View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Xiamen University Institutional Repository

User-oriented Materialized View Selection[*]

Ziyu Lin[*] Dongqing Yang[*] Guojie Song[‡][†] Tengjiao Wang[*]
[*]School of Electronics Engineering and Computer Science, Peking University, Beijing, China
[‡]State Key Laboratory of Machine Perception, Peking University, Beijing, China
cainiu@263.net;{dqyang, gjsong, tjwang}@pku.edu.cn

Abstract

The problem of materialized view selection has been long researched, and many approaches have been proposed to deal with this issue. However, all the methods proposed to date strive toward improving the overall query performance, instead of being user-oriented. In this paper, we propose a new user-oriented method, called SOMES (uSer-Oriented Materialized viEw Selection), aiming at achieving better performance for view selection problem. SOMES takes into account query characteristics of different users, in which, users are classified into different groups according to their query characteristics, and various user groups are provided with their own windows, user view windows containing the views involved in their own query process. Experimental results show that our method can achieve desirable performance improvements over other methods such as BPUS and FPUS.

1 Introduction

The problem of materialized view selection can be abstractly modeled as follows [1]: given a set of queries and a number of cost-determining parameters (e.g. query frequency), output a set of view definitions that minimizes a cost function and satisfies a number of constraints.

Many solutions as been proposed in recent years (e.g. [2, 4, 7, 8]), often suggesting elaborate greedy, heuristic or randomized optimization algorithms. However, all the methods proposed to date strive toward improving the overall query performance, instead of being user-oriented. In another word, they do not take into consideration the different types of users and specific characteristics of user queries. This will sometimes lead to undesirable results in real life application. More details about this will be discussed in Section 3.

Queries of certain type of users usually have distinct features. However, hardly will there be any feature if we put queries of different types of users together. Based on this, we here propose a new user-oriented method, called SOMES (uSer-Oriented Materialized viEw Selection), aiming at achieving better performance for view selection problem. SOMES takes into account query characteristics of different users, in which, users are classified into different groups according to their query characteristics, and various user groups are provided with their own windows, *user view windows* containing the views involved in their own query process. Experimental results show that our method can achieve desirable performance improvements over other available ones such as BUPS [5] and FPUS [7].

To better deal with materialized view selection problem, we proposed in our previous work a method called MUMW(Multi-User Multi-Window)[10] similar to SOMES. The differences between MUMW and SOMES mainly include:

- In MUMW, every user has a window, while in SOMES, every user group a window.
- In MUMW, users are classified into three types according to their roles in the organization, while in SOMES, the classification is based on query characteristics.
- The space allocation and view selection processes of the two methods are much different.

The remainder of this paper is organized as follows: Section 2 gives related work, followed by discussion on view selection methods in Section 3. Section 4 presents our method in detail. Experimental results are reported in Section 5. Finally, we conclude this paper in Section 6.

2 Related Work

There is a substantial body of work dealing with selecting views to materialize in data warehouses. In [9],

[*]Supported by the Natural Science Foundation of China under Grant No. 60473051 and China HP Co. and Peking University joint project (Scalable, Real-Time and Active MPP based Data Warehouse for Telecommunication Industry).

[†]Corresponding author

Theodoratos *et al* proposed a method for constructing search spaces for materialized view selection. In [5], lattice of data cube was presented for the first time, and the author presented a greedy algorithm called BPUS with time complexity of $\mathbf{O}(kn^2)$, which takes the benefits per unit space as the criterion of view selecting. In [2], the author discussed view selecting with B-tree indexing. An algorithm called PBS was presented in [4], which adopts view size as view selection criterion and its time complexity is $\mathbf{O}(kn^2)$.

All the methods above are based on such assumption that the probability distribution of queries is available, or all queries access the data with the same probability. In fact, however, such assumption is usually not the truth, because the users can hardly give the probability distribution of queries. For this reason, in [7, 8], the relating information about queries is maintained by systems, especially, in [7] the author proposed a new algorithm called FPUS with time complexity of $\mathbf{O}(nlog_2n)$, which takes as criterion the frequency of unit space. In [8], nearest materialized parent view was presented, and B^+-tree was used to create the index of aggregated views. In addition, other methods were proposed in [3, 6] etc.

3 Discussion on view selection methods

Definition 1.*Global view window :* *Global view window, denoted by GVW, is system-allocated space with an upper bound on size, which is used to store materialized views.*

Definition 2.*Accessed view set :* *Accessed view set of a query q, denoted by $R(q)$, is a set including all the views that q accesses. Accessed view set of a user u is defined as $R(u) = \bigcup_{i=1}^{n} R(q_i)$, where q_1, q_2, ... , q_n are n queries initiated by user u.*

Usually, application system may receive a lot of OLAP queries from different users. In all the traditional view selection methods, all users are seen to be with same characteristics and are not dealt with discriminatingly. In fact, however, there can be typically three types of users in organizations, i.e., decision makers, managers and ordinary workers, whose queries features may be much different from each other. In traditional methods, there is only one GVW, and all the views involved in query process compete with each other to go into GVW. Obviously, characteristics of different types of users are ignored, which may bring the following problems in real life application:

Frequent view adjusting: Views in user's accessed view set go into and out GVW repeatedly and unreasonably. For example, now we assume that there is no space left in GVW, and all users, except u_1 and u_2, continue querying with the same characteristics as before, whereas u_1 and u_2, with much different query characteristics, change their query behavior alternately. During the time period T_1, if u_1 starts queries more frequently than u_2, then the system would put more views of $R(u_1)$ into GVW. At the same time, some or all of the views from $R(u_2)$ would be deleted from GVW. While during the time period T_2, if u_2 starts queries more frequently than u_1, then the system would behave contrarily, i.e., selecting more views from $R(u_2)$ into GVW and deleting some or all of the views from $R(u_1)$. If such case occurs again and again, it would lead to views' continuously going into and out GVW, which will be much more frequent when there are many users in the system.

Never selected views: Views in the $R(u)$ of some users may seldom or even never get any chance to enter into GVW. For example, for those methods taking "large-size-first" as view selection criterion, views in $R(u)$ of user u can not enter into GVW at all, even though u starts queries frequently, because the size of views in $R(u)$ are usually smaller than that of the views in the accessed view sets of other users. Other methods involve the same issue unexceptionally, since they all ignore the respective query characteristics of different users and make views belonging to different $R(u)$ compete together with each other.

4 User-oriented Materialized View Selection

In this part, we will first give some definitions, followed by the description of our method. Then we present algorithms for space allocation and view selection. Finally, the advantages of our method are discussed.

4.1 Definitions

Definition 3.*Multidimensional data query :* *Query q on a set of multidimensional data **MD** can be seen as an operation getting a slice or block of data from **MD**. Here $q = \{(l_1, R_1), (l_2, R_2), ..., (l_d, R_d)\}$, where d denotes the number of dimensions of **MD**; l_i, certain level in dimension d_i; R_i, the selected range of l_i.*

In Definition 3, if there is no constraint on the selection range, then we can label R_i with "*ALL*". When R_i is $< r_{i1}, r_{i2} >$, we can get a block of data in the range of $< r_{i1}, r_{i2} >$. When R_i is $\{r_{i1}, r_{i2}, ..., r_{in}\}$, we get n slices of data in the range of $< r_{i1}, r_{in} >$. Furthermore, if $< r_{i1}, r_{i2} >$ is $dom(l_i)$, namely the range is the overall dimension, then there is no need to write the tuple (l_i, R_i), and it can be ignored. For example, $q = \{(l_1, R_1), (l_3, R_3), ..., (l_d, R_d)\}$, where (l_2, R_2) is ignored.

Definition 4.*Completely-aggregated view :* *In Definition 3, if, for every dimension, it satisfies such condition that $R_i = ALL$ or that $R_i =< r_{i1}, r_{i2} >$ is dom (l_i) , then we refer to q as a completely-aggregated view, denoted by CV.*

Definition 5.*Partly-aggregated view :* *In Definition 3, if there exists at least one i which satisfies $l_i \neq ALL$, then we* *refer to q as a partly-aggregated view, denoted by PV.*

Definition 6.*Public view window* : *Public view window, denoted by PVW, is referred to as system-allocated space with an upper bound of size. PVW includes CVD (Completely-aggregated View District) and PVD (Partly-aggregated View District). These two districts are used to store those materialized completely-aggregated views and partly-aggregated views selected by certain algorithm respectively.*

Definition 7.*User view window* : *User view window, denoted by UVW, is referred to as system-allocated space with an upper bound of size. It is used to store users' partly-aggregated views selected by certain algorithm.*

In order to help better understand the following content, we give a list for the acronyms of the terms used in this paper in Table 1.

Acronym	Term
GVW	Global View Window
PVW	Public View Window
UVW	User View Window
CV	Completely-aggregated View
PV	Partly-aggregated View
CVD	Completely-aggregated View District
PVD	Partly-aggregated View District
CVS	Completely-aggregated View Set
PVS	Partly-aggregated View Set

Table 1. The term acronym list

4.2 Description of SOMES

As Figure 1 shows, in our method, we design a PVW in the GVW. A PVW is used to store all materialized CVs and some qualified PVs. PVS contains all the candidate PVs in the system, and CVS all the CVs.

Users are grouped based on their query characteristics (in Figure 1, for simplification, we only give three different user groups). Here we assume that users have already been classified into different groups, and we will give in another paper the details on how to group users. Every user group is allocated with a UVW, which contains all the materialized views accessed by certain user group. When a CV is used by more than one groups, it will enter into CVD. Also, for a PV, it will enter into PVD if it is shared by more than one groups.

The reason for PVW being divided into PVD and CVD, is due to the difference between PV and CV. Usually, a CV can be used to answer more queries than a PV, since a PV is in fact a multi-dimensional range fragment. Sometimes, a query may get result only from several PVs combined. Therefore, under certain space constraint, it may bring more benefits if more space is allocated to CVD. However, this does not necessarily mean that all the space of PVW should always be for the used of CV. Sometimes, there are some frequently accessed PVs, and, if materialized, they will also bring great benefits.

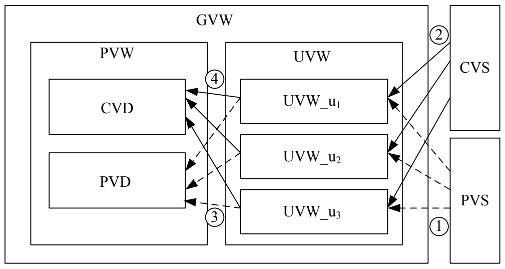


Figure 1. Different windows in SOMES

4.3 Space allocation

In our method, different types of user groups use their corresponding types of UVW$_-u$. This guarantees that query characteristics of different groups may not be "killed".

Table 2 shows the parameters for controlling space allocation among different types of windows. For these parameters, there exist $r_{UU1}+r_{UU2}+r_{UU3}=1$, $r_{UG}+r_{PG}=1$, and $r_{PP}+r_{CP}=1$. Additionally, the values of these parameters can be optimally defined only after a period of time of application. After these parameters are defined, we can further get other parameters such as S_{CVD_DOWN} (size lower bound of CVD), S_{PVD_UP} (size upper bound of PVD), S_{UVW} (unchangeable size of UVW), $S_{UVW_-u_1}$ (unchangeable size of UVW$_-u_1$), $S_{UVW_-u_2}$ (unchangeable size of UVW$_-u_2$) and $S_{UVW_-u_3}$ (unchangeable size of UVW$_-u_3$). Table 3 gives the computation methods of these parameters.

Parameter	Definition
r_{PG}	the size ratio of PVW to GVW
r_{UG}	the size ratio of UVW to GVW
r_{CP}	the size ratio of CVD to PVW
r_{PP}	the size ratio of PVD to PVW
r_{UU1}	the size ratio of UVW$_-u_1$ to UVW
r_{UU2}	the size ratio of UVW$_-u_2$ to UVW
r_{UU3}	the size ratio of UVW$_-u_3$ to UVW

Table 2. Parameter definition

Parameter	Computation method
S_{CVD_DOWN}	$S_{GVW}\times r_{PG}\times r_{CP}$
S_{PVD_UP}	$S_{GVW}\times r_{PG}\times r_{PP}$
S_{UVW}	$S_{GVW}\times r_{UG}$
$S_{UVW_-u_1}$	$S_{UVW}\times r_{UU1}$
$S_{UVW_-u_2}$	$S_{UVW}\times r_{UU2}$
$S_{UVW_-u_3}$	$S_{UVW}\times r_{UU3}$

Table 3. Parameter computation

The relationships of the sizes of different types of windows are as follows:

$S_{GVW}=S_{CVD}+S_{PVD}+S_{UVW}$;

$S_{UVW}=S_{UVW_-u_1}+S_{UVW_-u_2}+S_{UVW_-u_3}$;

Among all the windows, the space of CVD and PVD is changeable with S_{CVD_DOWN} as the size lower bound of CVD and S_{PVD_UP} as the size upper bound of PVD respectively. Determination on how to dynamically allocate space to PVD and CVD is based on the comparison

Algorithm 1: Dynamic space allocating

```
Input  : 1: $S_{PVD\_UP}$, the size upper bound of PVD
         2: $S_{CVD\_DOWN}$, the size lower bound of CVD
         3: $S_{PVW}$, the size of PVW
         4: $\alpha$, size changing rate
         5: $S_{PVD}$, the current size of PVD
         6: $S_{CVD}$, the current size of CVD
Output: 1:$S_{PVD}$, the changed size of PVD
         2:$S_{CVD}$, the changed size of CVD
1  begin
2      if AverageBenefit(CVD)≥AverageBenefit(PVD) then
3          $S_{CVD} \leftarrow S_{CVD} + \alpha \times S_{PVD\_UP}$;
4          $S_{PVD} \leftarrow S_{PVD} - \alpha \times S_{PVD\_UP}$;
5          if $S_{CVD} \geq S_{PVW}$ then
6              $S_{CVD} \leftarrow S_{PVW}$;
7              $S_{PVD} \leftarrow 0$;
8          end
9      else
10         $S_{CVD} \leftarrow S_{CVD} - \alpha \times S_{PVD\_UP}$;
11         $S_{PVD} \leftarrow S_{PVD} + \alpha \times S_{PVD\_UP}$;
12         if $S_{CVD} \leq S_{CVD\_DOWN}$ then
13             $S_{CVD} \leftarrow S_{CVD\_DOWN}$;
14             $S_{PVD} \leftarrow S_{PVD\_UP}$;
15         end
16     end
17     return $S_{PVD}$,$S_{PVD}$;
18 end
```

between the average benefits of PVD and CVD. When the average benefit of CVD is more than that of PVD, system will call back some space from PVD for the use of CVD, and vise versa. Algorithm 1 shows the dynamic space allocation process, in which, size changing rate α should be defined according to real application.

4.4 View selection within windows

Definition 8.*View selection rule M :* *View selection rule M is a rule used to select a view to materialize from candidate views.*

M is defined by system itself. Different systems have their own criteria, such as those based on size of view [4], benefit of unit space [7], or frequency of unit space [8], etc.

Definition 9.*View selection rule N :* *View selection rule N is a rule used to select a view from UVW to PVW, according to which, when two views compete, the one that is accessed by more users wins; if their numbers of users that access them are equal, the winner is the one that has larger total accessed times, namely, the sum of every user's times of accessing this view.*

Algorithm 2 shows the process of view selecting. It takes as input CVS, PVS and the available space of every window, namely S_{CVD}, S_{PVD} and S_{UVW_u}, and outputs the selected views for every window, namely V_{CVD}, V_{PVD} and V_{UVW_u}. In Algorithm 2, line 4 to 16 do the job of selecting views from PVS and CVS to UVW_u, which is denoted by process ① and ② in Figure 1. Line 17 to 27 are responsible for selecting views from UVW_u to PVD,

Algorithm 2: View selecting

```
Input  : $S_{CVD}$, $S_{PVD}$, $S_{UVW\_u}$, $CVS$, $PVS$
Output: $V_{CVD}$, $V_{PVD}$, $V_{UVW\_u}$
1  begin
2      $V_{CVD} \leftarrow \phi$;$V_{PVD} \leftarrow \phi$;
3      for each u do $V_{UVW\_u} \leftarrow \phi$;
4      for each u do
5          while $S_{UVW\_u} > 0$ do
6              VS=PVS $\cup$ CVS;
7              $v$ ←the view in VS satisfying rule M and accessed by $u$;
8              if $S_{UVW\_u} > |v|$ then
9                  $V_{UVW\_u} \leftarrow V_{UVW\_u} \cup \{v\}$;
10                 $VS \leftarrow VS - \{v\}$;
11                 $S_{UVW\_u} \leftarrow S_{UVW\_u} - |v|$;
12             else
13                 $S_{UVW\_u} \leftarrow 0$;
14             end
15         end
16     end
17     while $S_{PVD} > 0$ do
18         $v$ ←the PV in $UVW\_u$ satisfying both rule M and N;
19         if $S_{PVD} > |v|$ then
20             $V_{PVD} \leftarrow V_{PVD} \cup \{v\}$;
21             $V_{UVW\_u} \leftarrow V_{UVW\_u} - \{v\}$;
22             $S_{PVD} \leftarrow S_{PVD} - |v|$;
23             $S_{UVW\_u} \leftarrow S_{UVW\_u} + |v|$;
24         else
25             $S_{PVD} \leftarrow 0$;
26         end
27     end
28     while $S_{CVD} > 0$ do
29         $v$ ←the CV in $UVW\_u$ satisfying both rule M and N;
30         if $S_{CVD} > |v|$ then
31             $V_{CVD} \leftarrow V_{CVD} \cup \{v\}$;
32             $V_{UVW\_u} \leftarrow V_{UVW\_u} - \{v\}$;
33             $S_{CVD} \leftarrow S_{CVD} - |v|$;
34             $S_{UVW\_u} \leftarrow S_{UVW\_u} + |v|$;
35         else
36             $S_{CVD} \leftarrow 0$;
37         end
38     end
39     return $V_{CVD}$, $V_{PVD}$, $V_{UVW\_u}$;
40 end
```

which corresponds to process ③ in Figure 1. Line 28 to 38, corresponding to process ④ in Figure 1, achieve the task of selecting views from UVW_u to CVD.

Meanwhile, views in every window are to be adjusted according to the change of user queries. Since view adjusting process in every window is just the same as that in other methods such as FPUS [7], so we here do not give relating algorithms.

4.5 Advantages of SOMES

Compared with GVW, UVW$_u$ contains fewer views and is with smaller size. This results in faster speed in the view adjusting process of certain view window, since fewer views are involved. Furthermore, more improvements can be achieved if we adopt parallel technology to select and adjust views concurrently for all these windows.

In addition, the phenomena of views' going into and out view windows unreasonably are greatly reduced. When every user group is allocated with its own UVW$_u$, the use of views will have obvious feature, which is hard to observe if all user groups share the same view window, and this will reduce the unreasonable phenomena described above to a low level.

5 Empirical Study

In this section, we report the performance evaluation of our method. The algorithms are implemented with C++. All the experiments are conducted on 4*2.4GHz CPU (double core), 32G memory HP Proliant DL585 Server running Windows Server 2003 and Oracle 10g.

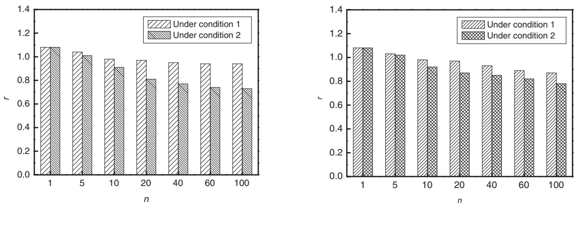
We design a multidimensional data as shown in Table 4. Here we assume that the data in ***DB*** is evenly distributed. In the multidimensional data, there are four dimensions, namely, D_0, D_1, D_2 and D_3, and they have 4, 3, 4 and 3 levels respectively. The basic fact table of the multidimensional data ***DB*** contains 500KB records, according to which we can estimate that the size of the cube ***DB*** is 15MB.

level	**dimension**			
	D_0	D_1	D_2	D_3
3	1	-	1	-
2	25	1	5	1
1	50	25	25	10
0	100	50	50	50

Table 4. Dimension member amount of *DB*

Here we will compare our method with other ones such as BPUS [5] and FPUS [7]. The values of the parameters in our method, if not given, is defined as follows: $r_{PG} = 0.4$, $r_{UG} = 0.6$, $r_{CP} = 0.6$, $r_{PP} = 0.4$, $r_{UU1} = 0.3$, $r_{UU2} = 0.3$ and $r_{UU3} = 0.4$.

EXPERIMENT 1 : Here we make the size of storage space 20 percent of that of the cube ***DB***. During the experiment process, there are 10 (simulated) users in each group continuously starting their own queries. In order to show the influence of query characteristics, we conduct the experiments under two much different conditions. Under the first condition, we make all the three user groups start queries with similar characteristics, as for the second condition, the characteristics of queries from different user groups are much different from each other. Let $r = t_1/t_2$, where t_1 is the total execution time of SOMES for the first n queries, and t_2 is that of BPUS or FUPS. Figure 2 shows how r changes with the variation of n. From Figure 2 we can get that, when under the second condition, SOMES can achieve more performance improvements over both BPUS and FPUS than when under the first condition. It means that SOMES can take fully advantage of query characteristics of different user groups, whereas this is too hard for BPUS and FPUS since they do not differentiate between queries from various users. Also it should be observed in Figure 2 that, when $n < 10$, r is larger than 1.0. Take SOMES and BPUS for example, when under the second condition, for $n = 1$, r=1.08, and for $n = 1$, r=1.01. It means that SOMES costs more than BPUS when $n < 10$. The reason is that, at the initial stages of SOMES, the benefits achieved are cut off by the costs involved in the space allocation process.



(a) SOMES / BPUS (b) SOMES / FPUS

Figure 2. The change of r when varying n

EXPERIMENT 2 : Figure 3 shows the change of query cost (see [7] for the cost computing method) for BPUS, FPUS and SOMES, when varying the ratio of the space of views to that of ***DB*** from 0 to 20%. It shows that available storage space (view space) plays an important role for all the three methods. Also, compared with BPUS and FPUS, SOMES can benefit more from the increase of storage space. For example, when storage space is 10KB, the query costs for BPUS, FPUS and SOMES are 300KB, 280KB and 322KB respectively; when storage space is 100KB, the query costs for them are 196KB, 140KB and 112KB respectively. Also we conduct experiment to see the effect on the performance when varying the parameters of our method, and the result is shown in Table 5.

EXPERIMENT 3 : We also design two simulated scenarios to show how SOMES greatly reduces the number of view exchange times. In the first scenario, we make three user groups start many queries freely. Figure 4 (a) shows that SOMES has less view exchange times than both BPUS and FPUS. In the second scenario, during the 3rd and 8th minutes, we make these three user groups initiate queries alternately. As the experimental result in Figure 4 (b) shows, both BPUS and FPUS will "flip" during the 3rd and 8th minutes, whereas SOMES exhibits a relatively

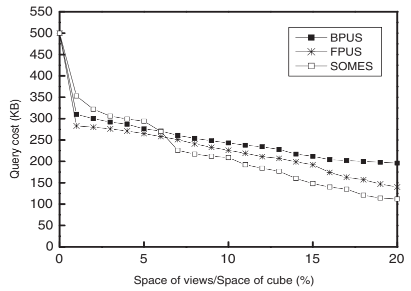


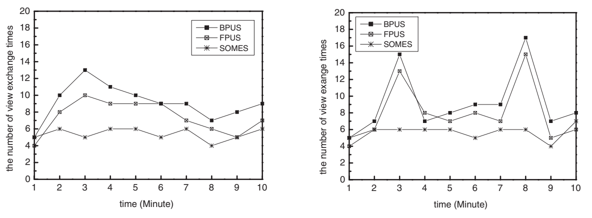
Figure 3. The change of query cost

Query Cost of SOMES / Query Cost of BPUS				
		r_{UU1}=0.20 r_{UU2}=0.40	r_{UU1}=0.40 r_{UU2}=0.30	r_{UU1}=0.60 r_{UU2}=0.20
r_{PG}=0.25	r_{CP}=0.25	56.41%	55.02%	57.75%
	r_{CP}=0.50	54.13%	53.67%	55.53%
	r_{CP}=0.75	57.43%	54.61%	58.45%
r_{PG}=0.50	r_{CP}=0.25	55.37%	53.09%	56.21%
	r_{CP}=0.50	53.18%	52.22%	54.23%
	r_{CP}=0.75	54.63%	53.65%	55.90%
r_{PG}=0.75	r_{CP}=0.25	56.18%	55.33%	57.06%
	r_{CP}=0.50	53.99%	53.05%	55.24%
	r_{CP}=0.75	57.58%	54.99%	58.13%

Query Cost of SOMES / Query Cost of FPUS				
		r_{UU1}=0.20 r_{UU2}=0.40	r_{UU1}=0.40 r_{UU2}=0.30	r_{UU1}=0.60 r_{UU2}=0.20
r_{PG}=0.25	r_{CP}=0.25	56.27%	54.22%	56.98%
	r_{CP}=0.50	53.87%	53.09%	55.17%
	r_{CP}=0.75	57.10%	54.14%	57.78%
r_{PG}=0.50	r_{CP}=0.25	54.89%	52.35%	55.83%
	r_{CP}=0.50	52.64%	51.77%	54.02%
	r_{CP}=0.75	54.08%	53.26%	55.06%
r_{PG}=0.75	r_{CP}=0.25	55.45%	55.01%	56.46%
	r_{CP}=0.50	53.17%	52.79%	54.67%
	r_{CP}=0.75	56.42%	54.54%	57.49%

Table 5. Effect on the performance of SOMES when varying the parameters

smooth curve during the whole process. It means that a lot of view exchanges occur during the 3rd and 8th minutes for both BPUS and FPUS, while the number of view exchange times for SOMES is more stable.



(a) Scenario 1 (b) Scenario 2

Figure 4. The change of the number of view exchange times in different scenarios

6 Discussion and Conclusion

In this paper, we have revisited materialized view selection problem, and propose a new user-oriented method called SOMES. It makes full use of query characteristics of different types of users so that more efficient view selection can be achieved. Experimental results show that our method can achieve desirable performance improvements over other available methods such as BPUS and FPUS.

In future, we will apply our method in the fields such as mobile communication to meet the real-time query requirements. Also, we will do more research work on how to better dynamically adjusting the view sets according to the change of query characteristic.

References

[1] D. Theodoratos and M. Bouzeghoub. A General Framework for the View Selection Problem for Data Warehouse Design and Evolution. In: *Proc. of the 3rd Intl. Workshop on Data Warehousing and OLAP*, 2000, pages:1-9.

[2] H. Gupta, V. Harinarayan, A. Rajaraman. Index Selection for OLAP. In: *ICDE'97, Proceedings of the 13rd International Conference on Data Engineering*. Birmingham, U.K. IEEE Computer Society Press, 1997, pages: 208-219.

[3] S. R. Valluri, S. Vadapalli, K. Karlapalem. View relevance driven materialized view selection in data warehousing environment. *Australian Computer Science Communications*, Vol 24(2), Jan, 2002, pages:187-196.

[4] A. Shukla, P. Deshpande, J.F. Naughton. Materialized view selection for multidimensional datasets. In: *VLDB'98, Proceedings of the 24th International Conference on Very Large Data Bases*. New York: Morgan Kaufmann Publishers, Inc., 1998. 488-499.

[5] V. Harinarayan, A. Rajaraman, J.D. Ullman. Implementing data cubes efficiently. In: *SIGMOD'96, Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*. Montreal: ACM Press 1996, pages:205-216.

[6] H. Gupta and I. S. Mumick. Selection of Views to Materialize in a Data Warehouse. *IEEE Transactions on Knowledge and Data Engineering*. Vol 17 (1), Jan. 2005. Pages 24-43.

[7] H.X. Tan, L.X. Zhou. Dynamic selection of materialized views of multi-dimensional data. *Journal of Software*, China, 2002, Vol.13(06), pages:1090-1096.

[8] U. Hidetoshi, R. Kanda, J.T. Toby. A progressive view materialization algorithm. In: *Proceedings of the second ACM international workshop on Data warehousing and OLAP*, 1999, pages:36-41.

[9] D. Theodoratos, W. G. Xu. Constructing search spaces for materialized view selection. In: *Proceedings of the 7th ACM international workshop on Data warehousing and OLAP*. Nov, 2004, pages: 112-121.

[10] Y.S. Xue, Z.Y. Lin, J.J. Duan, X.H. Lv and W. Zhang. Dynamic Selection of Materialized Views of Multi-Dimensional Data with Multi-Users and Multi-Windows Method. *Journal of Computer Research and Development*, China, Vol.41, No.10, 2004, pp:1703-1711.