Interest Centers to Improve the Maintenance Load of Materialized Views in Real-time Data Warehouse

Abdelaziz Abdellatif
Faculté des sciences de Tunis
Tunis El Manar University
Tunisia

Ali Ben Ammar
University College of Taima- Tabuk University
Arabie Saoudite.
Email: aamar [AT] ut.edu.sa

Amira Chouk
Faculté des sciences de Tunis
Tunis El Manar University
Tunisia

Abstract—**In this paper, we have developed an approach to specify when updating materialized views in real-time data warehouse. The aim is to improve the maintenance load which may have a bad impact on the query response time especially in a real-time environment where the data updates are frequent and the freshness of served data is highly required. We have introduced a new concept "Interest center" which is a frequent set of user queries i.e. common and frequent user demands. We identify the interest centers from the historic of user queries. Then, periodically, we analyze the recent asked queries to identify the more probably interest centers of the current users. This analysis allows us to estimate the next future user queries which will be part of the identified interest centers. Consequently, we affect the online update to the materialized views that are sources for the estimated queries. For the rest of materialized views we apply the on-demand update. Our experiments showed that the proposed hybrid approach guarantees strong consistency of data and allows reducing latency of updating materialized views. In addition, they prove that our solution decreases the maintenance load significantly better that the on-demand and the online policies.**

Keywords- Materialized View Maintenance; Interest Center; Online Update; On-demand Update; Real-time Data Warehouse.

I. INTRODUCTION

Materialized views in data warehouse and databases are a good way to increase query response time. This technique aims to reduce the cost ofgenerating data from databases to serve repetitive user queries. Its principle is to store and thenupdate the results,called *materialized views*, of some repetitive queries. But this technique generates an additional data maintenance load. To decrease this load, some approaches of materialized view selection have mainly integrated the maintenance cost of materialized views as a constraint for the selection [10,13,14,15]. But, this is insufficient to take advantage of view materialization. The maintenance approaches [1,3,4,5,6,7,8,9,11,12,16,18,19,22,23,25,26,27,28,29] propose techniques to optimize the update cost of materialized views. Mainly, these approaches address two topics: (i) how to update view? (ii) When to update view? The first question is about the refreshment or the recompilation of view. The second one is about the online, on-demand, or periodically maintenance. In the online policy, the updates to the source databases are directly reflected on the materialized views. This policy guarantee a strong consistency, i.e., the view will eventually reflect the final state of its sources. In the on-demand policy, the maintenance is performed when the materialized view is queried. This policy results in latency of going to the DBMS. In the periodic policy, the maintenance of the materialized views is performed on a regular basis (once a minute, once an hour, etc.). In this policy, a view may become inconsistent with its base data, representing a snapshot of the sources. Some approaches have mixed up different update policies in order to carry out hybrid maintenance [8,23,29]. The hybrid approaches apply a detailed policy to improve the maintenance cost but they are more complicated to be applied.

In this work we propose hybrid approach for the maintenance of materialized views in real-time data warehouse environment. In such environment, the data source updates are frequent and the data served to the user should be fresh. The aim of our approach is to found a compromise between quality of service (QoS) and quality of data (QoD) in real-time data warehouse environment. The quality of service (QoS) [15] represents the access rate to materialized data. It may be illustrated by the query response time. The quality of data (QoD)[15] represents the access rate to fresh data. QoD is decreased when there are later updates of materialized views. However, the QoS is decreased in two cases:

- When there is a lot of online and heavy updates of materialized views;
- When the materialized views are unused, either because they are not fresh or because theyare badly selected (not appropriate for the queries).

Our idea is to analyze the access historic in order to identify the main interests of users. We call these interests *"Interest centers"*(denoted IC) and they correspond to the frequent set of user queries i.e. the common and frequent user demands. Then and periodically, we try to estimate the interest centers (IC) of the near future. Our estimation is based on the fact that the recent user queries give us an idea about the interest centers to which the current user demands converge. Precisely, in the near future, the user accesses will converge to the interest centers (IC) having big part of the recent queries of the current

period. So, the queries of the estimated interest centers are considered the more probably next queries.

In our approach, the maintenance policies of materialized views are affected based on the estimated queries of the near future. The materialized views that constitute sources for these queries will be updated online in order to improve the QoS. The rest of materialized views will be updated on-demand. The early update of the materialized views of the estimated next queries reduces the delay of execution of these queries and so it improves the QoS. In the two cases, we guarantee a high level of QoD since the user queries will be served with fresh data.

Our contribution is the introduction and the use of interest center, as a new concept, to specify the appropriate maintenance policy for each materialized view. The use of interest centers allow us to early update the materialized views which are more probably to be accessed in the near future i.e. make them fresh. The aim is to continually improving the QoS of the near future queries by providing materialized views with fresh data.

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.

II. RELATED WORKS

The problem of maintenance has been widely addressed for materialized views in data warehouses environment[1,4,7,8,9,11,12,16,18,19, 25,26,27,28]. In the traditional environment, the maintenance approaches addressed the issue of finding a policy that limits the access load to the sources and minimize memory usage. Hence, they focused on how implementing an incremental maintenance or how to integrate intermediary structures, like global query plan or new materialized views, to accelerate materialized view update. Self-maintenance was the important technique used in such environment to incrementally update views [16,18,19,27]. However in real-time environment, the maintenance problem is more complex since the updates are frequent. Also, on real-time environment, there is a constraint on the data freshness. So, the maintenance approaches are mainly oriented to the query scheduling in order to improve the QoS and the QoD [4,11,22]. On the web, which is a dynamic environment, the view maintenance approaches are principally addressed the issue of improving the QoS and QoD. The main technique used in these approaches is the query scheduling [22].

In the literature the closest work to our approachis presented in [29]. The authors of this work proposed to specify the maintenance policy of each web view based on its state and on its access and update rates. They introduced the concept of state to simulate the web view behaviors. According to this approach, there are two main states: Active, Sleeping. The active statemeans that the view is always up-to-date and can be used directly. To keep the viewalways active, any update to the base data will lead to an immediate refreshingof this view which is in our approach the online update. Thesleeping state means that the view does not response to any update at all and willonly be evaluated on-demand. So, a state describes a specific status of a web view, while a state transfer graph is an abstraction model to describe how a web view moves among these states. Others intermediate states are added to the graph. The transition of view from a state to other one is arbitrated by the rates of view access and update.

The user session historic is widely used in materialized view selection but it rarely used to improve the maintenance load. In [23], the authors construct a navigation graph of web site in order to analyze the historic navigation between web views. Then they deduce the navigation rate between web views in order to specify those that should be early updated. We are not aware of any work that identify and cluster the interest of users and then exploit them to specify the maintenance policies of materialized views.

III. APPROACH PRESENTATION

The aim of our approach is to specify the appropriate maintenance policy for each materialized views in order to improve the query response time in real-time data warehouse. Our approach is based on the concept of interest center which is a set of common and frequent user queries i.e. the set of queries that are frequently asked by users in the same session. The procedure that we propose to specify view maintenance policies contain three main steps: (1) discover the main interest centers of users; (2) estimate the near future interest centers; and (3) affect policies to materialized views. So, the first two steps represent the analysis phase. However the third one represents the exploitation phase.

A. Discover the main interest centers

We start this task by analyzing the historic of user queries. Precisely, welook for the states of the queries (asked or not) duringsessions of each user. We have represented the result of this analysis in a matrix H as follow:

$$H(i,j) = \begin{cases} 1 \; if the query r_i was asked along \; a \; user \; session s_j \\ 0 \; if not \end{cases}$$

Example:

In this example we consider that we have 4 users and 10 queries.

In the table 1 we have presented an example of access historic of these 10queries. As an example, the query r_1 was asked by the user u_1 during his sessions s_1 and s_3 ♦

We use association rules as a data mining technique to identify the interest centers.

Association rules aims to discover correlations between items in alarge database. They are introduced by [2]. We will use the followingconcepts to formulate the definition of association rules:

- $I = \{i_1; i_2; \ldots; i_m\}$, a set called items. In the context of database these items may correspond to the table attributes;
- $D = \{T_1, \ldots, T_n\}$, a set of transactions, where each transaction T_j is a set of items (called itemset) such that$T_j \subseteq I$; In the context of database these items may correspond to the table rows.

An association rule is an implication of the form:

$R(A\%, B\%): X \rightarrow Y, where X \subseteq I\ , Y \subseteq I\ , And X \cap Y\ = \emptyset$

R is interpreted as follow:

- A% of the D' transactions contains both X and Y. So, A is called the support of the association rule R.
- B% of the D' transactions which contains X, contains also Y. So, B is called the confidence of the association rule R.

TABLE I

Users	Sessions	Queries									
		r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}
u_1	s_1	1	0	1	1	0	1	1	0	0	1
	s_2	0	0	0	1	0	1	1	0	0	0
	s_3	1	0	0	1	0	0	1	0	0	0
	s_4	0	1	0	0	1	0	0	0	1	0
u_2	s_1	1	0	0	0	0	1	1	0	0	1
	s_2	1	0	0	1	0	1	1	0	0	0
	s_3	1	1	0	0	1	0	0	0	1	0
	s_4	0	1	1	1	0	1	1	1	1	1
	s_5	0	1	0	1	1	0	0	0	1	1
	s_6	1	0	1	1	0	0	1	0	0	0
u_3	s_1	0	1	1	1	1	1	1	1	1	1
	s_2	0	1	1	1	1	0	0	1	0	1
	s_3	1	0	1	1	0	1	1	0	0	0
u_4	s_1	1	1	0	1	1	0	0	1	1	0
	s_2	0	1	1	1	0	1	1	1	0	1
	s_3	1	1	1	1	1	1	0	1	1	1
	s_4	1	0	1	0	0	1	1	0	0	0
	s_5	1	1	1	1	1	0	1	1	1	0

Given a set of transactions D, the problem of mining associationrules is to extract all association rules having a support and aconfidence greater than the user-specified minimum support(called $minSupp$) and minimum confidence (called $minConf$)respectively.

The problem of mining association rules can be divided into two sub-problems:

1. Find all frequent itemsets in the transaction database with respect to given support threshold. An itemset is called a frequent itemset if its support is no less than $minSupp$.
2. For each frequent itemset $X \cup Y$ founded, generate all association rules $X \rightarrow Y$, if its confidence is no less than $minConf$.

In our approach, the interest centers correspond to frequent query sets. So, we will address only the first sub-problem of the mining association rules. In other words, we search sets of queries which represent common and frequent interests of some users.

Several data mining algorithms are addressed the search of frequent data sets [2,17,20,21,24]. But according to [21], the requirement of mining the complete set of association rules leads to two problems: (1) there may exist a large number of frequent itemsets in a transaction database, especially when the support threshold is low, and (2) there may exist a huge number of association rules. It is hard for users to comprehend and manipulate a huge number of rules.

An interesting alternative, proposed in [20], to this problem is the mining of frequent closed itemsets and their corresponding association rules.

According to [21], an itemset X is a closet itemset if there exists no itemset X' such that (1) X' is a proper superset of$X$, and (2) every transaction containing X also contains X'. A closet itemset X is frequent if its support passes the given threshold $minSupp$.

Consequently, in this step we will mining the closet frequent sets of queries which represent common interest between users. So, each founded set will represent aninterest center.

Example:

In this example we suppose that the minSupp is 30% i.e. a closet set of queries is considered frequent only if its queries are asked in at least the 30% of the user sessions. Based on the table 1, the interest centers and their supports are presented in table 2.

TABLE II

Interest centers (IC)	Queries	Frequencies	Supports
IC_1	r_1,r_6	6	6/18=33.5%
IC_2	r_1,r_4,r_7	6	6/18=33.5%
IC_3	r_2,r_4	8	8/18=44.5%
IC_4	r_2, r_5, r_9	7	39%
IC_5	r_2, r_8	7	39%
IC_6	r_2, r_{10}	6	33.5%
IC_7	r_3, r_4, r_8	6	33.5%
IC_8	r_3, r_6, r_7	6	33.5%
IC_9	r_4, r_6, r_7	7	39%

For example, the frequent set {r_4,r_7} does not considered as an interest center although its frequency is 11 i.e. its support is 61%. This is because it's included in the frequent closed set {r_1,r_4,r_7}♦

B. Estimate the near future interest centers

The aim of this task is to estimate the user queries of the near future based on the recent access queries. Our idea is that the recent queries may conduct as to know the centers of interest of the current users. Then, we deduce the more probably set of future queries which obviously belong to the identified interest centers.

The main question in this step is how to specify the appropriate interest centers when the recent queries belong to more than one interest center? To resolve this problem we calculate a weight for each interest center containing part of the

recent queries. We call this weight *Interest Weight*, denoted $W(IC_j)$, and it corresponds to the probability that the current users are interested in the center IC_j. In other words, the current users are currentlyaskingqueries of IC_j and they will continue in the near future asking queries of the same center. We define the interest weight as follow:

$$W(IC_j) = \frac{|IC_j \cap Q|}{|IC_j|}$$

Where:

- Q: the set of recent queries;
- $|IC_j|$: cardinality of IC_j i.e. the number of its queries
- $|IC_j \cap Q|$: the number of recent queries belonging to IC_j

After calculating all the interest weights, the result of this step will be the interest centers with weights greater than a threshold θ i.e. $W(IC_j) \geq \theta$. This result represents the more probably interest centers i.e. theestimateduser queries of the current and near future time.

Example:

In this example we suppose that the threshold θ = 40% i.e. an interest center IC_j iskept only if $W(IC_j) \geq 40\%$. In other words, at least 40% of the IC_j queries should be recently asked to decide that IC_j represents a current interest center and its queries will be asked in the near future.

We suppose that the set queries which are recently asked is $Q = \{r_1, r_4\}$. So, based on table 2, we illustrate in figure 1, the interest centers and their weights.

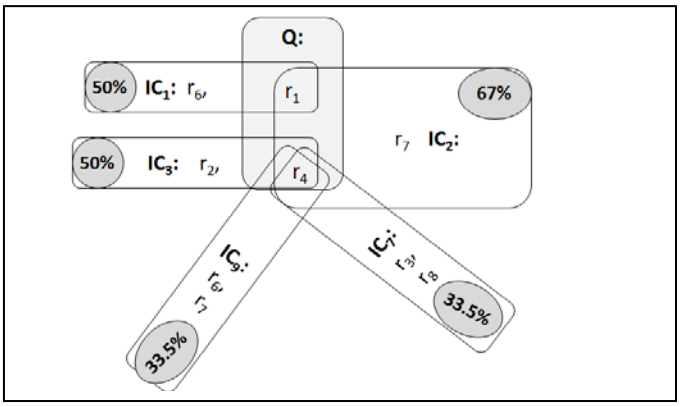

Figure 1. Example of interest centers and their weights

In figure 1:

- $W(IC_1) = 50\%$ because the half of IC_1 queries are included in Q, the set of recent queries.
- $W(IC_2) = 67\%$ because two queries of IC_2 from three (the total number of queries of IC_2), which are r_1 *and* r_4, belong to the set of recent queries Q i.e $W(IC_2) = 2/3$.

So since θ = 40%, the more probably interest centers will be IC_1, IC_2 *and* IC_3. Consequently, the queries $\{r_1, r_2, r_4, r_6, r_7\}$ of the identified interest centers $IC_1, IC_2 and IC_3$will be,more probably,asked in the near future♦

C. Affect update policies to materialized views

We apply the following simple rule to affect update policies to materialized views: we affect the online update to the materialized views that are sources for the user queries estimated in the previous step;for the rest of materialized views we apply the on-demand update.

Example:

Based on the results of the previous example, we affect the online policy to update the materialized views which are accessed by $\{r_1, r_2, r_4, r_6, r_7\}$ and applying the on-demand policy for the rest of materialized views.

Suppose that the queries of our examples are executed on 7 views as it is illustrated by the figure 2. Also, suppose that the materialized views are $\{v_2, v_4, v_5, v_6\}$(surrounded by rectangles in the figure). So, according to the results of the previous example, the online update will be applied for the materialized views $\{v_2, v_4, v_5\}$ and the on-demand update for the materialized view v_6♦

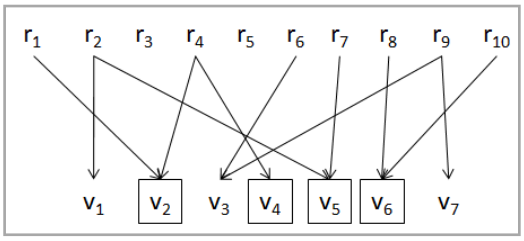

Figure 2. Example of source views

IV. 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 simulation we have construct the access historic of the 22 queries during 3 days i.e. about 72 hours. This historic is made to carry out the analysis phase of our approach i.e. the identification of the main interest centers. Also, in this simulation we have supposed 120 user sessions each with length of 10 minutes. At the end of the analysis phase we have executed the CLOSET algorithm [21] for mining the main interest centers.

During the exploitation phase, we have continued the simulation of access queries and view maintenance for 1 day. We have supposed that the results of queries represent the views to be materialized. The aim is to evaluate the ability of our approach to improve the maintenance cost of materialized views. In the two phases, the speedin which accesses can be processed, the incoming update stream,and the speed in which updates can be performed are inputs to thesimulator. We have supposed 5 access requests and 1 updaterequest per user session i.e. per 10 minutes. So, we have in total 720 accesses per day and about 150 updates.

Also, in the second phase we have executed,periodically, a greedy algorithm for the selection of materialized views. We have fixed the length of the selection period to 30 minutes i.e. the time interval between two successive executions of the selection algorithm. This algorithm applies a constraint of maintenance cost i.e. the maintenance cost of the materialized views should not exceed a specified value. This algorithm is iterative and 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. The materialization profit of a view v_i is the gain of query response time that should be produced if we materialize v_i. So, the problem of selection of materialized view is formulated as follow:

$$max \sum_{i=1}^{n} x_i * profit(v_i)$$

$$underconstraint \sum_{i=1}^{n} x_i * update\_cost\,(v_i) \leq CM$$

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 our tests and during the exploitation period, we have periodically identified the more probably interest centers and then decide which views should be updated online and which views should be updated on-demand. The maintenance policy of views is applied during a 30 minute time interval. In other words, the maintenance policy is affected at the beginning of each new 30 minute time interval.Our procedure of affecting policy maintenance of materialized views is executed as it described above in section 3.

To demonstrate the advantage of our approach, we have implemented the three maintenance policies: online, on-demand, and our hybrid approach. Then, we have compared the QoS, the maintenance cost of these three policies. We have illustrated the QoS by the query execution time. The figure 3 summarizes the results of these tests in milliseconds. The total cost is the sum of query execution cost and view maintenance cost.

According to the figure 3, our approach has improved the QoS of the on-demand approach by more than 5%, the maintenance cost of the online approach by more than 25%, the total cost of the online approach by more than 19% and the total cost of the on-demand approach by more than 11%. In real-time environment, this improvement of load and of QoS has a big impact on the user satisfaction.

In our tests, we have kept the QoD at its higher level i.e. 100%. In other words all the queries should be served with fresh data. For this reason we don't study the impact of our approach on the QoD.

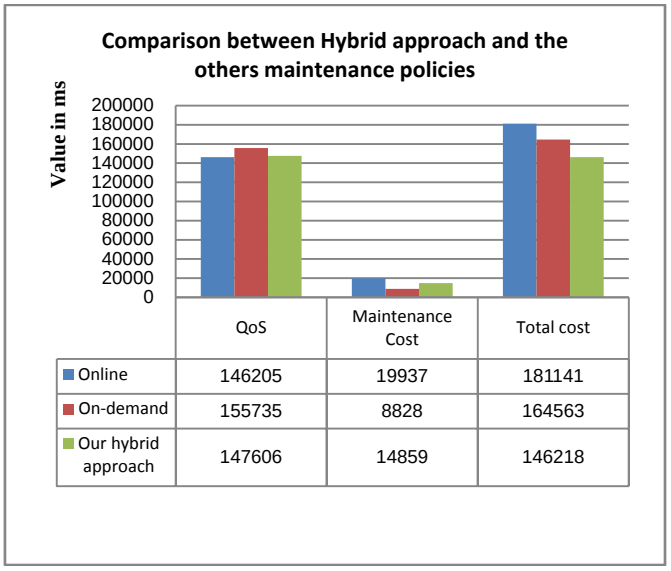

Figure 3. Comparison between 3 update policies

V. CONCLUSION

In this paper, we have proposed a hybrid approach for the maintenance of materialized views in real-time data warehouse. This approach affect to each materialized view one of the following two update policies: online updatewhen the view is estimated to be asked in the near future; and the on-demand update otherwise. We have introduced the concept of interest centers, which means the set of frequent user queries, to estimate the near future user requests.

Our solution guarantees strong consistency of data and allows reducing latency of updating materialized views. Compared to traditional online and on-demand updates, ourexperiments showed that this solution decreases the server load by more than 10%. In the future works we will attempt tointegrate this solution in a complete approach for the online selection and maintenance of materializedviews.

REFERENCES

[1] D. Agrawal, A. E. Abbadi, A. Singh, and T. Yurek."Efficient view maintenance at data warehouses". In Proceedings of SIGMOD, pages 417–427, May 1997.

[2] R.Agrawal,R.Skirant "Fast algorithms for mining association rules". In Proceedings of the 20th Intl Conference on Very Large Databases, pages 478-499, 1994.

[3] J. A. Blakeley, P. Larson, and F. W. Tompa. "Efficiently updating materialized views". Proceedings of the ACMSIGMOD International Conference on Management of Data, pages 61{71, June 1986.

[4] S.S.Boopathy, K.M.Subramanian."Improved Scheduling and Minimized Updates inData Warehouses". International Journal of Advanced Research in Data Mining and Cloud ComputingVol.1, Issue 2, August 2013.

[5] R. Chen and W. Meng."Precise detection and proper handling of view maintenance anomalies in a multidatabase environment". In Proceedings of the Second IFCIS International Conference on Cooperative Information Systems, pages 110–119, 1997.

[6] H.Engström, S.Chakravarthy, B. Lings."A Heuristic for Refresh Policy Selection in Heterogeneous Environments", ICDE 2003, pp. 674-676.

[7] H.Engström, S.Chakravarthy, B. Lings, "A Holistic Approach to the Evaluation of Data Warehouse Maintenance Policies". Retrieved July

24, 2006, From:http://citeseer.ist.psu.edu/cache/papers/cs/18347/http:zSzzSzwww.ida.his.sezSzidazSzresearchzSztech_rep ortszSzreportszSztr00zSzHS-IDA-TR-00-001.pdf/a-holisticapproach-to.pdf

[8] H.Engström."Selection of Maintenance Policies for a Data Warehousing Environment". Retrieved July 20, 2006, From:http://www.his.se/upload/15625/thesis.pdf#search=%22Selection%20of%20Maintenance%20Policies%20for%20a%20Data%20Warehousing%20Environment%22

[9] A. Gupta and I. S. Mumick. "Maintenance of materialized views: problems,techniques,and applications". IEEE Data Eng.Bull, 18(2):3–19, January 1995.

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

[11] R. Ismail, Omar H. Karam, Mohamed A. El-Sharkawy, Mohamed Said Abdel Wahaab. "Streaming Real Time Data In Data Warehouses For Just-In-Time Views Maintenance". IADIS International Conference Applied Computing 2009

[12] H. Jain, A. Gosain, "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

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

[14] P. P. Karde and V. M. Thakare, "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

[15] A. Labrinidis, Q. Luo, J. Xu, W. Xue. "Caching and Materialization in Web Databases", Foundationsand Trends in Databases Vol. 3, No. 2, pages 169-266. December 2009.

[16] H. Liu, Y. Tang, Q. Chen "The Online Cooperating View Maintenance Based on Source View Increment" Proceedings of the 11th International Conference on Computer Supported Cooperative Work in Design. 2007.

[17] H.Mannila, H.Toivonen, and A.I Verkamo. "Efficient algorithms for discovering association rules". In Proc. AAAI'94 Workshop Knowledge Discovery in Databases (KDD'94), pages 181-192, seattle, WA, July 1994.

[18] A. Mehwish, N. Shabnam, B. Pakeeza. "Efficiency Analysis of Materialized views in Data Warehouse Using Self-maintenance". International Journal of Computer Science and Information Security, Vol. 12, No. 6, 2014

[19] T.Nalini, A.Kumaravel, K.Rangarajan. "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

[20] N.Pasquier, Y.Bastide, R.Taouil, and L. Lakhal. "Discovering frequent closet itemsets for association rules". In Proc. 7th Int. Conf. Database Theory (ICDT'99), pages 398-416, Jerusalem, Israel, January 1999.

[21] J.Pei, J.Han, R.Mao, S.Nishio, S. Tang, D.Yang."CLOSET : An efficientalgorithm for mining frequent closed itemsets". Proceedings of the ACM SIGMODDMKD'00, Dallas,TX, 2002, p. 21-30.

[22] H.Qu, A. Labrinidis."Scheduling Update and Query Transactions under Quality Contracts in Web-Databases". Proceedings of the 5th Hellenic Data Management Symposium (HDMS'06), September 7-8, 2006, Thessaloniki, Greece.

[23] H. Saadi, A. Ben Ammar and A. Abdellatif. "Hybrid Approach for the Maintenance of Materialized Webviews". AMCIS 2010 Proceedings. 2010.

[24] Y.J.Tsay, J.Y.Chiang. "CBAR: an efficient method for mining association rules".Knowledge-Based Systems 18 (2005) 99–105, 2005.

[25] A. S. Varde and E. A. Rundensteiner."MEDWRAP:consistent view maintenance over distributed multi-relation sources". In Proceedings of the 13th International Conference on Database and Expert Systems Applications, pages 341–350, September 2002.

[26] J. L. Wiener, H. Gupta, W. J. Labio, Y. Zhuge, H. Garcia-Molina, and J. Widom."A system prototype for view maintenance". Proceedings of the ACM Workshop on Materialized Views , June 7, 1996, pp. 26-33.

[27] Xiaogang Zhang, Luming Yang, De Wang."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

[28] Y. Zhuge, H. Garcia-Molina, J. Hammer, and J. Widom. "View maintenance in a warehousing environment". In Proceedings of SIGMOD, pages 316–327, May 1995.

[29] Y. Zhang, X. Qin."State Transfer Graph: An Efficient Tool for Webview Maintenance". In the proceedings of WAIM2005 In Hangzhou, China, pages 513-525.