ClickHouse 聚合函数

作者：flowaters
原文链接：https://ld246.com/article/1516782830993
来源网站：链滴

背景

ClickHouse中内置许多标准SQL外的函数。

这里记录一下其中聚合函数的学习笔记。

聚合函数

常用函数

count

返回记录条数。

如

```
SELECT count() FROM table
```

注：如果求 COUNT(DISTINCT x)，则使用uniq函数

any(x)

返回遇到的第一个值

备注：待补充

anyHeavy(x)

通过 heavy hitters算法，得到一个经常出现的值.

示例

```
SELECT anyHeavy(AirlineID) AS res
FROM ontime
```

anyLast(x)

返回最后遇到的值

min(x)

返回最小值

max(x)

返回最大值

argMin(arg,val)

TBD

argMax(arg,val)

TBD

sum(x)

求和

仅对数值有效

sumWithOverflow(x)

求和

如果超过上限，则报错

sumMap(key,value)

对于相同的key，对其value求和

示例

输入

```
CREATE TABLE sum_map(
    date Date,
    timeslot DateTime,
    statusMap Nested(
        status UInt16,
        requests UInt64
    )
) ENGINE = Log;
INSERT INTO sum_map VALUES
    ('2000-01-01', '2000-01-01 00:00:00', [1, 2, 3], [10, 10, 10]),
    ('2000-01-01', '2000-01-01 00:00:00', [3, 4, 5], [10, 10, 10]),
    ('2000-01-01', '2000-01-01 00:01:00', [4, 5, 6], [10, 10, 10]),
    ('2000-01-01', '2000-01-01 00:01:00', [6, 7, 8], [10, 10, 10]);
SELECT
    timeslot,
    sumMap(statusMap.status, statusMap.requests)
FROM sum_map
GROUP BY timeslot
```

查询

```
┌────────────timeslot─┬─sumMap(statusMap.status, statusMap.requests)─┐
│ 2000-01-01 00:00:00 │ ([1,2,3,4,5],[10,10,20,10,10])               │
│ 2000-01-01 00:01:00 │ ([4,5,6,7,8],[10,10,20,10,10])               │
└─────────────────────┴──────────────────────────────────────────────┘
```

avg(x)

平均值

uniq(x)

求近似的排重值, 适用于数值，字符串，日期，日期时间，以及 multiple 和 tuple 参数

使用了自适合抽样算法：在 65535下非常准确和高效。

uniqCombined(x)

求近似的排重值，适用于数值，字符串，日期，日期时间，以及 multiple 和 tuple 参数

使用了组合的三个算法：数组，hash表和HyperLogLog表。

内存消耗比uniq要小几倍，同时精度高几倍，但是性能一般比uniq慢，虽然有时也会快。最大支持到6KiB个状态.

uniqHLL12(x)

使用HyperLogLog算法来做排重值的近似计算，支持 2.5KB个状态.

uniqExact(x)

计算精确的排重值.

groupArray(x), groupArray(max_size)(x)

创建一个数组值

我自己的示例

```
:) select str_nest.y from schema_nested;

SELECT str_nest.y
FROM schema_nested

┌─str_nest.y──────────────────────┐
│ ['search','mock1@a.com'] │
│ ['image','mock1@a.com']  │
│ ['video','mock2@a.com']  │
│ ['image','mock2@a.com']  │
└─────────────────────────────────┘

┌─str_nest.y──────────────────────────────────────┐
│ ['search','mock1@a.com','news']         │
│ ['image','mock1@a.com','search','news'] │
└─────────────────────────────────────────────────┘

┌─str_nest.y──────────────────────────────────────┐
│ ['video','mock2@a.com']                 │
│ ['image','mock2@a.com','music','news'] │
```

原文链接：ClickHouse 聚合函数

```
└──────────────────────────────────────┘

8 rows in set. Elapsed: 0.003 sec.

:) select groupArray(str_nest.y) from schema_nested

SELECT groupArray(str_nest.y)
FROM schema_nested

┌─groupArray(str_nest.y)──────────────────────────────────────────────────────┐
│ [['search','mock1@a.com'],['image','mock1@a.com'],['video','mock2@a.com'],['image','mock2@a.com'],['search','mock1@a.com','news'],['image','mock1@a.com','search','news'],['video','mock2@a.com'],['image','mock2@a.com','music','news']] │
└─────────────────────────────────────────────────────────────────────────────┘

1 rows in set. Elapsed: 0.003 sec.

:) select groupArray(3)(str_nest.y) from schema_nested

SELECT groupArray(3)(str_nest.y)
FROM schema_nested

┌─groupArray(3)(str_nest.y)──────────────────────────────────────────┐
│ [['search','mock1@a.com'],['image','mock1@a.com'],['video','mock2@a.com']] │
└────────────────────────────────────────────────────────────────────┘

1 rows in set. Elapsed: 0.002 sec.
```

groupArrayInsertAt

TBD

groupUniqArray(x)

生成排重的数组，内存消耗与uniqExact方法相同

```
:) select groupUniqArray(str_nest.y) from schema_nested

SELECT groupUniqArray(str_nest.y)
FROM schema_nested

┌─groupUniqArray(str_nest.y)──────────────────────────────────────────┐
```

```
| [['image','mock2@a.com','music','news'],['image','mock1@a.com'],['image','mock2@a.com'],[
video','mock2@a.com'],['image','mock1@a.com','search','news'],['search','mock1@a.com'],['se
rch','mock1@a.com','news']] |
└──────────────────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────────────────────
──────────────────────────────┘

1 rows in set. Elapsed: 0.004 sec.
```

quantile(level)(x)

估算百分位点值.

level是一个0至1之间的常数，但是不要为0或者1. 0或者1时，使用min,max方法最合适.

quantileDeterministic(level)(x, determinator)

TBD

quantileTiming(level)(x)

TBD

quantileTimingWeighted(level)(x, weight)

TBD

quantileExact(level)(x)

TBD

quantileExactWeighted(level)(x, weight)

TBD

quantileTDigest(level)(x)

TBD

median

求中位数

每一个百分位点方法，都对应着一个求中位数的方法，如

median, medianDeterministic, medianTiming, medianTimingWeighted, medianExact, medianExactWeighted, medianTDigest

quantiles(level1, level2, ...)(x)

求多个百分位点

也对应着一系列的方法 quantiles, quantilesDeterministic, quantilesTiming, quantilesTimingWeighted, quantilesExact, quantilesExactWeighted, quantilesTDigest

varSamp(x)

求方差，结果是随机变量的方差的无偏估计

varPop(x)

求总体方差

stddevSamp(x)

为 varSamp(x) 的平均根

stddevPop(x)

为 varPop(x) 的平均根

topK

使用了Filtered Space-Saving算法来计算 topK

同时使用了Parallel Space Saving算法来进行reduce and combine.

topK(N)(column)

```
:) select topK(3)(str_nest.y) from schema_nested

SELECT topK(3)(str_nest.y)
FROM schema_nested

┌─topK(3)(str_nest.y)─────────────────────────────────────────────────────────────────────┐
│ [['video','mock2@a.com'],['search','mock1@a.com'],['image','mock1@a.com']] │
└─────────────────────────────────────────────────────────────────────────────────────────┘

1 rows in set. Elapsed: 0.004 sec.
```

covarSamp(x,y)

计算协方差

covarPop(x,y)

计算总体协方差

corr(x,y)

计算Pearson相关系数

含参聚合函数

学习自官方文档

sequenceMatch(pattern)(time, cond1, cond2, ...)

按顺利先后匹配事件

- pattern: 类似正则表达式的匹配规则
- time: 事件发生时间
- cond1, cond2: 最多32个，来标识条件是否满足

返回值

- 0：不匹配
- 1: 匹配

示例

sequenceMatch ('(?1).*(?2)')(EventTime, URL LIKE '%company%', URL LIKE '%cart%')

是否存在事件链，即先访问了company的事件，再访问了cart的事件

也可以通过别的聚合方法来表示

minIf(EventTime, URL LIKE '%company%') < maxIf(EventTime, URL LIKE '%cart%')

sequenceCount(pattern)(time, cond1, cond2, ...)

逻辑与sequenceMatch函数一样，但是返回值为chain的编号.

TBD

uniqUpTo(N)(x)

返回N个排重值数目，如果实际值大于N，则返回N+1

聚合函数后缀

-If

任何聚合函数都可以通过增加后缀If, 来增加一个外部参数条件. 聚合函数仅处理满足条件的记录行. 如

条件一次也没有触发，则返回默认值(通常是0或者是空字符串)

这样可以一次处理多个聚合条件，而不需要转换为子查询或者JOIN来计算.

示例

sumIf(column, cond), countIf(cond), avgIf(x, cond), quantilesTimingIf(level1, level2)(x, cond), agMinIf(arg, val, cond)

-Array

任何聚合函数都可以通过增加后缀Array, 来使原来的参数类型T变为新的参数类型Array(T)。

在处理时，原来的聚合函数会依次处理Array中的每一项.

示例

sumArray(arr) : 对Arrays中的所有元素进行求和，即sum(arraySum(arr))

uniqArray(arr): 对Arrays中的所有元素进行排重值，即uniq(arrayJoin(arr))

-If 和 -Array 可以组合使用，但是Array必须在前，If在后。例如

uniqArrayIf(arr, cond)

quantilesTimingArrayIf(level1, level2)(arr, cond)

-State

返回的不是结果值，返回的是中间状态. 这个是与AggregatingMergeTree来配合使用的.

-Merge

聚合函数会把中间状态会为参数，进行Merge，来完成聚合，返回最终的结果值.

-MergeState

与-Merge类似，但是返回的不是结果值，而是类似于-State的中间状态.

-ForEach

将对table使用的聚合函数，转换为对数组的聚合函数。对数组的每一项进行处理，返回一个结果数。

如:

sumForEach([1,2],[3,4,5],[6,7])

结果为

[10,13,5]