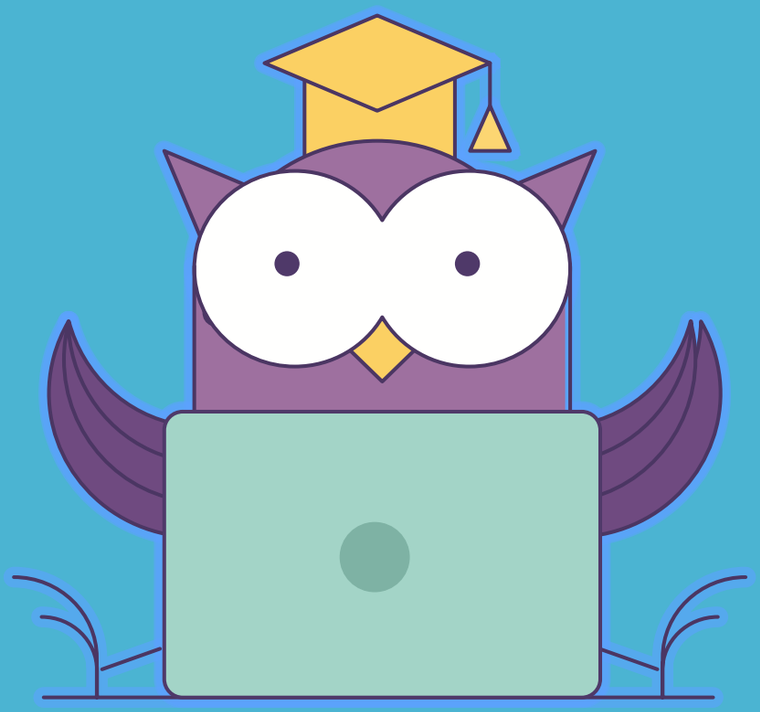
ОНЛАЙН-ОБРАЗОВАНИЕ

ClickHouse не тормозит!

Юрочко Юрий

Как меня слышно и видно?



> Напишите в чат

+ если все хорошо

– если есть проблемы со звуком или с видео

Преподаватель

Юрочко Юрий

- окончил МГТУ им. Н.Э. Баумана в 2016 (ИУ-7)
- 5+ лет разработки на C++/Go
- руководитель команды Go

Цель занятия

- Узнаем что такое ClickHouse
- Посмотрим где и как используется
- Познакомимся с архитектурой, особенностями работы с ним

- колоночная аналитическая БД
- написана на C++ в Яндексе
- open source (Apache License 2.0)
- не тормозит

Где используется

- Яндекс (много где)
- CloudFlare
- Qrator
- Badoo
- Avito

и еще большой список!

Строки vs колонки

Строка	WatchID	JavaEnable	Title	GoodEvent	EventTime
#0	89354350662	1	Investor Relations	1	2016-05-18 05:19:20
#1	90329509958	0	Contact us	1	2016-05-18 08:10:20
#2	89953706054	1	Mission	1	2016-05-18 07:38:00
#N	...	...	...	...	...

Строки vs колонки

Строка:	#0	#1	#2	#N
WatchID:	89354350662	90329509958	89953706054	...
JavaEnable:	1	0	1	...
Title:	Investor Relations	Contact us	Mission	...
GoodEvent:	1	1	1	...
EventTime:	2016-05-18 05:19:20	2016-05-18 08:10:20	2016-05-18 07:38:00	...

Что есть в ClickHouse?

- возможность переваривать огромные объемы информации
- векторная обработка запросов
- компрессия данных
- мощный SQL диалект
- репликация
- шардирование
- продвинутые типы данных
- работа с геоданными, машинным обучением

и еще большой список!

Чего нет в ClickHouse?

- транзакций
- изменения данных (update) (*)
- быстрого доступа по ключу

Пример с созданием таблицы

```
CREATE TABLE events (
        account_id UInt64,
        device_id UInt64,
        event_type Enum8(
                'Login' = 1,
                'Logout' = 2),
        country String,
        time_ns Int64,
        date_time DateTime MATERIALIZED toDateTime(time_ns / 1000000000)
)
ENGINE = MergeTree()
ORDER BY (account_id)
PARTITION BY toYYYYMM(date_time)
```

Пример с созданием таблицы

- в ORDER BY указывается ключ сортировки
- по нему ФИЗИЧЕСКИ будут уложены данные на диске
- можно отдельно указать PRIMARY KEY (если отличается)
- физически поменять сортировку она данный момент нельзя (*)

Пример с созданием таблицы

```
:/var/lib/clickhouse/data/default/events# ls
201910_1_1_0  201911_2_2_0  detached  format_version.txt
:/var/lib/clickhouse/data/default/events# cd 201910_1_1_0/
:/var/lib/clickhouse/data/default/events/201910_1_1_0# ls -lah
-rw-r----- 1 clickhouse clickhouse   34 Nov 26 20:10 account_id.bin
-rw-r----- 1 clickhouse clickhouse   48 Nov 26 20:10 account_id.mrk2
-rw-r----- 1 clickhouse clickhouse  555 Nov 26 20:10 checksums.txt
-rw-r----- 1 clickhouse clickhouse  176 Nov 26 20:10 columns.txt
-rw-r----- 1 clickhouse clickhouse   33 Nov 26 20:10 country.bin
-rw-r----- 1 clickhouse clickhouse   48 Nov 26 20:10 country.mrk2
-rw-r----- 1 clickhouse clickhouse    1 Nov 26 20:10 count.txt
-rw-r----- 1 clickhouse clickhouse   30 Nov 26 20:10 date_time.bin
-rw-r----- 1 clickhouse clickhouse   48 Nov 26 20:10 date_time.mrk2
-rw-r----- 1 clickhouse clickhouse   34 Nov 26 20:10 device_id.bin
-rw-r----- 1 clickhouse clickhouse   48 Nov 26 20:10 device_id.mrk2
-rw-r----- 1 clickhouse clickhouse   27 Nov 26 20:10 event_type.bin
-rw-r----- 1 clickhouse clickhouse   48 Nov 26 20:10 event_type.mrk2
-rw-r----- 1 clickhouse clickhouse    8 Nov 26 20:10 minmax_date_time.idx
-rw-r----- 1 clickhouse clickhouse    4 Nov 26 20:10 partition.dat
-rw-r----- 1 clickhouse clickhouse   16 Nov 26 20:10 primary.idx
-rw-r----- 1 clickhouse clickhouse   34 Nov 26 20:10 time_ns.bin
-rw-r----- 1 clickhouse clickhouse   48 Nov 26 20:10 time_ns.mrk2
```

Содержимое файлов

- *.bin - пожатые блоки данных
- primary.idx - разреженный индекс (каждое N-ое значение)
- *.mrk2 - файл засечек для соответствующего* .bin

Пример с созданием таблицы

```
insert into events values(1, 2, 'Login', 'USA', 1574800066263849999);
insert into events values(1, 3, 'Logout', 'USA', 1574800066263850999);

select name, active, partition_id from system.parts where table='events'
 name          active   partition_id
 201910_1_1_0 │      1 │ 201910
 201911_2_2_0 │      1 │ 201911
 201911_3_3_0 │      1 │ 201911
 201911_4_4_0 │      1 │ 201911
```

Пример с созданием таблицы

- на каждую вставку создается свой парт
- внутри парта для каждой колонки как минимум 1 файл
- парты мержатся в фоне
- частые мелкие вставки - плохо!

Рекомендации

Вставка:

- вставка больших кусков данных за раз (1 млн строк)
- чем реже - тем лучше (хотя бы раз в секунду)

Как достигнуть:

- на стороне приложения
- open source решения (kittenhouse, clickhouse-bulk, ...)
- kafka engine
- buffer table
- http-chunk (осторожно) и можно придумать еще!

Принудительное слияние

```
optimize table events

select name, active, partition_id from system.parts where table='events'
 name          active   partition_id
 201910_1_1_0 │      1 │ 201910
 201911_2_2_0 │      0 │ 201911
 201911_2_4_1 │      1 │ 201911
 201911_3_3_0 │      0 │ 201911
 201911_4_4_0 │      0 │ 201911

:/var/lib/clickhouse/data/default/events# ls
201910_1_1_0  201911_2_4_1  detached  format_version.txt
```

Таблица без партиционирования

```
insert into events_without_partitions values(1, 1, 'Login', 'Russia', 1572207029110549198)
insert into events_without_partitions values(1, 2, 'Login', 'USA', 1574800066263849999)

select name, active, partition_id from system.parts where table='events_without_partitions'
name       active   partition_id
all_1_1_0 │      1 │ all
all_2_2_0 │      1 │ all

:/var/lib/clickhouse/data/default/events_without_partitions# ls
all_1_1_0  all_2_2_0  detached  format_version.txt
```

Неактивные парты удалятся позже (~8 минут), но можно настроить.

index_granularity

- максимальное количество строк между засечками
- всегда читается не меньше index_granularity строк данных
- выбирать аккуратно под ваши нужды - это важно
- по умолчанию 8192
- adaptive index_granularity с 19.7, включен по умолчанию с 19.11

```
CREATE TABLE codec_example
(
    dt Date CODEC(ZSTD),
    ts DateTime CODEC(LZ4HC),
    float_value Float32 CODEC(NONE),
    double_value Float64 CODEC(LZ4HC(9))
    value Float32 CODEC(Delta, ZSTD)
)
ENGINE = <Engine>
...
```

- по умолчанию применяется алгоритм из конфига (секция <**compression**>)
- кодеки можно комбинировать
- можно менять в процессе работы

https://clickhouse.yandex/docs/ru/query_language/create/#kodeki-szhatiia-stolbtsov

Семейство MergeTree

- ReplacingMergeTree
- SummingMergeTree
- AggregatingMergeTree
- CollapsingMergeTree ...

https://clickhouse.yandex/docs/ru/operations/table_engines/#mergetree

- TinyLog
- StripeLog
- Log

https://clickhouse.yandex/docs/ru/operations/table_engines/#log

Пример с CollapsingMergeTree

```
CREATE TABLE users
(
    user_id UInt64,
    name String,
    age Int32,
    country String,
    sign Int8
)
ENGINE = CollapsingMergeTree(sign)
ORDER BY user_id
```

```
insert into users values(1, 'Alex', 30, 'Russia', 1)
insert into users values(2, 'Artem', 25, 'Russia', 1)
```

Пример с CollapsingMergeTree

```
SELECT * FROM users
┌─user_id─┬─name─┬─age─┬─country─┬─sign─┐
│       1 │ Alex │  30 │ Russia  │    1 │
└─────────┴──────┴─────┴─────────┴──────┘

┌─user_id─┬─name─┬─age─┬─country─┬─sign─┐
│       2 │ Artem│  25 │ Russia  │    1 │
└─────────┴──────┴─────┴─────────┴──────┘

insert into users values(2, 'Artem', 25, 'Russia', -1)
insert into users values(2, 'Artem', 26, 'Russia', 1)

SELECT *FROM users
┌─user_id─┬─name─┬─age─┬─country─┬─sign─┐
│       2 │ Artem│  25 │ Russia  │    1 │
└─────────┴──────┴─────┴─────────┴──────┘

┌─user_id─┬─name─┬─age─┬─country─┬─sign─┐
│       1 │ Alex │  30 │ Russia  │    1 │
└─────────┴──────┴─────┴─────────┴──────┘

┌─user_id─┬─name─┬─age─┬─country─┬─sign─┐
│       2 │ Artem│  25 │ Russia  │   -1 │
└─────────┴──────┴─────┴─────────┴──────┘

┌─user_id─┬─name─┬─age─┬─country─┬─sign─┐
│       2 │ Artem│  26 │ Russia  │    1 │
└─────────┴──────┴─────┴─────────┴──────┘
```

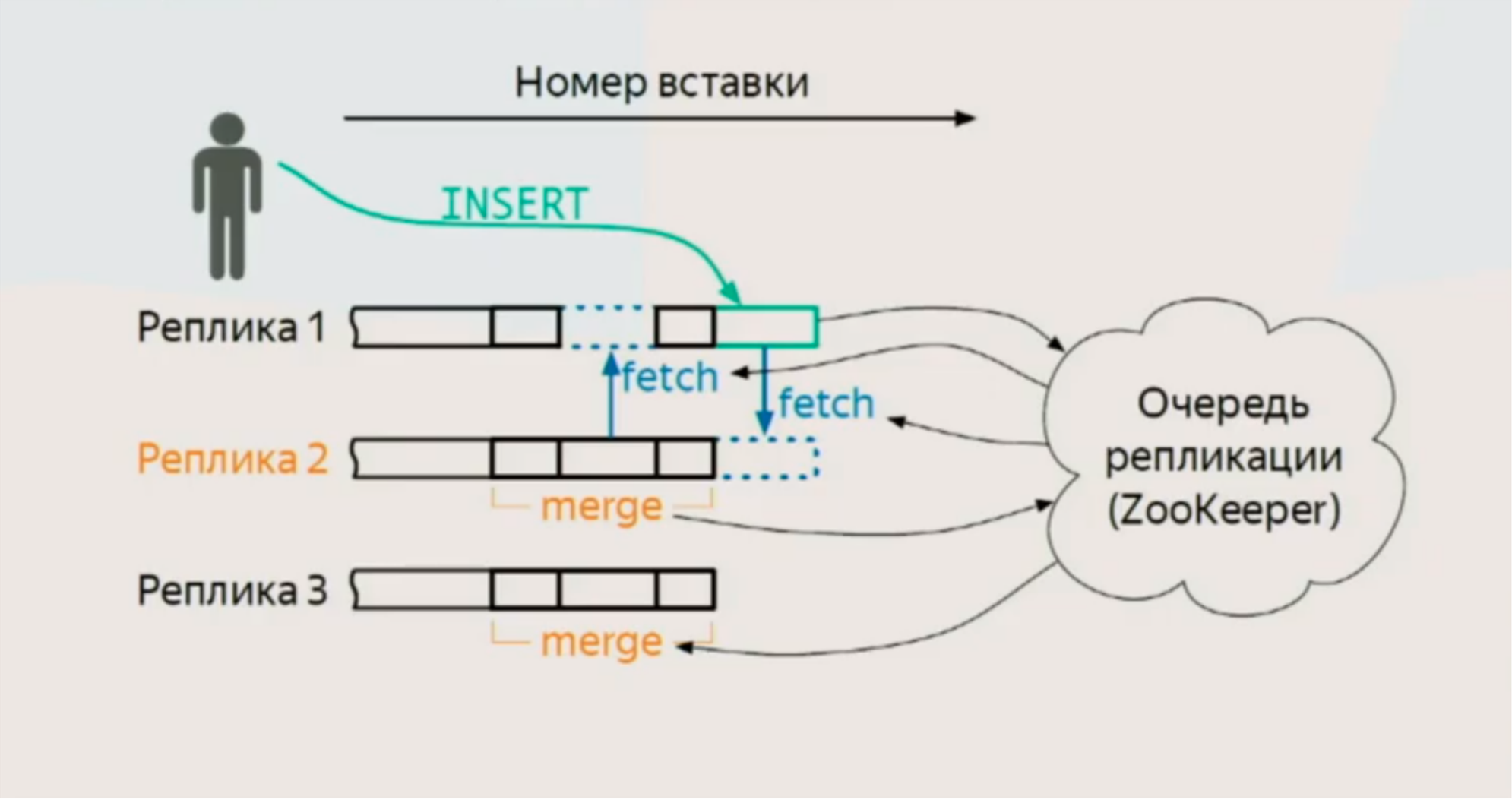
Пример с CollapsingMergeTree

```
SELECT * FROM users FINAL
┌─user_id─┬─name─┬─age─┬─country─┬─sign─┐
│       1 │ Alex │  30 │ Russia  │    1 │
└─────────┴──────┴─────┴─────────┴──────┘

┌─user_id─┬─name──┬─age─┬─country─┬─sign─┐
│       2 │ Artem │  26 │ Russia  │    1 │
└─────────┴───────┴─────┴─────────┴──────┘
```

```
optimize table users final

SELECT * FROM users
┌─user_id─┬─name──┬─age─┬─country─┬─sign─┐
│       1 │ Alex  │  30 │ Russia  │    1 │
│       2 │ Artem │  26 │ Russia  │    1 │
└─────────┴───────┴─────┴─────────┴──────┘
```

- только для семейства MergeTree
- на уровне отдельных таблиц
- репликация на уровне сжатых данных для **INSERT**, **ALTER**
- для координации используется **ZooKeeper** (привет CAP)
- асинхронная (привет CAP)
- master master (привет CAP)
- блоки записываются атомарно и дедуплицируются

https://clickhouse.yandex/docs/ru/operations/table_engines/replication/


Номер вставки
INSERT
Реплика 1
fetch
fetch
Реплика 2
merge
Реплика 3
merge
Очередь
репликации
(ZooKeeper)

Какие настройки бывают и где смотреть

Примеры:

- **max_memory_usage**
- **max_execution_time**
- **force_primary_key**
- ...

https://clickhouse.yandex/docs/ru/operations/settings/query_complexity/
https://github.com/ClickHouse/ClickHouse/blob/7a32ca057d244e3c278372d25030fd50005f511e/dbms/src/Core/Settings.h

Интересные функции

- json (**visitParamHas**, **isValidJSON**)
- умеет немного геоданные (**greatCircleDistance**, **pointInPolygon**)
- умеет немного машинки (**evalMLMethod**, **stochasticLinearRegression**)
- ...

https://clickhouse.yandex/docs/ru/query_language/functions/geo/
https://clickhouse.yandex/docs/ru/query_language/functions/machine_learning_functions/
https://clickhouse.yandex/docs/ru/query_language/functions/json_functions/

нативный:

- TCP
- используется, например, в clickhouse-client
- нет документации (исходники C++)
- порт 9000

HTTP:

- можно использовать в любом языке
- более ограничен
- гораздо проще
- порт 8123

1. Хочу постоянно искать пользователя по id?
2. Хочу писать логи телеметрии со 100 заранее известными параметрами?
3. Хочу писать информацию о банковских транзакциях?
4. Хочу строить месячные отчеты по широкой таблице, при этом самих данных очень много?

- https://clickhouse.yandex/docs/en/
- https://www.altinity.com/blog
- https://www.youtube.com/channel/UChtmrD-dsdpspr42P_PyRAw
- https://github.com/ClickHouse/ClickHouse

Итоги занятия

- Узнали что такое ClickHouse
- Посмотрили где и как используется
- Познакомились с архитектурой, особенностями работы с ним

Вопросы?

Пройдите, пожалуйста, опрос

https://otus.ru/polls/5539/

Спасибо за внимание!