Methodologies for Implementing Real Time Data Warehouses

Ricardo Santos[1], Jorge Bernardino[2]

[1]Departamento de Engenharia Informática – FCT – Universidade de Coimbra
[2]Dept. Engenharia Informática e de Sistemas – ISEC – Instituto Politécnico de Coimbra
e-mail: lionsoftware@iol.pt, jorge@mail.isec.pt

Abstract. *A data warehouse (DW) provides information for analytical processing, decision making and data mining tools. This information is updated periodically from transactional systems. Traditional DW systems have static structures of their data schemas and relationships, and therefore are not prepared to support the dynamics of real-time live data processing. As the concept of real time enterprise evolves, the synchronism between transactional data and DW, statically implemented, has been reviewed. For these purposes, Real-Time Data Warehouses (RTDW) seem to be very promising. This paper presents methodological indications for implementing RTDW, in which transactional data sources are available through common standard database access, allowing to minimize the time needed to accomplish extraction, transformation and loading (ETL) processes of that data, as well as its loading into the DW. The main method presented consists on using structural replicas of all fact tables without primary keys or index files, adapting those replica's data structures for housing, in real-time, all insert, edit and delete operations (data transactions) that occur within operational systems databases. This is accomplished using only append record operations towards those fact table replicas, allowing to minimize processing time, record locking and concurrency data access problems, both in transactional systems and the DW. Concurrently, this allows maintaining DW availability and keeping OLAP tools functioning properly, providing the most recent business data.*

1. Introduction

A data warehouse (DW) provides information for decision making and data mining tools. A DW can be seen as a set of summarized business data obtained periodically from transactional systems (OLTP – On-Line Transaction Processing) and used by analytical applications (OLAP – On-Line Analytical Processing) with different user requirements. Usage of OLAP tools retrieving DW data is the usual process for obtaining decision making information [4]. This data source consists in a set of tables distributed according to a star schema [6], whose records are updated periodically (usually on a daily or weekly basis). This implies that DW data is never up-to-date, because OLTP records saved between those updates are not included in its data area, thus getting excluded from OLAP tools supplied results. This absence of the most recent information has been considered as not relevant and not critical for traditional decision making. However, today's emerging enterprise advents such as e-commerce, m-commerce and health care systems, for instance, possess the need for real-time decision making, where it is necessary to react near real-time to transactions. They create a demand for valid, relevant, accurate and up-to-date information to support decision making that must be delivered as fast as possible to knowledge workers and decision makers, who rely on it. This necessity appears to present a growing tendency [7], making it imperative to evolve data warehouses towards their real-time integration with transactional operational systems (OS).

As shown in Figure 1, transactional data comes from OS, passing through transformations in the ETL Area and afterwards updates the Data Area, which contains the adequate data structure for supporting decision making, available for DW users through usage of OLAP tools.

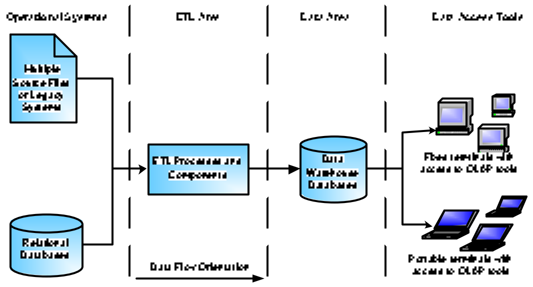

Operational Systems
ETL Area
Data Area
Data Access Tools
Multiple Source Files of Legacy Systems
Relational Databases
ETL Processes and Components
Data Warehouse Databases
Fixed terminals with access to OLAP tools
Portable terminals with access to OLAP tools
Data Flow Orientation

Fig. 1. Generic Data Warehouse Architecture

The OLTP systems are usually databases capturing all transactions that occur within an organization in a certain business area [6]. The ETL Area is an intermediate data housing and processing area which is not user-available, in which extracted transactional data goes through transformation processes such as error, inconsistency and redundancy correction, standard formatting, etc. Afterwards, the data is adequately moulded for Data Area updating purposes at the defined business detail level. If this update cannot be immediately executed, the data may stay housed in the ETL Area, usually named as Staging Area. The Data Area houses all relevant and accurate historical updated business information available to DW users. The Data Area is unavailable while being updated (alias loaded), because it has to be offline.

This paper refers ETL processes and data schema changes aiming to achieve real-time functionality, i.e., managing to continuously receive live transactional data and maintain high availability and processing performance, possessing up-to-date historical data at all times. This is done presenting methodological indications on how to implement those ETL processes in real-time, without significant prejudice of OS processing speed. Another goal is to maximize the DW's availability.

The remainder of this paper is organized as follows: In section 2, we provide an overview of related work in Real-Time Data Warehouses (RTDW) implementations and draw our main contributions. Section 3 characterizes a RTDW and its requirements. Section 4 proposes our methodology to implement RTDWs and section 5 describes modifications of the star schema and data updating in order to maximize RTDW availability. Section 6 presents considerations on how to use the RTDW for taking advantage of our proposal. Finally, in section 7 we conclude the paper and point out some future perspectives.

2. Contribution and Related Work

Although the database research community has studied extensively data warehousing (e.g. [4] and [14]), most of that research ignores temporal aspects in the updating area. DW users are often monitoring not only current information, but also analyzing the history in order to predict future trends. Therefore, real-world DWs are often temporal, but their temporal support is implemented in an ad hoc manner, difficult to automate. In practice, many OLTP systems are nontemporal, because they store only current data, not the complete history. Temporal DWs address the issue of supporting temporal information efficiently in DW systems [15]. Keeping them up-to-date is complex, because temporal views may need updating, not only when source data changes, but also as time progresses and these two dimensions of change interact. In [16], the authors present efficient techniques (e.g. temporal view self-maintenance) for maintaining a DW without disturbing source operations. A related challenge is supporting large-scale temporal aggregation operations in DW [17]. However, most research has focused on performance issues rather than higher-level issues, such as conceptual modeling [9]. In [2], 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 do not describe how to propagate the data efficiently. Theodoratos and Bouzeghoub discuss in [13] data currency quality factors in data warehouses and propose a DW design taking them in consideration.

Recently, a zero-delay DW is described with Gong [8], which assists in providing confidence in data available to every branch of the organization. Gong is a Tecco product [1] offering uni or bi-directional data replication between homogeneous and/or heterogeneous distributed databases. Gong's database replication enables zero-delay business for assisting daily decision making within the organization.

Our contribution in this paper is the characterization of RTDWs and identification of methods and techniques to support them. The prerequisite for a RTDW is a continual, near real-time live data propagation. As a solution for this problem, we propose methods which allow including real-time live transactional data in the DW without significant temporal negative aspects on its execution. Another goal is to make this phase faster, maximizing the availability of previously recorded data for querying and, consequently, the DW's global availability, which provides an efficient way of performing, controlling and monitoring the ETL processing tasks.

3. Requirements for Real-Time Data Warehouses

A RTDW aims for decreasing the time spent in obtaining accurate up-to-date decision making information and tries to attain zero latency between the cause and effect of a business decision, enabling analysis across corporate data sources. This effectively closes the gap between business intelligence systems and business processes. Business requirements may be different across various industries, but the underlying information requirements are similar – integrated, current, relevant and immediately accessible.

Transforming a standard DW using batch loading during update windows (where data access is not allowed) to a zero or near zero-latency analytical environment providing current live data involves addressing of various issues in order to enable (near) real-time dissemination of new information across an organization.

The requirements for this kind of analytical environment introduce a set of service level agreements that go beyond what is typical in a traditional DW. These service levels focus on three basic characteristics [3]:

Continuous data integration, enabling (near) real-time capturing and loading live data from different operational sources;

Active decision engines that can make recommendations or (rule-driven) strategic decisions for routine, analytical decision tasks encountered [11, 12];

Highly available analytical environments based on an analysis engine able to consistently generate and provide access to current business analyses at any time, not restricted by time loading windows, typical in the common batch approach.

An in-depth discussion of these characteristics from the analytical viewpoint is given in [3]. RTDWs try to represent history as accurately and up-to-date as possible (to enable strategic decision support).

Providing access to an accurate, integrated, consolidated view of the organizations' information and helping to deliver real-time information to RTDW users requires efficient ETL techniques enabling continuous data integration, which is the focus of this paper. Combining highly available systems with active decision engines allows near real-time information dissemination for DWs. Cumulatively, this is the basis for zero latency analytical environments [3].

4. Methods for Implementing Real-Time Data Warehouses

The main problems in maximizing the functionality of a RTDW are related with ETL processes needed for integrating new data [6, 10]. These processes, necessary for recording information in the OS and, at the same time, updating the Data Area, lead to two potential problems: on one hand, a significant amount of time is necessary for extracting and transforming OLTP data that affects the processing speed and availability of the OS; on the other hand, Data Area updating operations are complex and time consuming, lowering the DW's availability.

This paper presents a solution for supporting a functional RTDW, maintaining availability and processing speed levels for both OLTP and OLAP systems considered satisfactory for all users. In this section we will explain the methods to employ, through illustrated considerations and simple examples from a traditional DW.

The proposed methodology assumes that transactional systems use standard databases for housing information [5, 6]. Using this kind of data source, we assume that time for data extraction, transformation and cleaning can be considered minimal. Typically, this type of processing is extremely fast when compared to the execution of the same methods against other types of data sources. Furthermore, as the DW itself consists in a set of relational tables, the techniques to use for this purpose are extremely simple and direct, reducing the proposed solution's complexity.

In the remainder of the paper we consider as example an OS based on product sales, which stores transactional information in a database with tables referring to subjects such as Stores, Invoices, Invoice Details, Customers, Products, among others.

Suppose our aim is to obtain a DW that allows retrieving decision making information based on the quantity and value of daily product sales, for each customer, per store. We can consider as a possible Data Area solution a star schema similar to what is shown in Figure 2, which consists of four dimension tables (Customers, Products, Stores and Time) that represent business descriptors [6] and one fact table (Sales) which contains the business measures [6].

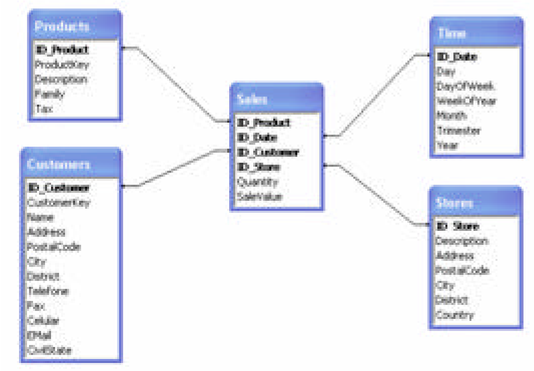


Fig. 2. Data Area star schema for the Data Warehouse

Notice that one of the factual attributes refers to the total sale value instead of the unitary price. This is necessary for attribute's addictivity purposes. This characteristic for factual attributes is considered relevant and extremely important, as we shall show further ahead in sections 5 and 6. In the next sections, we refer considerations and methods to use in each phase for implementing and supporting a RTDW, focused on achieving two major goals: maximizing Data Area availability and maintaining processing performance of OLTP systems at a high level.

4.1. Extracting and transforming transactional systems data

Assuming that OLTP systems use standard databases [5, 6], general use of standard industry interfaces, drivers (such as ODBC, EDA/SQL, gateways, etc) and owner scripts is allowed for acquiring this data. SQL triggers could be employed to accomplish this extraction, reacting to events occurred within OS data and initiating ETL processes according to those events relevance towards the DW.

The extracted data usually needs to be corrected and transformed before updating the DW's Data Area [5]. Since we pretend to obtain this data in real-time, the time gap between recording OS transactions and their extraction by ETL processes is minimal, occurring at the same time, which somewhat reduces error probability. We can also assume that the amount of intermediate “information deposits/buckets” which the data passes through in the ETL Area is also minimal, for temporary storage is not needed. Furthermore, instead of extracting a considerable amount of OS data, which is what happens in the “traditional” bulk loading DW, the volume of information extracted and transformed in real-time is extremely reduced (representing commonly a few dozen bytes), since it consists of only one transaction per execution cycle. All this allows assuming that the extraction and transformation phase will be cleaner and more time efficient.

4.2. Loading the transformed data into the Data Area

To maximize the Data Area’s availability, we must analyze the characteristics of each object that composes it which may cause any impact. Thus, we consider its following nuclear objects: dimension and fact tables, indexes and materialized views.

To deal with the mentioned problems in Data Area updating, the main method proposed consists on creating a structural replica of each fact table, initially empty of contents, with no defined index files or primary key, connected to the same dimension tables as the original fact tables (as shown in Figure 3). Each of these replicas, referred to as temporary fact table from this moment on, will be used for real-time storage of live OS transactions, resumed to the DW’s defined business granularity detail level, according to the original fact table from which it was obtained. The next section explains how to make use and take advantage of this for supporting a RTDW.

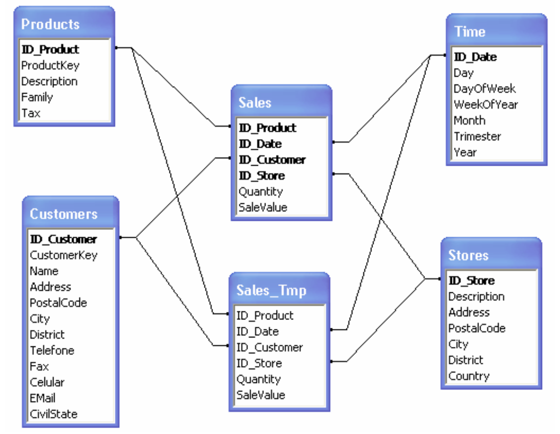


Fig. 3. Star schema for the Data Area of the Real-Time Data Warehouse

5. New Star Schema Implications

We will now explain how to deal with each of the Data Area’s nuclear objects when OS data changes occur, in order to achieve our RTDW goals.

5.1. Dimension Tables

OS records changes or insertions referring to dimension tables are usually few, and record erasure is even less usual or inexistent [6].

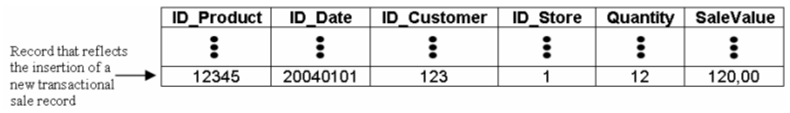
Deleting a transactional record in the OS which reports itself directly to a related record in the RTDW's Data Area should not imply any operation within the last, in order to maintain historical data.

The insertion or modification of a OS record directly related with a RTDW's dimension should trigger the creation of a new record with updated data in that dimension table, but maintaining any previously stored record for historical purposes.

5.2. Fact Tables

These tables contain the business measures, representing consumed OS transactions at the defined business granularity level. Given their usual huge size, these tables and their indexes are the main problem of data processing. Updating fact tables implies accessing and locking large amounts of information involving (at least) tables and complex index files, requiring a considerable processing and time consuming effort. Pretending a real-time transactional integration, the volume of updates will be massive and represents a problem. We will describe next the procedures to adopt for maximizing the availability of these tables, according to the type of operations that take place in OS.

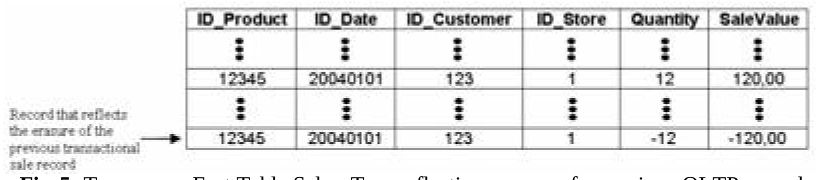
When a new OS record is stored, a new record related to that transaction should be added in the temporary fact table to which it concerns. Supposing that April 1st of year 2004, 12 units of a product identified by ID_Product 12345 were sold to a customer identified with ID_Customer 1, for 10 monetary units each. Immediately after recording this in the OS, it should trigger adding a new correspondent temporary sales fact table record, as shown in Figure 4.


Record that reflects the insertion of a new transactional sale record

ID_Product	ID_Date	ID_Customer	ID_Store	Quantity	SaleValue
⋮	⋮	⋮	⋮	⋮	⋮
12345	20040101	123	1	12	120,00

Fig. 4. Temporary Fact Table Sales_Tmp reflecting a new OLTP sale record

If the referred sale's transactional record is erased, a new record should be added in the temporary fact table recording the identifying (key) attributes and filling the factual attributes with the opposite arithmetic value of those recorded previously. An example of this can be viewed in Figure 5.

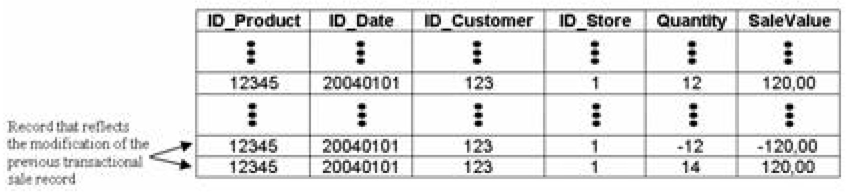

Record that reflects the erasure of the previous transactional sale record

ID_Product	ID_Date	ID_Customer	ID_Store	Quantity	SaleValue
⋮	⋮	⋮	⋮	⋮	⋮
12345	20040101	123	1	12	120,00
⋮	⋮	⋮	⋮	⋮	⋮
12345	20040101	123	1	-12	-120,00

Fig. 5. Temporary Fact Table Sales_Tmp reflecting erasure of a previous OLTP record

Grouping all the records by product, by customer, per store, per day, and using addictive functions such as Sum for the factual attributes, allows obtaining the true values referring to the sales that occurred.

Recurring to the same addictive characteristic, when a previously recorded transaction is modified, we should execute what was mentioned before to eliminate those previous values, and then make an insertion containing the new values. For instance, if the referred sale was modified by altering the sold quantity from 12 to 14, then we should eliminate the original sale, annulling it with a new record recurring to addictiveness, and subsequently insert a fact record with the correct values, according to what is illustrated in Figure 6.


Record that reflects the modification of the previous transactional sale record

ID_Product	ID_Date	ID_Customer	ID_Store	Quantity	SaleValue
⋮	⋮	⋮	⋮	⋮	⋮
12345	20040101	123	1	12	120,00
⋮	⋮	⋮	⋮	⋮	⋮
12345	20040101	123	1	-12	-120,00
12345	20040101	123	1	14	120,00

Fig. 6. Temporary Fact Table Sales_Tmp reflecting modification of a previous OLTP record

Once the factual attribute's addictivity is guaranteed, the temporary fact table record grouping at the defined granularity level and summing each of its factual attributes will represent all transactions with accuracy.

Notice that only record insertions are used for updating the fact table for all related occurrences of inserting, modifying and deleting records in the OS. Since this type of operation does not require any record locking in that table (except for the appended record itself) nor search operations for previously stored data, the time necessary to accomplish this is minimal. The issue of record locking is strongly enforced by the fact that the referred table does not have any indexes or primary key, implying absolutely no record locking, except for the appended record itself. This allows maximizing availability for both dimensional and factual data, contributing to effectively increase the RTDW's global availability.

By adopting this method for updating RTDW tables it is now easy to understand why it is crucial to guarantee that the addictivity condition referred to is satisfied. It will also be necessary for the Data Area's packing process exposed in section 6.2, essential for updating the original fact tables.

5.3. Materialized Views

Materialized views should also contain only addictive factual attributes. Materialized views that consider usage of non-addictive functions (such as an average function) should be redesigned to obtain those values from addictive functions – the average function result can be calculated by dividing a summing function result by a counting function result, for example.

5.4. Indexes

All fact and dimension table indexes should be rebuilt whenever the Data Area packing routine is executed.

6. Data Area of the Real-Time Data Warehouse

We will now present considerations in order to take advantage of the new schema archit ectured for the RTDW's Data Area.

6.1. Using the Data Area

Since there are now two fact tables for each factual subject, a join of both tables is needed to obtain the most recent RTDW's factual data, implying that all data extraction methods need to consider this to obtain real up-to-date decision making information.

What also needs to be considered is that it might be necessary to group its records according to the original fact table's granularity, since the temporary fact table does not have any defined primary key.

These usage principles are the same in what concerns materialized views.

6.2. Packing the Data Area

As new data is included in the temporary fact table, its functionality gets affected, for the increase of its size implies a decrease of its processing speed. The increasing number of records also has negative impact in OLAP query processing including the temporary fact table, due mainly to having no index files that could speed up the querying. When a significant volume of recorded data in the temporary fact table is attained which negatively affects OS performance or reveals to gain significance in OLAP querying time, it becomes necessary to execute the packing routine. This routine consists on using the temporary fact table to update the origi nal fact table. Afterwards, the temporary fact table is rebuilt with empty contents, so maximum processing speed is obtained once more.

When packing the Data Area, logging systems must be shut down, disabling user access. The appropriate moment for doing this is determined by the DW Administrator, or automatically, taking under consideration parameters such as defining a specific number of records as storage limit in the temporary fact table, the amount of disk space occupied by each temporary fact table, or yet a predefined period of time between packing.

The choice of the appropriate moment for packing should represent the best possible compromise related to its frequency of execution and time taken away from the DW's availability, for it represents the only time window in which it is unavailable for users. This evaluation will depend on physical, logical and practical characteristics inherent to the OS and DW themselves and is not object of discussion in this paper.

The packing method may be accomplished by executing the following steps:

1) Group all temporary fact table records according to the original fact table's primary key. In the presented RTDW, this represents grouping records per product, per day, per customer, per store, and summing the recorded values for each factual attribute.

This allows obtaining one record per factual primary key attribute values combination, corresponding to the granularity detail level of the original fact table. Once more, the importance of having addictive factual attributes becomes obvious, or else we cannot obtain the factual values through use of aggregate sum functions for the grouped records.

2) Use the grouped records for updating the original fact table, using a primary key match. Previously stored records in the original fact table whose primary key value is coincident with the same attribute combination in the temporary fact table are updated by adding factual data captured by the temporary table. Each grouped record with a related original fact table primary key attribute combination value that does not exist within the original fact table should be replicated in it. This inserts records in the original fact table that represent new transactions relatively to the prior packing moment.

3) Finally, empty the contents of the temporary fact table, thus maximizing its operational processing performance again and, consequently, the RTDW's also. Besides rebuilding indexes after the packing routine is completed, it would also be adequate to reconstruct all related aggregates and materialized views.

7. Conclusions and Future Work

Although a very simple example was used for illustrating our ideas (using a data warehouse consisting on a unique star schema), it can easily be extrapolated to more complex data warehouses that allow applying the proposed methods and techniques.

The data updating philosophy for efficiently supporting RTDWs seeks to use the least critical time and concurrency data manipulation operations, suppressing record searching, editing and deleting, preferring record insertion in tables without indexes and primary keys, replicated from original DW data structures. This can be accomplished efficiently using SQL triggers towards OLTP database events.

As future developments we would like to apply the proposed methodological indications to partitioned data warehouses. We are also considering implementing an ETL tool based on the considerations presented in this paper and use a real transactional system or a benchmark to test our proposal.

References

[1] Binder, T., Gong User manual, 2003/06/24, Tecco Software Entwicklung AG

[2] Bouzeghoub, M., Fabret, F., Matulovic, M.: Modeling Data Warehouse Refreshment Process as a Workflow Application. Intl. Workshop DMDW'99, Heidelberg, Germany, June 1999.
[3] Bruckner, R. M., Tjoa, A M.: Capturing Delays and Valid Times in Data Warehouses – Towards Timely Consistent Analyses. Journal of Intelligent Information Systems (JIIS), 19:2, 169-190, 2002.
[4] Chaudhuri, S., Dayal, U.: An Overview of Data Warehousing and OLAP Technology. Sigmod Record, Volume 26, Number 1, pp. 65-74, March 1997.
[5] Gatziu, S., Vavouras, A.: Data Warehousing: Concepts and Mechanisms, Informatik/Informatique 1, 1999.
[6] Kimball, R. "The Data Warehouse Toolkit – Practical Techniques for Building Dimensional Data Warehouses", Ed. J. Wiley & Sons, Inc, 1996.
[7] Kimball, R., Merz, R. "The Data Webhouse Toolkit", Ed. J. Wiley & Sons, Inc, 2000.
[8] Kühn, E.: The Zero-Delay Data Warehouse: Mobilizing Heterogeneous Databases. VLDB 2003, Berlin, Germany, pp.1035-1040
[9] Pedersen, T. B., Jensen, C. S., Dyreson, C. E.: A Foundation for Capturing and Querying Complex Multidimensional Data. Information Systems, 26(5), pp. 383-423, 2001.
[10] Scalzo, B. "Oracle DBA – Guide to Warehousing and Star Schemas", The Prentice Hall Ptr Oracle Series, Prentice Hall, 2003.
[11] Schrefl, M., Thalhammer, T.: On Making Data Warehouses Active. Proc. of the 2nd Intl. Conf. DaWaK, Springer, LNCS 1874, pp. 34–46, London, UK, 2000.
[12] Thalhammer, T., Schrefl, M., Mohania, M.: Active Data Warehouses: Complementing OLAP with Analysis Rules. Data & Knowledge Engineering, Vol. 39(3), pp. 241–269, 2001.
[13] Theodoratos, D., Bouzeghoub, M.: Data Currency Quality Factors in Data Warehouse Design. Intl. Workshop DMDW'99, Heidelberg, Germany, June 1999.
[14] Widom, J.: Research Problems in Data Warehousing. CIKM 1995, pp. 25-30, 1995.
[15] Yang, J.: Temporal Data Warehousing. Ph.D. Thesis, Department of Computer Science, Stanford University, 2001.
[16] Yang, J., Widom, J.: Temporal View Self-Maintenance. Proc. of the 7th Intl. Conf. EDBT2000, Springer, LNCS 1777, pp. 395–412, Konstanz, Germany, 2000.
[17] Yang, J., Widom, J.: Incremental Computation and Maintenance of Temporal Aggregates. In Proceedings of the 17th International Conference on Data Engineering (ICDE), Heidelberg, Germany, pp. 51-60, IEEE CS Press, 2001.