Яндекс

Яндекс

О чём не прочтёшь в документации

Мансурова Мария,
аналитик Яндекс.Метрики

ClickHouse в 2 словах

- Столбцовая СУБД
- Лучше всего подходит для OLAP
- Разработана в команде Яндекс.Метрики, с 2016 года в open source
- SQL-диалект на «стероидах»

Документация

ClickHouse

Quick search

Go

Switch to Russian

Single page documentation
Website home
Source code
Edit this page

Introduction
Getting started
Interfaces
Query language
Table engines

Documentation

- Introduction
  - What is ClickHouse?
  - Distinctive features of ClickHouse
    - True column-oriented DBMS
    - Data compression
    - Disk storage of data
    - Parallel processing on multiple cores
    - Distributed processing on multiple servers
    - SQL support
    - Vector engine
    - Real time data updates
    - Indexes
    - Suitable for online queries
    - Support for approximated calculations
    - Data replication and support for data integrity on replicas

https://clickhouse.yandex/docs/en/

10 трюков ClickHouse

#1 Большие данные и точные подсчеты

- `Memory limit exceeded` - одна из самых частых ошибок при работе с большими данными в ClickHouse
- Решение - семплирование

#1 Семплирование

```
SELECT
    count()*10 as visits,
    sum(PageViews)*10 as hits,
    uniq(UserID)*10 as users,
    URL as url
FROM visits_table SAMPLE 1/10
```

#1 Точность - наше всё

```
SELECT
    count() as visits,
    sum(PageViews) as hits,
    uniq(UserID) as users,
    URL as url
FROM visits_table SAMPLE 1/10 OFFSET {i}/10
-- где i = 0, 1, …, 9
GROUP BY url
```

#1 Это лишь одно из решений

- Доклад Алексея Миловидова **«GROUP BY и Memory limit exceeded»** содержит ещё много способов

http://bit.ly/ch_memory_limit

#2 Интервалы для дат

```
SELECT
    today(),
    today() - 7 AS prev_week_date,
    now(),
    now() - 7 * 24 * 60 * 60 AS prev_week_date_time
```

```
┌─today()─┬─prev_week_date─┬──────────────now()─┬─prev_week_date_time─┐
│ 2017-12-08 │     2017-12-01 │ 2017-12-08 16:00:27 │ 2017-12-01 16:00:27 │
└────────────┴────────────────┴─────────────────────┴─────────────────────┘
```

#2 Прошлый месяц

```
SELECT
    today(),
    toStartOfMonth(today() - toDayOfMonth(today())) AS prev_month_start,
    prev_month_start + toDayOfMonth(today() - 1) AS prev_month
```

```
┌─today()─┬─prev_month_start─┬─prev_month─┐
│ 2017-12-08 │       2017-11-01 │ 2017-11-08 │
└────────────┴──────────────────┴────────────┘
```

#2 Функция INTERVAL

```
SELECT
    today(),
    today() - INTERVAL 1 MONTH AS prev_month

┌─today()──┬─prev_month─┐
│ 2017-12-08 │ 2017-11-08 │
└──────────┴────────────┘
```

#3 Фильтрация по интервалу дат

```
SELECT
    count() as visits,
    sum(PageViews) as hits,
    uniq(UserID) as users
FROM visits_table
WHERE (Date >= toDate('2017-06-01'))
    AND (Date <= toDate('2017-08-31'))
```

#3 BETWEEN

```
SELECT
    count() as visits,
    sum(PageViews) as hits,
    uniq(UserID) as users
FROM visits_table
WHERE Date BETWEEN ‘2017-06-01’ AND ‘2017-08-31’
```

#4 Условия и преобразования

- `if(condition, true_value, false_value)` - простой условный оператор
- `transform(value, array_from, array_to, default)` - пригодится, если нужна простая трансформация
- встроенные словари и функции `dictGet*`

#4 Расшифруем UserAgent

```
SELECT
    if(UserAgent LIKE '%Yabrowser%', 'yabrowser',
      if(UserAgent LIKE '%Firefox%', 'firefox',
         if(UserAgent LIKE '%Opera%', 'opera',
            if(UserAgent LIKE '%Chrome%', 'google_chrome', 'other')
         )
      )
    ) as browser,
    count() as visits
FROM visits_table
GROUP BY browser
```

#4 Функция multiIf

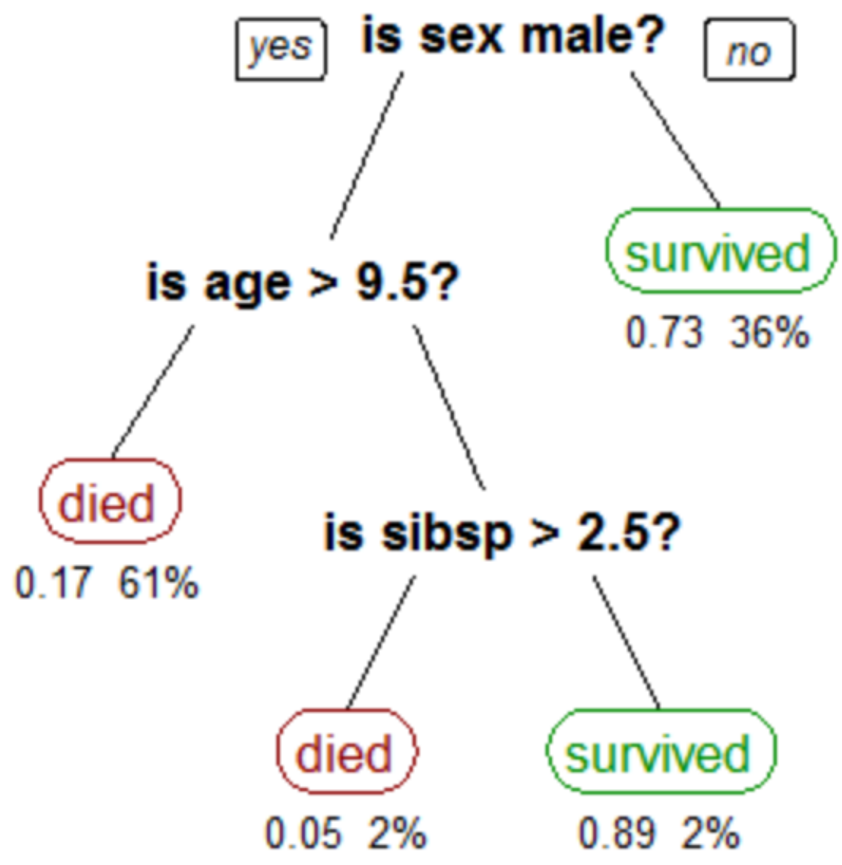
```
SELECT
    multiIf(
        UserAgent LIKE '%Yabrowser%', ‘yabrowser',
        UserAgent LIKE '%Firefox%', 'firefox',
        UserAgent LIKE '%Opera%', 'opera',
        UserAgent LIKE '%Chrome%', 'google_chrome',
        ‘other’
    ) as browser,
    count() as visits
FROM visits_table
GROUP BY browser
```

#5 ML в ClickHouse

- Было бы удобно применять модели, не выгружаю данные из ClickHouse

#5 Деревья принятия решений

- хорошо интерпретируемая модель - анкета/скрипт
- могут восстанавливать сложные нелинейные зависимости
- легко переобучаются



#5 Дерево принятия решений на CH

```
SELECT *,
    multiIf(
        Sex = ‘female’, 1,
        Age > 9.5, 0,
        SibSp > 2.5, 0, 1
    ) as survived
FROM titanic_data
```

#5 Градиентный бустинг

Градиентный бустинг - это ансамбль деревьев принятия решений.

- Можно также описать output в виде арифметических и условных операторов в ClickHouse, но будет много кода.

#5 Применение модель CatBoost

- `CatBoost` - open-source framework от Яндекса, основанный на градиентном бустинге
- применяем модель как `modelEvaluate( ‘model_name’, feature1, …, featureN)`

#5 Все подробности

- Доклад Николая Кочетова
  «Open source в Яндексе:
  Применение моделей CatBoost
  в ClickHouse»
  http://bit.ly/ch_and_catboost_pres
- **Tutorial** http://bit.ly/
  ch_and_catboost_tutorial

#6 Модификаторы агрегатных функций

- `If` - позволяет откинуть часть строк при расчете агрегатной функции
- `Array` - работает с элементами массивов как с значениями
- `ForEach` - работает на уровне элементов массива

#6 Модификатор -If

```
SELECT
    uniqIf(UserID, Browser = ‘YandexBrowser’) as browser_users,
    uniq(UserID) as total_users,
    countIf(Browser = ‘YandexBrowser’) as browser_visits,
    count() as total_visits,
    round(100*browser_users/total_users, 2) as users_share,
    round(100*browser_visits/total_visits, 2) as visits_share
FROM visits_table
```

#6 Модификатор -Array

```
SELECT
    sumArray(a) AS sum1,
    sumArrayArray(b) AS sum2
FROM
(
    SELECT
        [1, 2, 3, 4, 5, 6] AS a,
        [[1, 2], [3], [4, 5, 6]] AS b
)

┌─sum1─┬─sum2─┐
│   21 │   21 │
└──────┴──────┘
```

#6 Модификатор -ForEach

```
SELECT
    avgForEach(test_results),
    quantileForEach(0.5)(test_results)
FROM
(
    SELECT arrayJoin([[1, 1, 1, 1, 1], [1, 0, 1, 1, 1], [1, 0, 1, 0, 0], [1,
    1, 1, 1, 1], [1, 1, 1, 0, 0], [0, 0, 0, 0, 0], [1, 1, 0, 0, 1], [0, 1,
    1, 1, 0], [1, 1, 1, 0, 1], [1, 0, 0, 0, 0]]) AS test_results
)

┌─avg_results─────────────────┬─median_results─┐
│ [0.8,0.6,0.7,0.4,0.5] │ [1,1,1,0,0.5]  │
└─────────────────────────────┴────────────────┘
```

#7 arrayReduce

```
SELECT
    arrayReduce('avg', a) AS avg_arr,
    arrayReduce('median', a) AS median_arr,
    arrayReduce('groupUniqArray', a) AS uniq_arr
FROM
(
    SELECT arrayJoin([[1, 2, 1], [1, 2, 3, 2, 1]]) AS a
)

┌─────────────avg_arr─┬─median_arr─┬─uniq_arr─┐
│ 1.3333333333333333 │          1 │ [2,1]    │
│                1.8 │          2 │ [2,1,3]  │
└────────────────────┴────────────┴──────────┘
```

#8 WITH

```
WITH
    splitByChar(',', replaceRegexpAll(visitParamExtractRaw(data, 'products'), '[\\[\\]"]', ''))
        AS products,
    arrayMap(x -> toInt64(x), splitByChar(',', replaceRegexpAll(visitParamExtractRaw(data, 'prices'),
     '[\\[\\]"]', ''))) AS prices
SELECT
    products, prices, arraySum(prices) AS revenue, length(products) AS num_products
FROM
(SELECT arrayJoin([
     '{"id":1,"products":["coffee"],"prices":[300]}',
     '{"id":2,"products":["tea","croissant"],"prices":[200,150]}']) AS data)

┌─products──────────────────┬─prices──────┬─revenue─┬─num_products─┐
│ ['coffee']                │ [300]       │     300 │            1 │
│ ['tea','croissant']       │ [200,150]   │     350 │            2 │
└───────────────────────────┴─────────────┴─────────┴──────────────┘
```

#8* Более простой парсинг массивов

```
WITH
    CAST(replaceAll(visitParamExtractRaw(data, 'products'), '"', '\'') AS Array(String)) AS products,
    CAST(visitParamExtractRaw(data, 'prices') AS Array(Int)) AS prices
SELECT
    products, prices, arraySum(prices) AS revenue, length(products) AS num_products
FROM
(SELECT arrayJoin([
    '{"id":1,"products":["coffee"],"prices":[300]}',
    '{"id":2,"products":["tea","croissant"],"prices":[200,150]}']) AS data)


┌─products─────────────┬─prices─────┬─revenue─┬─num_products─┐
│ ['coffee']           │ [300]      │     300 │            1 │
│ ['tea','croissant'] │ [200,150] │     350 │            2 │
└──────────────────────┴────────────┴─────────┴──────────────┘
```

#9 Слияние массивов v.1

```
SELECT
    splitByChar(',',
        concat(
            arrayStringConcat(col1, ','),
            concat(',',
                arrayStringConcat(col2, ‘,’))
     )) AS array_concat
FROM
(
    SELECT ['a', 'b'] AS col1, ['c', 'd', 'e'] AS col2
)

┌─array_concat─────────┐
│ ['a','b','c','d','e'] │
└──────────────────────┘
```

#9 Слияние массивов v.2

```
SELECT
    arrayMap(
        i -> if(i <= length(col1), col1[i], col2[(i - length(col1))]),
        arrayEnumerate(range(length(col1) + length(col2)))
    ) AS array_concat
FROM
(
    SELECT
        ['a', 'b'] AS col1,
        ['c', 'd', 'e'] AS col2
)

┌─array_concat─────────┐
│ ['a','b','c','d','e'] │
└──────────────────────┘
```

#9 Слияние массивов v.3

```
SELECT groupArray(arrayJoin(arrayJoin([col1, col2]))) AS array_concat
FROM
(
    SELECT
        ['a', 'b'] AS col1,
        ['c', 'd', 'e'] AS col2
)
```

```
┌─array_concat─────────┐
│ ['a','b','c','d','e'] │
└──────────────────────┘
```

#9 Еще немного про слияние массивов

```
SELECT arrayReduce('groupArrayArray', [col1, col2])
FROM
(
    SELECT
        ['a', 'b'] AS col1,
        ['c', 'd', 'e'] AS col2
)
```

```
┌─array_concat──────────┐
│ ['a','b','c','d','e'] │
└───────────────────────┘
```

#9 Слияние массивов `arrayConcat`

```
SELECT arrayConcat(col1, col2) AS array_concat
FROM
(
    SELECT
        ['a', 'b'] AS col1,
        ['c', 'd', 'e'] AS col2
)
```

```
┌─array_concat─────────┐
│ ['a','b','c','d','e'] │
└──────────────────────┘
```

Хочу все знать!

Чтобы быть в тренде

- ClickHouse meetup’ы
- telegram-чат сообщества
- change log’и новых версий
- следить за коммитами на GitHub’е

And one more thing…

#10 ClickHouse все знает сам

```
SELECT *
FROM system.functions
WHERE lower(name) LIKE ‘%sort%'
```

```
┌─name─────────────┬─is_aggregate─┐
│ arraySort        │            0 │
│ arrayReverseSort │            0 │
└──────────────────┴──────────────┘
```

#10 Проверяем

```
WITH [4, 1, 5, 3, 2] AS arr
SELECT
    arr,
    arraySort(arr) AS sorted_arr,
    arrayReverseSort(arr) AS rev_sorted_arr
```

```
┌─arr─────────┬─sorted_arr──┬─rev_sorted_arr─┐
│ [4,1,5,3,2] │ [1,2,3,4,5] │ [5,4,3,2,1]    │
└─────────────┴─────────────┴────────────────┘
```

Контакты


https://clickhouse.yandex/


Google Group groups.google.com/forum/#!forum/clickhouse


clickhouse-feedback@yandex-team.ru


@clickhouse_ru


github.com/yandex/ClickHouse


@ClickHouseDB #clickhouse