Multi-source materialized views maintenance[1]

Josep Silva, Jorge Belenguer, Matilde Celma

Computer Science Department
Technical University of Valencia
Camino de Vera s/n
E-46021 Valencia – Spain
{jsilva, jorbefa, mcelma}@dsic.upv.es

Abstract. In many information systems, the databases that make up the system are distributed in different modules or branch offices according to the requirements of the business enterprise. In these systems, it is often necessary to combine the information of all the organisation's databases in order to perform analysis and make decisions about the global operation. This is the case of Data Warehouse Systems. From a conceptual point of view, a Data Warehouse can be considered as a set of materialized views which are defined in terms of the tables stored in one or more databases. These materialized views store historical data that must be maintained in either real time or periodically by means of batch processes. During the maintenance process the systems must perform selections, projections, joins, etc. that can affect several databases. This is a complex problem since making a join among several tables requires (at least temporarily) having the information from these tables in the same place. This requires the Data Warehouse to store auxiliary materialized views that in many cases contain duplicated information. In this article, we study this problem, and we propose a method that minimizes the duplicated information in the auxiliary materialized views and also reduces the response time of the system.

1 Introduction

The use of Data Warehouse systems is becoming one of the critical factors that determine the success of many companies and organisations. The information gathered in the Warehouse can be used to make decisions about the processes of the organization, and should therefore be consistent. The information should also be as up-to-date as possible. Having the information of the operational systems up-to-date makes the results of the queries carried out on the Warehouse Database to be closer to the reality of the organization.

[1] This work has been partially supported by the EU (FEDER) and the Spanish MEC under grant TIN2005-09207-C03-02, by the ICT for EU-India Cross-Cultural Dissemination Project ALA/95/23/2003/077-054, and by UPV under grant TAMAT.

Many previous approaches [Ding (1999). Kuchenhoff (1991), Moro (2001), Quass (1996), Widom (1995), Silva (2002), Zhuge (1996) and Zhuge (1995)] consider that the information of a Data Warehouse System consists of a set of materialized views that store information from one or more databases that the organization uses in its operational systems. The process of loading this information is usually done using daily batch processes at night in order to avoid the slowdown of the operation systems. However, in some cases, the organization needs to compare the historical information in the Warehouse with the most recent information available in the operational system and therefore the Warehouse must be maintained in real time.

Maintenance of materialized views in general, and of Data Warehouses in particular, is a very relevant problem that has been studied in many works. For instance, Zhuge (1996) and Zhuge (1995) deal with the problem of updating materialized views in real time, and Gupta (1995), Moro (2001) and Widom (1995) outline the general maintenance problem of materialized views. Unfortunately, none of these works have studied the case in which each independent view has been defined over multiple sources of data. In this case, the problem is more complex since each single materialized view can involve several operational databases. In Silva (2002) we proposed a qualitative solution to the problem; however the solution left the quantification of the different cases for future study. In this article we propose a quantitative approach to the problem and we analyze the most efficient solution for each case.

The article is organized as follows: In section 2, the statement of the problem is presented along with the current state of the art. In section 3, the different parameters for quantifying the problem are studied. Also, different ways of measuring the time and the space required in the maintenance of the materialized views are analyzed. In section 4, the different cases are analyzed and the most efficient solution for each one is presented. Section 5 concludes.

2 State of the art

The updating of materialized views in real time is usually performed by first establishing a communication channel between the Warehouse and the underlying operational systems. This is done so that every time a modification takes place in the tables of the operational systems, these systems inform the Warehouse of the changes by sending the updates that are necessary to maintain the consistency of the materialized views.

When a view is defined on tables from databases of different operational systems, neither the Warehouse nor any of the operational systems can make a join among the tables to solve the view. This is because all the necessary information must be (at least temporarily) in the same location. At first glance, this may appear to be a problem of small granularity; however, this is not always the case. Many times business enterprises have branch offices located in different parts of the world that share the same Data Warehouse. In this context, the volume of data needed for maintaining the views might contain millions of tuples.

The solutions that have been proposed [Moro (2001), Samtani (1998) and Widom (1995)] and used to solve the problem of the multi-source views are based on the duplication of information [Gupta (1995) and Kuchenhoff (1991)]; the definition of maintenance transactions Stanoi (1999); or the redefinition of the original views, in an attempt to avoid (if it is possible) the joining of tables from different databases Colby (1996); or establishing a hierarchy of auxiliary multi-level views Silva (2002).

As it was stated in Silva (2002), in many cases the most appropriate solution consists of extending the materialized views definition with a hierarchy of views whose top level are the original views. Although this proposal solves the problem, it has only been defined at a qualitative level, and has left the quantification for future work. An outline is presented in Figure 1, where the solution to this problem is presented.

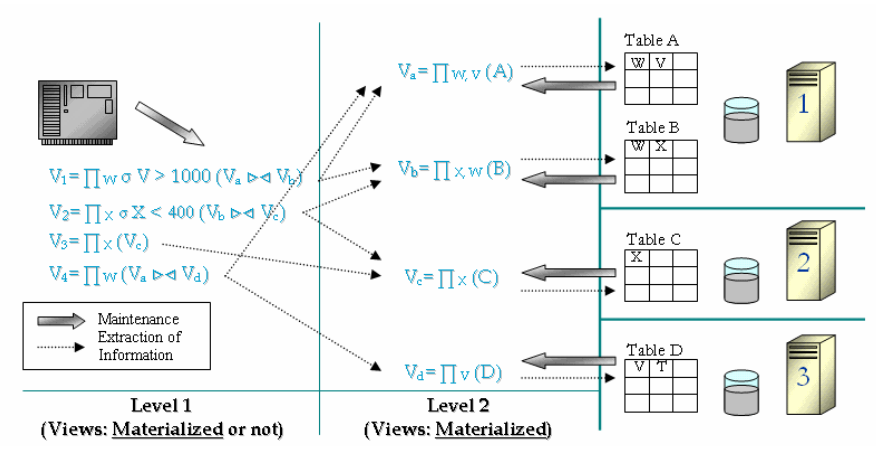

$V_1 = \prod w \, \sigma \, V > 1000 \, (V_a \bowtie V_b)$
$V_2 = \prod x \, \sigma \, X < 400 \, (V_b \bowtie V_c)$
$V_3 = \prod x \, (V_c)$
$V_4 = \prod w \, (V_a \bowtie V_d)$
$V_a = \prod w, v \, (A)$
$V_b = \prod x, w \, (B)$
$V_c = \prod x \, (C)$
$V_d = \prod v \, (D)$
Table A
W V
Table B
W X
Table C
X
Table D
V T
1
2
3
Maintenance
Extraction of Information
Level 1
(Views: Materialized or not)
Level 2
(Views: Materialized)

FIG. 1 – *Data warehouse maintenance using auxiliary multi-level views.*

Figure 1 shows three operational systems with tables (A, B, C, D) and a Data Warehouse with four materialized views (V1, V2, V3, V4) defined as follows:

- V1 = ΠW σ V>1000 (A∞B)
- V2 = ΠX σ X<400 (B∞C)
- V3 = ΠX (C)
- V4 = ΠW σ V>1000 (A∞D)

Views V1 and V2 introduce a problem because they include joins on tables from different databases that are in different locations. To solve this problem, we have defined four new views (Va, Vb, Vc, Vd) to extract the necessary information from tables A, B, C and D. The original views have been redefined from these new views (see definition in Figure 1) in such a way that none of the views include joins on tables from different databases. So the maintenance of the original views is now possible because they are defined over local views (single source); and also the maintenance of auxiliary views is possible because they are defined over a single (remote) source.

This leads to the question of how many level 2 views should be defined for each table of the operational systems in order to duplicate and maintain the smallest quantity of information possible, and at the same time maximize the response time of the system. This question does not have an easy answer, since the answer not only depends on the definition of the views, but on other parameters such as the transmission speed and the volume of data stored in the table. The main goal of this work is to find the answer to this question based on the study of all the possible cases where this problem occurs.

3 Problem statements: study of influential parameters

If we analyze the outline of views in Figure 1, it seems quite clear that there should be at least one view for each database in the operational systems. In the simplest case, this view belongs to the last level in the view hierarchy, and it contains information from one or more tables of this database. It is also clear that it is not always possible to represent all the necessary information from one database in a single view. In general, by defining a subview for each table included in the views of the first level, we have all the necessary information for maintenance. Each subview stores the minimum necessary information of each table, and by joining them, all the Data Warehouse views can be computed. Sometimes it is more appropriate to define more than one view for each table in order to avoid storing and maintaining unnecessary information. The following example shows that by using two views, V_1 and V_2, less information is stored than when using the single view V_3.

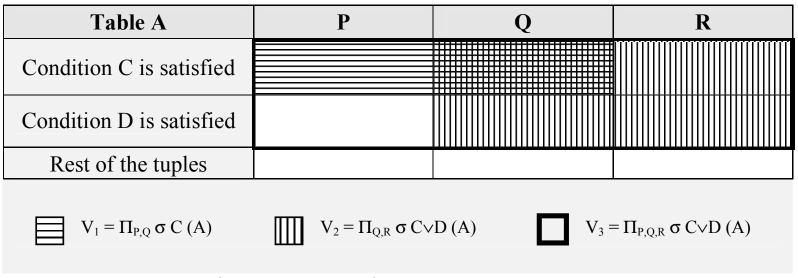


Table A
P
Q
R
Condition C is satisfied
Condition D is satisfied
Rest of the tuples
$V_1 = \Pi_{P,Q}\ \sigma\ C\ (A)$
$V_2 = \Pi_{Q,R}\ \sigma\ C \vee D\ (A)$
$V_3 = \Pi_{P,Q,R}\ \sigma\ C \vee D\ (A)$

FIG. 2 – *Removing spare data using several views.*

The information stored (and maintained) in views V1 and V2 does not include the tuples of the blank space (condition D/field P) which is stored by V3. A table can be represented as a space of two dimensions where dimension Y represents the tuples of the table and dimension X represents the fields of this table. Each view defined in this table represents a subspace that includes both the fields in its projection and the tuples defined in its selection. In this sense, two different subspaces defined by two different views can be grouped in a single new view. However, information that does not belong to any of them could be included in this new view.

Figure 2 shows two opposed problems that can appear during maintenance. When using a single view V3, unnecessary information can be stored (spare information). When using views V1 and V2 the same information is stored twice (redundant information).

Data Warehouse designers must determine the number of views for each database and each table of the operational systems. This decision is crucial to ensure that the volume of data stored in the auxiliary materialized views and the maintenance time of the Data Warehouse are minimized. To address this problem, we study quantitatively all the different cases that a designer can find. For this purpose, the designer should take into account the following variables:

Name	Description
M	Number of sent messages.
C	Global temporal maintenance cost (taking into account input-output operations, bits transference, etc.).
Card V	Cardinality of view V.
Grad V	Number of attributes (fields) in view V.
MT	Number of updates (insertions, modifications, deletions) on table T.
mT	Percentage of MT which corresponds with modifications on table T.
ρ	Number of updates that are relevant for several views.
k	Average of the maintenance cost (messages + index maintenance + etc) of an operation.
s-1	Number of updates cancelled before computing the view.

TAB. 1 – *Metrics and variables.*

In Zhuge (1995), authors propose to compute the number of messages between the operational systems and the Data Warehouse as a metric of the maintenance cost of a materialized view. This metric corresponds with M in Table 1. For its computation, they propose the following formula whose proof can be found in annex D in Zhuge (1995):

$$(1) \qquad M = \left\lceil \frac{M_T}{s} \right\rceil \times 2$$

In this metric, MT represents the total number of updates in the source; and for every ‘s’ updates the operational system sends (and receives) one message to the Data Warehouse. Then, assuming that the message is sent after each update (s=1), two messages are generated by each update in the source.

It is clear that measuring the maintenance cost by only considering the amount of sent messages can be too rough. For this reason, we propose a different metric that takes into account other factors such as the view size, the probability that the modifications in the sources will affect the materialized views, etc. The formula that defines this metric is:

$$(2) \quad C = M_T \cdot \frac{Card(V_1) \cdot Grad(V_1) + ... + Card(V_n) \cdot Grad(V_n)}{Card(V_T) \cdot Grad(V_T)} \cdot k + \rho \cdot k$$

where T is the table in the operational system on which the views V1..n are defined. The meaning of the rest of variables is explained in Table 1.

This metric takes into account the spare information as well as the redundant information. In the case that the intersection of two or more views is not the empty set, that is, there is redundant information, this redundancy is taken into account in the cost since it is computed several times in different pairs 'Card(Vi)·Grad(Vi)'. The spare information is computed in the corresponding pair in the same way.

The redundant maintenance is computed with the second expression of the formula; ρ·k represents the cost associated to all the operations caused by the change of a tuple from one view to another during maintenance. It is of special importance to realize that, in the case of disjoint views, this change of view cannot be caused by an insert or delete operation, but rather can only be caused by a modify operation. Therefore, when views do not have redundant tuples we use the parameter ρm which only refers to modifications. Assuming that every field in the views has the same probability to be modified, ρm can be computed by the formula:

$$(3)\ \rho_m = M_T \cdot m_T \cdot \sum_{n}^{i=1} \left(\frac{Card(V_i) \cdot Grad(V_i) \cdot Card(\bigcup_{j \neq i}^{j=1..n} V_j)}{Card(T) \cdot (Grad(T))^2} \right)$$

Parameter ρ represents the number of times that one update on a materialized view produces another update on any other view. In order to validate the correctness of figure (3), for simplicity, we can assume that every field in table T has the same probability to be updated by one of the MT updates (in practice, we use a statistical distribution or a weigh). Then, the probability that the view ‘i’ were affected by an update is:

$$(4) \qquad P_{Vi} = \frac{Card(V_i) \cdot Grad(V_i)}{Card(T) \cdot Grad(T)}$$

The probability that the update affects to the set of views V1…Vn:

$$(5) \qquad P_{V1..Vn} = \frac{Card(\bigcup_{n}^{i=1} V_i) \cdot Grad(\bigcup_{n}^{i=1} V_i)}{Card(T) \cdot Grad(T)}$$

Using (4), the number of updates which affect view ‘i’ is:

$$(6) \qquad N_{Vi} = M_T \cdot P_{Vi} = M_T \cdot \frac{Card(V_i) \cdot Grad(V_i)}{Card(T) \cdot Grad(T)}$$

The number of modifications which affect the view ‘i’ is:

$$(7) \qquad N'_{Vi} = M_T \cdot m_T \cdot P_{Vi} \cdot \frac{Card(\bigcup_{j \neq i}^{j=1..n} V_j)}{Card(T)}$$

The generalization of (7):

$$\rho_m = M_T \cdot m_T \cdot \sum_n^{i=1} \left(\frac{Card(V_i) \cdot Grad(V_i) \cdot Card(\bigcup_{j \neq i}^{j=1..n} V_j)}{Card(T) \cdot (Grad(T))^2} \right) \tag{8}$$

For the case of two views:

$$\rho_{1,2} = M_T \cdot \frac{Card(\bigcap V_1, V_2)}{Card(T)} \tag{9}$$

4 Case-based reasoning

There is not only one solution to the problem described in the previous section. The solutions depend on different factors that are related to the definition of the views (for example the number of fields and tuples of a table used in several views); and they also depend on architectural factors such as the average speed of the information transmission, the average number of transmissions which are necessary to perform maintenance, the volume of information to maintain, etc. In this section, we study all the possible combinations that can take place in the definition of diverse views on a table. For each case we analyze which solution is the most appropriate according to the rest of the parameters.

Conditions			**Field W**	**Field X**	**Field Y**	**Field Z**
Condition A	Condition C	Condition F				
	Condition D	Condition E				
Condition B	Condition F	Condition D				
	Condition E					

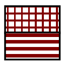
FIG. 3 – *Table T with 4 fields and 6 conditions defined over the tuples space.*

Figure 3 shows a hypothetical table T with 4 fields (W, X, Y, Z) and 6 conditions dividing the tuples space in 4 disjoint sets. Two views defined over this table necessarily hold in one of the following cases:

CASE 1: (Same projection, No disjoint conditions)

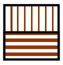
$V_1 = \Pi\ W,X\ \sigma\ A\ (T)$

$V_2 = \Pi\ W,X\ (T)$



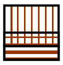
CASE 2: (Same projection, Disjoint conditions)

$V_1 = \Pi\ W, X\ \sigma\ A\ (T)$
$V_2 = \Pi\ W, X\ \sigma\ B\ (T)$



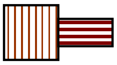
CASE 3: (Same projection, Partially disjoint conditions)

$V_1 = \Pi\ W, X\ \sigma\ A\ (T)$
$V_2 = \Pi\ W, X\ \sigma\ D\ (T)$



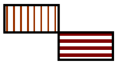
CASE 4: (Distinct projection, No disjoint conditions)

$V_1 = \Pi\ W, X\ \sigma\ B\ (T)$
$V_2 = \Pi\ Y, Z\ \ \sigma\ F \wedge D\ (T)$



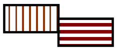
CASE 5: (Distinct projection, Disjoint conditions)

$V1 = \Pi\ W, X\ \sigma\ E\ (T)$
$V2 = \Pi\ Y, Z\ \ \sigma\ C\ (T)$



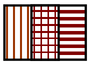
CASE 6: (Distinct projection, Partially disjoint conditions)

$V1 = \Pi\ W, X\ \sigma\ E\ (T)$
$V2 = \Pi\ Y, Z\ \ \sigma\ B\ (T)$



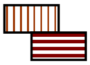
CASE 7: (Partially disjoint projections, Same conditions)

$V1 = \Pi\ W, X\ \sigma\ B\ (T)$
$V2 = \Pi\ X, Y\ \ \sigma\ B\ (T)$



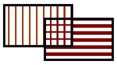
CASE 8: (Partially disjoint projections, Disjoint conditions)

$V1 = \Pi\ W, X\ \sigma\ F\ (T)$
$V2 = \Pi\ X, Y\ \ \sigma\ E\ (T)$



CASE 9: (Partially disjoint projections, Partially disjoint conditions)

$V1 = \Pi\ W, X, Y\ \sigma\ F\ (T)$
$V2 = \Pi\ X, Y, Z\ \ \sigma\ B\ (T)$



For each case, we have computed the cost taking into account all the possibilities (using different number of views). The whole analysis can be found in a publicly available technical report at: http://www.dsic.upv.es/~jsilva/research.htm#techs

From this analysis, we conclude that in cases 1, 2, 3 and 7 the best solution is to consider one single view. In cases 4, 5, 6 and 8 the best solution depends on the parameter ρ and on the cardinality of the views:

1.1. If $\rho >$ MT (Card V3 · Grad V3 – (Card V1 · Grad V1 + Card V2 · Grad V2)) / Card T · Grad T) then the cost of using one view is smaller than the cost of using two views.
1.2. If $\rho =$ MT (Card V3 · Grad V3 – (Card V1 · Grad V1 + Card V2 · Grad V2)) / Card T · Grad T) then the cost of using one view or two views is the same. In this case only one view is used.
1.3. If $\rho <$ MT (Card V3 · Grad V3 – (Card V1 · Grad V1 + Card V2 · Grad V2)) / Card T · Grad T) then the cost of using two views is smaller than the cost of using one view.

The case number 9 is more complex since it needs 3 views for maintaining all the information without redundancy. The computation of the cost for this case is also shown in the technical report at: http://www.dsic.upv.es/~jsilva/research.htm#techs

5 Conclusions and future work

There are many works [Ding (1999), Gupta (1995), Kuchenhoff (1991), Quass (1996), Stanoi (1999), Zhuge (1996) and Zhuge (1995)] which deal with the use of materialized views in Data Warehouse systems or with the problem of maintenance of these materialized views. However, surprisingly, not much effort [Moro (2001), Samtani (1998) and Widom (1995)] has been spent to address the problem of multi-source materialized views. This is an important problem since, for example, it is not possible —or it is really very expensive due to the necessity of duplicate the tables— to maintain in real-time a view that joins tables from different sources. In Silva (2002), we proposed to extend the original views with a set of intermediate views that extract information from just one determined source. But in that work we did not consider how to find out which is the optimal configuration of views taking into account the maintenance costs.

In this paper we have carried out this study analyzing each possible case from the point of view of required space and time consumed for maintenance. To achieve this goal ECV (Eager Compensating Algorithm) Zhuge (1995) have been used in order to maintain the auxiliary single-source (top level) views.

References

Ashish N., Knoblock C.A. and Shahabi C (1999). *Selectively Materializing Data In Mediators By Analyzing User Queries*. In Proceedings of the 4th IFCIS International Conference on Cooperative Information Systems (CoopIS), Edinburgh, Scotland, September 1999.

Colby L.S., Griffin T.G., Libkin L., Mumick I.S. and Trickey H. (1996). *Algorithms for Deferred View Maintenance*. In Proceedings of the SIGMOD: pp. 469-479, June 1996.

Ding L., Zhang X. and Rundensteiner E.A. (1999). *Enhancing Existing Incremental View Maintenance Algorithms Using the Multi-Relation Encapsulation Wrapper*. Technical Report WPI-CS-TR-99-23: pp. 9-11, Worcester Polytechnic Institute, Dept. of Computer Science, August 1999.

Gupta H. (1997). *Selection of Views to Materialize in a Data Warehouse*. In Proceedings of the International Conference on Database Theory (ICDT 1997): 98-112, Delphi, Greece; January 1997.

Gupta A. and Mumick I.S. (1995). *Maintenance of Materialized Views: Problems, Techniques, and Applications*. In the IEEE Data Engineering Bulletin Vol. 18 N^{er}. 2, June 1995.

Kuchenhoff V. (1991) *On the efficient computation of the difference between consecutive database states*. In Proceedings of the 2nd International Conference on Deductive and Object-Oriented Databases (DOOD), December 1991.

Moro G. and Sartorio C. (2001) *Incremental Maintenance of Multi-Source Views*. In Proceedings of the 12th Australasian of Electrical and Electronics Engineers (ADC 2001), 2001.

Quass D., Gupta A., Mumick I.S. and Widom J. (1996) *Making Views Self-Maintainable for Data Warehousing*. In Proceedings of the Conference on Parallel and Distributed Information Systems (PDIS '96), 1996.

Samtani S., Mohania M., Kumar V. and Kambayashi Y. (1998) *Recent Advances and Research Problems in Data Warehousing*. ER Workshops, 1998.

Stanoi I.R., Agrawal D. and El Abbadi A. (1999) *Modeling and Maintaining Multi-View Data Warehouses*. In Proceedings of the 18th International Conference on Conceptual Modelling (ER '99), November 1999.

Valluri S.R., Vadapalli S. and Karlapalem K. (2002) *View Relevance Driven Selection of Materialized Views in Data Warehousing Environment*. In Proceedings of the 13th Australasian Database Conference (ADC 2002), Melbourne, Australia 2002.

Widom J. (1995) *Research Problems in Data Warehousing*. Computer Science Department, Stanford University. In Proceedings of the 4th International Conference on Information and Knowledge Management (CIKM); November 1995.

Silva J., Belenguer J. and Celma M. (2002) *Materialización de Vistas Multi-Origen: Vistas Multinivel*". In Proceedings of the 7th Jornadas de Ingeniería del Software y Bases de Datos (JISBD'02), El Escorial, Madrid, Spain; 2002.

Zhuge Y., Garcia-Molina H. and Wiener J.L. (1996) *The Strobe Algorithms for Multi-Source Warehouse Consistency*. In Proceedings of the 4th International Conferences on Parallel and Distributed Information Systems (PDIS'96), Miami Beach, Florida, USA; 1996.

Zhuge Y., Garcia-Molina H., Hammer J. and Widom J. (1995) *View Maintenance in a Warehousing Environment*. CSD - Stanford University; SIGMOD Conference 1995: pp. 316-327; 1995.

Résumé

Dans beaucoup de systémes d'information, les bases de données que supportent ces systemes sont distribuées en différentes sections ou departements de l'organization. Dans ce type de systemes, trés souvent, il est necessaire integrer l'information en tels bases de données pour faire lánalyse de l'organization. C'est l'objectif des systémes de entrepôts de données. D'un point de vue théorique, on peut voir un entrepôt de données comme un ensemble de vues matérialisées, definis a partir de bases de données operationelles. Ces vues emmagasinent information historique qui doit être actualisée ou bien en temps real ou bien periodiquement. Durant le mantient de ces vues le systéme doit realiser des opérations de selection, projection, join, etc, sur les differentes bases de données, ce qui represente un processus complexe car réaliser un join sur des différentes tables exige disposer, au moins temporairement du contenu de ces structures dans un meme répositoire. Dans ce travail, on étudie ce probléme et on propose une méthode pour réduire le volume d'information auxiliaire qui doit ètre maintenue par le systéme en améliorant les temps de réponse.