ClickHouse для инженеров и архитекторов БД

Язык запросов SQL

otus.ru

Проверить, идет ли запись

Напишите «+» в чат, если меня слышно и видно

Правила вебинара

Активно
участвуйте

Задавайте вопросы
в чат

Вопросы видим в чате,
можем ответить не сразу

Об опыте

Руководитель направления анализа интернет-продаж — Альфа-Банк

Руководитель академических проектов — SAS

lamoda.ru мода с доставкой

ЭЛЬДОРАДО

ТЕХНОСИЛА

МТС БАНК

Руководитель направления продуктовой аналитики
Тимлид команды аналитики мобильного приложения и web

Титова Наталия Николаевна

*-**12 лет опыта в продуктовой аналитики и Data Science***
- Эксперт по товарной и клиентской аналитике в онлайн и рознице
-Управление проектами связанными с развитием аналитических решений онлайн данных

*- **6 лет опыта в преподавании***
-Старший преподаватель в МИЭМ НИУ ВШЭ в департаменте прикладной математики

- преподаватель курса “Nosql” и “Алгоритмы и структуры баз данных”в ОТУС

Контакты:
*Telegram: **@natalitics***

Цели вебинара

К концу занятия вы сможете

1. познакомиться с ClickHouse
2. рассмотреть возможности ClickHouse
3. научиться искать ответы на свои вопросы
4. узнать особенности реализации SQL в ClickHouse
5. изучить нюансы и специфики

Зачем нужен ClickHouse?

- Производительность.
  - ClickHouse очень быстрый, как и аналитика на нем
- Стоимость.
  - И в первую очередь стоимость масштабирования (антиподы - Vertica, Oracle, Microsoft). ClickHouse опенсорсный
- Операционная стоимость.
  - Это подход чуть-чуть с другой стороны. Примеры - RedShift, Google BigQuery.

Где используется ClickHouse сейчас?

- Аналитика веб-приложений
- AdTech компании
- Анализ операционных логов с разных источников.
- Мониторинг логов безопасности. Загрузка, хранение и визуализация в BI
- Финансовый анализ
- Телекоммуникационные компании
- Мониторинга производственных процессов. Пример - тестирование электроники и анализ получившейся продукции
- Блокчейн-аналитика

Диалекты

ClickHouse

ClickHouse поддерживает 2 языка запросов:
SQL и **PRQL**
и называет их диалектами

SQL - Structured Query Language

Классический язык запросов для баз данных.

ClickHouse поддерживает не все базовые операции.

UPDATE и DELETE представлены альтернативами через ALTER TABLE ... UPDATE|DELETE.

PRQL - Pipelined Relational Query Language

Альтернативный классическому SQL язык запросов.

Внутри ClickHouse преобразуется к SQL вокруг цепочки подзапросов.

Переключение диалекта

SET dialect=’clickhouse’; SET dialect=’prql’;
(default)

Переключение диалекта

SET dialect=’clickhouse’; SET dialect=’prql’;

(default)

```sql
-- SQL Equivalent:

SELECT
  product_id,
  SUM(quantity) AS total_quantity,
  AVG(price) AS average_price
FROM
  sales
WHERE
  date >= '2023-01-01'
GROUP BY
  product_id
ORDER BY
  total_quantity DESC
LIMIT
  10;
```

```
from sales
filter date >= '2023-01-01'
group product_id (
  total_quantity = sum quantity
  average_price = average price
)
sort total_quantity -1
take 10;
```

Основы SQL в ClickHouse

Синтаксис и ключевые слова

Создание таблиц

```
CREATE TABLE table_name (
    column1_name column1_type,
    column2_name column2_type,
    ...
) ENGINE = engine_type;
```

• ENGINE определяет тип движка для хранения данных (например, MergeTree, Log, Memory).

Вставка данных

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

Запросы на выборку

```
SELECT column1, column2
FROM table_name
WHERE condition
GROUP BY column
ORDER BY column
LIMIT n;
```

Синтаксис и ключевые слова

Обновление и удаление данных

• В ClickHouse нет стандартных команд **UPDATE и DELETE**, так как это колоночная база данных, оптимизированная для вставки и чтения. Для изменения данных используются подходы с помощью **ALTER TABLE** или создание новой таблицы с изменёнными данными.

Агрегатные функции

• ClickHouse поддерживает множество агрегатных функций: **SUM(), AVG(), MIN(), MAX(), COUNT()** и другие.

Функции работы с массивами и строками

• ClickHouse предлагает богатый набор функций для работы с массивами и строками, таких как **arrayJoin(), splitByChar(), concat()** и другие.

Отличия от стандартного SQL

Отсутствие поддержки транзакций

• ClickHouse не поддерживает транзакции в классическом понимании. Операции выполняются атомарно на уровне блоков данных.

Схема данных

• ClickHouse требует явного указания типов данных при создании таблиц, и они не могут быть изменены после создания таблицы.

Отсутствие индексов в традиционном смысле

• Вместо индексов ClickHouse использует сортировку данных по ключу в движке MergeTree, что позволяет ускорять запросы.

Ограниченная поддержка подзапросов

• Хотя ClickHouse поддерживает подзапросы, они не так мощны и гибки, как в некоторых других СУБД.

Специфические функции и оптимизации

• ClickHouse предлагает уникальные функции, такие как sample для работы с выборками данных и специализированные агрегатные функции для аналитики.

Различия в обработке NULL

• ClickHouse имеет специфическую обработку значений NULL и требует явного указания при создании столбцов.

Продвинутые функции ClickHouse

Продвинутые функции ClickHouse

1. **Колонко-ориентированное хранение;**
2. **Масштабируемость и распределенные запросы;**
3. **Материализованные представления;**
4. **MergeTree;**
5. **Асинхронные репликации и шардирование;**
6. **Поддержка SQL-подобного синтаксиса;**
7. **Функции для работы с временными рядами;**
8. **Поддержка JSON и других форматов;**
9. **Гибкая система индексов;**
10. **Функции машинного обучения;**
11. **Поддержка OLAP-анализа**

Возможности

ClickHouse

- горизонтальное масштабирование
- многоуровневое хранение
- высокая пропускная способность
- приближенные вычисления
- интеграция с другими системами
- преобразование данных

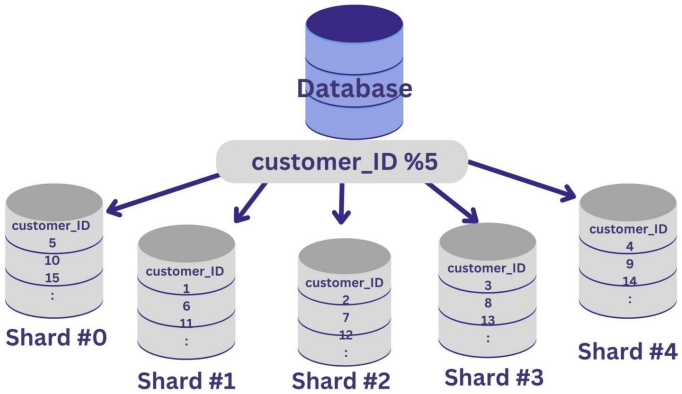
Горизонтальное масштабирование

Поддерживается:

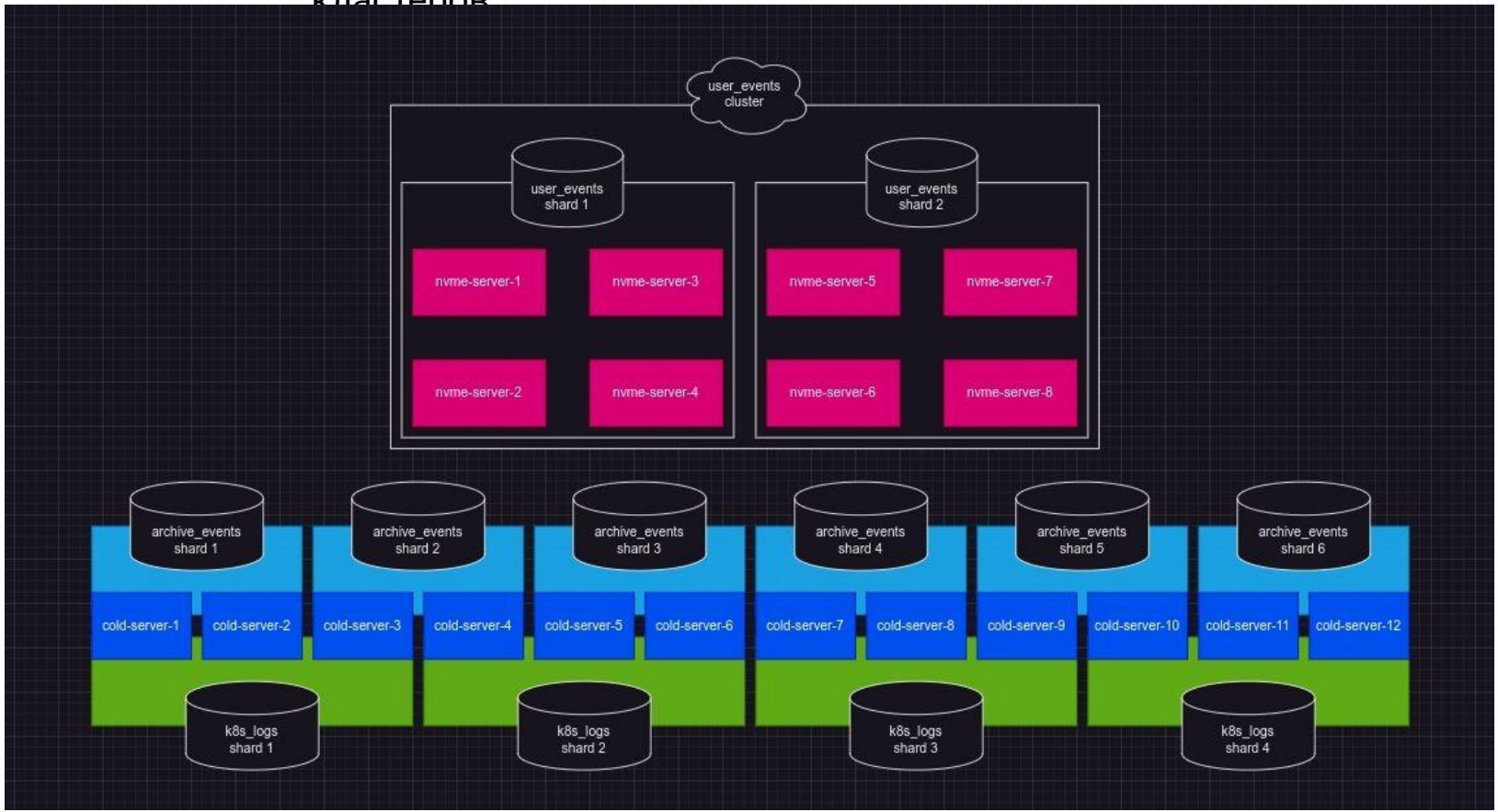
- добавление/удаление шардов
- добавление/удаление реплик

Особенность:

- кластер это топология, описанная в конфигурации;
- можно описать несколько топологий;
- сервер может быть частью нескольких кластеров

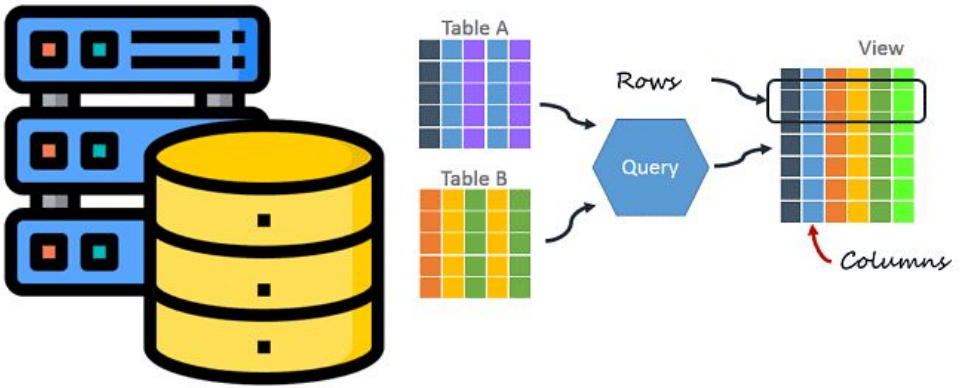


Сервер может быть частью нескольких кластеров



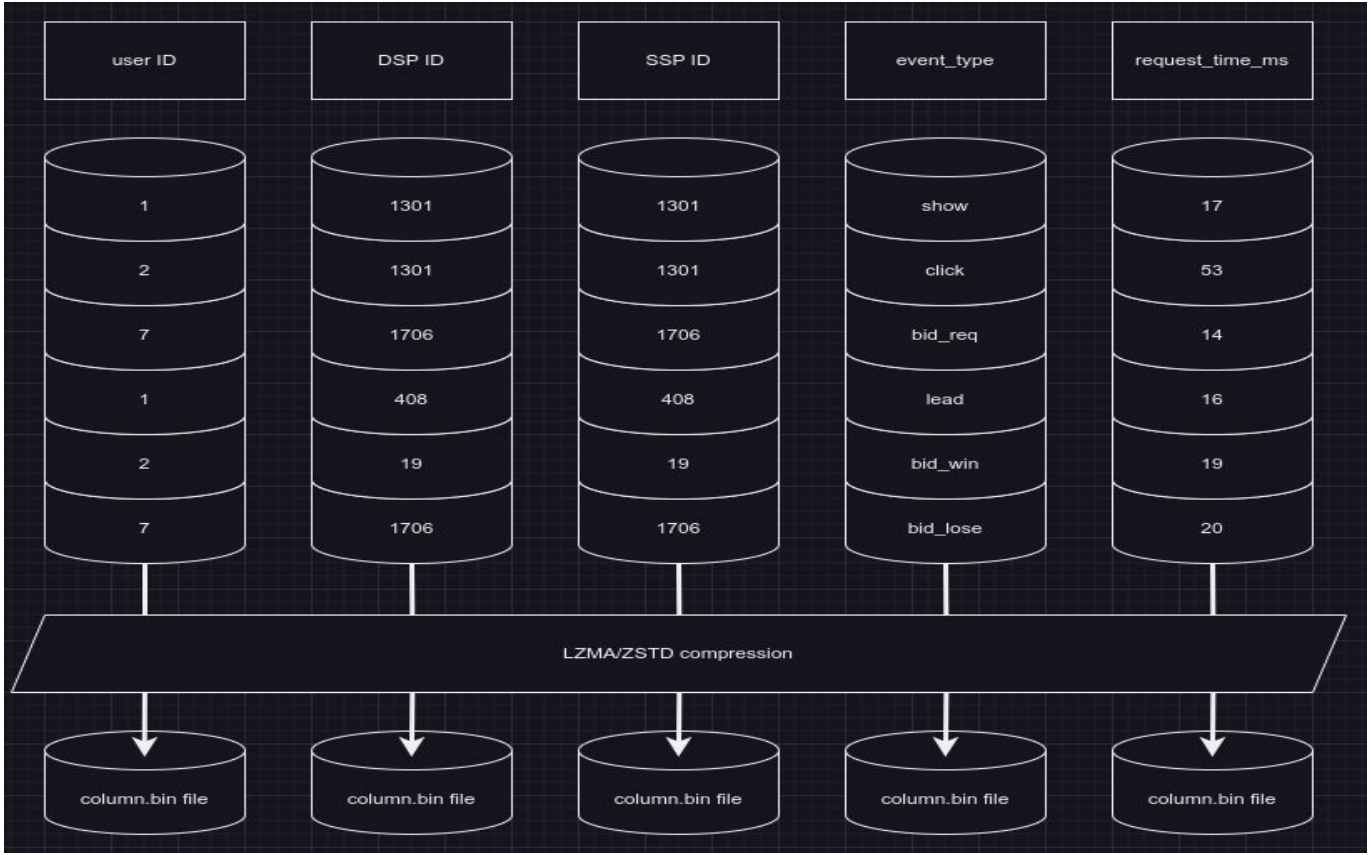
Высокая пропускная способность

- **колоночное хранение** обеспечивает высокую степень сжатия любыми кодеками
- **на простых запросах** достигается выдача данных на скорости дисковой системы умноженной на уровень сжатия
- **низкая производительность** на множестве точечных запросов



Колоночное хранение обеспечивает высокую степень сжатия любыми кодеками

- однородные данные хранятся рядом
- кардинальность чаще низкая, чем высокая
- даже на быстрых алгоритмах достигается сжатие в десятки раз



Интеграция с другими системами

С кем работает ClickHouse:

- облачные файловые системы: **s3 / gcs / azure/ hdfs**
- базы данных: **postgresql / mysql / mongo / rocksdb / redis**
- брокеры сообщений: **rabbitmq / kafka / NATS**
- **hive / iceberg / hudi / deltalake / sqlite**

Кто работает с ClickHouse:

- **redash / grafana / tableau / superset / vector**
- есть официальные библиотеки для основных языков: **python / golang / java / C++**
- а так же неофициальные для многих других

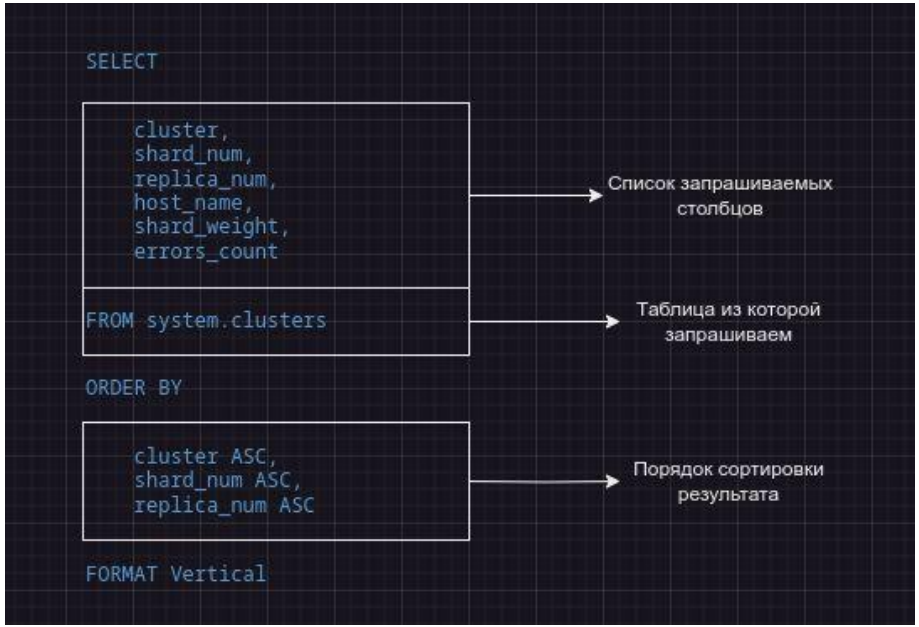
CRUD операции

SELECT
INSERT
UPDATE
DELETE

SELECT

Операция чтения данных. Позволяет получить данные из ClickHouse, выполнив над ними заданные преобразования.

Простой пример (запрос сконфигурированных топологий):



Результат:

```
Row 1:
──────
cluster:      my_cluster
shard_num:    1
replica_num:  1
host_name:    ch1
shard_weight: 1
errors_count: 0

Row 2:
──────
cluster:      my_cluster
shard_num:    1
replica_num:  2
host_name:    ch2
shard_weight: 1
errors_count: 0

Row 3:
──────
cluster:      my_cluster
shard_num:    2
replica_num:  1
host_name:    ch3
shard_weight: 1
errors_count: 0

Row 4:
──────
cluster:      my_cluster
shard_num:    2
replica_num:  2
host_name:    ch4
shard_weight: 1
errors_count: 0
```

SELECT

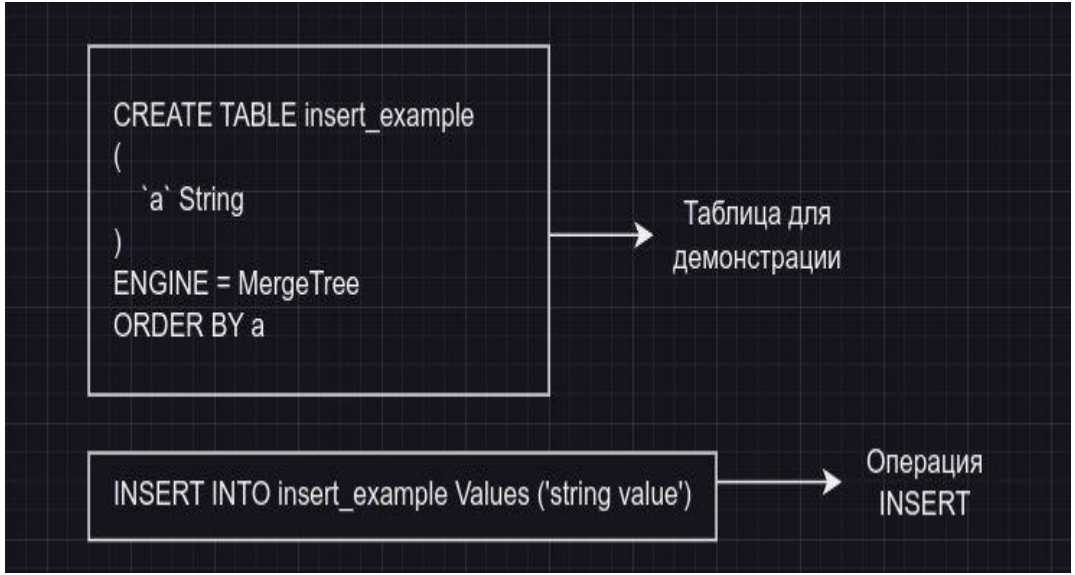
Полный синтаксис

```
[WITH expr_list|(subquery)]
SELECT [DISTINCT [ON (column1, column2, ...)]] expr_list
[FROM [db.]table | (subquery) | table_function] [FINAL]
[SAMPLE sample_coeff]
[ARRAY JOIN ...]
[GLOBAL] [ANY|ALL|ASOF] [INNER|LEFT|RIGHT|FULL|CROSS] [OUTER|SEMI|ANTI] JOIN (subquery)|table (ON <expr_list>)|(USING <column_list>)
[PREWHERE expr]
[WHERE expr]
[GROUP BY expr_list] [WITH ROLLUP|WITH CUBE] [WITH TOTALS]
[HAVING expr]
[ORDER BY expr_list] [WITH FILL] [FROM expr] [TO expr] [STEP expr] [INTERPOLATE [(expr_list)]]
[LIMIT [offset_value, ]n BY columns]
[LIMIT [n, ]m] [WITH TIES]
[SETTINGS ...]
[UNION  ...]
[INTO OUTFILE filename [COMPRESSION type [LEVEL level]] ]
[FORMAT format]
```

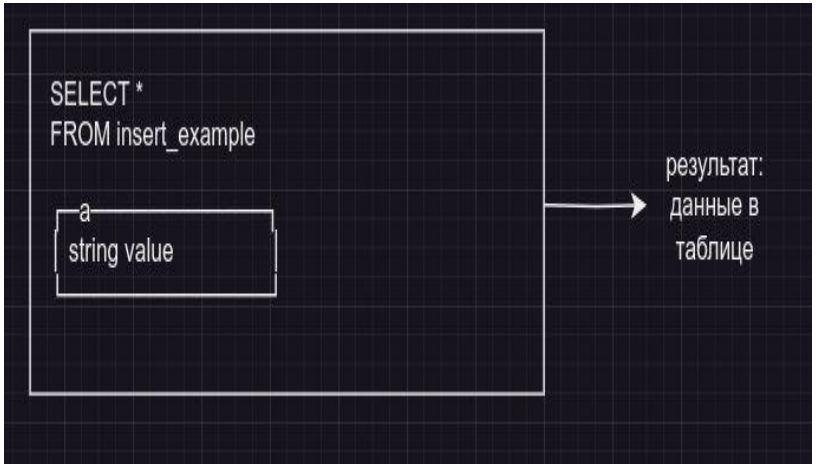
INSERT

Операция записи данных. Позволяет записать данные в ClickHouse.

Простой пример (вставить одну строку):



Результат:



INSERT

Полный синтаксис

```
INSERT INTO [TABLE] [db.]table [(c1, c2, c3)] [SETTINGS ...] VALUES (v11, v12, v13), (v21, v22, v23), ...
INSERT INTO [TABLE] [db.]table [(c1, c2, c3)] [SETTINGS ...] FORMAT format
INSERT INTO [TABLE] [db.]table [(c1, c2, c3)] [SETTINGS ...] SELECT ...
```

UPDATE

Операция обновления данных

Предостережение: в ClickHouse нет точечного обновления данных.
Операция обновления реализована как мутация, она требует свободного места не менее X2 от объема мутируемых партиций. Мутация требует достаточного времени и реализована - как полная перезапись всех данных партиций, затронутых обновлением.
Без указания IN PARTITION, будут перезаписаны ВСЕ данные таблицы.

Best practice для ClickHouse - не использовать ClickHouse для мутабельных данных.

Синтаксис:

```
ALTER TABLE [db.]table [ON CLUSTER cluster]
UPDATE column1 = expr1 [, ...]
[IN PARTITION partition_id] WHERE filter_expr
```

DELETE

Операция удаления данных

Предостережение: то же самое, что и с UPDATE

Best practice для ClickHouse - не использовать ClickHouse для мутабельных данных.

Синтаксис:

```
ALTER TABLE [db.]table [ON CLUSTER cluster]
DELETE
[IN PARTITION partition_id] WHERE filter_expr
```

Создание и удаление БД

синтаксис

Создание:

```
CREATE DATABASE [IF NOT EXISTS] db_name [ON CLUSTER cluster] [ENGINE = engine(...)] [COMMENT 'Comment']
```

Удаление:

```
DROP DATABASE [IF EXISTS] db [ON CLUSTER cluster] [SYNC]
```

Создание и удаление БД

модификаторы

IF NOT EXISTS / IF EXISTS: перед созданием удалением проверяет наличие отсутствие БД с таким именем. Не выдает ошибку и ничего не делает, если удаляемой базы уже нет, или если добавляемая база уже есть.

ON CLUSTER: требуется подключение к ZooKeeper. Добавляет в DISTRIBUTED DDL очередь задачу на выполнение такого же запроса на все реплики кластера cluster, описанного в конфигурации <remote_servers />.

ENGINE: Database Engine создаваемой базы.

SYNC: Меняет поведение по умолчанию при удалении таблиц для баз на Engine=Atomic. Поведение по умолчанию - запрос сразу возвращает успех, удаление происходит уже после выполнения запроса, в фоном режиме, через database_atomic_delay_before_drop_table_sec параметр конфигурации, секунд. Поведение с модификатором **SYNC** - отложенного удаления не происходит, запрос выполняется до полного удаления всех таблиц в базе, и затем самой базы.

Создание и удаление БД

ENGINE баз

Atomic: база данных по умолчанию. Структура хранения на диске через UUID таблиц, специальный запрос EXCHANGE TABLES, выполняющийся атомарно, неблокирующие операции RENAME/DROP TABLE, неблокирующие операции с партами и партициями.

Ordinary: legacy Engine для баз, простая структура хранения data/database/table/part, операции перечисленные выше в Atomic - блокирующие. Требует allow_deprecated_database_ordinary ключа конфигурации.

MySQL, **PostgreSQL**: специальные Engine, позволяющие отображать базы из других систем в ClickHouse.

MaterializedMySQL, **MaterializedPostgreSQL**: репликация данных из других систем.

Предупреждение:
- Engine экспериментальные
- операции удаления/изменения данных превращаются в фоновые операции ReplacingMergeTree, необходимо учитывать эту особенность при запросах.

Replicated: Engine с реплицируемыми запросами CREATE/DROP таблиц.

Предупреждение:
- экспериментальный

Lazy: только для таблиц Engine=Log. Держит в оперативной памяти самые свежие expiration_time_in_seconds данные

Создание и удаление таблиц

синтаксис

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1]

        [NULL|NOT NULL]

        [DEFAULT|MATERIALIZED|EPHEMERAL|ALIAS expr1]

        [compression_codec]

        [TTL expr1]

        [COMMENT 'comment for column'],
    name2 ...,
    ...
) ENGINE = engine
  COMMENT 'comment for table'
```

```
CREATE TEMPORARY TABLE [IF NOT EXISTS] table_name
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) [ENGINE = engine]
```

```
DROP [TEMPORARY] TABLE [IF EXISTS] [IF EMPTY] [db.]name [ON CLUSTER cluster] [SYNC]
```

Операции над партами и партициями

ALTER TABLE [db.]table ...

DROP PARTITION|PART: удаляет партицию/парт из таблицы. Поддерживает модификатор **SYNC** для database Engine=Atomic.

ATTACH PARTITION|PART: добавляет вручную подложенные в .../table_path/detached на файловой системе, партиции и парты

REPLACE PARTITION ... FROM table2: заменяет партицию копией из другой таблицы. Структуры таблиц должны совпадать.

DETACH PARTITION|PART: перемещает партицию/парт в каталог .../table_path/detached на файловой системе. **Предупреждение**:
- это произойдет на всех репликах шарда, легко об этом забыть и оставить мусор в .../table_path/detached реплик

Реальные примеры

ClickHouse

(https://clickhouse.com/customers) - здесь представлены реальные примеры использования ClickHouse в различных компаниях.

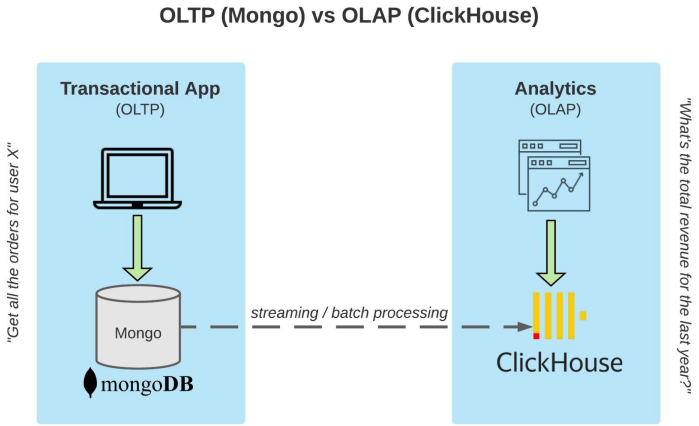
Реальные примеры. Object Storage

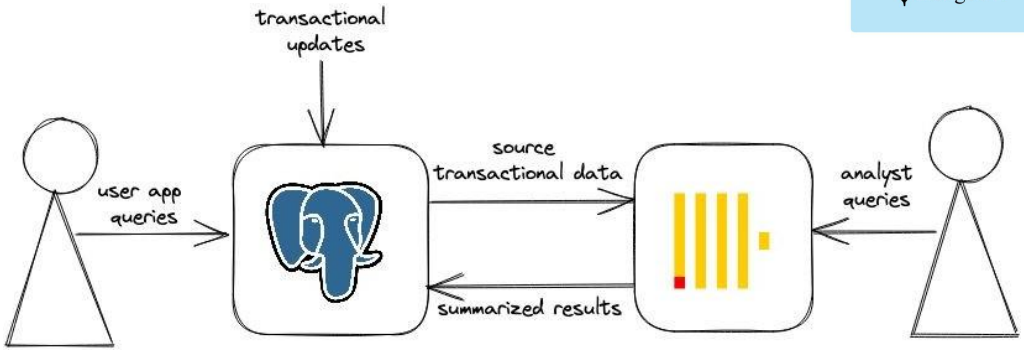
```
SELECT *
FROM
s3('https://datasets-documentation.s3.eu-west-3.amazonaws.com/nyc-taxi/trips_*.gz', 'TabSeparatedWithNames')
LIMIT 10;

CREATE TABLE s3_engine_table (name String, value UInt32)
    ENGINE = S3(path, [aws_access_key_id,
aws_secret_access_key,] format, [compression])
    [SETTINGS ...]
```

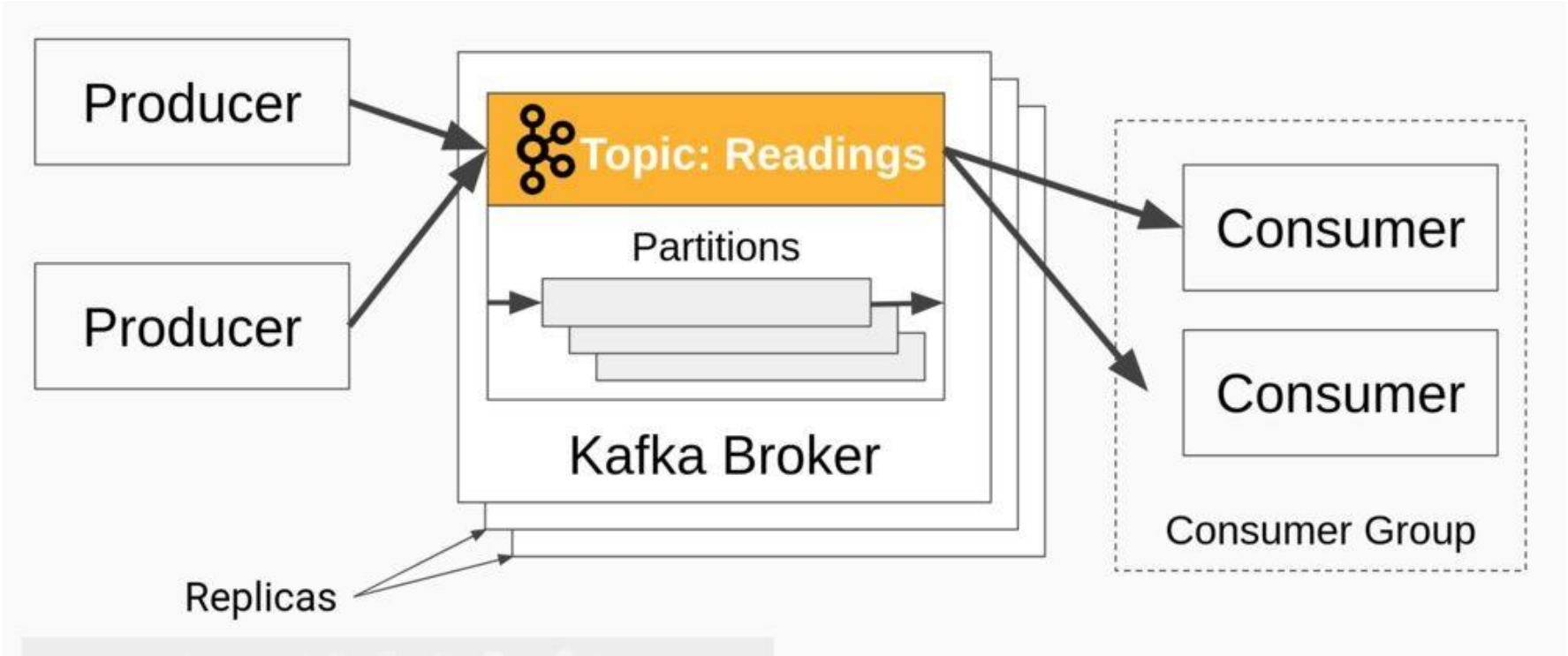
Реальные примеры. Интеграции с БД

- Postgres (Greenplum)
- MongoDB
- MySQL
- Redis

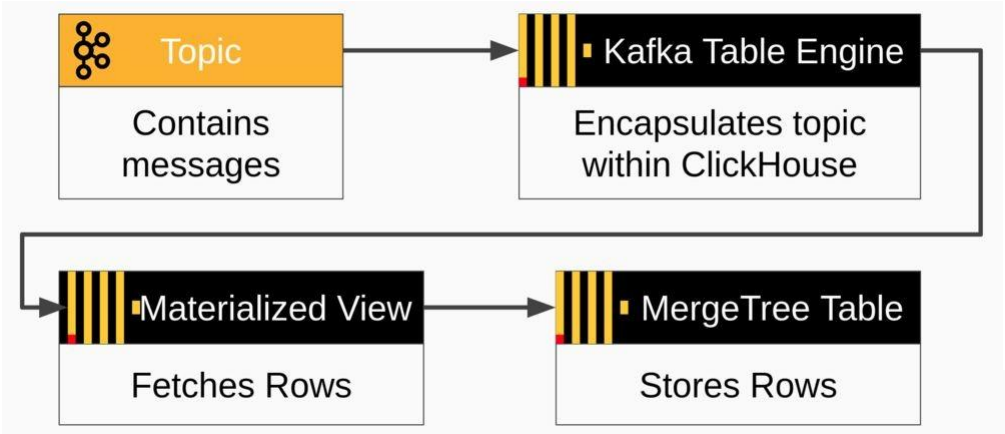


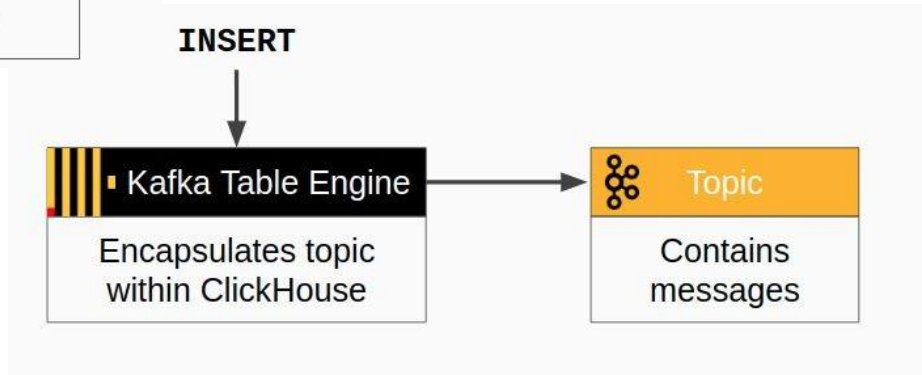


Реальные примеры. Apache Kafka



Реальные примеры. Apache Kafka





Реальные примеры. Визуализация данных

Многие из популярных BI-инструментов и средств визуализации подключаются к ClickHouse - как "из коробки", так и через установку коннектора.

Yandex Cloud CLI

- Install yc CLI
- yc CLI reference
- Yandex Object Storage CLI (AWS CLI)

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Скрипты для практики

https://disk.yandex.ru/d/BzSlenq0rwqJnA

Вопросы?

Задаем вопросы в чат

Ставим “–”, если вопросов нет

Рефлексия

Цели вебинара

Проверим достижение целей

1. познакомиться с ClickHouse
2. рассмотреть возможности ClickHouse
3. научиться искать ответы на свои вопросы
4. узнать особенности реализации SQL в ClickHouse
5. изучить нюансы и специфики

Список материалов для изучения

1. Shared Nothing Architecture Explained
2. Clickhouse vs. Greenplum. Какую MPP-базу данных выбрать? / Демо-занятие курса «Data Engineer»
3. What is an MPP Database? Intro to Massively Parallel Processing
4. Row vs Column Oriented Databases
5. https://clickhouse.tech/
6. https://www.altinity.com/blog
7. https://t.me/clickhouse_ru - официальный чат ClickHouse, самое популярное и активное место для получения поддержки и ответа на вопросы
8. https://clickhouse.com/docs - место, в которое отправляют в чате, когда не хотят отвечать на простые вопросы
9. https://github.com/ClickHouse/ClickHouse - код ClickHouse, а так же issue-трекер, с актуальными проблемами

Заполните, пожалуйста,
опрос о занятии

Спасибо за внимание!