Яндекс

Партиционирование по произвольному ключу в ClickHouse

Алексей Зателепин

Что такое партиционирование

Разделяем большую таблицу на непересекающиеся части

- Горизонтально – партиция это подмножество строк
- На одной машине
- Основная цель – удобство манипуляций с данными:
  - Удаление старых данных
  - Бэкап, восстановление
  - Перенос между серверами
- Для производительности тоже полезно

Традиционное партиционирование

```
CREATE TABLE table(…) ENGINE = MergeTree(Date, …)
```

Неявный ключ партиционирования: toYYYYMM(Date)

- Хорошо подходит для потока событий / clickstream
- Вставка обычно идёт в последнюю партицию
- Не слишком мелко для запросов по ~году
- Не слишком крупно для манипуляций с данными

Зачем нужно что-то другое?

Отсутствие партиционирования

- ClickHouse это не только clickstream
  Пример: аналитика данных генома

Партиционирование по дню/неделе

- Данных слишком много, хотим хранить 1 месяц

Партиционирование по кортежу (`Месяц, тип события`)

- хотим хранить “мусорные” события не больше N месяцев

Синтаксис (было)

```
CREATE TABLE table (…)
ENGINE = ReplicatedMergeTree(
    '/clickhouse/tables/table', 'replica1',
    Date, intHash32(UserID),
    (CounterID, Date, intHash32(UserID)),
    8192)
```

Синтаксис (стало)

```
CREATE TABLE table (…)
ENGINE ReplicatedMergeTree(
    '/clickhouse/tables/table', ‘replica1')
PARTITION BY toYYYYMM(Date)
ORDER BY (CounterID, Date, intHash32(UserID))
SAMPLE BY intHash32(UserID)
SETTINGS index_granularity=8192 --необязательно
```

Зачем?

- Необязательные ключи и настройки
- Ключи можно задавать в любом порядке
- Можно задавать потабличные настройки

Зачем?

- Необязательные ключи и настройки
- Ключи можно задавать в любом порядке
- Можно задавать потабличные настройки
- Так понятнее!
  В перспективе хочется побороться вот с таким:

```
ENGINE = Buffer(merge, hits,
    16, 10, 100, 10000, 1000000, 10000000,
    100000000)
```

Что можно делать с партициями

Пусть таблица партиционирована так:

```
PARTITION BY (toMonday(Date), EventType)
```

ALTER PARTITION:

- `ALTER TABLE table DROP PARTITION (toMonday(today() - INTERVAL 1 MONTH), 1)`
- `DETACH / ATTACH PARTITION`
- `FREEZE PARTITION`

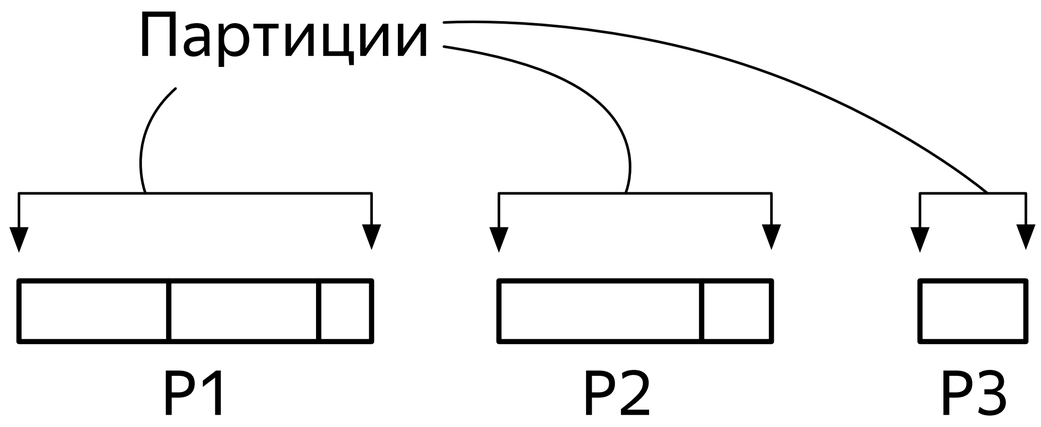
`OPTIMIZE TABLE PARTITION FINAL`

Совместимость

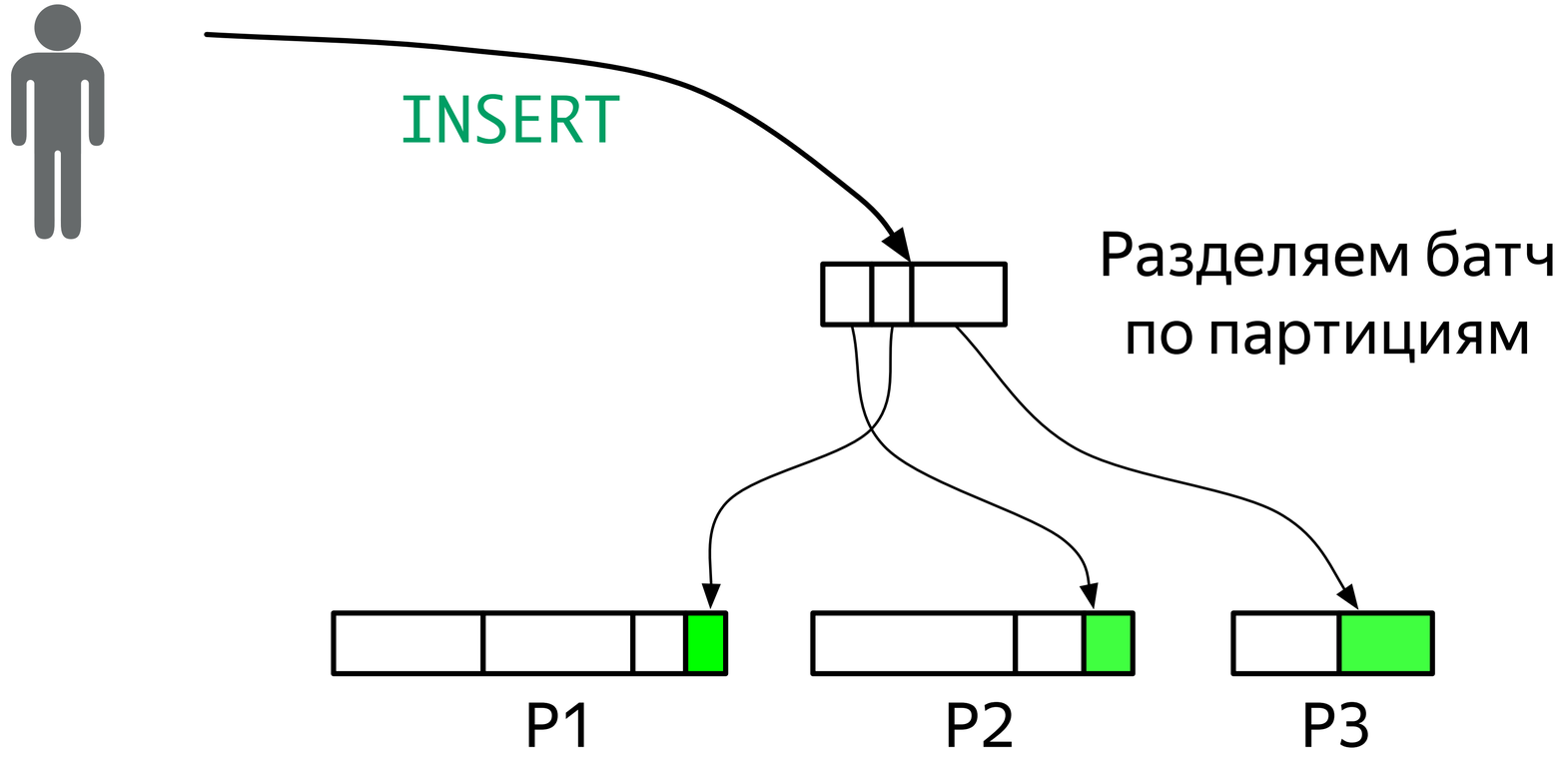
- Новый синтаксис доступен с версии 54310
  ```
  SET experimental_allow_extended_storage_definition_syntax=1
  ```
- Перенос данных через `INSERT SELECT`
- Старый сервер не подхватит таблицы нового стиля
- Но вновь созданные таблицы старого стиля будут работать

Данные таблицы на диске

Набор кусков, сортированных по первичному ключу

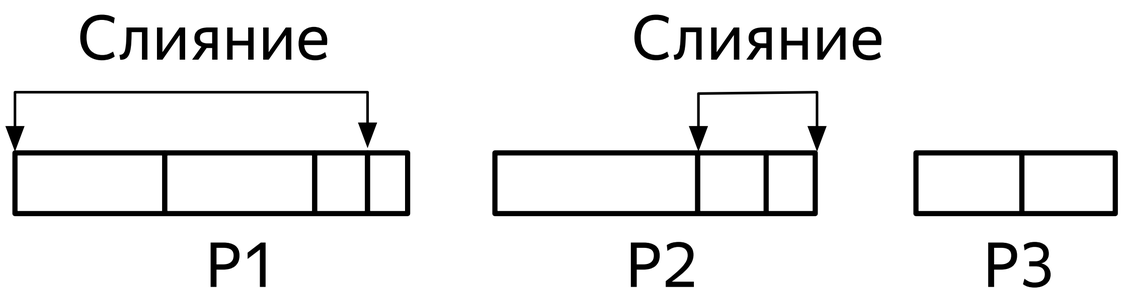


Вставка

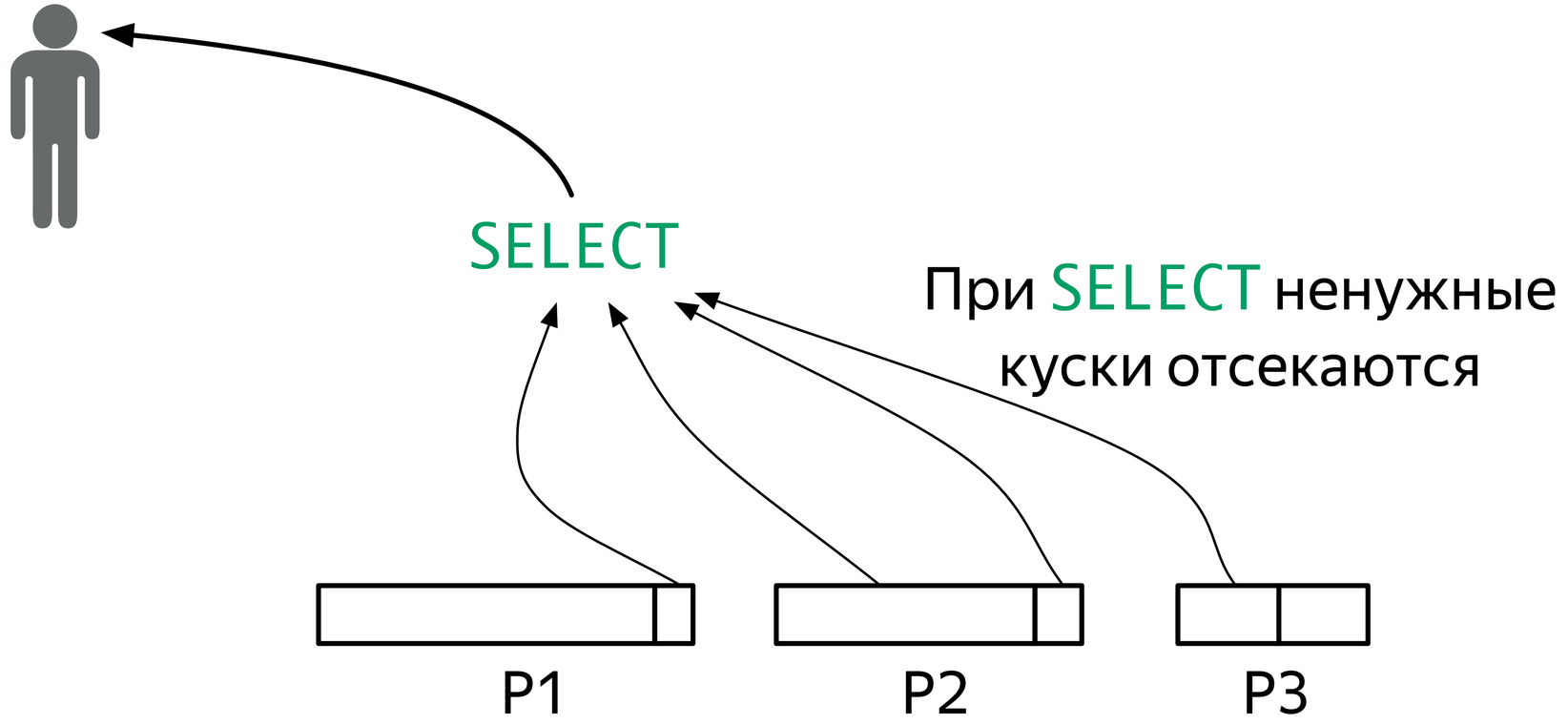


Слияние

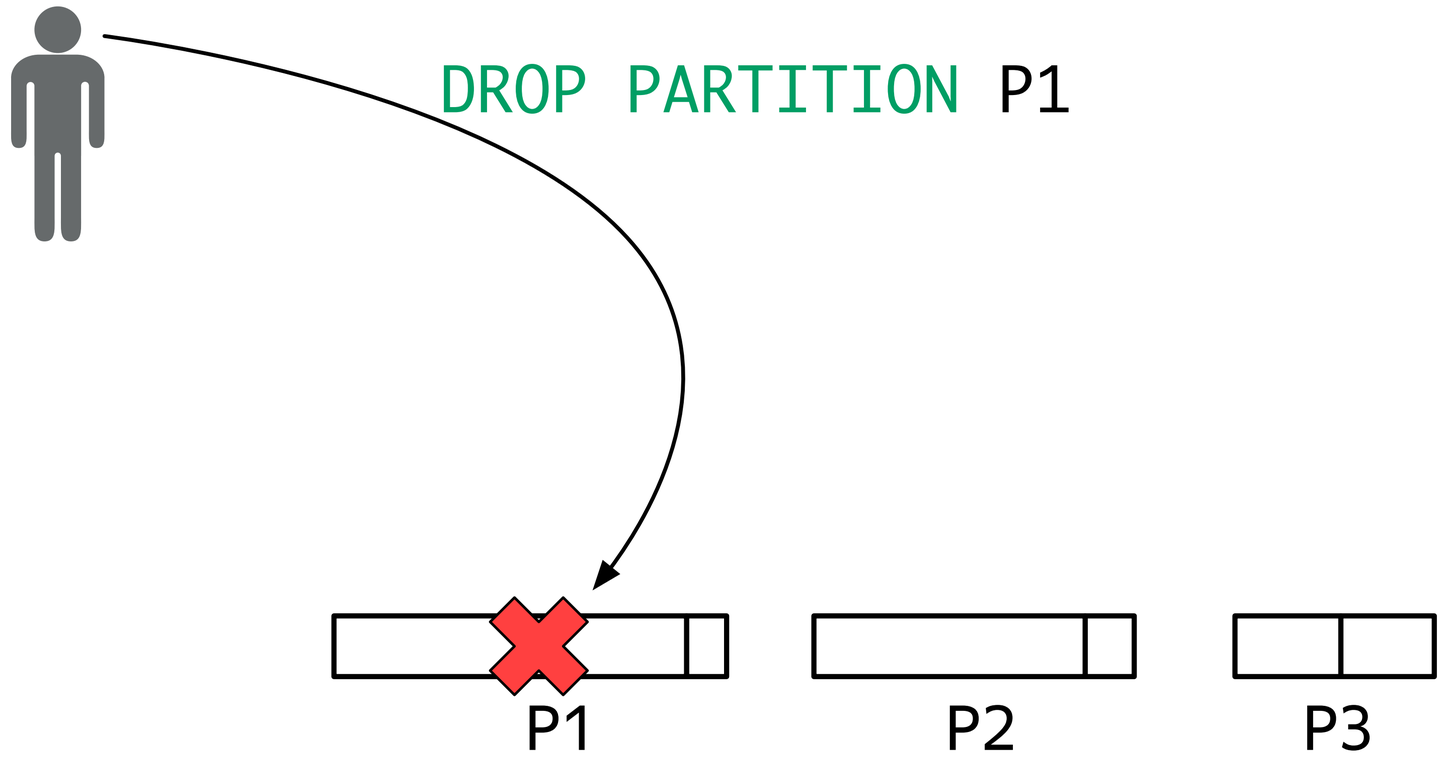
Для слияния выбираются
куски в одной партиции



Запрос



Удаление партиции



Как добиться лучшей производительности

Не делайте слишком много партиций!

- Оптимально не больше ~1000

Ключ партиционирования работает как “индекс”

- Отсечение кусков по min/max столбцов ключа
- Не зависит от первичного ключа
  Оптимальный запрос использует оба индекса!

Старайтесь вставлять в одну партицию

Замеры производительности

Таблица hits

- ~100 столбцов, 33.4M строк
- Данные за 1 месяц
- `ORDER BY (CounterID, EventDate, …)`
- Было: `PARTITION BY toYYYYMM(EventDate)`
- Стало: `PARTITION BY EventDate`

Производительность вставки

`INSERT SELECT` данных за месяц

- Было: 62.9с (530K строк/с)
- Стало: 86.6с (385K строк/с) – на 30% хуже

`INSERT SELECT` данных за день

- Было: 6.3с (1.23M строк/с)
- Стало: 6.3с (1.23M строк/с) – разницы нет

Производительность SELECT

`SELECT FROM hits WHERE EventDate = '2013-07-07'`
(Первый столбец первичного ключа не используется)

- Было: 0.49с (5.9M строк прочитано)
- Стало: 0.24с (0.37M строк прочитано) – на 50% лучше

`SELECT FROM hits WHERE`
`CounterID = 1234 AND EventDate <= ‘2013-07-07'`
(Оптимально для первичного ключа)

- Было: 0.31с (1.71M строк прочитано)
- Стало: 0.32с (1.71M строк прочитано) – разницы почти нет

Спасибо!

Вопросы?

Ещё про произвольное партиционирование:

- Документация: https://clck.ru/CC5UE
- Спрашивайте в чате: https://t.me/clickhouse_ru
- Больше примеров в тестах: https://clck.ru/CC5WL