A CONTINUOUS DATA INTEGRATION METHODOLOGY FOR SUPPORTING REAL-TIME DATA WAREHOUSING

Ricardo Jorge Santos [1] and Jorge Bernardino [1, 2]
[1] *CISUC – Centre of Informatics and Systems of the University of Coimbra - University of Coimbra*
[2] *ISEC – Superior Engineering Institute of Coimbra – Polytechnic Institute of Coimbra*
lionsoftware.ricardo@gmail.com, jorge@isec.pt

Keywords: real-time and active data warehousing, continuous data integration, refreshment loading process

Abstract: A data warehouse provides information for analytical processing, decision making and data mining tools. As the concept of real-time enterprise evolves, the synchronism between transactional data and data warehouses, statically implemented, has been reviewed. Traditional data warehouse systems have static structures of their schemas and relationships between data, and therefore are not able to support any dynamics in their structure and content. Their data is only periodically updated because they are not prepared for continuous data integration. For these purposes, real-time data warehouses seem to be very promising. In this paper we present a methodology on how to adapt data warehouse schemas and user-end OLAP (On-Line Analytical Processing) queries for efficiently supporting real-time data integration. To accomplish this, we use techniques such as table structure replication and query predicate restrictions for selecting data, managing to enable continuous data integration in the data warehouse with minimum impact in query execution time. We demonstrate the functionality of the method by analyzing its impact in query performance using benchmark TPC-H executing query workloads while simultaneously performing continuous data integration at various insertion time rates.

1 INTRODUCTION

A data warehouse (DW) collects data from multiple heterogeneous operational source (OLTP – On-Line Transaction Processing) systems and stores integrated information in a central repository, used by analytical applications (OLAP – On-Line Analytical Processing) with different user requirements. The common form of getting decision making information is using OLAP tools (Chaudhuri, 1997). The data source for these tools is the DW data area, where records are updated periodically using ETL (Extraction, Transformation and Loading) tools. ETL processes identify and extract relevant data from OLTP source systems, cleaning and molding it into an adequate integrated format and finally, loading the final formatted data into the DW's database (DB).

Executing this update periodically implies that most recent OLTP source records are not included into the data area, being excluded from the results supplied by OLAP tools. It has been assumed that data in the DW can lag at least a day if not a week or a month behind the actual operational data in the OLTP systems (Zurek, 2001). This has been based on the notion that business decisions do not require up-to-date information, but only the (recent) history. This still holds for a wide range of traditional businesses such as traditional retailing. However, advents like e-business, online telecommunications and health systems, for instance, information should be delivered as fast as possible to knowledge workers and decision systems which rely on it to react in a near real-time manner, according to the most recent data captured by an organization's information system (Inmon, 2001). In many health systems, all new data must be analyzed and coped with as a continuous data stream. It has to be immediately processed in order to trigger responses to knowledge workers and decision makers. In most cases, update delays greater than a few seconds may jeopardise the usefulness of the whole system. When using DWs in this kind of systems,

supporting real-time data warehousing (RTDW) is a vital issue. These scenarios suggest that the time between the moment operational data is recorded and the moment it is required for analytical purposes is dramatically reduced, making RTDW support a critical issue. Additionally, the real-time enterprise requires data to be always up to date.

DW refreshment (integration of new data) is traditionally performed in off-line fashion, implying that while processes for updating the data area are executed, OLAP users and applications cannot access any data. This set of activities takes place in a loading time window, usually during the night, in a daily, weekly or even monthly basis, to avoid overloading the operational OLTP source systems with the extra workload of this workflow. *Active Data Warehousing* refers a new trend where DWs are updated as frequently as possible, due to high demands of users for fresh data. *Real-Time Data Warehousing* (RTDW) is also referred for that reason in (White, 2002). The conclusions presented from a knowledge exchange network formed by major technological partners in Denmark (Pederson, 2004) refer that all partners agree real-time enterprise and continuous data availability is considered a short term priority for all business and general data-based advents.

In a nutshell, accomplishing near zero latency between OLTP and OLAP systems consists in insuring continuous data integration from the first type of systems to the other. To make this feasible, several issues need to be taken under consideration: (1) Operational OLTP systems are designed to meet well-specified (short) response time requirements, meaning that a RTDW scenario would have to cope with the overhead implied in those OLTP systems; (2) The DW tables directly related with transactional records (commonly named as fact tables) are usually huge in size, and therefore, addition of new data and consequent operations such as index updating would certainly have impact in OLAP systems' performance and data availability. Our work focuses on the DW perspective, presenting an efficient methodology for continuous data integration ETL loading process and techniques on how to adapt the DW's schemas for supporting continuous data integration and adapting OLAP queries for using all the integrated data.

The remainder of this paper is as follows. In section 2, we refer background and related work in real-time data warehousing. Section 3 explains our methodology, and in section 4 we present an experimental evaluation and demonstrate its functionality. The final section contains concluding remarks and future work.

2 RELATED WORK

The DW needs to be updated continuously to reflect source data updates. DW users are often not only interested in monitoring current information, but also in analyzing the history to predict future trends. Therefore, real-world DWs are often temporal, but their temporal support is implemented in an ad doc manner that is difficult to automate. In practice, many operational source systems are nontemporal, i.e., they store only the current state of their data, not the complete history. So far, research has mostly focused on the problem of maintaining the warehouse in its traditional periodically update setup (Yang, 2001B) (Labio, 2000). In a different line of research, data streams (Abadi, 2003) (Babu, 2001) (Lomet, 2003) (Srivastava, 2004) appear as a potential solution. Nevertheless, research in data streams has focused on topics concerning the front-end, such as on-the-fly computation of queries without a systematic treatment of the issues raised at the back-end of a DW (Karakasidis, 2005). Much of the recent work dedicated to RTDW is focused on conceptual ETL modelling (Vassiliadis, 2001) (Bruckner, 2002A) (Bouzeghoub, 1999) (Simitsis, 2005), lacking the presentation of specific extraction, transformation and loading algorithms along with their consequent OLTP and OLAP performance issues. Our contribution is the presentation of a methodology which efficiently enables continuous data integration in the DW and aims to minimize its negative impact in OLAP end user query workload executions. The issues focused in this paper concern the DW end of the system, referring how to perform the *loading* processes of ETL procedures and the DW's data area usage for efficiently supporting continuous data integration. *Extracting* and *transforming* of operational (OLTP) source systems data are not the focus of this paper.

In (Bouzeghoub, 1999) the authors describe an approach which clearly separates the DW refreshment process from its traditional handling as a view maintenance or bulk loading process. They provide a conceptual model of the process, treated as a composite workflow, but they do not describe how to efficiently propagate the date. In (Vassiliadis, 2001), authors describe ARKTOS ETL tool, capable of modeling and executing practical ETL scenarios by providing explicit primitives for capturing common tasks (such as data cleaning, scheduling and data transformations). ARKTOS uses a declarative language, offering graphical and declarative features for defining DW transformations optimizes execution of complex sequences for transformation and

cleansing tasks. Recently, (Kuhn, 2003) presents a zero-delay DW with Gong, which assists in providing confidence in the data available to every branch of the organization. Gong, a Tecco product (Binder, 2003), offers data uni/bi-directional replication between homogeneous and heterogeneous distributed DBs. Gong enables zero-delay business, assisting in daily running and decision making in the organization.

3 OUR METHODOLOGY

The main problems in maximizing functionality of a RTDW are related with ETL processes needed for integrating new data. These processes lead to two major problems: (1) a significant amount of processing time is necessary for extracting and transforming OLTP data, that affects the processing speed and availability of the OLTP source systems; (2) DW updating operations are complex and time consuming, lowering its availability to OLAP applications and end users. The major issue is how to enable continuous data integration, assuring that it minimizes negative impact in main characteristics of the system, such as:

- OLAP analytical most recent data availability;
- OLAP analytical environments' response time;
- OLTP operational systems' response time.

Therefore, we are motivated by the following requirements in real-time data warehousing:

- Maximizing the freshness of DW data by efficiently and rapidly integrating most recent OLTP data, preferably with continuous data integration;
- Minimizing OLAP instructions response time while simultaneously performing continuous data integration;

From the DW side, updating huge tables and related structures (such as indexes, materialized views and other integrated components) makes executing OLAP query workloads simultaneously with continuous data integration a very difficult task. Our methodology shows how to minimize the processing time and workload required for update processes. We also present how to adapt those OLAP workloads in order to take advantage of all the most recent data and minimize the impact caused by its integration in its execution time. Finally, our methodology allows to facilitate the DW off-line update time window, because the extraction and transformation issues are no longer present at that moment, for the data already lies within the DW and all ETL data extraction and/or transformation routines have been executed during the continuous data integration. Furthermore, the data structure of the replicated tables is exactly the same as the original DW schema. This minimizes the time window for packing the data area, since its update represents a one step process by resuming itself as a cut-and-paste operation from the temporary tables to the original ones, as we shall demonstrate further on.

Our methodology is focused on four major areas: (1) data warehouse schema adaptation; (2) ETL loading procedures; (3) OLAP query adaptation; and (4) DW database packing and reoptimization.

3.1 Adapting the DW Schema

For the area concerning DW schema adaptation, we adopt the method presented in Figure 2. By supplying empty or small sized tables without any kind of constraint or attached physical file related to it for supporting the record insertion operations inherent to continuous data integration, we guarantee the simplest and fastest logical and physical support for achieving our goals (Kimball, 2005). Transactional OLTP records should be loaded into the DW sequentially. The unique sequential identifier attribute present in each temporary table will allow discarding the rows which have been replaced for the identified OLTP transaction, as we shall demonstrate further on.

Data warehouse schema adaptation method for supporting real-time data warehousing: Creation of an exact structural replica of all the tables of the data warehouse that could eventually receive new data. These tables (referred from now on as temporary tables) are to be created empty of contents, with no defined indexes, primary key, or constraints of any kind, including referential integrity. For each table, an extra attribute must be created, for storing a unique sequential identifier related to the insertion of each row within the temporary tables.

Figure 2. Method for adapting the data warehouse's schema for supporting our real-time methodology.

3.2 ETL Loading Procedures

To refresh the DW, once the ETL application has extracted and transformed the OLTP data into the correct format for loading the data area, it shall proceed immediately in inserting that record as a new row in the correspondent temporary table, filling the unique sequential identifier attribute with the autoincremented number. This number starts at 1 for the first record to insert in the DW after executing the packing and reoptimizing technique (explained in section 3.4), and then autoincremented by one unit for each record insertion. The algorithm for continuous data integration by the ETL tool is similar to Figure 3.

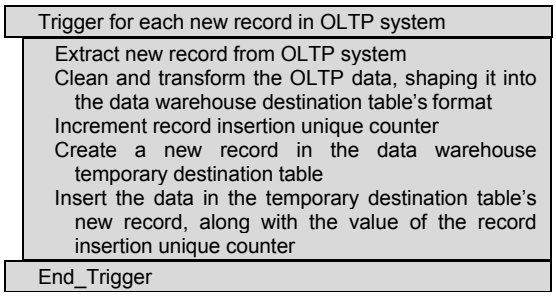

Figure 3. Continuous data integration algo in ETL tool.

3.3 OLAP Query Adaptation

Suppose a sales data warehouse has the schema illustrated in Figure 4, with two dimensional tables (Store and Customer, representing business descriptor entities) and one fact table (Sales, storing business measures aggregated from transactions). This DW stores sales value per store, per customer, per day.

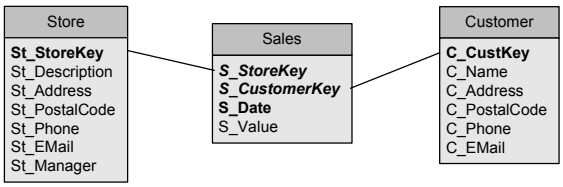

Figure 4. Sample sales data warehouse schema.

Consider the OLAP query presented in Figure 5, used for calculating the total revenue per store in the last seven days.

```
SELECT S_StoreKey,
       Sum(S_Value) AS Last7DaysSaleVal
FROM Sales
WHERE S_Date>=Date()-7
GROUP BY S_StoreKey
```

Figure 5. OLAP query for calculating the total revenue per store in last seven days.

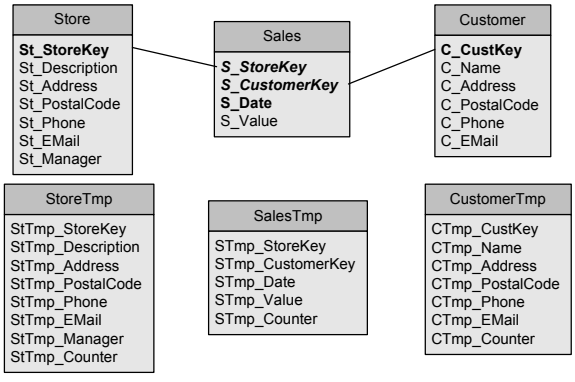

Figure 6. Sample sales data warehouse schema modified for supporting real-time data warehousing.

The modified schema for supporting RTDW based on our methodology is illustrated in Figure 6. To take advantage of our schema modification method and include most recent data in the OLAP query response, the queries should be rewritten taking under consideration the following rule: *the FROM clause should join all rows from the required original and temporary tables with relevant data, excluding all fixed restriction predicate values from the WHERE clause whenever possible.* The modification for the instruction presented in Figure 5 is illustrated in Figure 7, respecting our methods.

```
SELECT S_StoreKey,
       Sum(S_Value) AS Last7DaysSaleVal
FROM (SELECT S_StoreKey,
             S_Value FROM Sales
      WHERE S_Date>=Date()-7)
      UNION ALL
     (SELECT STmp_StoreKey,
             STmp_Value FROM SalesTmp
             WHERE STmp_Date>=Date()-7)
GROUP BY S_StoreKey
```

Figure 7. OLAP query for calculating the total revenue per store in last seven days.

It can be seen that the relevant rows from both issue tables are joined for supplying OLAP query answer, filtering the rows in the resulting dataset according to its restrictions in the original instruction.

3.4 Packing and Reoptimizing the DW

Since the data is integrated within tables without access optimization of any kind that could speed up querying, such as indexes, it is obvious that it implies a decrease of performance. Due to the volume of occupied physical space, after many insertions the performance becomes too poor to be considered acceptable. To regain performance optimization it is necessary to execute a pack routine for updating the original DW schema tables using the records in the temporary tables, and recreate those temporary tables empty of contents, along with rebuilding original tables' indexes and materialized views, so maximum processing speed is obtained once more.

For updating the original DW tables, the rows in the temporary tables should be aggregated according to the original tables' primary keys, maintaining the rows with highest unique counter attribute value for possible duplicate values, for they represent the most recent records. The time needed for executing these procedures represents the only period of time in which the DW in unavailable to OLAP tools and end users, for they need to be executed exclusively. The appropriate moment for doing this may be determined by the DW Administrator, or automatically,

considering parameters such as a fixed number of records in temporary tables, the amount of physically occupied space, or yet a predefined period of time. The definition of this moment is not object of discussion in this paper.

3.5 Final Remarks on Our Methodology

Notice that only record insertions are used for updating the DW for all related transactions in the OLTP source systems. Since this operation does not require record locking (except for the new appended record itself) nor search operations for previously stored data, the time required to do it is minimal. The issue of record locking is strongly enforced by the fact that the referred tables do not have any indexes or primary keys, implying no record locking, except for the appended record itself. Since they do not have constraints of any sort, including referential integrity and primary keys, there is no need to execute time consuming tasks such as index updating or referential integrity cross checks. This allows us to state that the data update time window is minimal for insertion of new data, maximizing its availability, and contributing to increase the DW's global availability and minimize negative impact in its performance.

The amount of "information buckets" which the data passes through in the ETL Area is also minimal, for temporary storage is almost not needed. Instead of extracting a large amount of OLTP data, what happens in "traditional" DW bulk loading, the volume of extracted and transformed real-time data is very reduced (few dozen bytes), since it consists of only one transaction per execution cycle, so we may assume that the extraction and transformation phase will be cleaner and more time efficient.

4 EXPERIMENT EVALUATION

Recurring to TPC-H decision support benchmark (TPC-H) we tested our methodology creating 5GB, 10GB and 20GB size DWs in ORACLE 10g RDBMS (Oracle, 2005). We also tested the system's response executing 1, 2, 4, 8 and 16 simultaneous query workloads to see how it reacted according to the number of simultaneous users executing those workloads. We used an Intel Celeron 2.8GHz with 2GB of SDRAM and a 7200rpm 160GB hard disk. The modified schema according to our methodology can be seen in Figure 8. Tables *Region* and *Nation* are not included as temporary tables because they are fixed-size and therefore do not receive new data.

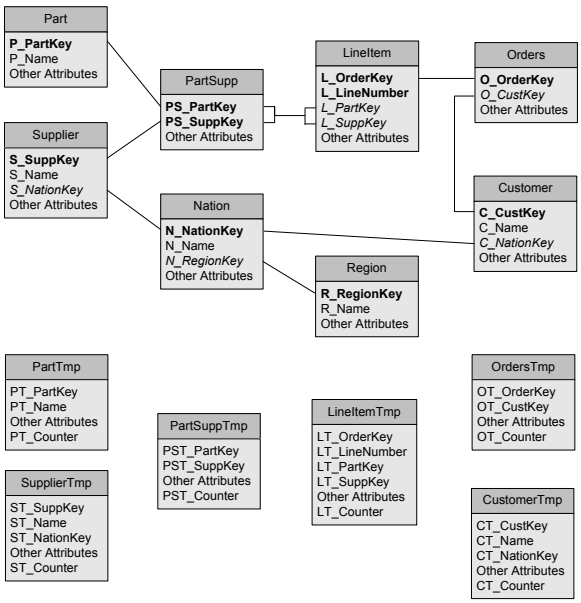

Figure 8. TPC-H schema modified for supporting RTDW.

The selected query workloads, TPC-H queries 1, 8, 12 and 20 (TPC-H), were executed in random order for each simultaneous user. The time interval between transactions (Transac. Interval) for each scenario is illustrated in tables 1 to 3. Each new transaction represents insertion of an average of four records in *LineItemTmp* and one row in each of the other temporary tables, continuously integrated for a period of 8 hours. Supporting RTDW capability in all scenarios is somewhat between 6.9% and 28.6% of query workload execution time, as shown in figures 9 to 11, reporting percentage of workloads execution overtime using RTDW, relatively to execution against standard workload without continuous integration.

Table 1. TPC-H 5GB data warehouse transaction real-time integration characteristics.

	TPC-H 5GB Data Warehouse		
	Scenario A	Scenario B	Scenario C
# Transactions	3.072	6.205	12.453
Transac. Interval	9,38 sec	4,64 sec	2,31 sec

Table 2. TPC-H 10GB data warehouse transaction real-time integration characteristics.

	TPC-H 10GB Data Warehouse		
	Scenario A	Scenario B	Scenario C
# Transactions	6.192	12.592	25.067
Transac. Interval	4,65 sec	2,29 sec	1,15 sec

Table 3. TPC-H 20GB data warehouse transaction real-time integration characteristics.

	TPC-H 20GB Data Warehouse		
	Scenario A	Scenario B	Scenario C
# Transactions	12.416	25.062	50.237
Transac. Interval	2,32 sec	1,15 sec	0,57 sec

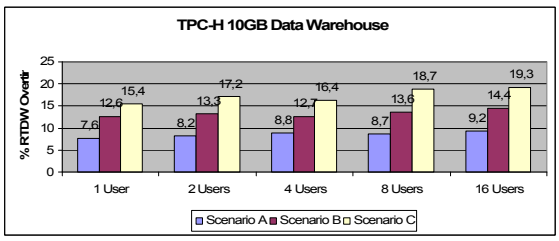

Figure 9. TPC-H 5GB DW overtime percentages

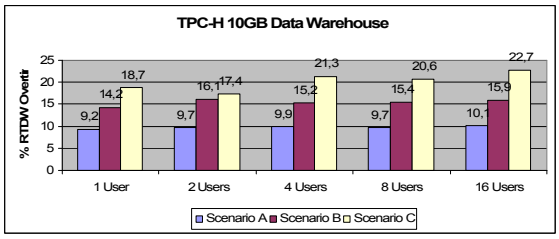

Figure 10. TPC-H 10GB DW overtime percentages

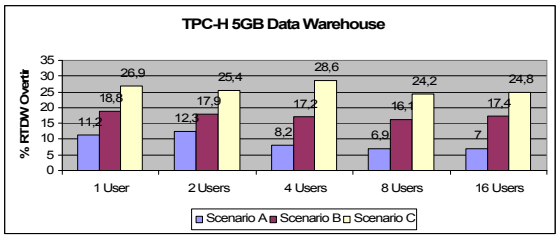

Figure 11. TPC-H 20GB DW overtime percentages

5 CONCLUSIONS

This paper refers the requirements for RTDW and presents a methodology for supporting it by enabling continuous data integration while minimizing impact in query execution on the DW end. This is done by data structure replication and adapting query instructions to take advantage of the new real-time data warehousing schemas.

We have shown its functionality, recurring to a simulation using the TPC-H benchmark, performing continuous data integration at various time rates against the execution of various simultaneous query workloads, for DWs with different scale sizes. All scenarios show that it is possible to achieve real-time data warehousing performance in exchange for an average increase of ten to thirty percent in query execution time. This should be considered the price to pay for real-time capability within the DW.

As future work we intend to develop an ETL tool integrating this methodology. There is also a huge space of research for optimizing query instructions used.

6 REREFENCES

Abadi, D. J., Carney, D., et al., 2003. *"Aurora: A New Model and Architecture for Data Stream Management"*, The VLDB Journal, 12(2), pp. 120-139.

Babu, S., Widom, J., 2001. *"Continuous Queries Over Data Streams"*, SIGMOD Record 30(3), pp. 109-120.

Binder, T., 2003. *Gong User Manual*, Tecco Software Entwicklung AG.

Bouzeghoub, M., Fabret, F., Matulovic, M., 1999. *"Modeling Data Warehouse Refreshment Process as a Workflow Application"*, Intern. Workshop on Design and Management of Data Warehouses (DMDW).

Bruckner, R. M., List, B., Schiefer, J., 2002 A. *"Striving Towards Near Real-Time Data Integration for Data Warehouses"*, International Conference on Data Warehousing and Knowledge Discovery (DAWAK).

Chaudhuri, S., Dayal, U., 1997. *"An Overview of Data Warehousing and OLAP Technology"*, SIGMOD Record, Volume 26, Number 1, pp. 65-74.

Inmon, W. H., Terdeman, R. H., Norris-Montanari, J., Meers, D., 2001. *Data Warehousing for E-Business*, J. Wiley & Sons.

Karakasidis, A., Vassiliadis, P., Pitoura, E., 2005. *"ETL Queues for Active Data Warehousing"*, IQIS'05.

Kuhn, E., 2003. *"The Zero-Delay Data Warehouse: Mobilizing Heterogeneous Databases"*, International Conference on Very Large Data Bases (VLDB).

Labio, W., Yang, J., Cui, Y., Garcia-Molina, H., Widom, J., 2000. *"Performance Issues in Incremental Warehouse Maintenance"*, (VLDB).

Lomet, D., Gehrke, J., 2003. *Special Issue on Data Stream Processing*, IEEE Data Eng. Bulletin, 26(1).

Oracle Corporation, 2005. www.oracle.com

Pedersen, T. N., 2004. *"How is BI Used in Industry?"*, Int. Conf. on Data W. and Knowledge Discov. (DAWAK).

Simitsis, A., Vassiliadis, P., Sellis, T., 2005. *"Optimizing ETL Processes in Data Warehouses"*, International Conference on Data Engineering (ICDE).

Srivastava, U., Widom, J., 2004. *"Flexible Time Management in Data Stream Systems"*, PODS.

TPC-H decision support benchmark, Transaction Processing Council, www.tpc.com.

Vassiliadis, P., Vagena, Z., Skiadopoulos, S., Karayannidis, N., Sellis, T., 2001. *"ARKTOS: Towards the Modelling, Design, Control and Execution of ETL Processes"*, Information Systems, Vol. 26(8).

White, C., 2002. *"Intelligent Business Strategies: Real-Time Data Warehousing Heats Up"*, DM Preview, www.dmreview.com/article_sub_cfm?articleId=5570.

Yang, J., 2001. *"Temporal Data Warehousing"*, Ph.D. Thesis, Dpt. Computer Science, Stanford University.

Yang, J., and Widom, J., 2001B. *"Temporal View Self-Maintenance"*, 7th International Conference on Extending Database Technology (EDBT).

Zurek, T., Kreplin, K., 2001. *"SAP Business Information Warehouse – From Data Warehousing to an E-Business Platform"*, (ICDE).