Pattern discovery with Clickhouse

Konstantin Ignatov

Qrator Labs

5 oct 2017

What is a pattern?

- Frequent subset...
- Frequent substring...
- Frequent subsequence...
- Frequent subgraph...

in a list of sets, strings, graphs or in a huge web-like graph

Toy example

```
SELECT *
FROM atest
```

```
┌─id─┬─es──────────┐
│  1 │ [2,3,4,5] │
│  2 │ [1,2,4,5] │
│  3 │ [4,5,7,8] │
│  4 │ [1,5,7,8] │
└────┴─────────────┘
```

Why mine them?

- To know what products are frequently bought together
- To know what users usually do
- To document business processes
- To get to know complex systems

Is it hard?

- Typical algorithm will do multiple full scans of the table
- Data are usually transformed using intermediate tables
- Most authors try to operate data with SQL avoiding large fetches
- ...mostly they fail
- ...some of them were/are trying to create DB suitable for mining

Let's do it in ClickHouse

First, transform data to ≪vertical format≫

```
CREATE TABLE vert ENGINE = TinyLog AS
SELECT
    arrayJoin(es) AS element,
    groupArray(id) AS row_ids
FROM atest
GROUP BY element
```

```
SELECT *
FROM vert

┌─element─┬─row_ids─┐
│       1 │ [2,4]   │
│       2 │ [1,2]   │
│       3 │ [1]     │
│       4 │ [1,2,3] │
│       5 │ [1,2,3,4] │
│       7 │ [3,4]   │
│       8 │ [3,4]   │
└─────────┴─────────┘
```

First iteration

Just to show the direction

```
CREATE TABLE vert1 ENGINE = TinyLog AS
SELECT
    [arrayJoin(es)] AS elements,
    groupArray(id) AS row_ids
FROM atest
GROUP BY elements
WHERE length(row_ids) > 1
```

```
SELECT *
FROM vert1

┌─elements─┬─row_ids─────┐
│ [7]      │ [3,4]       │
│ [5]      │ [1,2,3,4]   │
│ [4]      │ [1,2,3]     │
│ [1]      │ [2,4]       │
│ [3]      │ [1]         │
│ [8]      │ [3,4]       │
│ [2]      │ [1,2]       │
└──────────┴─────────────┘
```

Second iteration

Something useful this time

```
CREATE TABLE vert2 ENGINE = TinyLog AS
SELECT
    pattern AS elements,
    arrayFilter(elem -> has(row_ids, elem), row_ids_old) AS row_ids
FROM vert
ALL INNER JOIN
(
    SELECT
        arrayReduce('groupArrayArray', [elements, [element]]) AS pattern,
        element,
        row_ids AS row_ids_old
    FROM vert1
    ARRAY JOIN
        (
            SELECT groupArray(element) AS element
            FROM vert
        ) AS element
    WHERE arrayCount(elem -> (elem <= element), elements) = 0
) USING (element)
WHERE length(row_ids) > 1
```

```
SELECT *
FROM vert2

┌─elements─┬─row_ids─┐
│ [5,1]    │ [2,4]   │
│ [4,2]    │ [1,2]   │
│ [5,2]    │ [1,2]   │
│ [5,4]    │ [1,2,3] │
│ [7,5]    │ [3,4]   │
│ [8,5]    │ [3,4]   │
│ [8,7]    │ [3,4]   │
└──────────┴─────────┘
```

Third iteration

Almost there

```
CREATE TABLE vert3 ENGINE = TinyLog AS
SELECT
    pattern AS elements,
    arrayFilter(elem -> has(row_ids, elem), row_ids_old) AS row_ids
FROM vert
ALL INNER JOIN
(
    SELECT
        arrayReduce('groupArrayArray', [elements, [element]]) AS pattern,
        element,
        row_ids AS row_ids_old
    FROM vert2
    ARRAY JOIN
        (
            SELECT groupArray(element) AS element
            FROM vert
        ) AS element
    WHERE arrayCount(elem -> (elem <= element), elements) = 0
) USING (element)
WHERE length(row_ids) > 1
```

```
SELECT *
FROM vert3

┌─elements─┬─row_ids─┐
│ [5,4,2]  │ [1,2]   │
│ [8,7,5]  │ [3,4]   │
└──────────┴─────────┘
```

Last iteration

```
CREATE TABLE vert4 ENGINE = TinyLog AS
SELECT
    pattern AS elements,
    arrayFilter(elem -> has(row_ids, elem), row_ids_old) AS row_ids
FROM vert
ALL INNER JOIN
(
    SELECT
        arrayReduce('groupArrayArray', [elements, [element]]) AS pattern,
        element,
        row_ids AS row_ids_old
    FROM vert3
    ARRAY JOIN
        (
            SELECT groupArray(element) AS element
            FROM vert
        ) AS element
    WHERE arrayCount(elem -> (elem <= element), elements) = 0
) USING (element)
WHERE length(row_ids) > 1
```

```
:) SELECT * FROM vert4;

SELECT *
FROM vert4

Ok.

0 rows in set. Elapsed: 0.001 sec.
```

Result

```
SELECT
    elements AS pattern,
    length(row_ids) AS count
FROM vert_all
```

pattern	count
[7]	2
[5]	4
[4]	3
[1]	2
[3]	1
[8]	2
[2]	2

pattern	count
[5,1]	2
[4,2]	2
[5,2]	2
[5,4]	3
[7,5]	2
[8,5]	2
[8,7]	2

pattern	count
[5,4,2]	2
[8,7,5]	2

Thank you!

Konstantin Ignatov
@podshumok
kv@qrator.net