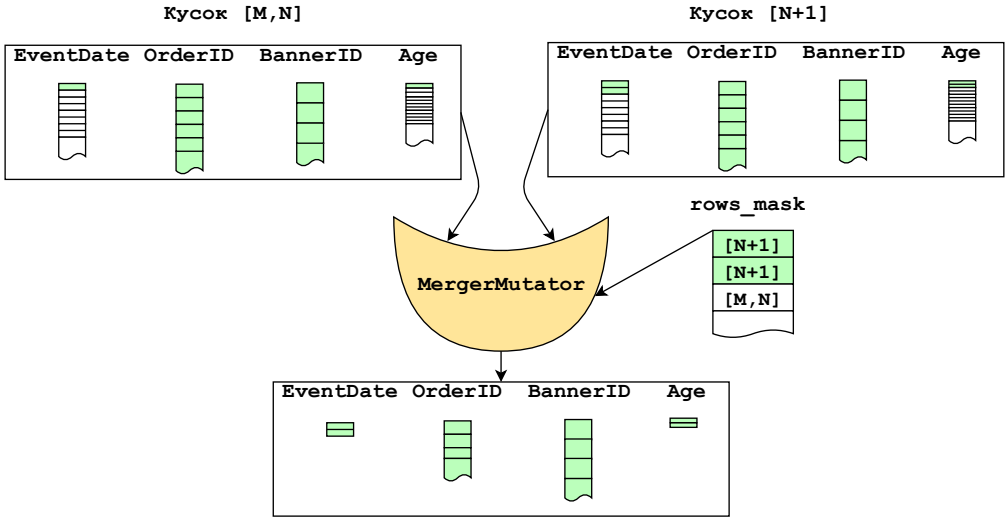
Вертикальное слияние



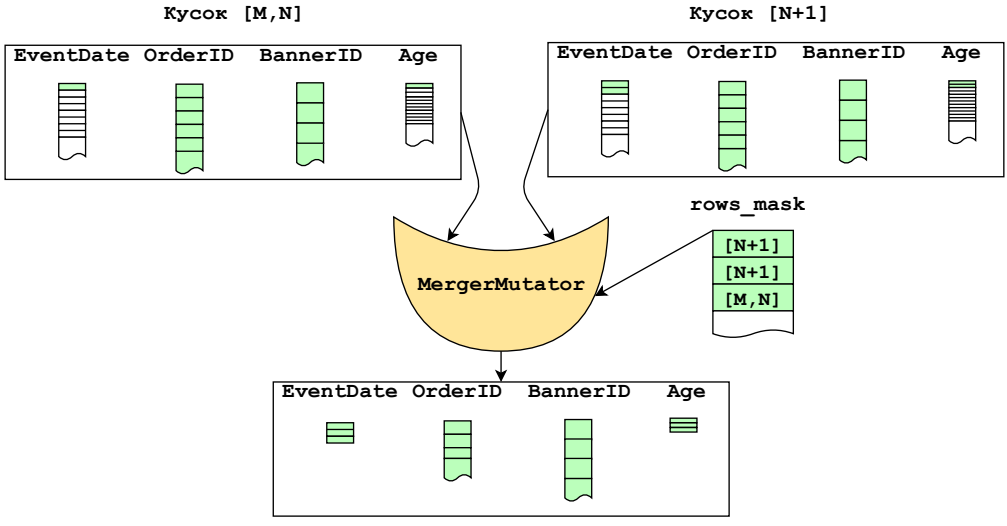
Вертикальное слияние



Вертикальное слияние



Вертикальное слияние



Вертикальное слияние