MARKOV CHAIN FOR THE RECOMMENDATION OF MATERIALIZED VIEWS IN REAL-TIME DATA WAREHOUSE

Abdelaziz Abdellatif[1], Ali Ben Ammar[2], and Chadlia Mazlout[3]

[1]Faculté des sciences de Tunis - Tunis El Manar University - Tunisia
[2]University College of Taima- Tabuk University. Arabie Saoudite.
[3] Faculté des sciences de Tunis - Tunis El Manar University – Tunisia

ABSTRACT

In this paper we propose an approach, which is based on Markov Chain, to cluster and recommend candidate views for the selection algorithm of materialized views. Our idea is to intervene at regular period of time in order to filter the candidate views which will be used by an algorithm for the selection of materialized views in real-time data warehouse. The aim is to reduce the complexity and the execution cost of the online selection of materialized views. Our experiment results have shown that our solution is very efficient to specify the more profitable views and to improve the query response time.

KEYWORDS

Query response time optimization, Materialized view selection, View clustering, candidate View recommendation, real-time data warehouse, Markov chain.

1. INTRODUCTION

View materialization is a technique to improve query response time in database, web database, and data warehouse environments. This technique aims to reduce the cost of generating data from databases to serve repetitive user queries. Its principle is to store and then update the results, called materialized views, of some repetitive queries. The mains issues of view materialization are: (i) the selection of the appropriate views to be materialized when there are resource constraints; (ii) the maintenance of the materialized views [16,17,26,32,34].

In this paper we will interest in the first issue i.e. the selection of materialized views. The majority of the research works [1,2,3,4,5,6,8,9,10,11,12,13,14,15,18,19,22,23, 24,25,27,28,30,31,33], addressing this issue, present a policy of materialized view selection. This policy should identify the views that improve the query response time and having a weak maintenance load. It should respond to three questions: (i) how identify the materialized views? (ii) what are the constraints to be applied in order to respect the resources capacities? and (iii) when execute and re-execute the selection algorithm? The result of each execution of the selection algorithm is called materialization plan. This plan specifies the state of each candidate view: materialized or not.

The first question is about the cost model which is used to evaluate the view contribution. In general, this contribution depends on the access and the maintenance costs which depend on the frequencies and the response time of the requests. We call materialization profit (mp) of a view v_i the gain of query response time that should be produced if we materialize v_i. The profit of a materialization plan is the sum of the materialized view profits. So, this question consists of defining how to calculate the materialization profit and how to identify the set of views that maximize this global profit.

The second question is about the resource constraints that should be respected when identifying materialized views. The mainly resource constraints are the memory space and the maintenance cost. These two constraints are widely used in the database and data warehouse environments. Another type of constraints is introduced in [21]. It allows selecting the set of views that respect a user-specified guarantee, θ, of data quality. The data quality is defined as the freshness of the data served to the user.

The third question of a selection policy is about the replacement period of the materialization plan that is the time interval between each two consecutive executions of the selection algorithm. In static environments, where the user needs are well known and the data updates are offline and less frequent, the materialization systems intervene at regular and long time periods to refresh the materialization plan. This make the intervention cost low because the executions of the selection algorithm are less frequent. However in real-time environments, where the data updates are online and high frequent, the materialized view selection should be dynamic that is depending on the variation of the access and maintenance cost. In such environments, the materialization system should intervene online, i.e. at short time periods, to calibrate the materialization plan. The advantage of this dynamic selection and refreshment of materialization plan is the online reaction to the variation of the materialization plan profitability.

Despite its impact on the total query response time in real-time environments, the intervention cost is less addressed in the literature [7]. Two ways may used to improve this intervention cost: (i) reduce the number of online interventions; and (iii) improve the complexity of the selection algorithm. The complexity of the selection algorithm depends on the manner in which the algorithm iterates to identify views and essentially on the number of candidate views.

In this paper we address the issue of improving the intervention cost in real-time data warehouse by reducing the number of candidate views at each execution of the selection algorithm. Real-time data warehouse is characterized by an online and frequent update of source data. So, the refreshment of the materialization plan should be frequent to ensure a high degree of data quality and to guarantee the efficiency of materialized views. Our contributions are: firstly we classify views based on their profitability historic; and secondly we filter the list of candidate views at each execution of the selection algorithm i.e. we consider only some classes of views. We analyze the access and the maintenance historic of the views in order to deduce for each view the more frequent duration of its profitability i.e. the more frequent time period during which the view produces a profit for the materialization. Our idea is that if a view is selected by the selection algorithm, then it spends its profitability duration materialized. In other words it will be added to the list of candidate views, for the next executions of the selection algorithm, only after its profitability duration and not at each execution.

In the next section we will present the related works. The section 3 presents our solution. The section 4 describes the experiment results. The section 5 is the conclusion.

2. RELATED WORKS

The materialization of views has been widely used in database and data warehouse environments as a technique to improve query response time. In the beginning, the research works are concentrated on the static selection of materialized views [4,8,12,13,15, 27, 31]. When the databases are largely used on the web and on real-time environments, the researchers were oriented to the dynamic selection of views [1,2,3, 5,6, 9, 10,11, ,14,18,19,20, 21, 22, 23,24,25,28,29,30, 33]. The majority of these works have addressed the issue of how the selection algorithm should identify the appropriate views that improve the query response time. Among these approaches, [1,9,10,24] propose to classify views in order to define the list of candidate views or to specify the list of views to be materialized. [9] proposes summarizing and then merging the user queries in order to construct general schemas of materialized views. The aim is to serve a lot of user queries by a limited number of materialized views. [7] proposes a cost model to filter the candidate views on the web.

So, after the analysis of these works we have remarked that the online intervention may take a long time because of the complexity of the selection algorithm, the high intervention frequency or the high number of candidate views. This long time may decrease the server performance and consequently it will have a negative impact on the query response time. In this paper we present an approach to improve this intervention load. Our solution consists of reducing the number of candidate views which affect the intervention load.

To the best of our knowledge, the intervention load has not yet been treated in the selection of materialized views in real-time data warehouses. Also, we are not aware of any work which continually cluster and filter views for the online selection.

3. APPROACH PRESENTATION

3.1 Principle

In this approach we consider that the selection algorithms of materialized views are executed at regular periods of time called selection period and denoted sp. Since the selection is online, these periods should be shorts. The aim of our approach is to specify, at the beginning of each sp, a specific set of candidate views for these algorithms so that the materialization profit is improved.

To recommend candidate views for the selection algorithms we intervene at regular period of time p, called recommendation period. p is a multiple of sp i.e. $p = n \times sp$ with n an integer greater than 1. For example p may be 1 hour and $sp = 10$ minutes that is $n = 6$. At the beginning of each recommendation period, we classify the candidate views into n profitability classes, $pc_1, pc_2, \dots, pc_n$ so that pc_k contains the views which if they are materialized they produce a materialization profit during k successive selection periods. This clustering is based on the access and maintenance historic of the views. Consequently, the views of the class pc_k, they will not be presented as candidate views for the selection algorithm only after k selection periods.

Our approach follows 3 steps to recommend candidate materialized views in real-time data warehouse environment:

1. Analyze the access and the maintenance historic of views in order to deduce the profitability historic of views.
2. Construct the transition matrix of view profitability.
3. Recommend candidate views for the next recommendation period.

3.2 Analysis of view historic

We start this analysis by specifying the states of the views (profitable for the materialization or not) during each selection period sp of the historic interval. A view is considered profitable for the materialization when its access cost is greater than its maintenance one during the specified sp i.e. it has been frequently accessed and less updated. So, in this step we verify if a view v_i has produced a materialization profit during a past selection period sp_t. We have represented the result of this analysis in a matrix H as follow:

$$H(i,j) = \begin{cases} 1 \; if\ the\ view\ v_i\ was\ profitable\ along\ sp_j \\ 0 \; if\ not \end{cases}$$

Example:

In this example we consider that:

- We refresh the lists of candidate views after each 2 hours, which represent the length of the recommendation period i.e. $p = 2$ hours;
- We refresh the materialization plan after each 30 minutes, which represent the length of the selection period i.e. $sp = 30$ minutes. So the recommendation period will contain 4 selection periods i.e. $p = 4 \times sp$.
- We have an historic of 6 recommendation periods.
- We have 5 materialized views.

In the table 1 we have presented an example of profitability historic of these 5 views. As an example, v_1 has produced a profit during sp_1 and sp_3 and it has produced a lost during sp_2 and sp_4.

Table 1. Example of profitability historic

Recommendation periods	**Selection periods**	v_1	v_2	v_3	v_4	v_5
p_1	sp_1	1	0	1	0	0
	sp_2	0	0	1	1	0
	sp_3	1	0	0	1	1
	sp_4	0	1	1	1	1
p_2	sp_5	1	1	1	1	1
	sp_6	0	0	1	0	1
	sp_7	1	1	0	0	1
	sp_8	0	1	1	1	0
p_3	sp_9	0	0	0	0	0
	sp_{10}	1	0	0	0	1
	sp_{11}	1	0	0	1	0
	sp_{12}	0	0	1	1	1
p_4	sp_{13}	1	1	1	1	0
	sp_{14}	0	1	1	1	1
	sp_{15}	0	0	0	1	1
	sp_{16}	1	1	0	1	1
p_5	sp_{17}	1	1	1	1	1
	sp_{18}	0	1	1	0	1
	sp_{19}	0	1	1	0	1
	sp_{20}	0	1	0	0	0
p_6	sp_{21}	0	1	1	0	1
	sp_{22}	0	0	0	1	1
	sp_{23}	1	1	1	1	0
	sp_{24}	1	0	1	1	0

We call profitability class, pc_b, the set of views which are kept profitable during b successive selection periods. So, during a recommendation period p_j, a view v_i may have several pc. In table 1 and during the recommendation period p_1, v_3 was profitable during 2 successive selection periods sp_1 and sp_2. Then, it was profitable during sp_4. So, v_3 pc_2; and v_3 pc_1. However during the recommendation period p_1, v_1 pc_1; v_2 pc_1; v_4 pc_3; v_5 pc_2. When the number of selection periods per recommendation period is greater than 6, we can found a view which belongs to more than two classes. For example if the number of selection periods is 6, a view may be in pc_1, pc_2 and pc_3.

In our approach we need that each view belongs to only one profitability class during a recommendation period. So, the aim of this analysis is to specify the appropriate profitability class for each couple (view, recommendation period). For this reason we calculate the average duration of profitability (ADP) of each view v_i during each recommendation period p_j. Then, we specify the profitability class based on the ADP value. We apply the following formula to calculate the ADP of the couple (v_i, p_j):

$$\mathrm{ADP}(v_i, p_j) = \frac{\sum_{k=1}^{n} \mathrm{frequency}(v_i, pc_k, p_j) \times k}{\sum_{k=1}^{n} \mathrm{frequency}(v_i, pc_k, p_j)} \qquad (1)$$

With:

- n: the number of selection periods per recommendation period;
- $frequency(v_i, pc_k, p_j)$: the number of sequences with k successive selection periods, within the recommendation period p_j, on which the view v_i was profitable. For example, from the table 1, $frequency(v_3, pc_1, p_1)$ =1, $frequency(v_3, pc_2, p_2)$ =1, $frequency(v_3, pc_3, p_1)$ =0 and $frequency(v_3, pc_4, p_1)$ =0, $frequency(v_1, pc_1, p_1)$ =2, ……
-

Based on the table1, we present in table2 the ADP of all the views.

Table 2. ADP values of the 5 views presented in table1.

p_j/v_i	v_1	v_2	v_3	v_4	v_5
p_1	1	1	1.5	3	2
p_2	1	1.5	1.5	1	3
p_3	2	0	1	2	1
p_4	1	1.5	2	4	3
p_5	1	4	3	1	3
p_6	2	1	1.5	3	2

Table 3. profitability classes of the 5 views presented in table1.

p_j/v_i	v_1	v_2	v_3	v_4	v_5
p_1	1	1	1	3	2
p_2	1	2	1	1	3
p_3	2	1	1	2	1
p_4	1	1	2	4	3
p_5	1	4	3	1	3
p_6	2	1	1	3	2

The profitability classes of the 5 views are deduced from the ADP values in table 2 and they are presented in table 3. The profitability class is the integer part of the ADP value. It represents the more frequent period of time in which the view was continually profitable. If the ADP value is 0 then we affect the profitability class pc_1 to the corresponding view in order to be recommended as a candidate view at each execution of the selection algorithm. For example, during p_1 the view v_4 was frequently profitable for 3 successive selection periods. However during p_3, it was frequently profitable for 2 successive selection periods. So, the profitability classes of v_4 for these two recommendation periods are respectively pc_3 and pc_2.

3.3 Construct the transition matrix

This matrix summarizes the transition probabilities of views between profitability classes during the past recommendation periods. We use these probabilities to recommend the candidate materialized views. For example in table 3, v_1 was kept in the class pc_1 when we move from p_1 to p_2. The same things when we move from p_4 to p_5. So, among the 5 transitions (total number of transitions of v_1 in table 3) between profitability classes, v_1 was moved 2 times from pc_1 to pc_1. This makes the transition probability of v_1 from pc_1 to pc_1 be 2/5 .The figure 1.(a) presents the transition probabilities of v_1. Those of v_4 are presented in figure 1.(b).

$$\begin{array}{c|cccc} & pc_1 & pc_2 & pc_3 & pc_4 \\ \hline pc_1 & 2/5 & 2/5 & 0 & 0 \\ pc_2 & 1/5 & 0 & 0 & 0 \\ pc_3 & 0 & 0 & 0 & 0 \\ pc_4 & 0 & 0 & 0 & 0 \end{array}$$

(a)Transition matrix of v_1

$$\begin{array}{c|cccc} & pc_1 & pc_2 & pc_3 & pc_4 \\ \hline pc_1 & 0 & 1/5 & 1/5 & 0 \\ pc_2 & 0 & 0 & 0 & 1/5 \\ pc_3 & 1/5 & 0 & 0 & 0 \\ pc_4 & 1/5 & 0 & 0 & 0 \end{array}$$

(b) Transition matrix of v_4

Figure 1. Examples of transition matrixes.

3.4 Recommend candidate views

To recommend candidate views we use the Markov chain as a technique to identify the future profitability class of a view v_i during the future recommendation period.

Markov chain is described as follows: We have a set of states, $\mathrm{S} = \{\mathrm{s}_1, \mathrm{s}_2, \ldots, \mathrm{s}_\mathrm{r}\}$. The process starts in one of these states and moves successively from one state to another. Each move is called a step. If the chain is currently in state s_i, then it moves to state s_j at the next step with a probability denoted by p_{ij}, and this probability does not depend upon which states the chain was in before the current state. The probabilities p_{ij} are called transition probabilities. The process can remain in the state it is in, and this occurs with probability p_{ii}. An initial probability distribution, defined on S, specifies the starting state. Usually this is done by specifying a particular state as the starting state.

In our case, the states represent the profitability classes and the transition probabilities are represented by the transition matrix. The figure 2 illustrates the transitions of the view v_1 by a Markov chain.

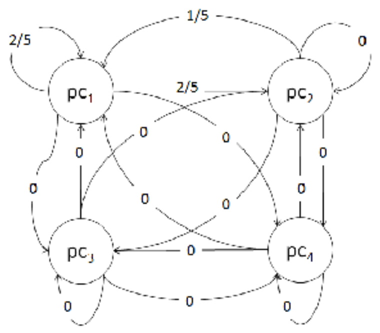

Figure 2. Representation of v_1 transitions by a Markov chain.

Our idea to recommend candidate view is to select the next profitability class for each view v_i based on its current state and the transition probabilities outgoing from it. Precisely, we select the higher outgoing probabilities. For example, suppose that we are at the end of the recommendation period p_6 and we want to recommend candidate views for the recommendation period p_7. So, from the figure 2, the next profitability class of v_1 will be more probably pc_1 because it is currently (i.e. at the end of p_6) in the state pc_2 and the higher outgoing probability from pc_2 is 1/5 which correspond to the transition oriented to the state pc_1.

When there are more than 2 outgoing transitions with the same value we select the profitability class having the long duration. This is because when we choose the longer class, we reduce the number of candidate views used in the refreshment of the materialization plan during the recommendation period. Consequently, we join the aim of our approach which is improving the refreshment cost of the materialization plan.

Based on the affectation in table3, we present, in the table 4, the recommended profitability classes during the next period p_7 and theirs sets of views.

Table 4. Recommended profitability classes of the 5 views for p_7.

pc_j	$sets\ if\ views\ of\ \ pc_j$
cp_1	$\{v_1, v_4\}$
cp_2	$\{v_3\}$
cp_3	$\{v_5\}$
cp_4	$\{v_2\}$

So, the candidate views of the selection algorithm during p_7 will be organized as follow:

- At the beginning of sp_1, the list of recommended views $V = \{v_1, v_2, v_3, v_4, v_5\}$. Suppose that, at this step, the selection algorithm materialize the list of views $M = \{v_1, v_2\}$
- At the beginning of sp_2, $V = \{v_1, v_3, v_4, v_5\}$. Since v_1 cp_1 , it should be added to the list of candidate views at the beginning of each selection period. Since v_2 cp_4 , it should be added to the list of candidate views only after 4 selection periods i.e. in this example it is kept materialized during the recommendation period. Since v_3, v_4 *and* v_5 are not materialized in the previous selection period, they should be added to the list of candidate views. Suppose that, at this step, the selection algorithm materializes the list of views $M = \{v_3, v_4, v_5\}$.

- At the beginning of sp_3, $V = \{v_1, v_4\}$. Since v_4 □ cp_1 , it should be added to the list of candidate views at the beginning of each selection period. Since v_3 □ cp_2 , it should be added to the list of candidate views only after 2 selection periods i.e. in this example it is kept materialized during the selection periods sp_2 and sp_3 . Since v_5 □ cp_3 , it should be added to the list of candidate views only after 3 selection period i.e. in this example it is kept materialized during the selection periods sp_2 , sp_3 and sp_4 . Since v_1 is not materialized in the previous selection period, it should be added to the list of candidate views. Suppose that, at this step, the selection algorithm materialize the views $M = \{v_4\}$.
- At the beginning of sp_4, $V = \{v_1, v_3, v_4\}$.

For the next recommendation period we specify the profitability classes of the views during the current recommendation period and then we refresh the transition matrix. In other words we continually refresh the transition matrix and redefine the sets of candidate views. It's a continually clustering and recommendation of candidate views.

Suppose that at the end of p_7, the profitability class of v_1 is pc_1. Consequently we should firstly update the transition probability of v_1 and then reselect its new profitability class for the period p_8.

In the table 5 we present an example of profitability classes of the 5 views in the recommendation period p_7 and in the figure 3 we present the update of the transition matrixes of v_1 and v_4. So, the next profitability classes of these two views will be respectively $pc_2$and pc_3.

Table 5. profitability classes of the 5 views during p_7.

p_j/v_i	v_1	v_2	v_3	v_4	v_5
p_7	1	3	4	3	3

$$\begin{array}{c|cccc} & pc_1 & pc_2 & pc_3 & pc_4 \\ \hline pc_1 & 2/6 & 2/6 & 0 & 0 \\ pc_2 & 2/6 & 0 & 0 & 0 \\ pc_3 & 0 & 0 & 0 & 0 \\ pc_4 & 0 & 0 & 0 & 0 \end{array}$$

(a) Update of the transition matrix of v_1

$$\begin{array}{c|cccc} & pc_1 & pc_2 & pc_3 & pc_4 \\ \hline pc_1 & 0 & 1/6 & 1/6 & 0 \\ pc_2 & 0 & 0 & 0 & 1/6 \\ pc_3 & 1/6 & 0 & 1/6 & 0 \\ pc_4 & 1/6 & 0 & 0 & 0 \end{array}$$

(b) Update of the transition matrix of v_4

Figure 3. Examples of updates of transition matrixes.

4. EXPERIMENTS

To evaluate our approach, we have simulated, by using the TPC-H benchmarking [http://www.tpc.org/tpch/], the load of a real-time data warehouse. This benchmark proposes 22 OLAP queries. In this experiment we have considered the following:

- For each query of the benchmark we have a corresponding materialized view.
- The length of the recommendation period is 4 hours.
- Simulate the historic of these 22 views for 18 recommendation periods $p_1, p_2, \ldots, p_{18}$ i.e. an historic of 3 days.

- The length of the selection period is 30 minutes i.e. 8 selection periods per recommendation period $sp_1, sp_2, \dots, sp_8$.
- The speed in which accesses can be processed, the incoming update stream, and the speed in which updates can be performed are inputs to the simulator. We have supposed 15 access requests and 3 update requests per selection periods. So, we have in total 720 accesses per day.

After the analysis of this historic and the constructions of transition matrixes, we have recommended for each view a profitability class. Then, we have defined the sets of candidate views for the next recommendation period i.e. for p_{19}.

During the recommendation period number, p_{19} , we have executed a greedy algorithm for the selection of materialized views. This algorithm applies a constraint of maintenance cost. It identifies, at the end of each iteration, the view v_i with the best (higher) ratio $\frac{materialization\_profit(v_i)}{update\_cost(v_i)}$ until it reaches the maintenance constraint. So, the problem of selection of materialized view is formulated as follow:

$$max \sum_{i=1}^{n} x_i \cdot profit(v_i)$$
$$under\ constraint \sum_{i=1}^{n} x_i \cdot update\_cost(v_i) \le CM$$

With:

- $n = number\ of\ candidate\ webviews$
- $x_i = \begin{cases} 1\ if\ v_i\ is\ materialized \\ 0\ if\ not \end{cases}$
- $CM = the\ value\ of\ the\ maintenance\ constraint\ in\ seconds.$

In order to evaluate our approach we have calculate the materialization profit at the end of the recommendation period p_{19} as follow:

$$mp = ec_{(-m)}(R) - \left[ec_{(+m)}(R) + uc(M) + rc\right]$$

With:

- mp: the total materialization profit at the end of p_{19}.
- $ec_{(-m)}(R)$: the execution cost of the 22 queries of the benchmark during p_{19} without materializing $(-m)$ any view. $R = \{r_1, r_2, \dots, r_{22}\}$.
- $ec_{(+m)}(R)$: the execution cost of the 22 queries of the benchmark during p_{19} when there is a materialized plan (M) i.e. some views are materialized.
- $uc(M)$: the update cost of the materialized views of the plan M during p_{19}.
- rc: the refreshment cost of the materialization plan M during p_{19} i.e. the total execution cost of the selection algorithm during the 8 selection periods of p_{19}.

In the first experiment we have varied the maintenance constraint and we have calculated the materialization profit. Then we have compared the results of our approach with those of the approach DMVSA presented in [9]. This approach consists of reducing the number of candidate views by summarizing and then integrating the user queries. The idea of this approach is to construct general schemas of materialized views.

The results of this experiment are presented in figure 4. In this figure we have demonstrate that our approach produces a materialization profit better than the DMVSA approach for the restrictive values of the selection constraint. This approves that, in such conditions, our approach is more efficient than the DMVSA approach to identify the appropriate candidate materialized views and to reduce the refreshment cost.

Another obvious result is that the materialization profit increases with largest values of the maintenance constraint. In such conditions, the approach DMVSA produces better materialization profits because its lists of candidate views are larger than those of our approach. However, by applying our approach and with a large maintenance constraint, the query response time may be improved by 180 seconds during a recommendation period which contains 120 access queries i.e. about 1.5 seconds per query.

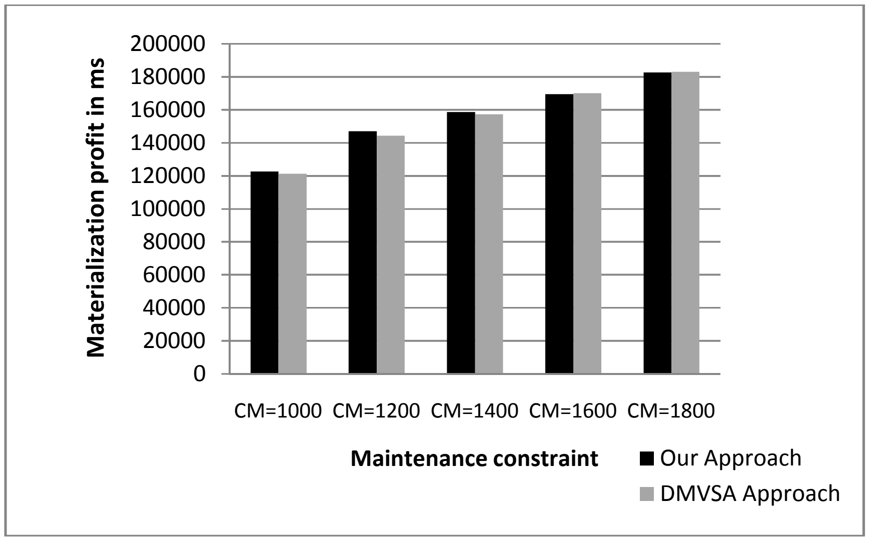

Figure 4. Impact of the maintenance constraint on the materialization profit.

In the second experiment we have varied the historic length from 6 recommendation periods to 18 recommendation periods. We have considered a maintenance constraint equal to 1000 seconds. The aim is to evaluate the impact of the historic on the materialization profit.

The results of this experiment are presented in figure 5. In this figure we have demonstrated that our approach produces a materialization profit better than the DMVSA approach especially when the number of the historic periods is small. This is proves that our approach is efficient to identify the appropriate candidate materialized views and to improve the refreshment cost even if the historic is limited.

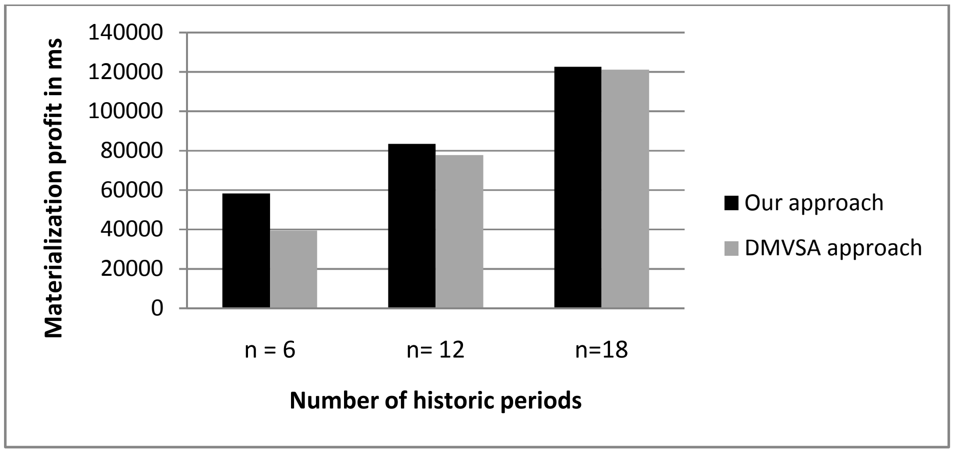

Figure 5. Relationship between the materialization profits and the historic length.

In the figure 5 we have demonstrated that the materialization profit increases when we extend the analyzed historic of views. This is because a largest historic makes the recommendation of profitability classes more efficient.

5. CONCLUSION

In this paper we have presented an approach for the recommendation of candidate materialized views in real-time data warehouse environment. Our main contribution is to cluster the candidate views and to affect classes of them for each execution of the selection algorithm. The aim is to improve the query response time by reducing the refreshment cost of the materialization plan. Our experiment results have shown that the proposed approach can improve the query response time by than 1 second per query. Compared to an existing approach and for restrictive values of the selection constraints, our solution was demonstrated more efficient for the recommendation of materialized views. In the future works we will attempt to integrate this solution in a complete approach for the online selection of materialized views.

REFERENCES

[1]. K. Aouiche, P. Jouve, and J. Darmont.(2006) "Clustering-based materialized view selection in datawarehouses", *In ADBIS'06, volume 4152 of LNCS, pages 81–95, 2006*.

[2]. B.Ashadevi, R.Balasubramanian, (2008) "Cost Effective Approach for Materialized Views Selection in Data Warehousing Environment", *proc. of the International Journal of Computer Science and Network security* Vol. 8, No. 10, pp. 236-242, 2008.

[3]. B.Ashadevi and R.Subramanian, (2009) " Optimized Cost Effective Approach for Selection of Materialized views in Data Warehousing", *International Journal of Computer Science and Technology,* Vol.9 No.1 ,April 2009.

[4]. X. Baril, Z. Bellahsene. (2003) "Selection of Materialized Views: A Cost-Based Approach". *CAiSE* 2003, pages 665-680. 2003

[5]. Z. Bellahsene, M. Cart, and N. Kadi. (2010) "A cooperative approach to view selection and placement in P2P systems". *In OTM*, pages 515–522, 2010.

[6]. L. Bellatreche, K. Boukhalfa.(2005) "Une repartition statique et dynamique de l'espace entre les vues matérialisées et les index dans les entrepôts de données". *International Symposium on Programming and Systems (ISPS'05),* USTHB - Alger, 2005

[7]. A. Ben Ammar, A. Abdellatif. (2012) "Cost Model for the Recommendation of Materialized Webviews". *International Journal of Data Mining & Knowledge Management Process*. Vol 1, N° 2. 2012

[8]. H. Ben Ghezala, A. Abdellatif, A. Ben Ammar. (2005) "An Approach to Specify When Reselecting Views to be Materialized". *1ères journées francophone sur les Entrepôts de Données et l'Analyse en ligne (EDA 2005)*, Lyon, Juin 2005; RNTI, Vol. B-1, Cépaduès, Toulouse, pages 161-176.

[9]. Manoj S. Chaudhari and Chandrashekhar Dhote. (2010). "Dynamic materialized view selection algorithm: a clustering approach". In *Proceedings of the Second international conference on Data Engineering and Management* (ICDEM'10)

[10]. Yogeshree D. Choudhari & S. K. Shrivastava. (2012) "Cluster Based Approach for Selection of Materialized Views ",*in International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 7*, July 2012

[11]. Gang Gou, Jeffery Xu Yu and Hongjun Lu, (2006) "A* Search: An Efficient and Flexible Approach to Materialized View Selection", *IEEE Trans. on Systems, Man and Cybernetics – Part C: Appl. And Reviews*, Vol. 36, No. 3, May 2006.

[12]. H. Gupta, (1997) "Selection of Views to Materialize in a Data Warehouse," *Proceedings of ICDT*, pp.98-112, 1997.

[13]. H. Gupta, V. Harinarayan, A. Rajaraman, and J.D. Ullman. (1997) "Index Selection for OLAP". *ICDE. IEEE Computer Society, Washington, DC, pages 208-219.* 1997

[14]. H. Gupta, and I. S. Mumick. (2005) "Selection of Views to Materialize in a Data Warehouse". *IEEE Trans. on Knowl. and Data Eng. 17(1)*, pages 24-43. Jan. 2005.

[15]. H. Gupta, I. S. Mumick. (1999) "Selection of Views to Materialize Under a Maintenance-Time Constraint". *ICDT*. 1999.

[16]. Hemant Jain, AnjanaGosain (2012) "QOP: Proposed Framework for Materialized View Maintenance in Data Warehouse Evolution" *International Journal of Scientific & Engineering Research*, Volume 3, Issue 7, July-2012 1 ISSN 2229-5518

[17]. P. P. Karde and V. M. Thakare, (2010) "Selection & Maintenance of Materialized View and It's Application for Fast Query Processing A survey", *International Journal of Computer Science & Engineering Survey (IJCSES)* Vol.1, No.2, November 2010

[18]. Deepika Kirti, Jaspreeti Singh, (2014) "Assortment of Materialized View: A Comparative Survey in Data Warehouse Environment", *International Journal of Computer Applications (0975 – 8887), Volume 96– No.7, June 2014*

[19]. Y. Kotidis, N. Roussopoulos. (1999) "Dynamat: A dynamic view management system for data warehouses". *In ACM SIGMOD International Conference on Management of Data (SIGMOD 1999), pages 371-382. Philadelphia, USA, 1*999.

[20]. Labrinidis, Q. Luo, J. Xu, W. Xue. (2009) "Caching and Materialization in Web Databases", *Foundations and Trends in Databases Vol. 3, No. 2, pages 169-266*. December 2009.

[21]. Labrinidis, N. Roussopoulos. (2001) "Adaptive WebView Materialization". *WebDB'01*, pages 85-90. 2001

[22]. Xin Li, Xu Qian, Junlin Jiang and Ziqiang Wang, (2010) "Shuffeled Frog Leaping Algorithm for Materialized Views Selection", *Second IEEE International Workshop on Education Technology and Computer Science 2010.*

[23]. Zhou Lijuan, Ge Xuebin, Wang Linshuang, Shi Qian, (2009) "Efficient Materialized View Selection Dynamic Improvement Algorithm", *Sixth IEEE International Conference on Fuzzy Systems and Knowledge Discovery*, 2009.

[24]. H. Mahboubi, K. Aouiche et J. Darmont. (2006) "Materialized View Selection by Query Clustering in XML Data Warehouses". *4th International Multiconference on Computer Science and Information Technology (CSIT 06), Amman, Jordan, volume 2, pages 68–77,* April 2006.

[25]. Imene Mami, Zohra Bellahsene, Remi Coletta (2013) "A Declarative Approach to View Selection Modeling". *T. Large-Scale Data- and Knowledge-Centered Systems* 10: 115-145 (2013)

[26]. T.Nalini, A.Kumaravel, K.Rangarajan (2012) "A Novel Algorithm With Im-Lsi Index For Incremental Maintenance Of Materialized View" *Journal of Computer Science & Technology* (JCS&T);2012, Vol. 12 Issue 1, p32

[27]. P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobe. (2000). "Efficient and extensible algorithms for multi query optimization". *In SIGMOD Conference*, pages 249–260, 2000.

[28]. T. V. Vijay Kumar. (2013) "Answering query-based selection of materialised views". *IJIDS* 5(1): 103-116 (2013)

[29]. S. Saidi, Y. Slimani, and K. Arour.(2007) "Webview selection from user access Patterns". *In PIKM '07, pages 171-176. Lisboa, Portugal* . November 2007.

[30]. Jin-Hyuk Yang and In-Jeong Chung.(2006). “Materialized View Selection Algorithm in Data Warehouse” *International Journal of Information Processing Systems, Vol.2, No.2*, June 2006
[31]. J.Yang, K. Karlapalem, and Q. Li. (1997) “A framework for designing materialized views in data warehousing environment”. *Proceedings of 17th IEEE International conference on Distributed Computing Systems, Maryland, U.S.A.*, May 1997.
[32]. Xiaogang Zhang; Luming Yang; De Wang (2010), "Incremental view maintenance based on data source compensation in data warehouses," *International Conference on Computer Application and System Modeling (ICCASM)*, 2010, vol.2, no., pp.V2-287,V2-291, 22-24 Oct. 2010
[33]. J. Zhou, P. Larson, J. Goldstein, and L. Ding.(2007). “Dynamic materialized views”. *In ICDE, pages 526–535, 2007.*
[34]. Y. Zhuge, H. Garcia-Molina, J. Hammer, and J. Widom (1995). “View maintenance in a warehousing environment”. *In Proceedings of SIGMOD, pages 316–327,* May 1995.