ClickHouse для инженеров и архитекторов БД

Репликация и другие фоновые процессы

otus.ru

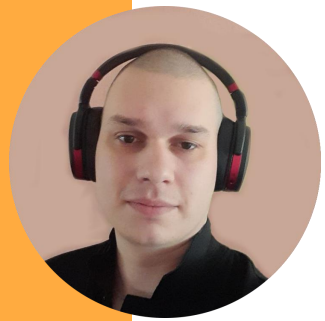
Проверить, идет ли запись

Меня хорошо видно && слышно?

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

Тема вебинара

Репликация и другие фоновые процессы



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-2024-04

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

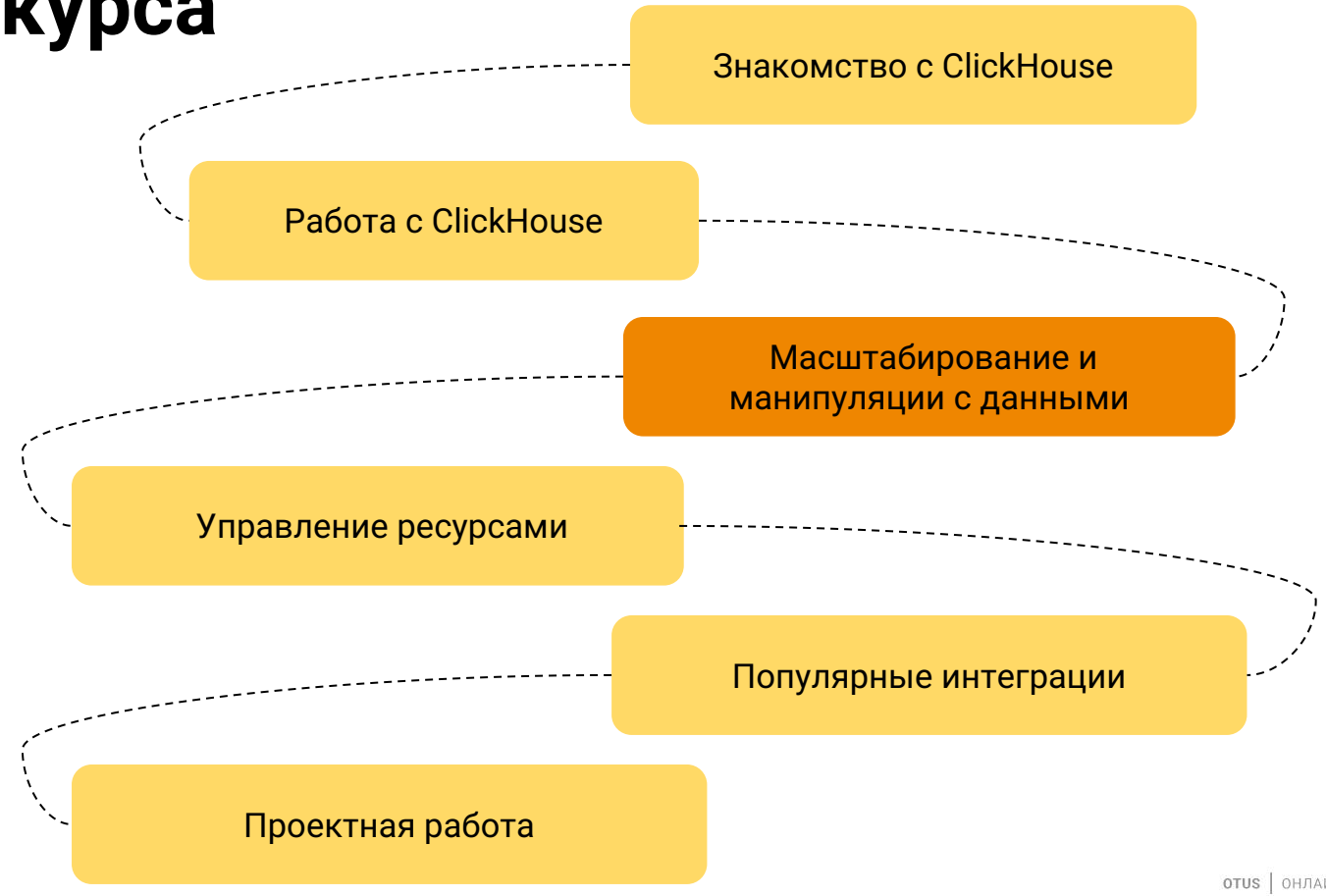
Пишем в чат

Говорим голосом

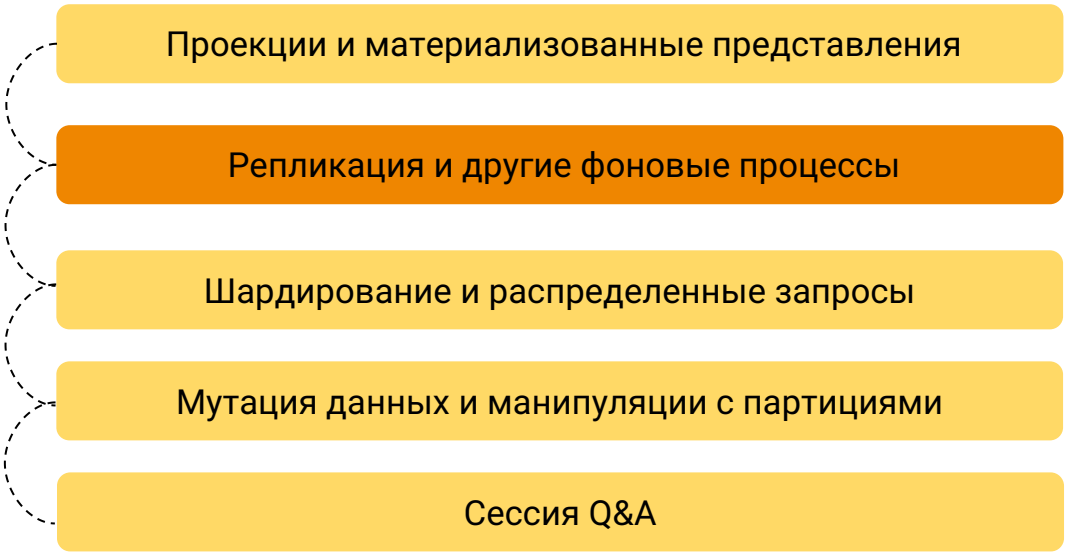
Документ

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
SQL, движки и другие особенности



Маршрут вебинара

Цели вебинара

К концу занятия вы сможете

1. настраивать репликацию в ClickHouse
2. настраивать отложенное удаление данных
3. ограничивать пропускную способность репликации

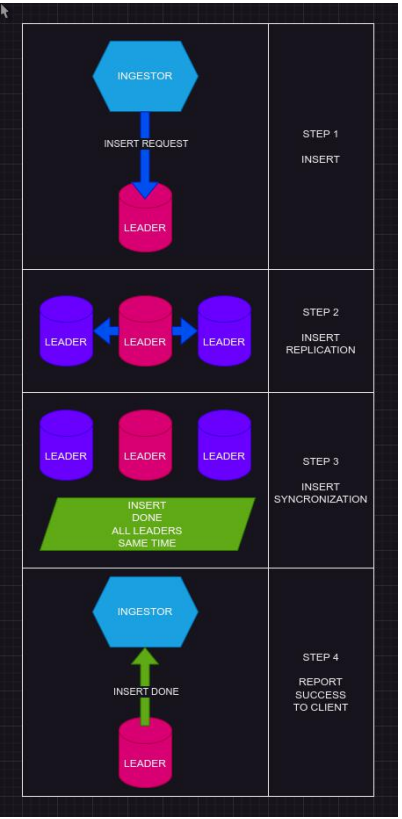
Смысл

Зачем вам это уметь

1. обеспечивать высокую доступность и резервирование
2. хранить только последние N дней востребованных данных
3. исключать полную утилизацию ClickHouse пропускной способности сети, необходимой для других приложений

Виды репликации в разных системах

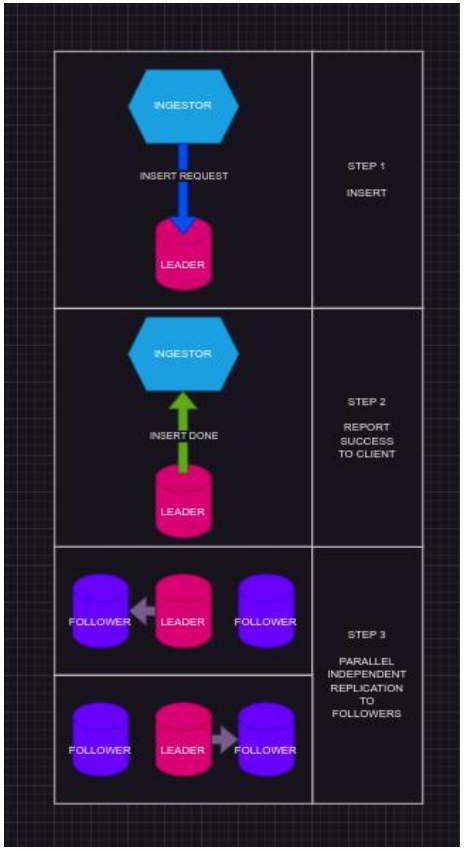
Синхронная



Запрос считается выполненным тогда, когда он выполнен на всех репликах, как правило все реплики лидеры.

Дополнительные механизмы синхронизации выполнения запроса на репликах.

Асинхронная



Запрос считается выполненным тогда, когда он выполнен на принимающей запрос реплике.

Репликация запроса выполняется независимо на все реплики, в фоновом режиме.

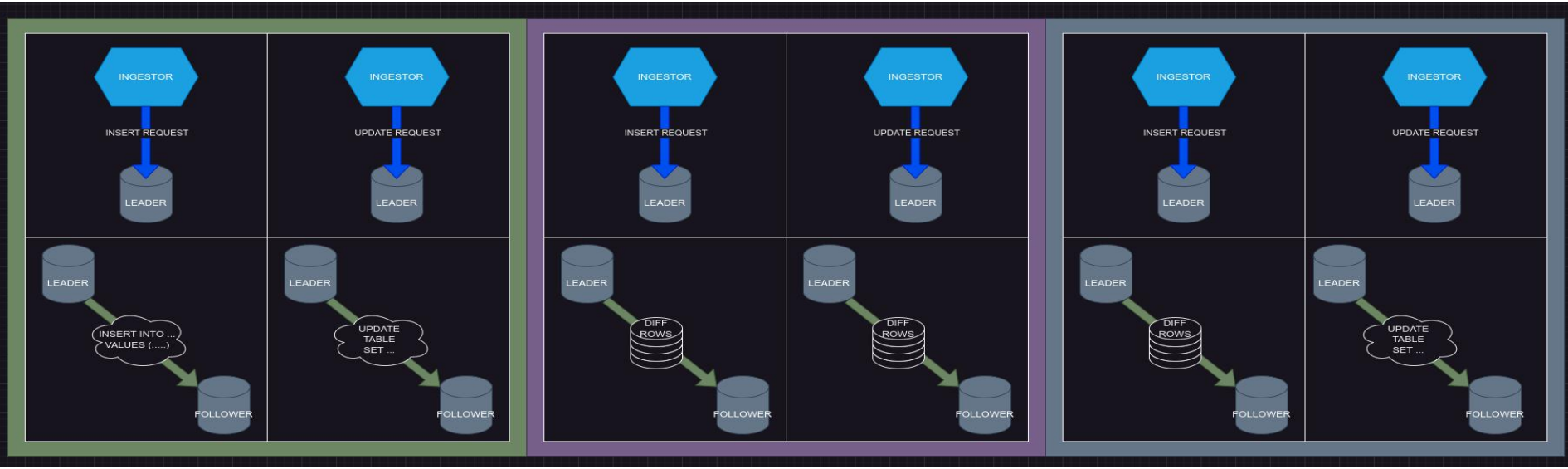
Как правило один лидер, остальные реплики.

Однако мульти-лидер так же возможен с дополнительными механизмами обеспечения консистентности.

Statement

Row

Mixed



Реплицируются запросы на выполнение

Реплицируются сами данные и изменения этих данных

Совмещение подходов

Вопрос на подумать:

Какая репликация больше подходит для ClickHouse?

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Репликация в ClickHouse

Асинхронный мульти-лидер

Каждая реплика самостоятельно принимает данные

- данные о новых партах записываются в метаданные в zookeeper/clickhouse-keeper (далее *keeper), и в очередь *keeper как событие GET_PART
- каждая реплика слушает http-порт заданный в конфигурации как interserver_http_port И/ИЛИ interserver_https_port, а так же анонсирует себя в *keeper
- каждая реплика разбирает очередь и выполняет GET_PART, скачивая с соседних реплик недостающие данные

Архитектурно исключены проблемы с асинхронной репликацией

- нет требований к уникальности Primary Key
- нет автоинкремента

Row или Statement? Mixed!

Ближайший аналог для репликации данных - ROW.

Репликация данных происходит партами, их можно считать блоками строк, но правильнее будет их называть блоками колонок, а репликацию PART-Based.

Кроме результатов фоновой операции MERGE_PARTS, можно однако настроить и эти данные перевыкачиваться готовыми с реплик, а не пересчитываться.

Для репликации метаданных - Statement.

Для репликации мутаций - Statement.

Требования к репликации

1. <zookeeper>...</zookeeper> секция в конфигурации, смотрящая на живой *keeper кластер

2. Replicated приставка к Engine таблицы

3. дополнительные параметры для Engine:
- path - путь в *keeper
- replica - уникальное имя реплики

4. уникальный путь в *keeper для каждой таблицы

Пример

Конфигурация

```
<zookeeper>
    <node index="1">
        <host>zk-server1.zone</host>
        <port>2181</port>
    </node>
    <node index="2">
        <host>zk-server2.zone</host>
        <port>2181</port>
    </node>
    <node index="3">
        <host>zk-server3.zone</host>
        <port>2181</port>
    </node>
</zookeeper>
```

Таблица

```
CREATE TABLE replication_test
(
    `a` String
)
ENGINE = ReplicatedMergeTree('/some_uniq_zookeeper_path', 'some_replica_name')
ORDER BY a
```

Макросы

Применяются для упрощения задания параметров репликации

Конфигурация:

<macros>

<macros_name>macros_value</macros_name>

</macros>

наиболее часто используемые:

<shard> и <replica> либо <host>

Встроеные макросы:

<database> - подставит имя текущей базы

<table> - создаваемой таблицы

Пример использования макросов:

CREATE TABLE ....

ENGINE=Replicated...('/clickhouse/shard_{shard}/{database}/{table}','{replica}', ...)

Добавление реплики

- разворачивается ещё один сервер вашей системой управления конфигурациями
- идентичная конфигурация ClickHouse, за минусом секции <macros> и параметров идентифицирующих сервер, таких как <inserserver_host>, если вас не устроили значения по умолчанию, в противном случае только <macros>.
- создаются все таблицы как на соседней реплике, с такими же path в *keeper, с отличным значением параметра <replica>.
- дальше ClickHouse самостоятельно выкачает по репликации все данные

Конвертация таблицы в реплицируемую

- создаем новую таблицу

CREATE TABLE new AS old ENGINE=Replicated*(2 параметра реплиации, ...) остальное так же как в оригинальной таблице

- выполняем ALTER TABLE new ATTACH PARTITION ID ‘...’ FROM old

для всех партиций старой таблицы, можно их найти как ”SELECT DISTINCT partition_id FROM system.parts where database || ‘.’ || table == ‘database.table’;”

- “EXCHANGE TABLES new AND old; DROP TABLE new;” или “DROP TABLE old; RENAME TABLE new TO old;” или просто удаляем старую если новое имя нас устраивает.

Зачем нужен leader в ClickHouse?

- назначает операции MERGE_PARTS
- можно крутить ручкой can_become_leader
- может быть несколько лидеров или все реплики лидерами, в зависимости от версии ClickHouse

Скорость репликации

Она высокая, никакого пережатия и дообработки данных не происходит, только подсчет контрольных сумм, узкое место - скорость дисковой системы, иногда *keeper если он перегружен или данные поступают часто мелкими батчами, накапливая большое количество партов.

Если *keeper в порядке, а дисковая подсистема позволяет, можно утилизировать любой сетевой интерфейс. Например, забить аплинк между датацентрами, одновременно поставив заливаться десяток-другой реплик с 25Gbit интерфейсами.

Ограничивать скорость можно и нужно параметрами:

- max_replicated_sends_network_bandwidth_for_server (предпочтительно)

или

- max_replicated_fetches_network_bandwidth_for_server (тоже можно)

Состояние репликации

system.replicas - системная таблица с состоянием репликации каждой таблицы.

полезные поля:

- is_readonly - 0 когда всё хорошо, 1 когда репликация не работает, таблица блокируется на запись
- last_queue_update_exception и zookeeper_exception - ошибка, смотреть если is_readonly
- absolute_delay / queue_size / {inserts,merges,part_mutations}_in_queue - лаг репликации и состояние очередей
- {total,active}_replicas / replica_is_active - сколько всего и где реплики есть у таблицы

Состояние репликации

system.replication_queue - состояние очереди репликации

полезные поля:

- type - тип события репликации: GET_PART, REPLACE_RANGE, MERGE_PARTS

- postpone_reason - обычно это «Not executing fetch ... because 8 fetches already executing» и это нормально

- num_postponed - сколько раз откладывалось

- num_tries - попыток; last_exception - почему не получилось, например не хватило диска/памяти, сетевые обрывы

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

TTL (time-to-live)

TTL - отложенное удаление данных

Главный кейс - удаление.

Объявляется для столбца

CREATE TABLE table (... column ТИП ***TTL timestamp + INTERVAL 1 hour*** ...)

Или для таблицы

CREATE TABLE table (...) ... ***TTL timestamp + INTERVAL 1 hour***

timestamp - любая колонка Date или DateTime

INTERVAL 1 hour - описание специального типа interval

для столбцов удаляет значение столбца когда now()>=timestamp+интервал

для таблицы целиком строку

TTL - *горячее и холодное хранение*

При наличии storage_policy, можно вместо удаления перещать данные между дисками
синтаксис: ***TTL timestamp + INTERVAL 1 hour TO VOLUME вольюм_из_конфигурации***
объявляется на таблицу

TTL - *альтернатива*

Важно понимать, что сравнение происходит во время Merge партов, и стоит ресурса в размере сравнения timestamp с каждой строкой, на нагруженной системе Merge парта может случиться сильно позже объявленного TTL или не случиться вовсе при достижении макс. размера парта.

Есть ручка, меняющая это поведение на близкое к крону дропающему партиции - ***ttl_only_drop_parts***

Более надежным/эффективным/дешевым способом является простой cron с ALTER TABLE DROP PARTITION или ALTER TABLE MOVE PARTITION

Mutations

фоновые операции мутации данных

Виды и применимость мутаций в теме «Мутация данных и манипуляции с партициями» курса.

Мутации так же являются фоновыми операциями и статус их выполнения можно наблюдать в таблице system.mutations, важные колонки:

- command - запрос описывающий мутацию
- is_done - становится =1 когда успешно выполнена
- latest_fail_reason - последняя ошибка
- parts_to_do_names / parts_to_do - прогресс выполнения

Merge операции

фоновые операции Merge

Когда иное не подразумевается контекстом, или type операции Merge не является чем то отличным от MERGE_PARTS, под Merge понимают объединение партов.

Есть 2 алгоритма объединения, Horizontal и Vertical.

Horizontal - по умолчанию, выбирает столько, сколько удовлетворяет настройке max_bytes_to_merge_at_max_space_in_pool (150Гб default)

Vertical - выбирает строго 2 парта для объединения, включается автоматически при достижении одного из критериев:

- vertical_merge_algorithm_min_rows_to_activate = 131072
- vertical_merge_algorithm_min_bytes_to_activate = inf по умолчанию
- vertical_merge_algorithm_min_columns_to_activate = 11 столбцов не-ключа

фоновые операции Merge

Кроме MERGE_PARTS есть типы Merge-операций:

- MUTATE_PART (отображается как пустая строка в system.merges) для мутаций
- REPLACE_RANGE для ALTER TABLE REPLACE PARTITION
- TTL

Выделение ресурсов под фоновые операции

пул фоновых операций

Количество одновременных фоновых задач репликации, Merge партов и TTL определяется пулами тредов, а именно:

background_pool_size (default: 16, в старых версиях 8) - количество тредов для операций Merge и мутаций

background_move_pool_size (default: 8) - пул для Merge-операций типа MOVE_PART

background_fetches_pool_size (default: 16) - пул для GET_PART

В старых версиях выставляется пользователю default, в новых в основной конфиг сервера.

Список и назначение пулов меняется от версии к версии, актуальный можно получить в метриках запросом

```
SELECT *
FROM system.metrics
WHERE metric LIKE 'Background%'
```

чаще всего страдают TTL-операции

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Домашнее задание

1) Взять любой демонстрационный DATASET, не обязательно полный набор данных:
https://clickhouse.com/docs/en/getting-started/example-datasets
2) Конвертировать таблицу в реплицируемую, используя макрос replica
3) Добавить 2 реплики
4) отдать результаты запросов как 2 файла
SELECT
getMacro(‘replica’),
*
FROM remote(‘разделенный запятыми список реплик’,system.parts)
FORMAT JSONEachRow;

SELECT * FROM system.replicas FORMAT JSONEachRow;
5) Добавить/выбрать колонку с типом Date в таблице, добавить TTL на таблицу «хранить последние 7 дней». Предоставить результат запроса «SHOW CREATE TABLE таблица» на проверку.

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Рефлексия

Цели вебинара

Проверка достижения целей

1. настраивать репликацию в ClickHouse
2. настраивать отложенное удаление данных
3. ограничивать пропускную способность репликации

Вопросы для проверки

1.	как добавить реплику
2.	как настроить удаление данных старше 7 дней для таблицы с партиционированием по столбцу date типа Date
3.	ClickHouse утилизирует всю пропускную способность сервера, создавая проблемы в работе других приложений. Какие следует принять меры?

Рефлексия

С какими впечатлениями уходите с вебинара?

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

Следующий вебинар

20 июня 2024

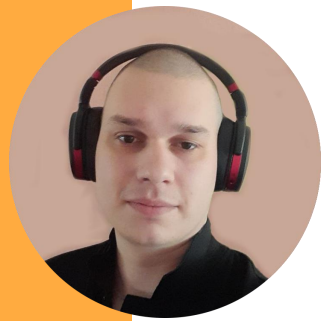
Проекции и материализованные представления

- 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.