CIT. Journal of Computing and Information Technology, Vol. 24, No. 1, March 2016, 79–89
doi: 10.20532/cit.2016.1002694

Application of Clustering Algorithm CLOPE to the Query Grouping Problem in the Field of Materialized View Maintenance

Kateryna Novokhatska and Oleksii Kungurtsev

Odessa National Polytechnic University, System Software Department, Odessa, Ukraine

In recent years, materialized views (MVs) are widely used to enhance the database performance by storing pre-calculated results of resource-intensive queries in the physical memory. In order to identify which queries may be potentially materialized, database transaction log for a long period of time should be analyzed. The goal of analysis is to distinguish resource-intensive and frequently used queries collected from database log, and optimize these queries by implementation of MVs. In order to achieve greater efficiency of MVs, they were used not only for the optimization of single queries, but also for entire groups of queries that are similar in syntax and execution results. Thus, the problem stated in this article is the development of approach that will allow forming groups of queries with similar syntax around the most resource-intensive queries in order to identify the list of potential candidates for materialization. For solving this problem, we have applied the algorithm of categorical data clustering to the query grouping problem on the step of database log analysis and searching candidates for materialization. In the current work CLOPE algorithm was modified to cover the introduced problem. Statistical and timing indicators were taken into account in order to form the clusters around the most resource intensive queries. Application of modified algorithm CLOPE allowed to decrease calculable complexity of clustering and to enhance the quality of formed groups.

ACM CCS (2012) Classification: Information systems ⟶ Data management systems ⟶ Database management system engines ⟶ Query optimization;
Computing methodologies ⟶ Cluster analysis

Keywords: query, grouping, clustering, materialized view, CLOPE, categorical data

1. Introduction

Materialized view (MV) is one of the effective ways to enhance the database performance. MV represents the pre-calculated results of the queries stored in the physical memory. Implementation of MV makes it possible to significantly reduce the query execution time by decreasing the number of calls to the physical memory and eliminating the resource-intensive operations such as sorting and joining [1-2].

In order to achieve greater efficiency of MVs, they were used not only for the optimization of single queries, but also for entire groups of queries that are similar in syntax and execution results. Thus, the creation of automated tools for clustering queries with similar syntax became a live issue of computer science [3].

The evolution of automated creation of MVs brought up a number of problems for resolution. It is recommended to create MVs for the most resource-intensive and frequently executed queries. In addition, whenever base tables are updated, MVs require refresh based on the information about these modifications. Thereby, materialization of frequently updated data is ineffective. These statements should be taken into account during development of clustering algorithms in order to provide an opportunity to form clusters around the most intensive and frequently executed queries and reduce maintenance costs of future MVs.

Query clustering is performed on the full volumes of input data. In case transaction log for a long period of time requires analysis, the clustering becomes computationally difficult as

number of queries in the transaction log can be significant. Thus, clustering algorithms and approaches require reducing of the computational complexity and resource intensity.

The family of SQL dialects is constantly enhanced by adding new language constructions, modifying existing statements or making obsolete certain phrases. Therefore, the clustering algorithms should be scalable in terms of database manipulation language evolution. Support of the full syntax of the SQL language improves the quality of the formed clusters and, as a result, allows creation of more effective MVs [4].

2. Analysis of Recent Research and Publications

The grouping algorithm for MVs was firstly proposed in [5]. In the context of MVs, grouping means the selection of queries similar by the following criteria:

1) Data manipulation operator (SELECT, INSERT, UPDATE, DELETE) is identical for all queries in a group;
2) The same tables are accessed by the queries;
3) The filtering conditions of one query covers the similar filtering conditions of another query;
4) Query result sets are intersecting for all queries in a group;
5) Queries have similar syntax (the same grouping construction, aggregation or analytic functions, etc. are used).

The method [5] runs through comparing the tables, columns, and conditional predicates involved in the queries. The authors proposed the following main stages of query comparison:

1) Comparing the sets of tables by analyzing FROM clauses of each query;
2) Comparing the sets of fields obtained from the SELECT and WHERE clauses;
3) Comparing the WHERE conditions:

a. Casting WHERE clause to the canonical form;
b. Analyzing the logical structure of the WHERE clause in disjunctive form;
c. Comparing the relation expressions;
d. Comparing the individual arithmetic expressions.

The main disadvantages of this method are:

1) Lack of consideration of statistical and timing indices of query execution. It is recommended to create MV for the most resource-intensive and frequently executed queries. Withal, the method [5] considers only syntactic similarity of queries during clustering, without taking into account their execution time and frequency, as well as the amount of consumed resources;
2) High computational complexity of the task:

a. In order to form the groups of queries as described in [5], an exhaustive search of queries must be performed at each stage of the algorithm. Thus, the complexity of the algorithm equals $(N-1)^N$, where N is the number of unique queries in the transaction log. N in turn can reach significant orders (thousands and tens of thousands values),
b. Algorithm is implemented based on comparison of the textual data that is not effective in terms of consumption of computing resources,
c. All steps of the algorithm are performed on the full arrays of input data;

3) Fixed width of obtained clusters. Since the clusters are formed based on the similarity of queries, there is no possibility to expand the clustering criteria (to take into account the number of resources allocated for the MVs, etc.).

3. Unsolved Aspects of the Problem

We propose to apply more generic algorithms of clustering the categorical data to the task of query grouping.

However, existing algorithms for data clustering do not allow to fully cover the specifics of the MV creation problem. In particular, they do not completely meet the following requirements:

1) Input data should be textual;

2) There should be a transformation of textual data into numeric vectors to decrease a resource use during clustering;
3) The algorithm should group queries, distinguishing them by the syntactic structure;
4) Statistical and timing indicators of query execution should be considered during clustering;
5) Result groups should be suitable for the inverse transformation of the numeric vectors into textual data;
6) Algorithm should be productive and consume minimum amount of resources to store the service structure, as it will be applied to very large data sets;
7) Algorithm must be easy to implement.

4. Analysis of the Existing Algorithms

We have investigated the following families of algorithms for categorical data clustering:

I. Density-Based Clustering (e.g. DBSCAN [6], OPTICS [7]);
II. Hierarchical algorithms (e.g. COBWEB [8], BIRCH [9], ROCK [10], LIMBO [11]);
III. Histogram algorithms (e.g. k-ANMI [12], CLOPE [13], K-Histograms [14]);
IV. Genetic algorithms (e.g. G-ANMI [15], TCSOM [16], GSC [17]);
V. Spectral clustering (e.g. SpectralCAT [18], COOLCAT [19]).

Table 1 shows how the investigated algorithms meet the requirements for grouping queries.

After the comparative analysis, the algorithm CLOPE was chosen [20]. The essence of this algorithm lies in comparing the histograms of the input queries.

Algorithm CLOPE was chosen due to the following factors:

1) The algorithm is applicable to the input data represented as a numeric vector, each element of which corresponds to the SQL token;
2) The algorithm operates histograms that allow the most accurate comparison of SQL queries;

Table 1. Comparative analysis of data clustering algorithms.

Evaluation criteria	I	II	III	IV	V
Input data in SQL terms	–	+	+	+	+
Transformation of textual data into numerical vectors	–	–	+	+	+
Grouping queries with taking SQL syntax into account	–	+	+	+	–
Considering statistical and timing indicators of query execution	–	–	–	–	+
Inverse transformation of the numerical vectors into textual data	–	+	+	+	–
High performance and low resource consumption	+	–	+	–	+
Simplicity of implementation	+	–	+	–	+

3) Generalized query for MV creation can be obtained on the basis of calculated histograms;
4) Algorithm is productive, simple to implement and requires a minimum number of resources for storing intermediate data.

It was decided to modify the algorithm for taking into account statistical and timing indicators of the input queries.

5. Statement of the Problem

The aim of this work is the application of a clustering algorithm CLOPE to the problem of grouping queries to reduce the computational complexity of the problem and to improve the quality of formed clusters by taking into account the statistical and timing indicators of query execution.

6. Input Data

After the analysis of transaction log of information system for a large period of time T, the set of non-unique queries of different types (*SELECT, INSERT, UPDATE, DELETE*) was formed with the time of their execution and the amount of consumed resources:

$$Q = \bigcup q\langle t, \tau, b\rangle ,$$

where:

t – query text;

τ – query execution time;

b – number of data blocks processed by the query.

We introduce the coefficient K that shows how query is "suitable" for materialization. In general, normalized value of b can be used as K. In [21] a more precise and multifactorial algorithm of calculating K was proposed, which took into account the statistical and timing indicators of the query execution.

We divide the set of queries Q into 4 groups according to the type of data manipulation operator. We consider only N queries of *SELECT* type from the set Q during further analysis. Other types of queries are excluded from consideration as MVs are not created on their basis:

$$S = \bigcup_{i=1}^{N} s_i \langle t_i, \tau_i, b_i \rangle$$

In order to perform the conversion of the query from SQL text to a numeric vector, we divide the query into atomic tokens. An atomic token is one or more SQL expressions such as field names, table names and aliases, constants, functions, operators that are minimal semantic units in the formation of *SELECT, FROM, WHERE* clauses, sorting conditions, grouping conditions, etc.

Vocabulary of tokens V is the set of unique tokens found while parsing S. Each vocabulary entry is described by deuce: the current token and its numeric identifier (serial number) in the vocabulary V.

For each query s_i, $i = 1, \ldots, N$ we will form the vector D_i having a set of tokens from the vocabulary V found while parsing the query. To simplify the data processing, while forming the vector D_i, we will operate with serial numbers from the vocabulary V instead of tokens themselves. The number of occurrences of token in vector D_i may be greater than 1. The detailed algorithm of the formation of tokens for grouping queries is presented in [22]. In the next step we transform the set S to the following form:

$$S = \bigcup_{i=1}^{N} s_i \langle t_i, D_i, K_i \rangle$$

Below an example of parsing query S_1 into numerical vector D_1 is provided:

$S_1 :=$ *select* $s1.rt\ object\_id,\ s1.reference,$
$np.object\_id\ value$
from $params\ np,$
$(select\ connect\_by\_root$
$nr.object\_id\ rt,\ nr.reference$
$from\ references\ nr$
$where\ nr.attr\_id\ =\ 1$
$start\ with\ nr.object\_id\ in\ (2,\ 3)$
$and\ nr.attr\_id\ =\ 4$
$connect\ by\ prior\ nr.reference$
$=\ nr.object\_id\ and\ nr.attr\_id$
$=\ 5$
$order\ by\ LEVEL)\ s1$
$where\ np.object\_id$
$=\ s1.referenceand\ np.attr\_id$
$=\ 6\ and\ rownum = 7;$

For clarity, a phrase in which the token was found ('select', 'from', etc.) was added as a prefix to each token.

Table 2. Example of dividing a query into set of tokens.

N	***l***
1	*select rt*
2	*select reference*
3	*select object_id*
4	*from params*
5	*select connect_by_root object_id*
6	*from references*
7	*where attr_id = @NUMBER*
8	*start with object_id in (@NUMBER, @NUMBER)*
9	*start with attr_id = @NUMBER*
10	*connect by prior reference = object_id*
11	*connect by prior attr_id = @NUMBER*
12	*order by LEVEL*
13	*where object_id = reference*
14	*where attr_id = @NUMBER*
15	*where rownum=@NUMBER*

After analysis of the query, the following numeric vector is obtained:

$$D_1 = \{1, 2(2), 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}.$$

As it can be seen, the token

$$L_2\langle \text{"}select\ reference\text{"}\rangle$$

occurs in vector D_1 twice.

Similarly to the previous example, we parse the query S_2 using the existing vocabulary V:

$$\begin{array}{lll} S_2 := & select & s1.rt\ object\_id \\ & from & references \\ & where & attr\_id = 9 \\ & and & rownum = 10; \end{array}$$

As a result we obtain the following vector:

$$D_2 = \{3, 6, 14, 15\}.$$

7. Data Clustering

By a cluster we mean a group of queries that meet the previously mentioned criteria of similarity. Cluster can be considered as an independent unit with certain characteristics.

A set of clusters C is a partition of S such that:

$$\left\{ C = \bigcup_{j=1}^{E} c_j = S | c_k \bigcap c_j = \emptyset, k = 1, ..., E, k \neq j \right\}$$

As soon as the cluster $c_j, j = 1, \ldots, E$ is a subset of S, its elements can be described as triples of the form $\langle t, D, K\rangle$.

Each cluster is characterized by the following parameters:

- $W(c_j)$ – cluster width. Corresponds to the number of unique tokens L in the cluster c_j;
- $O(c_j)$ – the number of queries in the cluster c_j;
- $Occ(L_n, c_j)$ – number of occurrences of a unique token $L_n, n = 1, \ldots, W(c_j)$ in the cluster c_j;
- $P(c_j)$ – cluster power, corresponds to the total number of tokens in the cluster c_j:

$$P(c_j) = \sum_{n=1}^{W(c_j)} Occ(L_n, c_j) \qquad (1)$$

- $H(c_j)$ – cluster height:

$$H(c_j) = \frac{P(c_j)}{W(c_j)} \qquad (2)$$

Histogram of cluster c_j is called a graphical representation of its characteristics: the horizontal axis shows tokens, in descending order of $Occ(L_n, c_j)$, the vertical axis shows the value $Occ(L_n, c_j)$ itself. Cluster power $P(c_j)$ geometrically corresponds to the area of the histogram.

As an example, consider the cluster:

c_1=[{1, 6}, {2, 4}, {3, 2}, {4, 1}, {5, 1}, {6, 1}].

The histogram of the cluster c_1 is shown in Figure 1.

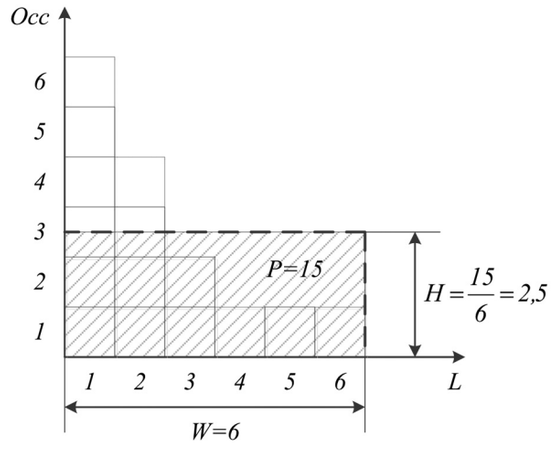

Figure 1. Histogram of the cluster c_1.

We propose to split the clustering algorithm CLOPE into several stages:

Stage 1: Initialization of clusters

In order to form the clusters around the most resource intensive and frequently executed queries, we propose to introduce an additional step for cluster initialization. For this purpose, we sort the queries in descending order by coefficient K, and choose V queries that match its maximum value. The number of queries V, based on which the clustering would be performed, may

be arbitrarily selected regarding the administrative restrictions on the number of MV in the system.

For each query s_i, $i = 1, \ldots, V$ we form the cluster $c_j, j = 1, \ldots, E, E = V$ and calculate its initial characteristics based on the parameters of the vector D_i:

- $W(c_j)$ – the number of unique tokens in vector D_i;
- $O(c_j) = 1$;
- $P(c_j) = P(D_i)$;
- $Occ(L_n, c_j) = Occ(L_n, D_j), n = 1, \ldots, W(c_j)$.

An additional empty cluster c_{E+1} is created, for which all parameters are initialized with zero value.

To the above mentioned query notation s_i we add the fourth element $Cnum_i$ – number of cluster assigned to the current query:

$$s_i\langle q_i, D_i, K_i, Cnum_i\rangle, i = 1, \ldots, N$$

If the cluster has not been assigned yet, $Cnum_i$=0.

We distribute the remaining queries from the sorted set S across the clusters:

1. Retrieve the query $s_i, i = V+1, dots, N$ from the set S.
2. For the current s_i iterate through the clusters.
3. For the current value of s_i and c_j calculate profit function Q_{ij}. This function shows how the inclusion of query s_i to cluster c_j affects the characteristics of the cluster. The higher the value of Q_{ij}, the more common tokens were found when comparing the query with cluster histogram. Thus, the purpose of the clustering is to maximize Q_{ij}.

   For calculation of Q_{ij} we determine the updated characteristics of cluster c_j including query s_i:

$$W(c_j)^{new} = W(c_j) + |\{L_m | L_m \in D_i, L_m \notin c_j\}, m = 1, \ldots, W(D_i)| \quad (3)$$

$$P(c_j)^{new} = P(c_j) + P(D_i) \quad (4)$$

   In the classical interpretation of algorithm CLOPE height $H(c_j)^{new}$ can be calculated as follows:

$$H(c_j)^{new} = \frac{P(c_j) + \sum_{m=1}^{W(D_i)} Occ(L_m, D_i)}{W(c_j)^{new}} \quad (5)$$

   In order to regulate the impact of the new entry of s_i into the cluster c_j according to the "usefulness" of the query from the perspective of materialization, we add coefficient K to the formula 5. For this purpose we represent the cluster c_j as a set of queries s_v, $v = 1, \ldots, O(c_j)$, each of which is characterized by its own value K_v and vector D_v. Expand $P(c_j)$ as:

$$P(c_j) = \sum_{v=1}^{O(c_j)} P(D_v) \quad (6)$$

   Thus we obtain:

$$H(c_j)^{new} = \frac{\sum_{v=1}^{O(c_j)} K_v P(D_v) + K_i \sum_{m=1}^{W(D_i)} Occ(L_m, D_i)}{W(C_j)^{new}} \quad (7)$$

   Calculate the profit function Q_{ij}:

$$Q_{ij} = \frac{H(c_j)^{new}(O(c_j)+1)}{W(c_j)^{new}} - \frac{H(c_j)O(c_j)}{W(c_j)} \quad (8)$$

   For the case when $O(c_j)$=0:

$$Q_{ij} = \frac{H(c_j)^{new}(O(c_j)+1)}{W(c_j)^{new}} \quad (9)$$

4. If $j \leq E$, proceed to step 2. Otherwise, proceed to step 5.
5. Include query s_i into the cluster $c_{\max}$, which corresponds to the maximum value of profit function Q_{ij}:

$$Cnum_i^{new} = \{\max | Q_{i\max} = MAX_{j=1}^{E+1}(Q_{ij})\}, s_i\langle q_i, D_i, K_i, Cnum_i^{new}\rangle, i = 1, \ldots, N. \quad (10)$$

6. Update the histogram of cluster $c_{\max}$. For this purpose, for each token of the vector D_i check the existence of an appropriate token in the cluster $c_{\max}$. If the token was found:

$$Occ(L_n, c_{\max})^{new} = Occ(L_n, c_{\max}) + Occ(L_m, D_i) \quad (11)$$

Otherwise, add a new token to the cluster $c_{\max}$:

$$Occ(L_{M+1}, c_{\max})^{new} = Occ(L_m, D_i), \quad M = W(c_{\max}) \tag{12}$$

7. Update the cluster characteristics according to formulas (3), (4) and (7).
8. If the current value of $O(c_{\max}) = 0$, then initialize a new empty cluster: $E = E + 1$. Increment the number of queries in the selected cluster:

$$O(c_{\max}) = O(c_{\max}) + 1 \tag{13}$$

9. If $i \leq N$, proceed to step 1. Otherwise, proceed to step 10.
10. Complete the initialization stage.

Stage 2: Iterative distribution of queries across clusters

1. Initialize the values $i = 1, j = 1$, a counter of permutations $Z = 0$.
2. Retrieve the query $s_i, i = 1, \ldots, N$ from the set S.
3. For the current s_i iterate through the clusters.
4. If $Cnum_i = j$, then the query must be removed from the cluster before the new cluster is calculated for it.

4.1. Recalculate histogram of the cluster. For each token $L_n, n = 1, \ldots, W(c_j)$ of the cluster c_j and $L_m, m = 1, \ldots, W(D_i)$ of the vector D_i, if $L_n = L_m$:

$$Occ(L_n, c_j)^{remove} = Occ(L_n, c_j) - Occ(L_m, D_i) \tag{14}$$

If $Occ(L_n, c_j)^{remove} = 0$, exclude the token from the cluster c_j:

$$W(c_j)^{remove} = W(c_j) - 1 \tag{15}$$

Otherwise:

$$W(c_j)^{remove} = W(c_j) \tag{16}$$

4.2. Update the cluster characteristics:

$$P(c_j)^{remove} = P(c_j) - P(D_i);$$
$$O(c_j)^{remove} = O(c_j) - 1;$$
$$H(c_j)^{remove} = \frac{P(c_j)^{remove}}{W(c_j)^{remove}}$$

5. According to the formula (8) or (9), calculate the current value of Q_{ij}. If $j \leq E$, proceed to step 3. Otherwise, proceed to step 6.
6. According to the formula (10) for the current s_i, select cluster $c_{\max}$ corresponding to the maximum value of Q_{ij}.
7. Calculate new characteristics of the selected cluster according to the formulas (3), (4) and (7).
8. If the current value of $O(c_{\max}) = 0$, initialize a new empty cluster: $E = E + 1$. Increment the number of queries in the selected cluster according to the formula (13).
9. If $Cnum_i \neq Cnum_i^{new}$, then increment counter of permutations: $Z = Z + 1$.
10. If $Cnum_i \neq Cnum_i^{new}$ and $O(c_{Cnumi}) = 0$, remove the empty cluster c_{Cnumi}.
11. If $i \leq N$, proceed to step 2. Otherwise, proceed to step 12.
12. If $Z > 0$, move to step 1. Otherwise, output the result.

8. Experiment

All tests have been executed in 64-bit JVM, JDK v.7.0, 1 GB heap space on Intel® Core™ i3-4000M CPU, 2.4 GHz PC with 8 GB RAM and 500 GB SATA disk. For the experiment, a transaction log has been used, which contained more than 2,000 non-unique queries involving aggregation operations, multiple joins, hierarchical queries, analytic functions etc. It was derived from commercial production database deployed on Oracle 10g EE DB server. The data was collected during one-week period. As a result of filtering, 502 unique queries were obtained. After lexical processing of data, a vocabulary was formed containing 393 tokens.

The following measures were chosen as quality indicators of the formed clusters:

1) The mean similarity of elements – dynamic value which is calculated during the formation of clusters. This is the number of common tokens in the vector D and the histogram of the cluster C averaged over all clusters divided to the width of the vector D:

$$\overline{M_{similarity}}=\frac{1}{E}\sum_{i=1}^{E}\left(\frac{1}{O(C_i)}\cdot\sum_{j=1}^{O(C_i)}\frac{|\{L_m|L_m\in D_j,L_m\in C_i\},m=1,\ldots,W(D_j)|}{W(D_j)}\right)\cdot 100,$$

where E – total number of clusters.

It indicates how much "similar" queries form a cluster. For values tending to 100%, it can be concluded that the groups will be formed for virtually identical queries which makes them narrowly applicable. For small values of the index "broad", clusters will be obtained, which in the future may not be optimal in terms of maintenance and storage. It has been experimentally established that the optimum value of the index ranges from 70% to 90%;

2) The mean distance between clusters – the average value of "Manhattan distance" as a measure of the difference of two clusters:

$$\overline{M_{Manh}}=\frac{\sum_{i=1}^{E}\sum_{j=1}^{E}\left|Occ(L_n,C_i')-Occ(L_n,C_j')\right|}{(E-1)^2},$$
$$i\neq j,n=1,\ldots,W(C_i')=1,\ldots,W(C_j'),$$

where C_i' and C_j' – clusters aligned in width:

$$C_i'=C_i\bigcup\{\langle L_k,0\rangle\},L_k\in C_j,L_k\notin C_i,$$
$$k=1,\ldots,W(C_j);$$
$$C_j'=C_j\bigcup\{\langle L_m,0\rangle\},L_m\in C_i,L_m\notin C_j,$$
$$m=1,\ldots,W(C_i).$$

As a percentage the value is calculated as follows:

$$\overline{M_{Manh,\%}}=\frac{1}{E}\cdot\sum_{i=1}^{E}\sum_{j=1}^{E}\frac{\left|Occ(L_n,C_i')-Occ(L_n,C_j')\right|}{P(C_i+C_j)}\cdot 100$$

3) The minimum distance between clusters:

$$M_{\min}=\min\left|Occ(L_n,C_i')-Occ(L_n,C_j')\right|,$$
$$i=1,\ldots,E,j=1,\ldots,E,i\neq j,$$
$$n=1,\ldots,W(C_i')=1,\ldots,W(C_j')$$

4) The maximum distance between clusters:

$$M_{\max}=\max\left|Occ(L_n,C_i')-Occ(L_n,C_j')\right|,$$
$$i=1,\ldots,E,j=1,\ldots,E,i\neq j,$$
$$n=1,\ldots,W(C_i')=1,\ldots,W(C_j')$$

5) The percentage of single clusters – the ratio of the number of clusters that contain only one query to the total number of calculated clusters. From the perspective of grouping queries, this indicator identifies the number of queries that are not of interest in the problem of creating MV. Since the purpose of grouping is not a complete coverage of queries coming into the system, the optimal value of this index is the range of $40-60\%$;

6) The percentage of single tokens in the cluster – the ratio of the number of tokens in the cluster for which the value *Occ(L, C)* is equal to one. It shows which part of the cluster could potentially be excluded from participation in the further processing of the group;

7) The number of iterations of the algorithm;

8) Data processing time – is a measure of performance of the algorithm;

9) Amount of memory consumed.

Based on the selected criteria, a comparative analysis of the algorithm from the work [5] ("Algorithm 1") and the modified algorithm CLOPE ("Algorithm 2") was performed. The table below shows the results.

Figures 2 and 3 show histograms of clusters obtained by applying the algorithms 1 and 2 (without single clusters). The vertical axis represents the number of queries that form a cluster, while the horizontal axis represents the number of clusters.

Figure 1 shows that most part of queries is concentrated in 7 huge clusters. The rest of the data forms very small clusters. That means that algorithm 1 completed a rather rough analysis of queries without taking into consideration query syntax. Lower average distance between the

clusters and lower rate of similarity of elements within the cluster show that scattered queries form the groups. As a result, "broad" materialized views will be created, which will not be optimal in terms of maintenance and storage.

Table 3. Results of the comparative analysis.

Evaluation criterion	Algorithm 1	Algorithm 2
Number of input queries	502	502
Number of unique tokens	N/A	393
Quantity of initial clusters	N/A	10
Quantity of result clusters	46	154
Mean similarity of elements	61.44%	70.8%
Mean distance between clusters	273.17 (75.73%)	79.07 (84.89%)
Minimum distance between clusters	8 (25%)	3 (25.3%)
Maximum distance between clusters	2267 (100%)	785 (100%)
Percentage of single clusters	54.35%	48%
Percentage of single tokens in the cluster	42.38%	28.5%
Number of iterations of the algorithm	1	10
Data processing time	22 181 ms	1493 ms
Amount of memory consumed	252 510 kbytes	11 370 kbytes

Figure 2 shows in turn a wider range of clusters. Groups of queries are much smaller and their size is more uniform. It gives the grounds to consider that the clusters were formed more precisely. As soon as the average distance between the clusters and the rate of similarity of elements within the cluster were increased, more common queries from the syntax perspective formed the clusters. Thus, the maintenance cost of future MVs created for the formed clusters will be decreased.

Based on the experimental data, the following conclusions can be made:

1. The performance of the modified algorithm CLOPE is 15 times better than the previous solution;
2. Resource consumption of Algorithm 2 is 22 times lower than the previous solution;
3. The quality of the clusters has increased:

3.1. The average distance between clusters increased by 9.16%;

3.2. The average similarity of the elements within the cluster increased by 9.36%;

3.3. The percentage of single clusters decreased by 6.35%;

3.4. The percentage of single tokens in the cluster decreased by 13.88%;

3.5. The number of clusters increased 3.16 times.

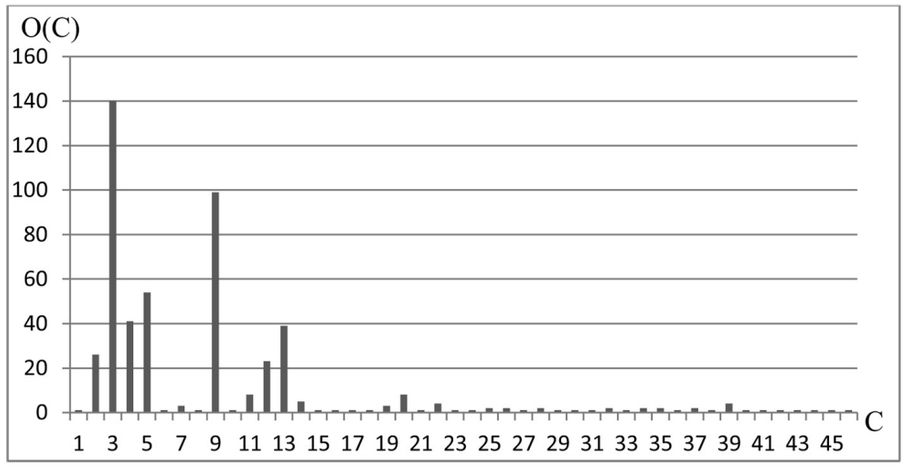

Figure 2. The distribution of queries across clusters obtained by the Algorithm 1.

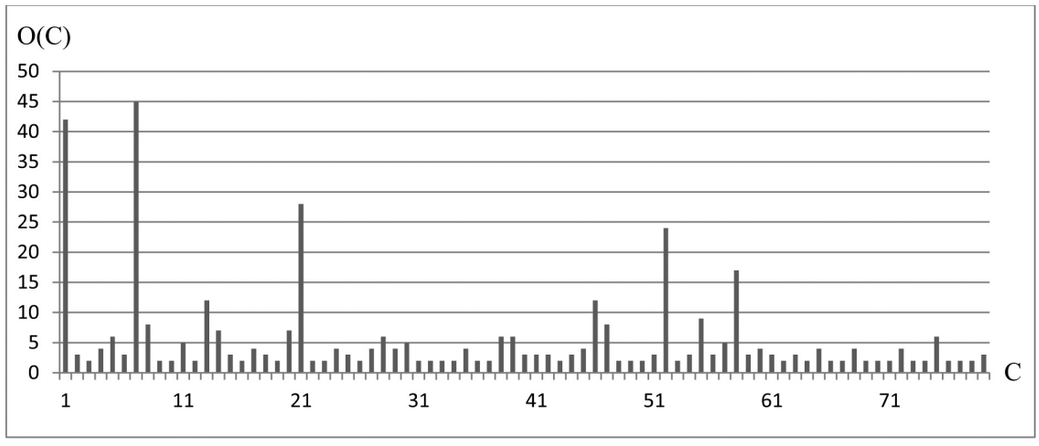

Figure 3. The distribution of queries across clusters obtained by the Algorithm 2.

9. Conclusion

In this paper, clustering algorithm for textual data has been applied to the problem of grouping queries during selection of MVs. Based on the results of the comparative analysis, CLOPE algorithm was chosen from the set of existing solutions. It was further improved to meet the requirements for the quality of future MVs in the context of MV selection problem.

Modification of the algorithm allowed:

1. To consider statistical and timing indicators of the query execution. Thus, the MVs will be formed around the most resource-intensive and frequently executed queries;
2. To reduce the computational complexity of the task through the use of more efficient algorithms of clustering;
3. To reduce the resource consumption of the algorithm by avoiding operating textual data and transitioning to numerical operations;
4. To group queries more precisely in terms of the query syntax, as evidenced by the decrease in the percentage of single clusters and the increase in the total number of formed groups;
5. To form higher-quality MVs by increasing the average distance between the clusters and the rate of similarity of elements within the cluster. The data stored in the MVs will intersect less and more closely match the queries accessing them.

References

[1] A. B. Kungurtsev and Y. N. Vozovikov, "Materialized views management in information systems", *Eastern European Journal of Advanced Technologies*, vol. 1/4, no. 43, pp. 18–21, 2010.

[2] A. B. Kungurtsev and Y. N. Vozovikov, "Supporting the effectiveness of the control of materialized views", *Electrotechnic andComputer Systems*, vol. 4, no. 80, pp. 136–140, 2011.

[3] K. A. Novokhatska and Y. N. Vozovikov, "The method of update function generation for the incremental maintenance of materialized views". *Visnyk SumDU*, Seriya "Tehnichni nauky", vol. 3, pp. 82–96, 2011.

[4] A. B. Kungurtsev and Quoc Vinh Nguyen Tran, "The analysis of feasibility of applying the materialized views in information systems", in *Proc. of OPU*, vol. 2, no. 2, 2003, pp. 102–106.

[5] A. B. Kungurtsev *et al.*, "Comparison of queries in relational databases to build materialized views", in *Proc. of UNDIRT*, vol. 3, no. 39, 2004, pp. 35–38.

[6] M. Ester *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise", in *Proc. of the KDD*, vol. 96, 1996, pp. 226–231.

[7] M. Ankerst *et al.*, "OPTICS: Ordering Points To Identify the Clustering Structure", *Proc. of ACM-SIGMOD International Conference on Management of Data*, pp. 49–60, 1999. http://dx.doi.org/10.1145/304181.304187

[8] D. Fisher, "Knowledge AcquisitionVia Incremental Conceptual Clustering", *Machine Learning*, vol. 2, 1987, pp. 139–172. http://dx.doi.org/10.1007/BF00114265

[9] T. Zhang *et al.*, "An efficient data clustering method for very large Databases", *Proc. of ACM SIGMOD International Conference on Management of Data*, 1996, pp. 103–114. http://dx.doi.org/10.1145/233269.233324

[10] S. Guha *et al.*, "ROCK: a robust clustering algorithm for categorical attributes", *Information System*, vol. 25, no. 5, pp. 345-366, 2000. http://dx.doi.org/10.1016/S0306-4379(00)00022-3

[11] P. Andritsos *et al.*, "LIMBO: Scalable Clustering of Categorical Dana", *Proc. of EDBT'04*, 2004, pp. 123–146. http://dx.doi.org/10.1007/978-3-540-24741-89

[12] Z. He *et al.,* "A mutual information based clustering algorithm for categorical dana", *Information Fusion*, vol. 9, no. 2, 2008, pp. 223–233. http://dx.doi.org/10.1016/j.inffus.2006 .05.006

[13] Y. Yang *et al.*, "CLOPE: a fast and effective clustering algorithm for transactional dana", *Proc. of KDD'02*, 2002, pp. 682–687. http://dx.doi.org/10.1145/775047.775149

[14] Z. He *et al.*, "K-Histograms: An Efficient Clustering Algorithm for Categorical Dataset", *Technical Report*. Tr-2003-08, Harbin Institute of Technology, 2003.

[15] S. Deng *et al.*, "A mutual information based genetic clustering algorithm for categorical dana", *Knowledge-Based Systems*, vol. 23, no. 2, 2010, pp. 144–149. http://dx.doi.org/10.1016/j.knosys.2009.11.001

[16] Z. He *et al.*, "TCSOM: Clustering Transactions Using Self–Organizing Map", *Neural Processing Letters*, 2005. http://dx.doi.org/10.1007/s11063-005-8016-3

[17] H. Wang *et al.*, "A Genetic Spectral Clustering Algorithm", *Journal of Computational Information Systems*, vol. 9, pp. 3245–3252, 2011.

[18] G. David and A. Averbuch, "SpectralCAT: Categorical spectral clustering of numerical and nominal dana" *Pattern Recognition*, vol. 45, pp. 416–433, 2010. http://dx.doi.org/10.1016/j.patcog.2011.07.006

[19] D. Barbara *et al.*, "COOLCAT: an entropy–based algorithm for categorical clustering", *Proc. of the 11th International Conference on Information and Knowledge Management*, 2002, pp. 582–589. http://dx.doi.org/10.1145/584792.584888

[20] K. A. Novokhatska and A. B. Kiper, "Reducing the computational complexity of query grouping using the algorithm CLOPE", *Proc . of Intellectual Technologies in System Programming*, 2013, pp. 120–121.

[21] K. A. Novokhatska and A. B. Kungurtsev, "Calculating the materialization factor in query evaluation during the maintenance of materialized views", *Visnyk KhNTU*, vol. 2, 2015.

[22] K. A. Novokhatska and A. B. Kungurtsev, "Formation of tokens in query grouping in the method of incremental maintenance of materialized views", *Visnyk ChDTU. Seriya "Tehnichni nauky"*, vol. 1, no. 71, 2014, pp. 193–199.

Received: February, 2015
Revised: November, 2015
Accepted: December, 2015

Contact addresses:

Kateryna Novokhatska
Odessa National Polytechnic University
System Software Department
Odessa
Ukraine
e-mail: katherinaniv@gmail.com

Oleksii Kungurtsev
Odessa National Polytechnic University
System Software Department
Odessa
Ukraine
e-mail: abkun@te.net.ua

KATERYNA NOVOKHATSKA: She graduated from Odessa National Polytechnic University in 2012 with a Master's degree in Computer Science. She is currently a PhD student at ONPU. Her research interests include database performance, advanced query optimization, and materialized views.

OLEKSII KUNGURTSEV: He is a professor at Odessa National Polytechnic University, System Software Department. His research interests include development of methods for improving database productivity.