ClickHouse для инженеров и архитекторов БД

Словари, оконные и табличные функции

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

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

Тема вебинара

Словари, оконные и табличные функции

Кариев Нурсултан

Clickhouse DBA в компании Азия Ритейл

*Преподаватель курса **ClickHouse для инженеров и архитекторов БД** в OTUS*

nursultankariev98@gmail.com

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в учебной группе
#OTUS ClickHouse-2024-08

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

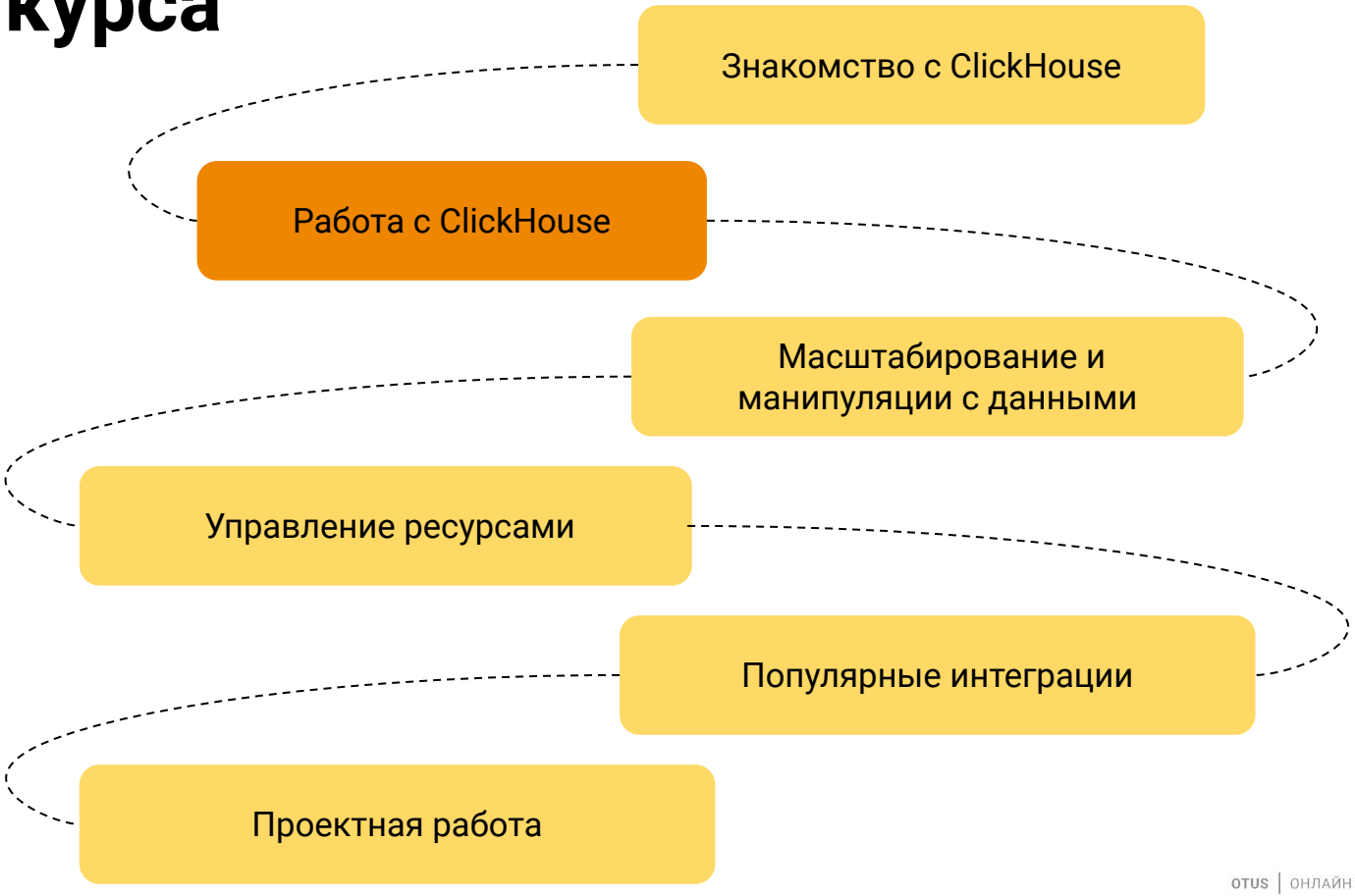
Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

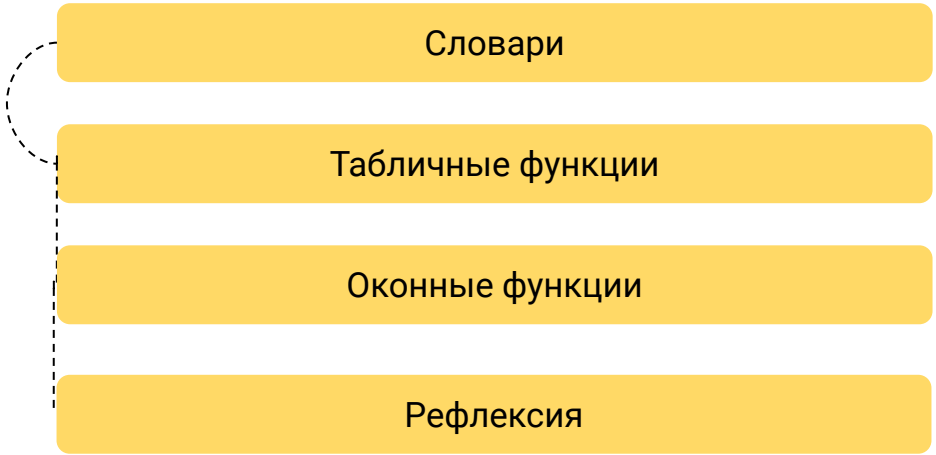
Карта курса


Знакомство с ClickHouse
Работа с ClickHouse
Масштабирование и манипуляции с данными
Управление ресурсами
Популярные интеграции
Проектная работа

Темы модуля «Работа с ClickHouse»

- Язык запросов SQL
- Функции для работы с типами данных, агрегатные функции и UDF
- Движки MergeTree Family
- Другие движки
- Джоины и агрегации
- Словари, оконные и табличные функции
- Сессия Q&A

Маршрут вебинара



Цели вебинара

К концу занятия вы сможете

1. создавать словари из разных источников
2. познакомиться с оконными функциями

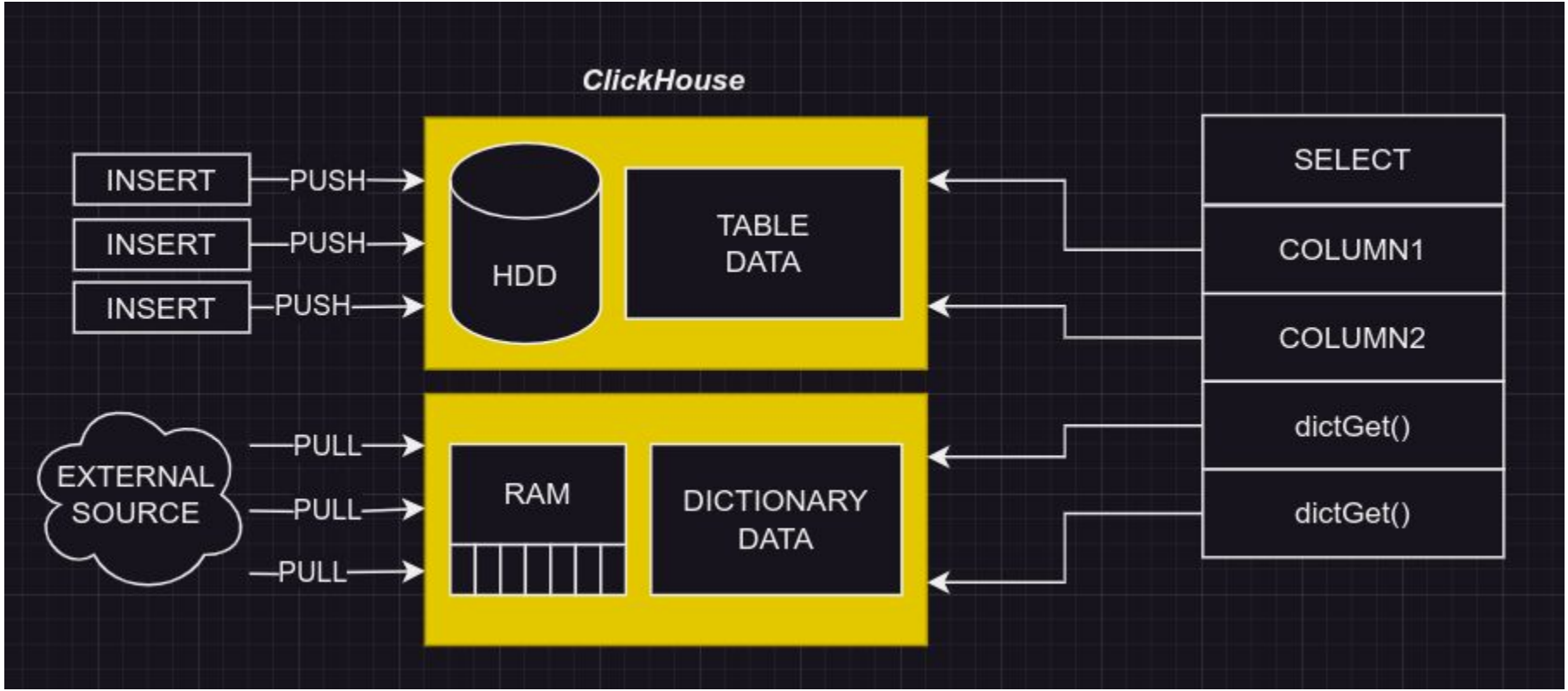
Смысл

Зачем вам это уметь

1. использовать данные других систем совместно с ClickHouse
2. использовать оконные функции в аналитических запросах

Словари

Что такое словарь



Словарь - данные из внешних систем, которые регулярно забираются ClickHouse-ом в оперативную память (как правило в оперативную, но не только), для использования совместно с собственными данными в запросах.

Структура словаря

KEY	ATTR1	ATTR2	ATTR3
key1	value	value	value
key2	value	value	value
key3	value	value	value
key4	value	value	value

Словарь представляет собой KEY-value хранилище, ключ может быть простым или составным.

Ключ при обращении к словарю указывается целиком.

Атрибутов может быть от 1 и больше, их значения можно получать по ключу.

Зачем нужны словари?

1) Мутабельные данные

В ClickHouse сохраняются события, например действия пользователя, за долгий период. События не подлежат изменению, они уже произошли, поэтому они отлично подходят для хранения и анализа в ClickHouse.

Однако, у пользователя есть так же меняющиеся свойства, такие как статус активности, баланс, семейное положение, и другие. Такие свойства в силу изменчивости не подходят к хранению в ClickHouse, однако так же относятся к этому пользователю.

Для совместного использования тех и других данных и применяются словари.

Зачем нужны словари?

2) Интеграции с другими системами

Описаный в п.1 сценарий, подразумевает хранение мутабельных данных в других системах, и как правило там они изначально и находятся, словари подходят для интеграции этих данных с данными ClickHouse.

Зачем нужны словари?

3) Производительная альтернатива JOIN

Даже если данные уже в ClickHouse, однако в разных таблицах, построение словаря вокруг меньшей таблицы или вокруг её агрегата является более производительным решением.

В реляционных системах мы привыкли JOIN-ить сотни и тысячи строк, иногда десятки тысяч, не принимая в расчет накладные расходы на построение hash-таблицы JOIN-а, и размер оперативной памяти и издержки по CPU необходимые для такой операции. Работая в ClickHouse с миллионами и миллиардами строк, очень быстро становится очевидна дороговизна такого подхода.

Как выглядит запрос

с использованием джоина:

```
select
  p.Id AS PostId,
  p.Title,
  u.username AS AuthorName
(SELECT * FROM posts
where date between ‘2024-01-01’ and today()) p
INNER JOIN users u ON p.OwnerUserId = u.id;
```

с использованием словарей:

```
SELECT
   Id AS PostId,
   Title,
   dictGet('users_dict', 'username', toUInt64(OwnerUserId)) AS AuthorName
FROM posts
WHERE date between ‘2024-01-01’ and today();
```

Как создать словарь

XML-конфигурация

```
<dictionary>
  <name>
    ИМЯ СЛОВАРЯ
  </name>
  <structure>
    КЛЮЧ И АТРИБУТЫ
  </structure>
  <source>
    ИСТОЧНИК
  </source>
  <layout>
    СХЕМА ХРАНЕНИЯ (ОБЫЧНО В ПАМЯТИ)
  </layout>
  <lifetime>
    КАК ЧАСТО ОБНОВЛЯТЬ
  </lifetime>
</dictionary>
```

SQL-синтаксис

```
CREATE DICTIONARY ИМЯ СЛОВАРЯ
(
   АТРИБУТЫ
)
PRIMARY KEY КЛЮЧ
SOURCE(ИСТОЧНИК)
LAYOUT(СХЕМА ХРАНЕНИЯ)
LIFETIME(КАК ЧАСТО ОБНОВЛЯТЬ)
```

В чем разница?

XML-конфигурация

Требует доступа к файловой системе сервера и может потребовать перезагрузки конфигурации сервера.

Требует прав на чтение файлов конфигурации на сервере

Если доступ к серверу ограничен, то этот вариант безопаснее со стороны иб

Изменения требуют редактирования файла и перезагрузки конфигурации.

Можно автоматизировать создание словарей через ansible

SQL-синтаксис

Управляется как обычный объект базы данных, не требует доступа к файловой системе.

Управляется через систему прав доступа SQL, как для других объектов базы данных

Можно изменять и обновлять словарь с помощью SQL-команд, как обычные объекты базы данных.

Работает в Clickhouse Cloud

Как создать словарь

XML-конфигурация

```
<clickhouse>
    <dictionary>
        <name>users_dict</name>
        <structure>
            <id>
                <name>id</name>
            </id>
            <attribute>
                <name>username</name>
                <type>String</type>
                <null_value></null_value>
            </attribute>
        </structure>
        <source>
            <clickhouse>
                <host>localhost</host>
                <port>9000</port>
                <user>default</user>
                <password></password>
                <db>default</db>
                <table>users</table>
            </clickhouse>
        </source>
        <layout>
            <flat />
        </layout>
        <lifetime>
            <min>300</min>
            <max>360</max>
        </lifetime>
    </dictionary>
</clickhouse>
```

SQL-синтаксис

```
CREATE DICTIONARY users_dict_1
(
    `id` UInt64,
    `username` String
)
PRIMARY KEY id
SOURCE(CLICKHOUSE(HOST 'localhost' PORT 9000
USER 'default' TABLE 'users' PASSWORD '' DB
'default'))
LIFETIME(MIN 300 MAX 360)
LAYOUT(FLAT())
```

Как создать словарь / ключ словаря

Простой ключ: одно поле, тип всегда UInt64.

Объявляется в XML как ***<id><name>имя</name></id>***

Составной ключ: несколько полей, заданные типы.

Объявляется в XML как ***<key>несколько полей</key>***, где каждое поле объявляется так же как атрибут:

```
<attribute>
    <name>поле</name>
    <type>тип</type>
</attribute>
```

Для SQL простой и составной ключи объявляются одинаково в списке полей и дополнительно перечисляются в PRIMARY KEY:

```
CREATE DICTIONARY имя (
    key1 String,
    key2 String
    ...
)
PRIMARY KEY key1, key2
...
```

Как создать словарь / атрибуты

Для XML: так же, как и атрибуты составного ключа, но уровнем выше:

```
<structure>
<key>
  <attribute><name>ключ</name><type>тип</type></attribute>
  ...
</key>
<attribute><name>атрибут1</name><type>тип</type></attribute>
<attribute><name>атрибут2</name><type>тип</type></attribute>
...
</structure>
```

Для SQL: просто перечисляем как не состоящие в ключе поля.

Как создать словарь / источник

В качестве источников поддерживается:

1) файл на файловой системе

для XML:

```
<source>
  <file>
    <path>/opt/dictionaries/os.tsv</path>
    <format>TabSeparated</format>
  </file>
</source>
```

для SQL:

```
SOURCE(FILE(path './user_files/os.tsv' format 'TabSeparated'))
```

Как создать словарь / источник

2) файл доступный по http/https

XML:

```
<source>
   <http>
      <url>http://url/os.tsv</url>
      <format>TabSeparated</format>
      необязательно: <credentials><user>...</user><password>...</password></credentials>
      необязательно: <headers> одна или несколько секций <header><name>...</name><value>...</value></header>
   </http>
</source>
```

SQL: SOURCE(HTTP(

```
   url 'http://[::1]/os.tsv'
   format 'TabSeparated'
   credentials(user 'user' password 'password')
   headers(header(name 'API-KEY' value 'key'))
))
```

Как создать словарь / источник

3) другие базы данных

здесь конфигурация через SQL и XML однотипна

3.1) odbc коннектор

```
SOURCE(ODBC(
    db 'DatabaseName'
    table 'SchemaName.TableName'
    connection_string 'DSN=some_parameters'
    invalidate_query 'SQL_QUERY'
    query 'SELECT id, value_1, value_2 FROM db_name.table_name'
))
```

db — имя базы данных. Не указывать, если имя базы задано в параметрах. <connection_string>.

table — имя таблицы и схемы, если она есть.

connection_string — строка соединения.

invalidate_query — запрос для проверки статуса словаря. Необязательный параметр.

query – пользовательский запрос. Необязательный параметр.

Поля table и query не могут быть использованы вместе. Также обязательно должен быть один из источников данных: table или query.

Как создать словарь / источник

3.2) mysql

```
SOURCE(MYSQL(
    port 3306
    user 'clickhouse'
    password 'qwerty'
    replica(host 'example01-1' priority 1)
    replica(host 'example01-2' priority 1)
    db 'db_name'
    table 'table_name'
    where 'id=10'
    invalidate_query 'SQL_QUERY'
    fail_on_connection_loss 'true'
    query 'SELECT id, value_1, value_2 FROM db_name.table_name'
))
```

port — порт сервера MySQL. Можно указать для всех реплик или для каждой в отдельности (внутри <replica>).

user — имя пользователя MySQL. Можно указать для всех реплик или для каждой в отдельности (внутри <replica>).

password — пароль пользователя MySQL. Можно указать для всех реплик или для каждой в отдельности (внутри <replica>).

replica — блок конфигурации реплики. Блоков может быть несколько.

db — имя базы данных.

table — имя таблицы.

where — условие выбора. Синтаксис условия совпадает с синтаксисом секции WHERE в MySQL, например, id > 10 AND id < 20. Необязательный параметр.

invalidate_query — запрос для проверки статуса словаря. Необязательный параметр. Читайте подробнее в разделе Обновление словарей.

fail_on_connection_loss – параметр конфигурации, контролирующий поведение сервера при потере соединения. Если значение true, то исключение генерируется сразу же, если соединение между клиентом и сервером было потеряно. Если значение false, то сервер повторно попытается выполнить запрос три раза прежде чем сгенерировать исключение. Имейте в виду, что повторные попытки могут увеличить время выполнения запроса. Значение по умолчанию: false.

query – пользовательский запрос. Необязательный параметр.

Как создать словарь / источник

3.3) postgresql

```
SOURCE(POSTGRESQL(
    port 5432
    host 'postgresql-hostname'
    user 'postgres_user'
    password 'postgres_password'
    db 'db_name'
    table 'table_name'
    replica(host 'example01-1' port 5432 priority 1)
    replica(host 'example01-2' port 5432 priority 2)
    where 'id=10'
    invalidate_query 'SQL_QUERY'
    query 'SELECT id, value_1, value_2 FROM db_name.table_name'
))
```

параметры как у mysql

Как создать словарь / источник

3.4) ClickHouse

```
SOURCE(CLICKHOUSE(
    host 'example01-01-1'
    port 9000
    user 'default'
    password ''
    db 'default'
    table 'ids'
    where 'id=10'
    secure 1
    query 'SELECT id, value_1, value_2 FROM default.ids'
))
```

аналогично mysql и postgresql, можно включить ssl флагом secure, нельзя указать несколько реплик, предлагается вместо этого использовать Distributed-таблицу

Как создать словарь / источник

3.5) MongoDB

```
SOURCE(MONGODB(
    host 'localhost'
    port 27017
    user ''
    password ''
    db 'test'
    collection 'dictionary_source'
))
```

ещё меньше параметров подключения, есть параметр collection - аналог таблицы в MongoDB

Как создать словарь / источник

3.6) Redis

```
SOURCE(REDIS(
    host 'localhost'
    port 6379
    storage_type 'simple'
    db_index 0
))
```

host – хост Redis.

port – порт сервера Redis.

storage_type – способ хранения ключей. Необходимо использовать simple для источников с одним столбцом ключей, hash_map – для источников с двумя столбцами ключей. Источники с более, чем двумя столбцами ключей, не поддерживаются. Может отсутствовать, значение по умолчанию simple.

db_index – номер базы данных. Может отсутствовать, значение по умолчанию 0.

Как создать словарь / источник

3.7) Cassandra

SOURCE(CASSANDRA(

параметры описываются аналогичным образом

))

host – имя хоста с установленной Cassandra или разделенный через запятую список хостов.

port – порт на серверах Cassandra. Если не указан, используется значение по умолчанию: 9042.

user – имя пользователя для соединения с Cassandra.

password – пароль для соединения с Cassandra.

keyspace – имя keyspace (база данных).

column_family – имя семейства столбцов (таблица).

allow_filtering – флаг, разрешающий или не разрешающий потенциально дорогостоящие условия на кластеризации ключевых столбцов. Значение по умолчанию: 1.

partition_key_prefix – количество партиций ключевых столбцов в первичном ключе таблицы Cassandra. Необходимо для составления ключей словаря. Порядок ключевых столбцов в определении словаря должен быть таким же, как в Cassandra. Значение по умолчанию: 1 (первый ключевой столбец - это ключ партицирования, остальные ключевые столбцы - ключи кластеризации).

consistency – уровень консистентности. Возможные значения: One, Two, Three, All, EachQuorum, Quorum, LocalQuorum, LocalOne, Serial, LocalSerial. Значение по умолчанию: One.

where – опциональный критерий выборки.

max_threads – максимальное количество тредов для загрузки данных из нескольких партиций в словарь.

query – пользовательский запрос. Необязательный параметр.

Схема хранения

объявляется

XML:
<схема>
параметры
</схема>

SQL: LAYOUT(схема(параметр1 значение параметр2 значение ...))

Схема хранения

flat схема

предназначена для хранения словарей с простым UInt64 ключом

параметры:

* initial_array_size - по умолчанию 1024, под сколько ключей резервировать память сразу

* max_array_size - по умолчанию 5000000, максимальное количество ключей, при превышении словарь не создается

самая простая и производительная схема, не эффективна по памяти

hashed

строит хеш-таблицу вокруга хеша ключа, параметров нет, несколько эффективнее по памяти

sparse_hashed

тоже самое, но ещё меньше потребление памяти ценой большей нагрузки на CPU

complex_key_hashed, **complex_key_sparse_hashed** - нет параметров, строит хеш-таблицу вокруг составного или не-UInt64 ключа

Схема хранения

cache

Словарь хранится в кэше, состоящем из фиксированного количества ячеек. Ячейки содержат часто используемые элементы.

При поиске в словаре сначала просматривается кэш. На каждый блок данных, все не найденные в кэше или устаревшие ключи запрашиваются у источника с помощью SELECT attrs... FROM db.table WHERE id IN (k1, k2, ...). Затем, полученные данные записываются в кэш.

Наименее производительный словарь, однако позволяет держать не полный набор данных, а только часто используемые.

параметры

* size_in_cells - размер кеша
* allow_read_expired_keys - возвращать устаревшие
* max_update_queue_size - очередь на обновление
* update_queue_push_timeout_milliseconds - таймаут для очереди
* query_wait_timeout_milliseconds - таймаут для обновления
* max_threads_for_updates - тредов для обновления

Схема хранения

complex_key_cache

аналогично **cache**, для составных ключей

ssd_cache, complex_key_ssd_cache

ключи в памяти, значения по ним на ssd, есть дополнительные параметры относящиеся к хранению на ssd:

* block_size
* file_size
* read_buffer_size, write_buffer_size
* path

Автообновление

объявляется

XML:

<lifetime>секунд</lifetime>

или

<lifetime> <min>секунд</min> <max>секунд</max> </lifetime>

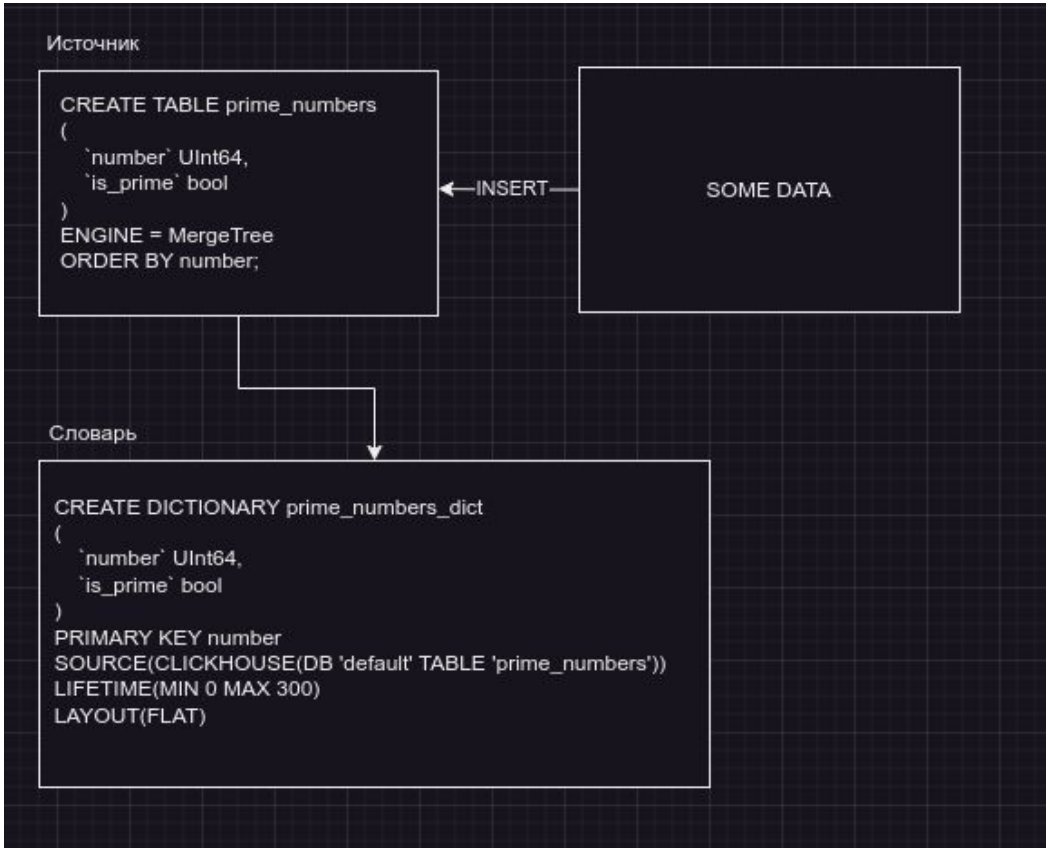
SQL: LIFETIME(секунд) или LIFETIME(MIN секунд MAX секунд)

задает период обновления словаря, в случае задания через min/max, период обновления рандомизируется между min и max

Как обращаться к словарю

Обращение к словарю выполняется функцией dictGet(), принимающей параметры:
dict_name - имя словаря
attr_names - имя атрибута, или Tuple имен атрибутов
id_expr - значение ключа по которому хотим получить атрибуты

Пример



Словарь с простыми числами.

Можно использовать в запросе

SELECT
dictGet(‘prime_numbers_dict’,’is_prime’,
number_column) AS is_prime,
другие колонки
FROM any_table_with_numbers

Информации о словарях

Детальная информация о существующих словарях можно узнать из таблицы system.dictionaries

dictionaries 1

```
select * from system.dictionaries
```

Введите SQL выражение чтобы отфильтровать результаты

Таблица

	database	name	uuid	status	origin	type	key.names	key.types
1	default	users_dict_1	ba7efaa1-2c4e-493d-b6ef-1f276051c64b	NOT_LOADED	ba7efaa1-2c4e-493d-b6ef-1f276051c64b		id	UInt64
2		users_dict	00000000-0000-0000-0000-0000000000	NOT_LOADED	/etc/clickhouse-server/users_dict_dictionary		id	UInt64

Структуру словаря созданного через SQL запрос можно получить через команды:

Show create dictionary dictionary_name;

Select create_table_query from system.tables
where name=’dictionary_name’

как рассчитать обьем словаря

Предварительно обьем словаря рассчитать невозможно

Для этого лучше всего построить небольшой словарь заполнив его данными

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Табличные функции

Что такое табличная функция

Табличная функция создает временную таблицу на лету, для выполнения над ней запроса, и может использоваться вместо таблицы в секции FROM запроса. Некоторые табличные функции позволяют так же вставку, например функция file().

Популярные табличные функции:

file('путь', 'формат', структура, сжатие)

пример: SELECT * FROM file('path/file.csv', 'CSV', 'column1 UInt32, column2 UInt32')

делает таблицу из файла по пути относительно <user_files> каталога в конфигурации.

путь - относительный путь до файла

формат - любой из понимаемых ClickHouse (https://clickhouse.com/docs/en/interfaces/formats)

структура - список «имя тип» для колонок, разделенный запятыми

сжатие - поддерживаются «gz, br, xz, zst, lz4, bz2», если не указано то никакое

fileCluster(cluster_name, ...) - то же самое что и file, но объединит файлы с каждой реплики указанного кластера

merge(['db_name',] 'tables_regexp') - позволяет поработать с несколькими таблицами отобранных по tables_regexp, из указанной базы, как с единой таблицей. Базу можно не указывать, тогда будет текущая. Можно указать в место базы так же regexp.

url(URL [,format] [,structure] [,headers]) - то же, что и file(), но позволяет получить по http, можно передавать свои заголовки в формате 'headers('key1'='value1', 'key2'='value2')'

s3(
URL,
[NOSIGN | access_key_id, secret_access_key, [session_token]],
[format] [,structure] [,compression_method])
) - как URL, только позволяет работать с s3

hdfs(URI, format, structure) - то же самое для HDFS

remote(addresses_expr, [db, table, user [, password], sharding_key]) - позволяет подключиться к другому ClickHouse или нескольким, объединив результаты с таблиц из нескольких ClickHouse

addresses_expr - разделенный запятыми список host:port строк

cluster, clusterAllReplicas - то же самое, но вместо ***addresses_expr*** топология кластера из <remote_servers> конфигурации. Для cluster берется одна реплика каждого шарда, для clusterAllReplicas все реплики.

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Оконные функции

Что такое оконная функция

Функция, позволяющая вычислять значения относительно диапазона других строк. Диапазон называется окном.

В ClickHouse объявляется конструкцией вида любая агретаная функция (например sum) и окно OVER

```
agg_func(столбец) OVER (
    PARTITION BY category
    ORDER BY purchases
    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
)
```

позволяющая вывести результат аггрегации в пределах OVER, где каждое уникальное значение PARTITION BY - самостоятельное окно, ORDER BY сортировка окна, ROWS диапазон. PARTITION BY и ROWS можно не указывать.

ROWS можно смотреть в диапазоне между «UNBOUNDED PRECEDING», «UNBOUNDED FOLLOWING», «CURRENT ROW»

Вместо UNBOUNDED можно указать количество строк вперед/назад

Пример

```
WITH subq AS
    (
        SELECT
            number,
            number % 2 AS odd
        FROM system.numbers
        LIMIT 10
    )
SELECT
    number,
    odd,
    sum(number) OVER (PARTITION BY odd ORDER BY number ASC) AS cum_sum_odd
FROM subq
ORDER BY number ASC

Query id: 9a55424e-92a7-484d-8da9-49326cb30499

┌number─┬─odd─┬─cum_sum_odd─┐
│      0 │   0 │           0 │
│      1 │   1 │           1 │
│      2 │   0 │           2 │
│      3 │   1 │           4 │
│      4 │   0 │           6 │
│      5 │   1 │           9 │
│      6 │   0 │          12 │
│      7 │   1 │          16 │
│      8 │   0 │          20 │
│      9 │   1 │          25 │
└────────┴─────┴─────────────┘

10 rows in set. Elapsed: 0.006 sec.
```

subq - подзапрос возвращающий 10 первых чисел, odd - признак четности

на основании признака четности odd строим окна сортированные по number

получаем коммулятивную сумму для четных, коммулятивную сумму для нечетных

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Домашнее задание

1) Создать таблицу с полями:

user_id UInt64,

action String,

expense UInt64

2) Создать словарь, в качестве ключа user_id, в качестве атрибута email String, источник словаря любой вам удобный, например file.

3) Наполнить таблицу и источник любыми данными, с низкоардинальными (неуникальными, повторяемыми) значениями для поля action и хотя бы по несколько повторящихся строк для каждого user_id

4) написать SELECT, возвращающий:

- email при помощи dictGet,

- аккамулятивную сумму expense, с окном по action

- сортировка по email

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Рефлексия

Цели вебинара

Проверка достижения целей

1. создавать словари из разных источников
2. познакомиться с табличными функциями
3. познакомиться с оконными функциями

Вопросы для проверки

1. какой layout вы выберете для составного ключа в условиях нехватки оперативной памяти
2. что такое «окно» в понятии оконной функции
3. где мы можем посмотреть информацию о словарях

Рефлексия

С какими впечатлениями уходите с вебинара?

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

Следующий вебинар

11 ноября 2024

Сессия Q&A

- 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

Кариев Нурсултан

Clickhouse DBA в компании Азия Ритейл

*Преподаватель курса **ClickHouse для инженеров и архитекторов БД** в OTUS*

nursultankariev98@gmail.com