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

Abdelaziz Abdellatif¹, Ali Ben Ammar², and Chadlia Mazlout³
¹Faculté des sciences de Tunis - Tunis El Manar University - Tunisia
²University College of Taima- Tabuk University. Arabie Saoudite.
³Faculté 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 main 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, \( \theta \), 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 (ii) 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 short. 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, \( pc_1, pc_2, ..., pc_n \) 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 $sp_e$. We have represented the result of this analysis in a matrix $H$ as follow:

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

**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, $v_4$ 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**

<table>
<thead>
<tr>
<th>Recommendation periods</th>
<th>Selection periods</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
<th>$v_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>$sp_1$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$sp_2$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$sp_3$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$sp_4$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$p_2$</td>
<td>$sp_5$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$sp_6$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$sp_7$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$sp_8$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$p_3$</td>
<td>$sp_9$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$sp_{10}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$sp_{11}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$sp_{12}$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$p_4$</td>
<td>$sp_{13}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$sp_{14}$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$sp_{15}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$sp_{16}$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$p_5$</td>
<td>$sp_{17}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$sp_{18}$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$sp_{19}$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$sp_{20}$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$p_6$</td>
<td>$sp_{21}$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$sp_{22}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$sp_{23}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$sp_{24}$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
We call profitability class, \( pc \), 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, \( v_3 \in pc_2 \) and \( v_3 \in pc_1 \). However, during the recommendation period \( p_1 \), \( v_1 \in pc_1 \); \( v_2 \in pc_1 \); \( v_4 \in pc_3 \); \( v_5 \in pc_2 \). When the number of selection periods per recommendation period is greater than 6, we can find 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 \((v_i, p_j)\):

\[
ADP(v_i, p_j) = \frac{\sum_{k=1}^{n} frequency(v_i, pc_k, p_j) \times k}{\sum_{k=1}^{n} frequency(v_i, pc_k, p_j)}
\]

(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 \( frequency(v_3, pc_3, p_1) = 0 \), \( frequency(v_1, pc_1, p_1) = 2 \), ……

Based on the table 1, we present in table 2 the ADP of all the views.

<table>
<thead>
<tr>
<th>( p_j / v_i )</th>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>( v_3 )</th>
<th>( v_4 )</th>
<th>( v_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_1 )</td>
<td>1</td>
<td>1</td>
<td>1.5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>( p_2 )</td>
<td>1</td>
<td>1.5</td>
<td>1.5</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>( p_3 )</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>( p_4 )</td>
<td>1</td>
<td>1.5</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>( p_5 )</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>( p_6 )</td>
<td>2</td>
<td>1</td>
<td>1.5</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3. profitability classes of the 5 views presented in table 1.

<table>
<thead>
<tr>
<th>( p_j / v_i )</th>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>( v_3 )</th>
<th>( v_4 )</th>
<th>( v_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_1 )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>( p_2 )</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>( p_3 )</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>( p_4 )</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>( p_5 )</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>( p_6 )</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
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 $p_4$ 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 $p_4$ to $p_5$. 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 $pc_2$. This makes the transition probability of $v_1$ from $pc_1$ to $pc_2$ be $2/5$. The figure 1.(a) presents the transition probabilities of $v_1$. Those of $v_4$ are presented in figure 1.(b).

![Transition matrix of $v_1$](image1.png)

![Transition matrix of $v_4$](image2.png)

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 $v_1$ during the future recommendation period.

Markov chain is described as follows: We have a set of states, $S = \{s_1, s_2, \ldots, s_5\}$. 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.
Our idea to recommend candidate view is to select the next profitability class for each view \( v \) 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 \( v \) will be more probably \( p_{c_1} \) because it is currently (i.e. at the end of \( p_6 \)) in the state \( p_{c_2} \) and the higher outgoing probability from \( p_{c_2} \) is \( 1/5 \) which correspond to the transition oriented to the state \( p_{c_1} \).

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 table 3, we present, in the table 4, the recommended profitability classes during the next period \( p_7 \) and theirs sets of views.

<table>
<thead>
<tr>
<th>( p_{c_j} )</th>
<th>sets if views of ( p_{c_j} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_{c_1} )</td>
<td>{ ( v_1, v_4 ) }</td>
</tr>
<tr>
<td>( p_{c_2} )</td>
<td>{ ( v_3 ) }</td>
</tr>
<tr>
<td>( p_{c_3} )</td>
<td>{ ( v_5 ) }</td>
</tr>
<tr>
<td>( p_{c_4} )</td>
<td>{ ( v_2 ) }</td>
</tr>
</tbody>
</table>

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 \( v_1 \in p_{c_1} \), it should be added to the list of candidate views at the beginning of each selection period. Since \( v_2 \in p_{c_4} \), 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 \( v_3, v_{4and} v_5 \) 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 $sp_2$, $V = \{v_1, v_4\}$. Since $v_4 \in cP_1$, it should be added to the list of candidate views at the beginning of each selection period. Since $v_3 \in cP_2$, 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 $v_5 \in cP_3$, 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 $p_7$.

<table>
<thead>
<tr>
<th>$p_7$</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
<th>$v_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_7$</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

(a) Update of the transition matrix of $v_1$

(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 $s_{p_{1}}, s_{p_{2}}, \ldots, s_{p_{8}}$.

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 of materialization profit $\frac{\text{profit}(v_{i})}{\text{update cost}(v_{i})}$ until it reaches the maintenance constraint. So, the problem of selection of materialized view is formulated as follow:

$$\max \sum_{i=1}^{n} x_{i} \cdot \text{profit}(v_{i})$$

under constraint

$$\sum_{i=1}^{n} x_{i} \cdot \text{update cost}(v_{i}) \leq CM$$

With:

- $n = \text{number of candidate views}$
- $x_{i} = \begin{cases} 1 & \text{if } v_{i} \text{ is materialized} \\ 0 & \text{if not} \end{cases}$
- $CM = \text{the value of the maintenance constraint in seconds}$

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

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

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. $R = \{r_{1}, r_{2}, \ldots, r_{22}\}$.
- $ec_{+m}(R)$: 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.
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.](image)

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.
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


