Розглядається проблема вибору матеріалізованих представлень (МП) в технології автоматизованого створення МП. Запропонована методика вибору МП складається з двох етапів. Групування запитів здійснюється за допомогою порівняння абстрактних синтаксичних дерев. Для вибору МП з множини груп однотипних запитів був застосований генетичний алгоритм. Запропонована цільова функція, що враховує вартість виконання запиту і вартість обслуговування МП

Ключові слова: матеріалізоване представлення, оцінка запиту, групування запитів, центральний запит, генетичний алгоритм

Рассматривается проблема выбора материализованных представлений (МП) в технологии автоматизированного создания МП. Предложена методика выбора МП, состоящая из двух этапов. Группировка запросов осуществляется посредством сравнения абстрактных синтаксических деревьев. Для выбора МП из множества групп однотипных запросов был применен генетический алгоритм. Предложена целевая функция, учитывающая стоимость выполнения запроса и стоимость обслуживания МП

Ключевые слова: материализованное представление, оценка запроса, группировка запросов, центральный запрос, генетический алгоритм

UDC 004.658.2
DOI: 10.15587/1729-4061.2016.68737

DEVELOPING METHODOLOGY OF SELECTION OF MATERIALIZED VIEWS IN RELATIONAL DATABASES

K. Novokhatskaya
Postgraduate Student*
E-mail: katherinaniv@gmail.com
A. Kungurtsev
PhD, Professor*
E-mail: abkun@te.net.ua
*Department of System Software
Odessa National Polytechnic University
Shevchenko ave., 1, Odessa, Ukraine, 65044

1. Introduction

There are two kinds of views used in modern DBMS (Database Management System): materialized and virtual. Materialized views (MV) are the physical objects of the databases containing the result of query execution. MV allows significantly accelerating query execution that accesses a large number of entries. This is achieved through the use of pre-calculated summary data and the results of table joins. The disadvantage of MV is that they require additional maintenance costs because MV must be updated every time the changes in the data sources take place.

An alternative to this approach is the use of virtual views. Each time they are accessed, a selection of base tables is required. In the case of resource-intensive queries, this method is not efficient and it is mostly used in order to implement a more flexible scheme of allocation of access rights, rather than query optimization. But in contrast to MV, the virtual views do not require significant resources for the service data storage.

Thus, the question arises what views should be materialized in the system and which should remain virtual. The materialization of views makes it possible to achieve the best performance at the highest maintenance cost. If one leaves all the views virtual, the maintenance cost will be minimal, but the performance gain will not be observed. Certain views can be materialized, some may remain virtual. With this approach, one can find a balance between the growth in productivity and the cost of the views maintenance. Thus, the decision should be made, taking into account two types of costs: the cost of queries processing and the cost of views maintenance.

The relevance of the work in this direction consists in defining the configuration of views, in which the maximum query capacity at the lowest maintenance cost is achieved.

2. Analysis of scientific literature and the problem statement

The task of MV selection out of the pre-known set of views is often found in the literature [1–4]. Its input parameters are:

– a set of queries that require optimization;

– a set of views on the basis of which the result can be calculated;

– the cost of the query based on a particular MV;

– MV maintenance cost.

The aim of the task is to select a subset of the MV, in which the execution of queries and maintenance costs will be minimal [5, 6]. The problem is NP-hard and requires exhaustive search of possible solutions [7].

In the paper [8] the authors suggest reducing the solution space through the use of heuristic algorithms of MV selection. With this aim the so-called “greedy” algorithms are used. However, this method has low productivity and cannot be applied to large volumes of data.

In [9], a heuristic algorithm of MP selection was improved. The concept of multiple view processing plan is used (MVPP). Query cost is introduced, which includes the resource estimate costs required for MV service. Search for optimal global plan is processed via binary matrices.

In the paper [10], the method of MV selection is improved, using MVPP by applying a Parallel Simulated Annealing algorithm (PSA) – a general algorithmic method of solving the problem of global optimization, simulating physical process of metal annealing. The use of PSA increased the scalability of the previous solution. However, this algorithm allows finding only locally optimal solutions, assuming that the final solution is also optimal. Thus, the best solutions may be missed out.

In the paper [11], a process of MV selection was divided into two phases – the selection of an optimal global query execution plan out of many possible plans performing a single query and the MV selection out of the optimal global plans of all set of queries. To solve the task, a hybrid technique is used: in various stages of analysis, heuristic as well as genetic algorithms are used. Despite the fact that this approach allows finding the best solutions, rather than heuristic algorithms, it is much less productive.

The following main disadvantages of solutions were identified as a result of scientific literature analysis:

– there is no a query grouping method, which could reduce the number of MVs, and thus reduce MV service resource-intensity;

– objective functions at MV selection are not perfect and do not provide optimal balance between virtual and materialized views;

– existing MV selection procedures based on heuristic algorithms, PSA, etc., are hardly scalable and unproductive. In addition, they do not provide accurate solutions, stopping at locally optimal solutions.

3. The purpose and objectives of the study

The purpose of this work is development of the method of MV selection out of a set of queries entered into the information system, to achieve maximum query execution performance at the lowest cost of physical resources for the views service.

To achieve the set goal, the following tasks must be solved:

– design a similar view grouping algorithm to reduce their number in the system;

– to propose the objective function to find the optimal subset of MVs;

– to adapt a genetic algorithm for the MV selection task;

– to develop a criteria to determine which views will be materialized and which will be created as virtual.

4. Materials and methods of the study of MV selection task

4. 1. Materials of the study

As the input data, a variety of views-candidates for materialization are proposed:

$$Q=\bigcup q\left\langle \text{text},\tau,b,K\right\rangle, \qquad (1)$$

where text is the query text, forming a view; τ is the average query running time; b is the average number of data blocks, read from the disk or the buffer at the given query execution; n is the query occurrences number; K is the materialization coefficient, displaying to which degree the query “fits” the materialization [12]. It can be described in general view by the following expression:

$$K=\frac{\tau bn}{F_\Delta}, \qquad (2)$$

where F_Δ is the refresh rate of base tables involved in this query.

4. 2. Query grouping

The decision about the materialization for individually considered views does not provide the optimal solution at the level of the information system as the MV can be created for syntactically similar queries. By similar queries we understand those in which query accesses the same tables, the set of conditions of filtering the data of the query covers a similar set of another query, the sets of query results fields intersect, the queries have the same syntax structures (groups, aggregations, analytical functions, and so on). Materialization of similar queries causes the data duplication in different MVs and their number may exceed what is required. This leads to unnecessary excess resources for MV service. To avoid such problems, the grouping stage is introduced. The grouping task is to divide the set of views into subgroups of views with similar structure by analyzing the syntax trees.

The first step we perform is a query parse to present them in the form of abstract syntax trees (AST). A query grouping will be subsequently implemented by comparing the obtained trees.

To parse the query, we will use the context-free LL(1) grammar of SQL language. For this class of grammar, the predictive parsers are applied, performing scanning of input stream from left to right. At each step of the lookahead, one character is used for the decision of the parser actions. The phrases based on LL-grammar are formed by taking the left occurings [13].

We shall construct a SQL parser for the LL(1)-grammar language. We use ANTLR 4 [14] for this – a parser generator which allows automatically creating a parser–program according to the description of the LL-grammar.

Each query which forms the view q_i, $i=1..N_Q$, will be represented in the form of AST through its parse means. AST is a finite oriented tree in which the inner tops of the SQL language operators are compared, and the leaves correspond to the constants, table fields, variables, etc. AST example for the following query is shown in Fig. 1:

SELECT a+b FROM T WHERE c=9 AND d>3

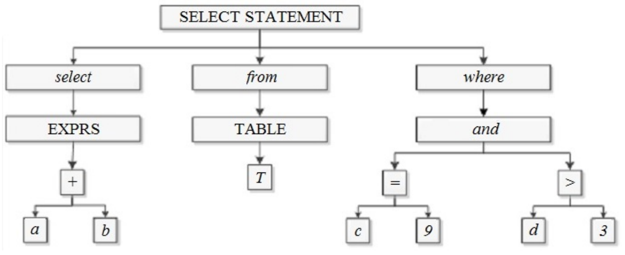

Fig. 1. Abstract syntax query tree

Let us describe the grouping algorithm. We select the following group settings:

G is the set of received groups of views;

N_G is the number of groups;

$AST(G_j)$ are the AST groups G_j, $j=1..\ N_G$.

Let us define a query transformation dictionary P, each entry p_z, $z=1..N_P$ of which can be represented by a triple of

<predicate condition Operation>, where N_P is the number of dictionary P entries. The predicate describes the SQL language phrase which is to be converted. Condition determines in which case the conversion may be applied to a given phrase. The operation determines the conversion itself.

Let us enter a number of possible operations:

– “exclude” – conversion is impossible. In this case, the current AST cannot be combined with AST of the subgroup G_j, $j=1..N_G$;

– “include” – the top of the current AST should be included in the AST of the subgroup G_j;

– “modify” – the top of AST of the subgroup G_j should be changed, taking into account the value of the top of the current AST;

– “skip” – inclusion of the top of the current AST into the AST of the subgroup G_j is not required.

Partial example of the P query transformation dictionary is given in Table 1.

Table 1

Example of P query transformation dictionary

Predicate $v_k(G_j)$	**Condition** $v_m(q_i)$	**Procedure**
WHERE	new filter for existing table	skip
WHERE	new filter of new table	include
WHERE field>@NUMBER1	field>@NUMBER2	modify: field> >MIN(@NUMBER1, @NUMBER2)
WHERE field = @NUMBER1	field=@NUMBER2	modify: field IN (@NUMBER1, @NUMBER2)
FROM	new combination: INNER JOIN	include: LEFT JOIN
FROM	new combination: LEFT JOIN	include: LEFT JOIN
FROM	new combination: RIGHT JOIN	include: FULL JOIN
SELECT	new field	include
GROUP BY	new field	include
QUERY	GROUP BY	skip
QUERY	WITH	include

Let us create a table of exceptions Exp with a signature similar to the signature of P query transformation dictionary. If the necessary transformation was not found in the dictionary P during the grouping, we form a record of exception in the form <predicate condition, NULL> in the Exp table. After finalizing the grouping, it is needed to analyze and add a conversion procedure for each record of the exception table Exp. Then the contents of the Exp table can be added to the dictionary P. Thus, the algorithm can be extended to new language constructions, encountered in the analyzed queries.

Let us distribute a set of Q views into the subgroups according to the algorithm described below. The algorithm contains two phases: initialization and iteration.

I. Initialization

1. Let us sort out the set of Q views in the descending order of the values of materialization coefficient K. This step is necessary in order for the groups to form the queries that are the most valuable for materialization.

2. Let us form a starting subgroup including a query view, corresponding to the maximum value of the coefficient K:

$N_G=1$;

$G_1=\{q_{k\ max}|K=MAX(K)\}$;

$AST(G_1)=AST(q_{k\ max})$.

II. Iteration

1. Reading the $AST(q_i)$, $i=2..O(c)$ out of sets of Q queries.

2. Initializing the subgroup iterator $j=1$.

3. If $j>N_G$, we form a new subgroup and proceed to p.1:

$N_G=N_G+1$;

$G_j=\{q_i\}$;

$AST(G_j)=AST(q_i)$.

4. Reading G_j.

5. Combining the tops of $AST(G_j)$ and $AST(q_i)$ in the following order:

– the FROM phrase;

– the WITH phrase;

– the hierarchy conditions START WITH and CONNECT BY;

– the conditions of combining and filtering of WHERE phrase;

– the conditions of grouping of GROUP BY... HAVING;

– the field list of the SELECT phrase.

5. 1. We read the next child node $v_m(q_i)$, $m=1..M_{NODE}$ of the analyzed phrase FROM, WITH and so on from the $AST(q_i)$ tree, where M_{NODE} is the number of the child nodes of the phrase.

5. 2. For every $v_m(q_i)$ we do sequentially read the child nodes $v_k(G_j)$, $k=1..N_{NODE}$ of the analyzed phrase FROM, WITH and so on from the $AST(G_j)$ tree where N_{NODE} is the number of child nodes of this phrase.

5. 3. We look up in the P dictionary a corresponding transformation p_z, $z=1..N_P$:

– <predicate> p_z corresponds to $v_k(G_j)$;

– $v_m(q_i)$ meets the <condition> p_z.

5. 4. If the conversion p_z is found and the <procedure> p_z is not equal to “exclude”, we modify the top $v_k(G_j)$ from $AST(G_j)$ according to the <procedure> p_z based on the $v_m(q_i)$ from $AST(q_i)$ and we proceed to p. 5. 1.

If the <procedure> p_z is equal to “exclude”, we form a new group and proceed to p. 1:

$N_G=N_G+1$;

$G_j=\{q_i\}$;

$AST(G_j) = AST(q_i)$.

5. 5. If the transformation p_z is not found, we read the next top $v_k(G_j)$ and proceed to p. 5.3.

5. 6. If $k=N_{NODE}$ and the p_z transformation is not found, we read the parent node $r_k(G_j)$ of the $v_k(G_j)$ top, add a record $<r_k(G_j), v_k(G_j), NULL>$ to the table of exceptions Exp. Increment the counter of groups $j=j+1$. Then proceed to p. 3.

As a result of the grouping, a set of Q views will be divided into N_G-groups, each of which will be represented as a generalized AST.

4. 3. MV selection

The problem of MV selection can be formulated as follows – on the basis of syntactically similar G groups, each of which covers a certain number of Q_G queries, a set of MV must be selected in order to achieve:

– a query performance increase through the use of the MVs is the highest;

– MV maintenance cost is minimal.

The following system limitations must be considered:

– the volume of disk space allocated to the MV storage;

– MV update rate;

– number of queries in the system covered by MV.

Let us represent a G query input groups as a matrix, whose lines represent groups, columns – the queries. Each matrix element can take one of the three values:

– 0 – a query is included in a group, but is not materialized;

– 1 – a query is included in a group and is materialized;

– –1 – a query is not included in a group and is not materialized.

Example. Assume three query groups were entered at the input: G_1 {q_1, q_2}, G_2 {q_3, q_4, q_5}, G_3 {q_6, q_7, q_8}. Then the initial matrix will take the following form:

	1	2	3
G_1	0	0	–1
G_2	0	0	0
G_3	0	0	0

We decided to materialize G_1 group on the basis of q_1, q_2 queries, and G_3 group on the basis of q_7, q_8 queries. Then the matrix will take the following form:

	1	2	3
G_1	1	1	–1
G_2	0	0	0
G_3	0	1	1

In the paper [7], it is proved that the MV selection task is NP-hard. For its solution a genetic algorithm (GA) is applied, which is one of the most commonly used heuristic search algorithms, used to solve NP-hard optimization problems. GA uses such methods of natural evolution as inheritance, mutation, selection and crossover [15].

The objective function in the MV selection problem can be defined as a the ratio of increase in query performance through the use of the selected MV to the cost of MV maintenance:

$$\mathrm{Profit} = \sum_{j=1}^{M} \frac{\sum_{i=1}^{N} \left(S_{q_{ji}} - S_{q_{ji}\text{МП}_j} \right) * n_{ji}}{V_{\text{МП}_j} * f_{\Delta_j}}, \tag{3}$$

where $S_{q_{ji}}$ is the cost of execution of q_{ji} query out of G_j group without using MV_j; $S_{q_{ji}MV_j}$ is the cost of execution of q_{ji} query out of G_j group using MV_j; n_{ji} is the number of executions of q_{ji} query during the studied period; V_{MV_j} is the MV_j size, created for G_j group; f_{Δ_j} is the MV_j upgrade rate; M is the number of G query groups; N is the number of queries in G_j group.

As an initial population for each group of queries G_j, j=1..M, we shall select the queries whose materialization coefficient K exceeds a certain threshold value. Thus, the initial population will receive the queries which are the most valuable in terms of creating MV.

Let us describe genetic operations used in GA:

– The selection is a process in which individuals "survive" in accordance with the value of the objective function. There are different methods of selection: tournament selection, ranking method, sigma clipping, and so on. In this paper we use the method of roulette, in which the probability of selecting an individual is more likely the better his/her fitness function value is.

– Crossover connects intermediate good solutions to obtain the best results. It allows defining new starting points for the search of optimal solutions. In this paper, we use a one-point crossover. In this case the gene "–1" cannot be subjected to the crossover.

Here are two individuals, for example:

0	1	1	0	0	1	0	1	1	–1
1	0	1	1	0	0	1	1	0	1

As a break point, we select the fifth position. As a result of the crossover we obtain the following individuals:

0	1	1	0	0	0	1	1	0	–1
1	0	1	1	0	1	0	1	1	1

– The mutation operation is a random change of some gene (from "0" to "1" and from"1" to "0"; the gene "-1" does not mutate within the current problem. Mutations form individuals with characteristics that may be absent in the original population. As an example, we can assume that the fourth gene was selected for mutation:

$0 \quad 1 \quad 1 \quad \underline{0} \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1$

The result of the operation is the following mutation genotype:

$0 \quad 1 \quad 1 \quad \underline{1} \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1$

The GA work is an iterative process that continues until a specified number of generations is reached. Each iteration is in the successive application of the crossover and mutation operations. Then the selection of the best results is performed – the selection.

As a result of this step, a subset of MV will be received out of the sets of G groups, which will be optimal for the materialization. Moreover, the optimal subset of queries of Gj group (which forms a given MV) will be selected for each final MV. On the basis of the groups that were not included in the final subset of views for materialization, it is advisable to create virtual views.

4. 4. Creating the central query

For each group, formed in the previous step, we shall perform an inverse transformation from AST into SQL query by means of a parser. A formed query is central. The final MV can be created on its basis.

5. Results of the study of the problem of MV selection

To perform the experiment, a transactions log which contained more than 502 unique queries was used. The analysis of the consumed resources was conducted for the DBMS transactions log Oracle 11g Release 2 EE, 64-bit JVM.

As part of the experiment, comparative analysis of the four algorithms was carried out: Parallel Simulated Annealing algorithm (PSA) [10], heuristic MV selection algorithm using multi-view processing plan (HV_MVPP) [9], evolutionary algorithm using the multi view-processing plan (EV_MVPP) [11] and a genetic algorithm using a group of queries, suggested in this work.

The experiment's results are summarized in Table 2. The following metrics are used for comparison:

$\overline{S_{MV}}$ is the average cost of query execution, entered into the system, using MV;

$\overline{C_{MV}}$ is the average maintenance cost of created MV:

$$\overline{C_{MV}} = \frac{V_{MV} * f_{\Delta}}{N_{MV}}, \qquad (4)$$

where V_{MV} is the size of created MVs; f_{Δ} is the MV upgrade rate; N_{MV} is the number of created MVs.

Table 2

Results of comparative analysis of algorithms

Algorithm	$\overline{S_{MV}}$	$\overline{C_{MV}}$	N_{MV}
PSA	781.89	602.12	34
HV_MVPP	731.73	672.81	25
EV_MVPP	605.28	773.17	24
GA_GR	502.73	638.32	22

Based on the results in Table 2, it can be concluded that the parallel simulated annealing algorithm displayed the worst result by the cost of the query execution. In this case the maintenance cost of the created MV for PSA is minimal. Heuristic MV selection algorithm and evolutionary algorithm showed the best result by the cost of query execution in comparison to PSA. However, the MV service cost, using these algorithms, is higher.

6. Discussion of the results of the study of the problem of MV selection

A genetic algorithm, proposed in this paper, using a query grouping displayed the best result by the cost of query execution. Despite the fact that the maintenance cost of created for the given algorithm MV is higher than the Parallel Simulated Annealing algorithm, it is compensated by the query execution cost value. Thus, the maximum efficiency of a query execution at the lowest cost of physical resources for the views service was achieved.

The solution of the problem of MV selection is part of a more general problem of the design of technology of the automated creation of MV. The proposed method can be applied in information systems using relational databases, to improve the performance of their work by optimizing resource-intensive and frequently-executed queries.

The conducted study is an improvement over the previously proposed technology of the automated creation of MV [16], and it clarifies the query grouping phase and the phase of MV selection out of the set of candidate groups. In the future, we plan to continue the research in this area by clarifying the methods of generation of the central query.

7. Conclusions

A study of the problem of MV selection in the technology of automatic creation of MV was conducted. As a result of the research:

1. The introduction of the query grouping phase based on the comparison of abstract syntax trees helped to group the uniform queries more accurately and to reduce a number of created MVs, and to decrease the total volume of physical resources required for their maintenance.
2. The objective function of the MV selection is the ratio of a query performance increase through the use of chosen MV to the cost of MV servicing.
3. The MV selection criterion formalizes for which query groups the MV creation is recommended and which of them should be created as virtual.
4. The application of a genetic algorithm for solving the problem of MV selection made it possible not only to cover the mentioned criterion, but also to define the queries within one group which will form the next central query.

Experimental data displayed that by using the proposed algorithm it is possible to obtain such a set of MVs, at which the maximum efficiency of query execution at the lowest consumption of physical resources is achieved for the MVs servicing.

References

1. Karde, P. P. An Efficient Materialized View Selection Approach for Query Processing in Database Management [Text] / P. P. Karde, V. M. Thakare // International Journal of Computer Science and Network Security. – 2010. – Vol. 10, Issue 9. – P. 26–33.
2. Nalini, T. A comparative study analysis of materialized view for selection cost [Text] / T. Nalini, A. Kumaravel, K. Rangarajan // International Journal of Computer Science & Engineering Survey. – 2012. – Vol. 3, Issue 1. – P. 13–22. doi: 10.5121/ijcses.2012.3102
3. Ashadevi, B. Cost Effective Approach for Materialized Views Selection in Data Warehousing Environment [Text] / B. Ashadevi, R. Balasubramanian // International Journal of Computer Science and Network Security. – 2008. – Vol. 8, Issue 10. – P. 236–242.
4. Ashadevi, B. A Framework for the View Selection Problem in Data Warehousing Environment [Text] / B. Ashadevi, P. Navaneetham, R. Balasubramanian // International Journal on Computer Science and Engineering. – 2010. – Vol. 2, Issue 9. – P. 2820–2826.

5. Jogekar, R. N. Design and Implementation of Algorithms for Materialized View Selection and Maintenance in Data Warehousing Environment [Text] / R. N. Jogekar, A. Mohd // International Journal of Emerging Technology and Advanced Engineering. – 2013. – Vol. 3, Issue 9. – P. 134–140.
6. Chaudhuri, S. Self-Tuning Database Systems: A Decade of Progress [Text] / S. Chaudhuri, V. Narasayya // Proc. of the 33rd International Conference on Very Large Data Bases, 2007. – P. 3–14.
7. Shukla, A. Materialized View Selection for Multidimensional Datasets [Text] / A. Shukla, P. Deshpande, J. F. Naughton // Proc. of the 24rd International Conference on Very Large Data Bases, 1998. – P. 488–499.
8. Baralis, E. Materialized Views Selection in a Multidimensional Database [Text] / E. Baralis, S. Paraboschi, E. Teniente // Proc. of the 23rd International Conference on Very Large Data Bases, 1997. – P. 156–165.
9. Yang, J. Algorithms for Materialized view design in Data Warehousing Environment [Text] / J. Yang, K. Karlapalem, Q. Li // Proc. of the 23rd International Conference on Very Large Data Bases, 1997. – P. 136–145.
10. Derakhshan, R. Parallel Simulated Annealing for Materialized View Selection in Data Warehousing Environments [Text] / R. Derakhshan, B. Stantic, O. Korn, F. Dehne // Algorithms and Architectures for Parallel Processing. – 2008. – Vol. 5022. – P. 121–132. doi: 10.1007/978-3-540-69501-1_14
11. Zhang, C. An Evolutionary Approach to Materialized Views Selection in a Data Warehouse Environment [Text] / C. Zhang, X. Yao, J. Yang // IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews). – 2001. – Vol. 31, Issue 1. – P. 282–294. doi: 10.1109/5326.971656
12. Novokhatskaya, E. A. Calculation the materialization factor in query evaluation during the maintenance of materialized views [Text] / E. A. Novokhatskaya // Vestnik KhNTU. – 2015. – Vol. 2, Issue 53. – P. 128–133.
13. Aho, A. V. Compilers: Principles, Techniques, and Tools [Text] / A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman. – Addison-Wesley, 2007. – 1000 p.
14. Parr, T. The Definitive ANTLR Reference [Text] / T. Parr. – San Francisco, 2013. – 328 p.
15. Whitney, D. A genetic algorithm tutorial [Text] / D. Whitney // Statistics and Computing. – 1994. – Vol. 4, Issue 2. – P. 65–85. doi: 10.1007/bf00175354
16. Novokhatskaya, E. A. Development of technology for automated creation of materialized views [Text] / E. A. Novokhatskaya // Eastern-European Journal of Enterprise Technologies. – 2015. – Vol. 5, Issue 4 (77). – P. 64–73. doi: 10.15587/1729-4061.2015.50892