Performance Tests in Data Warehousing ETLM Process for Detection of Changes in Data Origin

Rosana L. A. Rocha, Leonardo Figueiredo Cardoso, Jano Moreira de Souza

COPPE/UFRJ, Federal University of Rio de Janeiro, PO BOX 68511, Rio de Janeiro, RJ - Brazil
(55)(21) 2590-2552
{rosana, cardoso, jano}@cos.ufrj.br

Abstract. In a data warehouse (DW) environment, when the operational environment does not posses or does not want to inform the data about the changes that occurred, controls have to be implemented to enable detection of these changes and to reflect them in the DW environment. The main scenarios are: i) the impossibility to instrument the DBMS (triggers, transaction log, stored procedures, replication, materialized views, old and new versions of data, etc) due to security policies, data property or performance issues; ii) the lack of instrumentation resources on the DBMS; iii) the use of legacy technologies such file systems or semi-structured data; iv) application proprietary databases and ERP systems. In another article [1], we presented the development and implementation of a technique that was derived for the comparison of database snapshots, where we use signatures to mark and detect changes. The technique is simple and can be applied to all four scenarios above. To prove the efficiency of our technique, in this article we do comparative performance tests between these approaches. We performed two benchmarks: the first one using synthetic data and the second one using the real data from a case study in the data warehouse project developed for Rio Sul Airlines, a regional aviation company belonging to the Brazil-based Varig group. We also describe the main approaches to solve the detection of changes in data origin.

1 Introduction

The detection of changes in data origin issue is well known in the DW area. As mentioned by DO, DREW et al. [2], most research work on DW update focuses on the problem that, given a differential relation, how do we refresh the DW efficiently, approach defined by ÖZSU & VALDURIEZ [3]. This approach capture the after and before images, for all lines affected by each operation. Some of these researches are based on the existence of an instrumentation resource of differential relation on the DBMS. The difference between them occurs in terms of DW capabilities, such as: convergent DW consistency [4; 5], replication of some source relations [6], full replication [7], versioning [8], etc.

According INMON & KELLEY [9], a DW is a repository of integrated information, available for queries and analysis (e.g., decision support, or data mining). The DW came to meet this demand for a possibility of fast analysis of business information.

In this paper, to prove the efficiency of our technique, we do comparative performance tests between two database snapshots approaches (with and without signatures) for detection of changes in data origin. We performed two benchmarks: the first one using synthetic data and the second one using the real data. The rest of the paper is organized as follows: In section 2, we have described the main approaches which may be used for solving the problem of detecting change in data origin. In section 3, we have described the implementation of our technique. In section 4, we have presented performance tests among our technique and database snapshots approach. In the section 5 and last part, we present our final considerations and current works aiming at the improvement of this implementation.

2 Detection of Changes

There are several challenges which would be overcome in the development of a DW environment. However, in this article, the main focus will gravitate around the detection of changes in data origin. According to LABIO, YERNENI et al. [10], one of the main problems is updating derived data when the remote information sources change. For the cases in which the operational environment does not bear or does not want to inform the storage of changing history taking place on data, controls have to be implemented in order to meet this need.

There are several approaches which may be implemented in detection of changes in data origin. In order to choose among them, we have to take into consideration the following factors: there are advantages, disadvantages, and the existence of the necessary features in the operational and DW environments. Below, we will describe some of the most of used approaches for the solution of the change detection problem in data origin. We will use as a basis for this description, the framework proposed for the DW environment in [1].

2.1 New table with all the performed changes

Operational: for each table in the data origin area in which the mapping of changes is necessary, we have created a "daughter table" related to the originating table, identifying the following: the change operation which was performed, when it occurred, and in the case of alterations, which fields have been the altered and their values prior to the change.

Necessary features: that the data from the origin area be stored in a DBMS; that the a DBMS bearing a trigger mechanism, with the approach of "old" and "new" versions implemented, for the operations of insert, update and delete. This approach was mentioned WIDOM & CERI [11]; that the using of triggers be permitted.

Advantages: assurance the entire mapping of changes performed in the originating table have been stored; facility and greater speed in the extraction process, as only the lines changed from the last extraction performed would be consulted.

Disadvantages: overload on the operational environment, due to the need treatment in inclusion of mapping of each change taking place; increase in the need for space and storage of this new data and indexes; need for cleaning management, from time to time, of the mapping tables which beard a very large growth.

When to use: when the mapping of all changes performed in the data origin becomes really necessary.

2.2 Marking on originating table for storing the last change carried out

Operational: for each table in the data origin area in which the mapping of changes becomes necessary, we have created some columns identifying the change operation performed and when it took place. In the case of the change operation, we only mark when there has been a change in one of the fields relevant to the DW extraction process. Approach presented in [12].

Necessary features: as abovementioned "approach 2.1".

Advantages: decrease in operational environment overloading this *à vis*, "approach 2.1"; decrease in the need for storage space regarding "approach 2.1"; is an greatest speed in the extraction process, as only the lines changed as from the last extraction performed will be consulted.

Disadvantages: the entire mapping of changes taking place in the originating table is not stored, only the last change; increase in the need for storage space of these new fields; the removals must be logical; the need for cleaning management, from time to time, on the originating table for the lines marked as logical removals which have already been extracted for the DW.

When to use: when the mapping of all changes performed in data origin is not necessary, only the last change.

2.3 Interpretation of transaction log on the DBMS

Operational: as from the DBMS transactions log, we interpret all transactions carried out for each table in the data origin area, in which the mapping of changes performed is necessary. Approach presented in [2; 12; 13].

Necessary features: that the data from the origin area be stored in a DBMS; this DBMS should have implemented the transactions log control; that the translation from the DBMS transactions log of SQL commands for insert, update and delete should be possible. This translation maybe performed through a specific tool from DBMS or through the development of a proprietary translator.

Advantages: there is no overload on the operational environment; decrease the necessity of space for storage, communication and processing; ease and greater speed in the extraction process, as only the lines changed since the last extraction performed would be consulted.

Disadvantages: need for controlling the transaction log area size by the database administrator (DBA), in order to prevent transaction lost. This may occur when the cycling of the transaction log takes place, that is, an area from the transaction log, with transactions already commit, maybe reused by the DBMS; the possibility of a physical or human failure in the transaction log file copy in a segregated area before the beginning of the backup process.

When to use: when the mapping of all or part of the changes occurred in data origin is necessary; possibility risk of lost of changes because of human, physical failure or by an unexpected operation by the DBA.

2.4 Comparison of database snapshots

A number of methods may be used to perform a copy of data, from the origin area to the work area, necessary for the implementation of this approach. As an example of these methods we have: bulk copy, replication or trigger (two phase commit protocol). We present below the items common to all the methods of this approach. This approach was mentioned in [12-15].

Operational: for each table in the data origin area, in which the mapping of changes is necessary, we will perform a copy of the table of the origin area, for the work area. In this copy only the fields pertinent to the DW extraction process will be considered. We have identified this new table under "<name of table>_current". In case the extraction process is being carried out for the second time, there will be the analogous table to this new table, identified as "<name of table>_previous". These two tables will be used to verify the changes taking place. These changes will be found in the comparison between their data. In case the extraction process is being performed for first time it will not be necessary to carry out the comparisons.

Necessary features: that the data from the origin area be stored in a DBMS; a large storage space in the work area for the two versions the "current" and the "previous", for each table in which the mapping of changes is necessary.

Disadvantages: use of processing and recording time in data copy from the origin area to the work area and on data comparison, for the detection of changes an each of the tables; use of large storage space in the work area for the two versions the "current" and the "previous", of each table in which the mapping of changes is necessary.

When to use: in environments in which the DBMS in the origin area does not bear trigger mechanisms; or when the implementation of triggers is not allowed; or where there is no transaction log in DBMS; or where there is no possibility for translating the transaction log; or when there are problems and/or low performance access to the origin area data.

2.5 Refresh tables

Operational: for each table in the origin of data, in which the mapping of changers it is necessary, we have removed all data from the fact and dimension tables and performed the entire process of extraction and load again. This approach was mentioned in [13; 16; 17].

Necessary features: none in special.

Advantages: there is no need for implementing any control for change detection both in the origin and work areas.

Disadvantages: use of a large processing time of each new extraction, transformation / validation, load and materialization process as the process would be redone each time; the mapping of changes performed is not fully done as we only have the last position; lost of DW information history, as at each new ETLM process we would be recreating its data as from the last position in the operational environment.

When to use: when the mapping of all changes performed in data origin is not actually necessary; there is no need for keeping DW information history; there is no time restriction, as we will recreate entire DW environment at each new processing.

3 An Improved Approach

In our DW development and implementation at Rio-Sul Airlines, we have used as a basis the framework previously presented for the DW environment [1]. For the choice of the approach to be implemented in the solution of problem of change detection at data origin, we have taken into consideration the features found in the Rio-Sul Airlines operational and DW environment.

The main feature of this environment refers to DBMS in which the originating data was stored. The system has serious limitations both technical and policy-wise, such as: i) the inexistence of a instrumentation resource of triggers, replication and transaction log; ii) low performance in querying processing, leading to overload in the operational environment; iii) limitations in data access via ODBC; iv) concurrency problems; v) proprietary application database; vi) great difficulty for DBMS maintenance; vii) high data volatile; viii) an interrupted use of DBMS by users (24x7).

The technique we have developed and implemented derivate from the comparison of database snapshots (“approach 2.4”), in which we use signatures to mark and detect changes. We describe below, in a step by step manner, how we have solved each of the problems found.

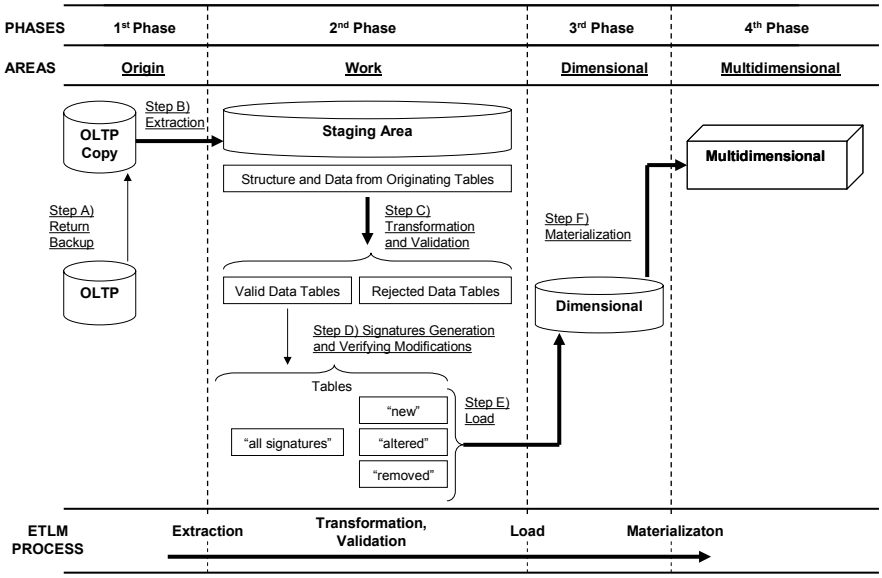


Fig. 1. The DW environment framework used at Rio-Sul Airlines

In this environment, we use a smaller machine. Every weekend, we secured the backup from the previous night and we went up in this new machine (step A, Figure 1). As from this data, we started the extraction process (step B, Figure 1) using the method of bulk copy (approach 2.4, bulk copy method). At the end of this process, we had the tables and the data relevant to the DW project in the relational model, without foreign keys and with some indexes created, aiming at facilitating queries to be performed next.

As from this data, we carried out the data validation and transformation process (step C, Figure 1), having as a result, the information of valid lines and of rejected lines. For the valid line information, we marked the lines which had gone through all validations, using only a new filed created in the tables charged in the extraction process themselves. For the rejected line information, we created a new table with the relevant information from the problem table / line, identifying also the reason for rejection, aiming at facilitating the adjustment of incorrect data in the operational environment later.

Due to limitations of the DBMS, we use the approach of comparison of database snapshots ("approach 2.4"). This approach could be used as there was no need to map all the changes which had occurred in data origin, only the last situation met the company needs.

With the purpose of improving the two main disadvantages of the approach of comparison of database snapshots ("approach 2.4"), which are, namely, the processing time for comparisons in the verification of changes which have occurred and the space for storage of table copies; we implemented the following improvement: for each table which is the origin of data in fact and dimension tables and in which the detection of change in data origin was necessary, we created a new table in which we had the primary key of the data originating table and a signature field.

This signature is calculated using the BCS (Binary Checksum) algorithm, taking into consideration the bringing together of all relevant fields on each table. In our implementation, the occurrence of collisions was very unlikely, as the queries performed on the tables with the suffix of "all signatures" were always carried out as from the primary key of the table, and not as from the signatures created. Therefore, the signatures were only used to detect whether there has been any change to the original data.

Depending on the diversity of data, the use of a BCS (Binary Checksum) for calculating the signature may have a greater probability of causing collisions. In this case, the use of a more

complex algorithm is necessary to minimize this risk. In most cases, given the statistical nature of the use of warehouses, the missing of one operation will be of no harm.

In order to help in the identification and the query of the changes which took place, for each table in the process we have created four new tables: the three first are comprised by the primary key of the originating table plus the signature field of the line ("all signatures", "new", "altered"); and the fourth is comprised only by the primary key of the originating table ("removed"). We describe the use of each one as follows.

We will describe, as follows, how the process of calculating and verifying signatures works (step D, Figure 1). For each table in which the detection of changes in data origin was necessary we obtained the lines marked as valid and carried out the calculation of the signature using the BCS algorithm. At each line calculated, we checked on the reference table with the suffix "all signatures", using the primary key as a basis for queries. If the result of the query does not return any line, we include the data in the reference table with the suffix "new". If returns, we compare the signature calculated with the signature stored in the reference tables with the suffix "all signatures". In case the signature is different, we include a new line in the reference table with the suffix "altered" otherwise the line is the same as the last load. Finally, we move to the next line.

At the end of the process for checking the inclusion and alteration modifications, we started the process to identifying the removals. For all tables with the suffix "all signatures", we queried as from the primary key, whether the line existed in the table of data origin referring to the table which is being checked. In case the query does not return any value, this means that the original line has being removed. In the end we include a new line in the reference table with the suffix "removed".

At the end of this process, as from data existing on tables with suffix "new", "altered" and "removed", we will start the load process (step E, Figure 1). In this process, we have follow the logical order, aiming at avoiding failures and the adjustments which will be performed on the dimensional model data, according to each case. As we describe below: i) Processing of new lines (dimensions), ii) Processing of altered lines (dimensions), iii) Processing of new lines (facts), iv) Processing of altered lines (facts), v) Processing of removed lines (facts), vi) Processing of removed lines (dimensions).

5 Performance Tests

As mentioned before, the two main disadvantages of the approach of comparison of database snapshots ("approach 2.4") are: the space for storage of table copies and the processing time for comparisons in the verification of changes. For solving these problems, we implemented the improvement of signatures. With the purpose of verifying the better results of our approach, we present our considerations for each disadvantage.

For the first one, the space for storage of table copies, the improvement reached is clear. Instead of store all fields, which we will verify the changes taking place, we transform these fields in one field for the signature of them, for each table involved.

For the second one, the processing time for comparisons in the verification of changes, we did performance tests to demonstrate the improvement reached. In these tests we performed two benchmarks: the first one using synthetic data and the second one using the real data from Rio Sul Airlines.

The tests were done to detect changes. The inclusion and removal changes follow the same operation process, for both approaches, so the performance tests were not necessary. The performance tests were done using a computer Pentium III 600 Mhz, 256 Mb RAM, SQL Server 2000 DBMS. In all tests, we followed the structure presented on Figure 2 and detailed as below.

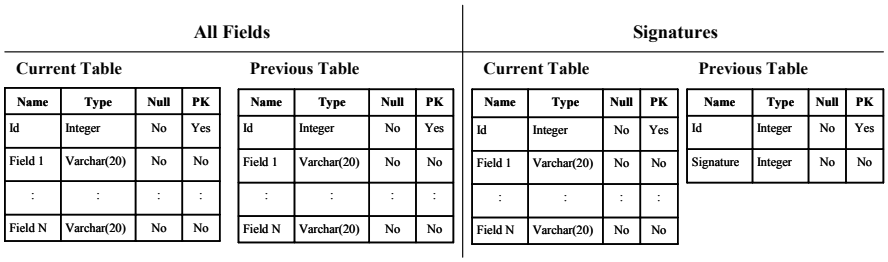


Fig. 2. Structures used on performance tests

For the tests using synthetic data, we prepared two types of tables, using the structures presented in Figure 2. They are named respectively "table10" and "table20". For all of them we have one field as primary key, and respectively nine and nineteen other fields as complementary data. These tests were done in three steps: i) for each one of the tables current and previous, we carried out, 100.000, 200.000 and 500.000 lines; ii) we did 10%, 20% and 90% of alteration changes in current table. iii) we did the comparisons in the verification of changes.

For the tests using real data, we used one table, named "real table". This table has one field char(12) type as primary key, nine fields decimal (19,4) type, seven fields as char type, all of them as complementary data. These tests were done in three steps: i) for each one of the tables current and previous, we had already around, 19.000.000 lines. ii) we did 1%, 2% and 9% of alteration changes in current table. iii) we did the comparisons in the verification of changes.

To organize the presentation of our performance tests we named our approach as "signatures", and the comparison database snapshots approach as "all fields". The pattern queries used by each approach for comparisons in the verification of changes were: i) All fields approach: Perform a number of comparisons according to the number of the fields for the table.

```
SELECT count(C.id)
FROM table_current as C, table_previous as P
WHERE   (C.id = P.id) AND
 ((C.field 1 <> P.field 1) OR (C.field 2 <> P.field 2) OR (…)
 (C.field  N <> P.field N));
```

ii) Signatures approach: Perform a number of comparisons according to the number of the fields for the primary key and one execution program, to calculate the signature.

```
SELECT count(C.id)
FROM table_current as C, table_previous as P
WHERE  (C.id = P.id)  AND (P.signature <> F_CalculateSignature
(C.Field 1,..,C.Field N));
```

The use of the clause *count(C.id)*, has the objective to obtain the spend time for comparisons, only after all results were computed. In a real situation, we obtain only the fields that compose the primary key. In our tests we did the alteration changes only in one field, that was the second field used in the *where* clause. The main objective for this scenario is permitting, always the best case in difference comparisons for the "all fields" approach. The results of our tests are presented and explained as follows.

5.1 Performance tests using synthetic data

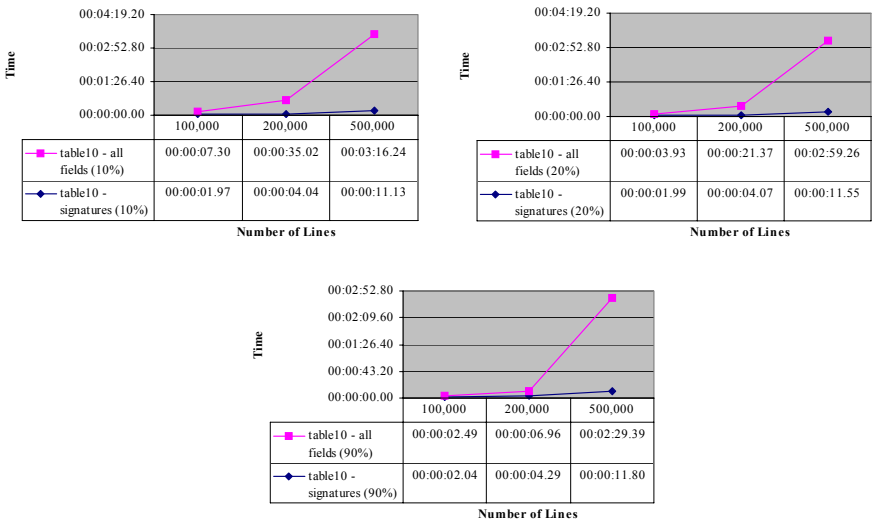


Fig. 3. "table10" with 10%, 20% and 90% of alteration changes

According to the graphs depicted in Figure 3, we may observe that the "signatures" approach is practically constant for each modification tax, this occurs because the algorithm always do the same number of comparisons and function calls. The "all fields" approach predisposes to be slowly with a minor number of changed lines. Because this approach has always to compare all fields to discover if one modification occurred when in fact it not occurred. The better performance for "signatures" approach, should take place also, because the "previous table" is considerably smaller than in "all fields" approach, and can easily fit in main memory for join process.

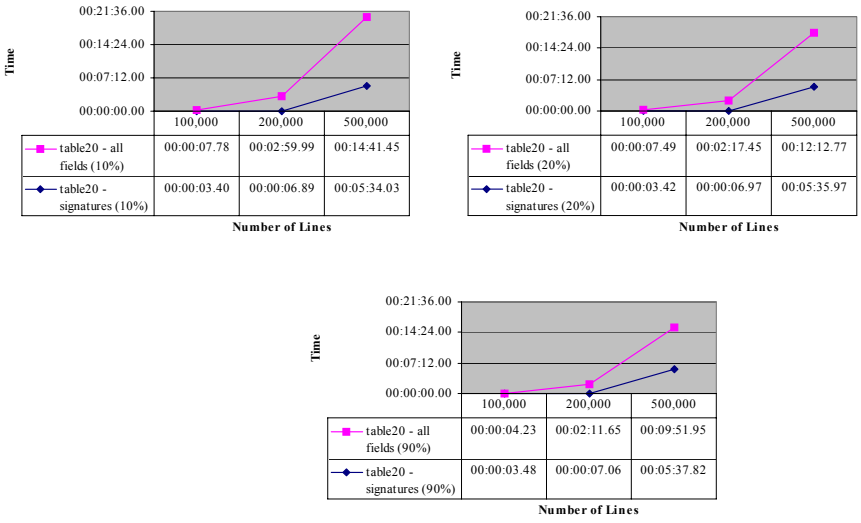
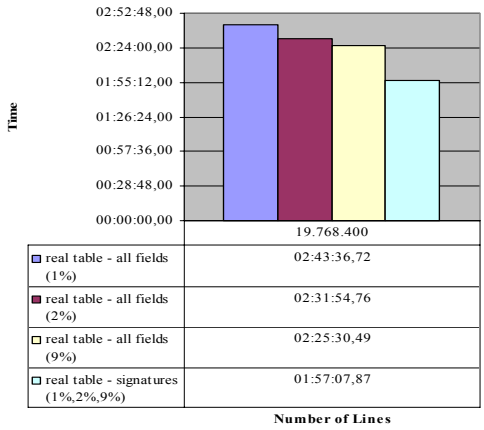


Fig. 4. "table20" with 10%, 20% and 90% of alteration changes

According to the graphs depicted in Figure 4, the results prove that the processing time for "signatures" approach is directly proportional to the number of fields involved in the comparisons and the number of lines of tables.

5.2 Performance tests using real data



According to the graph depicted in Figure 5 we may observe that the tests with real data only confirm the better performance of "signatures" approach. Although, the computed times are considerable, we can notice the gain of approximately 30 minutes for a huge table. In our real DW environment we have a great number of such type of huge tables, so accumulated gain of time became more important with the use of "signatures" approach.

Fig. 5. "real table" with 1%, 2% and 9% of alteration changes

6. Conclusion

According to LABIO, YERNENI et al. [10], due to the constantly increasing size of warehouses and the rapid rates of change, there is increasing pressure to reduce the time taking for warehouse update. To meet this demand, this article had as its main purpose presenting our performance tests to prove the efficiency of our technique, with signatures, against the approach of comparison database snapshots. In our performance tests we should conclude that, in the greater part of the cases our technique had a better performance; especially in cases that the modifications occurred in a small number of lines. In real situations, usually the number of modifications is smaller than presented in our tests, favoring the use of our "signature" approach.

Thus, we conclude and prove that our "signature" approach is more efficient and economic in relation the storage space and comparison time, and should be used in situations where the use of instrumentation resource on the DBMS in operational environment is not possible.

7 References

[1]. ROCHA, R. L. A., CARDOSO, L. F., SOUZA, J. M., 2003, *An Improved Approach in Data Warehousing ETLM Process for Detection of Changes in Data Origin*. COPPE/UFRJ, Report Nº ES-593/03.

[2]. DO, L., DREW, P., JIN, W., et al, 1998, "Issues in Developing Very Large Databases". In: *Proceedings of the 24th VLDB Conference*, pp. 633-636, New York, USA, August.

[3]. ÖZSU, M. T.,VALDURIEZ, P., 1991, *Principles of Distributes Database Systems*. 1st Ed, New Jersey, USA, Prentice Hall Inc.

[4]. ZHUGE, Y., GARCIA-MOLINA, H., HAMMER, J., et al, 1995, "View Maintenance in a Warehousing Environment". In: *Proceedings of ACM SIGMOD International Conference on Management Data*, pp. 316-327, San Jose, California, USA, June.

[5]. ZHUGE, Y., GARCIA-MOLINA, H., WIENER, J. L., 1996, "The Strobe Algorithms for Multi-Source Warehouse Consistency". In: *Proceedings on Parallel and Distributed Information Systems*, pp. 146-157, Miami Beach, Florida, USA, December.

[6]. QUASS, D.,WIDOM, J., 1997, "On-Line Warehouse View Maintenance". In: *Proceedings of ACM SIGMOD International Conference on Management Data*, pp. 405-416, Tucson, Arizona, USA, May.

[7]. HULL, R.,ZHOU, G., 1996, "Towards the Study of Performance Trade-offs Between Materialized and Virtual Integrated Views". In: *Proc. Workshop on Materialized Views: Techniques and Applications (VIEWS 96)*, pp. 91-102, Montreal, Canada, June.

[8]. QUASS, D., GUPTA, A., MUMICK, I. S., et al, 1996, "Making Views Self-Maintainable for Data Warehousing". In: *Proceedings on Parallel and Distributed Information Systems*, pp. 158-169, Miami Beach, Florida, USA, December.

[9]. INMON, W. H.,KELLEY, C., 1993, *Rdb/VMS, developing the data warehouse*, Boston, QED Pub. Group.

[10]. LABIO, W. J., YERNENI, R., GARCIA-MOLINA, H., 1999, "Shrinking the Warehouse Update Window". In: *Proceedings of ACM SIGMOD International Conference on Management Data*, pp. 383-394, Philadelphia, USA, June.

[11]. WIDOM, J.,CERI, S., 1996, "Active Databases Systems: Triggers and Rules for Advanced Database Processing.", San Francisco, California, USA.

[12]. CRAIG, R. S., VIVONA, J. A., BERKOVITCH, D., 1999, *Microsoft data warehousing building distributed decision support systems*, New York, Wiley.

[13]. WIDOM, J., 1995, "Research Problems in Data Warehousing". In: *Proceedings of ACM CIKM International Conference on Management Data*, pp. 25-30, Baltimore, USA, November.

[14]. HAMMER, J., GARCIA-MOLINA, H., WIDOM, J., et al, 1995, "The Stanford Data Warehousing Project", *IEEE Quarterly Bulletin on Data Engineering; Special Issue on Materialized Views and Data Warehousing*, v. 18, n. 2, pp. 41-48.

[15]. CHAWATHE, S. S.,GARCIA-MOLINA, H., 1997, "Meaningful Change Detection in Structured Data". In: *Proceedings of ACM SIGMOD International Conference on Management Data*, pp. 26-37, Arizona, USA, May.

[16]. KIMBALL, R., 1996, *Data Warehouse Toolkit*, New York, USA, John Wiley & Sons, Inc.

[17]. KIMBALL, R., 1998, *The Data Warehouse Lifecycle Toolkit. Expert Methods for Designing, Developing, and Deploying Data Warehouses*, New York, USA, John Wiley & Sons, Inc.