24/7 Real-Time Data Warehousing: A Tool for Continuous Actionable Knowledge

Ricardo Jorge Santos
CISUC – DEI – FCTUC
University of Coimbra
Coimbra, Portugal
lionsoftware.ricardo@gmail.com

Jorge Bernardino
CISUC – DEIS – ISEC
Polytechnic Institute of Coimbra
Coimbra, Portugal
jorge@isec.pt

Marco Vieira
CISUC – DEI – FCTUC
University of Coimbra
Coimbra, Portugal
mvieira@isec.pt

Abstract - **Technological evolution has redefined many business models. Many decision makers are now required to act near real-time, instead of periodically, given the latest transactional information. Decision-making occurs much more frequently and considers the latest business data. Since data warehouses (DWs) are the core of business intelligence, decision support systems need to deal with 24/7 real-time requirements. Thus, the ability to deal with continuous data loading and decision support availability simultaneously is critical, for producing continuous actionable knowledge. The main challenge in this context is to efficiently manage the DW's refreshment, when data sources change, to recapture consistency and accuracy with those sources, while maintaining OLAP availability and database performance. This paper proposes a simple, fast and efficient solution based on database replication and temporary tables to change a traditional enterprise DW into a real-time DW, enabling continuous data loading and OLAP availability on a 24/7 schedule. Experimental evaluations using a real-world DW and the TPC-H decision support benchmark show its advantages and analyze its impact in OLAP performance.**

Keywords: Real-time data warehousing, ETL, Continuous data loading, OLAP, Availability, Actionable knowledge

I. INTRODUCTION

Data warehouses (DWs) are the main decision-support assets in many business fields. Using data mining on DWs enables producing business knowledge. DWs typically store the complete business history, producing decision support information by using On-Line Analytical Processing (OLAP) tools. Transactional systems (TS) store only current information for supporting business transactions. Extraction, Transformation and Loading (ETL) tools are used for extracting TS business data, transforming and cleansing it into an analytical format, and finally loading it into the DW database(s). It has been well accepted that DW databases have been updated periodically, typically in a daily, weekly or even monthly basis [12]. This update policy was defined as common practice due to two major assumptions:

1) Since it stores the whole business history, it has huge tables and complex optimization data structures (such as indexes, partitions, etc). Executing data updating procedures continuously and frequently (similar to TS), while simultaneously executing OLAP, would create huge bottlenecks, due to time and resource consumption implied in executing those actions simultaneously; and

2) Business decisions were made on a daily, weekly or monthly basis, dismissing recent real-time business data.

This type of update policy meant that DW data was never up-to-date; transactional data saved between those updates was not included in its databases, excluding the most recent transactional data from OLAP results. Today, technological evolution has pushed forward and redefined business models operating on a 24/7 schedule, requiring decision makers to act almost immediately after the original transactional data is written, instead of making decisions on a daily or weekly basis [3, 26]. A paper by Oracle [2] states that the "Holy Grail" of data warehousing is to support analysis at the speed of the business, affirming the concept of *"on-time information"* as having information available whenever it is needed. This means that decision-making occurs much more frequently and continuously.

DW refreshment (integration of new data) has traditionally been done in an offline fashion. This meant that while updating the DW, OLAP applications could not access any data. Since DWs are currently the core back-end of decision support and Business Intelligence (BI) systems [13], they need to deal with 24/7 real-time enterprise requirements, coping with both continuous data integration and decision support availability in these business scenarios. In some scenarios, update delays greater than a few seconds or minutes may jeopardize the whole system's usefulness.

Thus, the 24/7 real-time enterprise requires a decision support tool able to continuously acquire and process the latest business data, supplying business knowledge at all times for aiding decision making that can take effect while the business transactions occur. For example, airlines and government agencies need to be able to analyze the most current information when detecting suspicious passengers or potentially illegal activity. Fast-paced changes in financial markets may make personalized suggestions on a stockbroker's website obsolete by the time they are viewed. Effectively, supplying decision support in useful time is what we consider *actionable knowledge*.

Our work focuses on the DW perspective, and for that reason, our contribution is a simple and easy-to-implement solution that enables efficient 24/7 continuous data loading in DW, while simultaneously keeping the database available for OLAP execution at all times, *i.e.*, enabling everlasting availability. Much research on Real-Time DW (RTDW) has been published, but it has mainly focused on ETL workflow or data integration issues, lacking OLAP performance and availability aspects. We focus on *Loading* procedures, using a middleware tool to accomplish our aims, which uses fast

data insertion techniques for minimizing impact in database performance. Issues on *Extracting* and *Transforming* transactional data are not within the scope of this paper. Our tool acts as a data manager between the DW ETL tool and its databases, accomplishing three functions: managing the DW database schemas, updating the databases, and aiding OLAP querying against them.

The remainder of this paper is organized as follows. In Section 2, we present requirements for 24/7 RTDW and explain our methodology. Section 3 presents an experimental evaluation of our proposal. Section 4 presents background and related work in RTDW, and the final section summarizes our conclusions and actions for future work.

II. Designing a 24/7 Real-Time Data Warehouse

One of the most difficult parts of building any DW is the process of extracting, transforming, cleansing, and loading the data from the source system. According to [12], 70% of the warehouse implementation and maintenance effort is spent on ETL procedures. Real-time ETL brings additional challenges. Almost all ETL tools and systems operate in batch mode, assuming data becomes available as some sort of extract file on a certain schedule, usually nightly, weekly, or monthly. Then, the system transforms and cleanses the data and loads it into the DW. This process typically involves DW downtime, so users are unable to access it while the load takes place. In what concerns DW, near zero latency between TS and OLAP systems consists on ensuring continuous data integration in the DW and guaranteeing that decision makers may request decision support information at all times [3]. Therefore, in RTDW, we need to cope with two radical data state changes:

1) Performing continuous data update actions, which should mostly concern row insertions;

2) These updates must be performed simultaneously with OLAP execution, which – due to its new real-time nature – will probably be requested much more often.

This implies that the DW should be able to cope with database access failure, involving fault tolerance procedures to ensure its continuous availability. Given the purpose of the 24/7 real-time DW, besides the needs and issues behind traditional periodic updated DW and used ETL tools, the main new requirements that must be considered are:

- Loading the new data into the DW databases as soon as possible, as quickly and as frequently as possible;
- Minimizing the impact in OLAP performance due to the simultaneous execution of data update procedures;
- Assuring OLAP is always possible and available on a 24/7 schedule/strategy.

Figure 1 shows a classic DW architecture, illustrating the data flow between transactional operational systems and the DW databases and OLAP users and applications. The Database Administrator (DBA) is responsible for managing and monitoring the ETL processes, as well as the databases.

To use our tool, the DW database schemas need to be adapted, using a replica of each fact table, empty of contents and without any optimization structure such as primary keys or referential integrity constraints, for loading the new factual DW data, as described in detail in previous work [18, 19]. We shall now summarily explain how this adaptation should be made, as explained in [18, 19].

Suppose a simple sales DW, with a schema as shown in Figure 2; two dimensional tables (`Store` and `Customer`, representing business descriptor entities) and one fact table (`Sales`, storing business facts aggregated from transactions). To simplify the figure, the Date dimension is not shown. This DW stores the sales value per store, per customer, and per day. Primary keys are in bold, while the referential integrity constraints with foreign keys are shown in italic. The factual attribute `S_Value` is additive. This property in fact data is very important for our methodology, as we shall demonstrate further on. For the data area concerning DW schema, we adopt the following method:

> **DW schema adaptation for supporting Real-Time DW:** Creation of a structural replica of each original table of the DW that could eventually receive new data. These tables (referred also as temporary tables) are created empty of contents, with no defined indexes, primary key, or constraints of any kind, including referential integrity. For each table, an extra attribute is created, for storing a unique sequential identifier related to the insertion of each row within the temporary tables.

Figure 3 shows the sub-schema to be appended to the original schema.

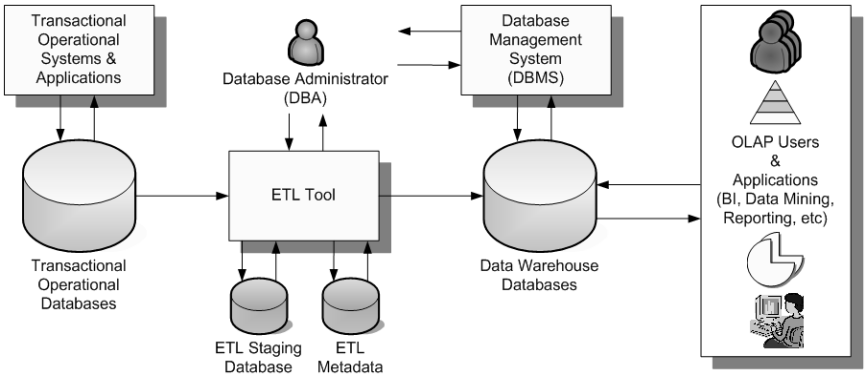


Figure 1. A classical data warehouse architecture.

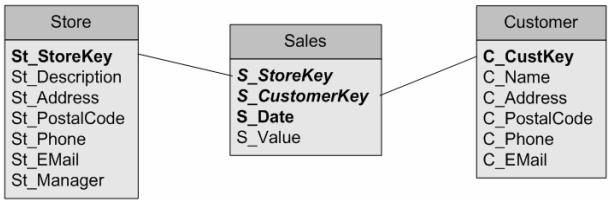


Figure 2. Sample sales data warehouse schema.

A temporary table is created for each original schema table. The unique sequential identifier attribute in each temporary table (`xTmp_Counter`) records the sequence in which each row is appended in the database. This allows identifying the exact sequence for each new inserted row, useful for restoring prior data states in disaster recovery procedures, and also for discarding dimensional rows which have more recent updates. For instance, if one customer has had two updates in the OLTP systems which, consequently, lead to the insertion of two new rows in the temporary table `CustomerTmp`, only the most recent one is relevant. This is done by considering as most recent the row with highest `CTmp_Counter` for that same customer (`CTmp_CustKey`).

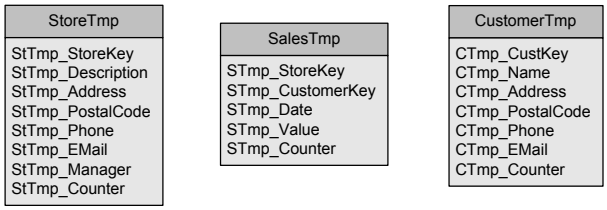


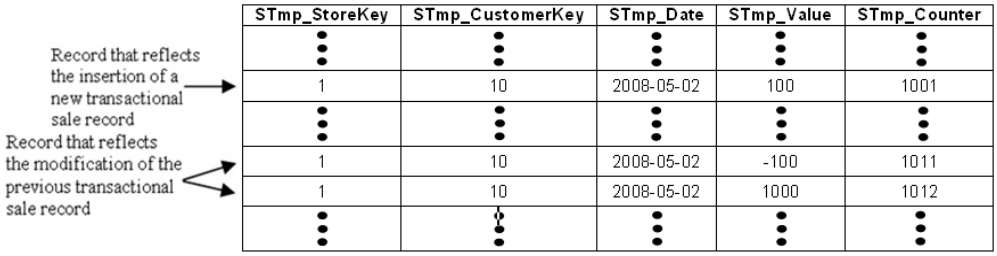
Figure 3. Sample sales DW appended sub-schema.

As demonstrated in [18,19], making the factual attributes additive is crucial for our solution, in spite of eventual drawbacks, for enabling the update of DW fact tables just by inserting new records, instead of using UPDATE or DELETE actions. However, Kimball refers in [12] that many ETL tools use a *UPDATE ELSE INSERT* function for loading data, considering this as a performance killer. With our method, any appending, updating or eliminating factual data tasks on OLTP systems only generate as new record insertions in the DW, allowing to minimize row, block and table locks and other concurrent data access problems. Physical database tablespace fragmentation is also avoided, once there is no deletion of data, only sequential increments. INSERT is much faster than UPDATE or DELETE row operations, since these need to previously perform a lookup in order to know which rows to update, allowing us to state that our method uses the fastest methods to refresh the DW [12, 19]. In the solution proposed in our paper, all dimensional data is to be updated or inserted directly in the existing tables. The definition of which attributes are additive and which are not should be the responsibility of the DW design team. According to [11], the most useful facts in a DW are numeric and additive. Our method for data loading uses the simplest method for writing data: appending new records. Any other type of writing method needs to execute more time consuming and complex tasks.

Regarding the sample sales DW of Figures 2 and 3, we shall now describe an example for loading new data. Figure 4 presents the insertion of a row in the data warehouse temporary fact table for the recording of a sales transaction of value 100 which took place at 2008-05-02 in store with `St_StoreKey` = 1 related to customer with `C_CustKey` = 10, identified by `STmp_Counter` = 1001. Meanwhile, other transactions occurred, and the organization's OLTP system recorded that instead of a value of 100 for the mentioned transaction, it should be 1000. The rows in the temporary fact table with `STmp_Counter` = 1011 and `STmp_Counter` = 1012 reflect this modification of values. The first eliminates the value of the initial transactional row and the second has the new real value, due to the additivity of the `STmp_Value` attribute.

Many issues involving the use of ETL tools do not focus only on performance problems (as would be expected), but also in aspects such as complexity, practicability and price [12]. By using only record insertion procedures to enable continuous data integration, using empty or small sized tables without any kind of constraint or attached physical file related to it, we guarantee the simplest and fastest logical and physical support for achieving our goals [12].

The fact that the only significant change in the logical and physical structure of the DW's schema is the simple adaptation shown in Figure 3, allows implementing ETL procedures in a manner to maximize its operability. Data loading may be done by simple standard SQL instructions or DBMS batch loading software such as SQL*Loader [16], with a minimum of complexity. There is no need for developing complex routines for updating the data area, since the needed data is easily accessible, independently from the used ETL tools.



STmp_StoreKey	STmp_CustomerKey	STmp_Date	STmp_Value	STmp_Counter
⋮	⋮	⋮	⋮	⋮
1	10	2008-05-02	100	1001
⋮	⋮	⋮	⋮	⋮
1	10	2008-05-02	-100	1011
1	10	2008-05-02	1000	1012
⋮	⋮	⋮	⋮	⋮

Figure 4. Partial contents of temporary fact table SalesTmp with exemplification record insertions.

Since the data schemas are modified, OLAP queries need to be adapted in order to take advantage of the most recent integrated data, which resides in the temporary tables. For the sample sales DW, consider the following query, calculating the total revenue per store in the last 7 days:

```
SELECT S_StoreKey,
       Sum(S_Value) AS Last7DaysSV
FROM Sales
WHERE S_Date>=SystemDate()-7
GROUP BY S_StoreKey
```

To take advantage of our method and include the most recent data in the OLAP query response, 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 prior instruction is illustrated below. It can be seen that the relevant rows from both issue tables are joined for supplying the query answer, filtering the rows used in the resulting dataset given its restrictions in the original instruction.

```
SELECT S_StoreKey,
       Sum(S_Value) AS Last7DaysSV
FROM (SELECT S_StoreKey, S_Value
      FROM Sales
      WHERE S_Date>=SystemDate()-7)
      UNION ALL
     (SELECT STmp_StoreKey, STmp_Value
      FROM SalesTmp
      WHERE STmp_Date>=SystemDate()-7)
GROUP BY S_StoreKey
```

An interesting and relevant aspect of the proposed methodology is that if users wish to query only the most recent information, they only need to do so against the temporary replicated tables. For instance, if the temporary tables are meant to be filled with data for each business day before they are recreated, and we want to know the sales value of the current day, per store, the adequate response could be obtained from the following SQL instruction:

```
SELECT STmp_StoreKey,
       Sum(STmp_Value) AS TodaysValue
FROM SalesTmp
WHERE STmp_Date=SystemDate()
GROUP BY STmp_StoreKey
```

This way, our method aids in processing the DW's most recent data, since this kind of data is stored within the temporary replica tables, assumed to be small in size. This minimizes CPU, memory and I/O costs involved in most recent data query processing. Theoretically, this would enable it to deliver the most recent decision making information while the business transaction itself occurs.

In our previous work, we also point out other advantages and disadvantages of our loading method, which are not included here due to lack of space, since the current paper is focused on engineering its applications. The performance and functional issues of this method for DW data loading are thoroughly presented and discussed in the mentioned previous work [18, 19]. Our data loading and database schema managing procedures lie between the used ETL tool and the DW databases. Since the original database schemas need to be modified, we also supply methods for querying the new schemas, making the OLAP querying procedure public for DW end users and applications. The proposed architecture for our 24/7 real-time DW is shown in Figure 5.

To enable 24/7 OLAP availability, we use the following method: *every DW database schema will be replicated, creating exact duplicates of all their components (tables, indexes, materialized views, etc). Both databases will be updated simultaneously by our 24/7 RTDW Tool, but only one at a time will be made available for OLAP users and applications.*

Since our data loading is done mainly into fact table replicas lacking performance optimization structures of any kind, as their size increases, OLAP performance decreases. This happens until it is time to reoptimize the system, which is done by transferring the data in the fact table replicas into the original ones, as explained in [18, 19], recreating the fact tables replicas empty of contents once more, and rebuilding the performance optimization structures, for the database which is offline to OLAP users. After this, the reoptimized database is made available to OLAP users, switching places with its duplicate, and vice-versa, continuously rotating the availability of both databases. The next section explains how *24/7 RTDW Tool* works and presents its components.

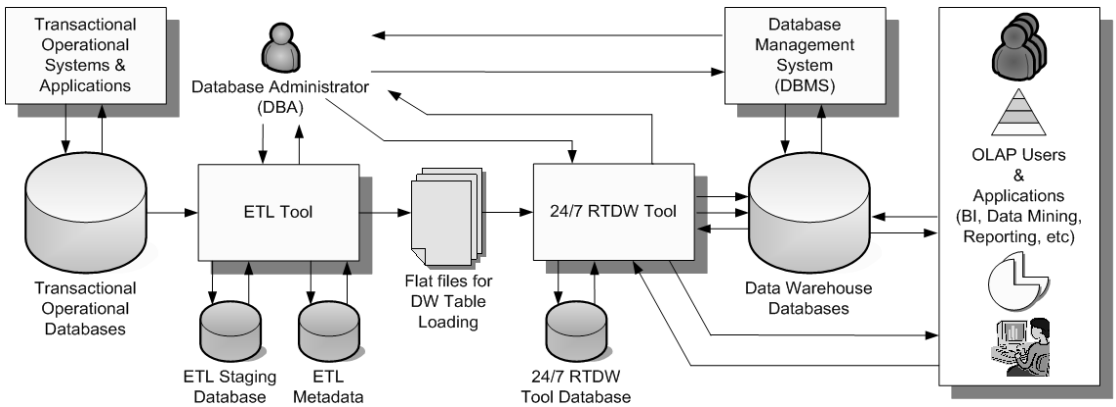


Figure 5. The 24/7 real-time data warehouse architecture.

III. THE 24/7 REAL-TIME DATA WAREHOUSE TOOL

Our 24/7 RTDW Tool has 3 main components: the *24/7 RTDW Tool Manager*, the *Loader* and the *Query Executor*. The first manages the DW database schemas and supplies an interface for monitoring loading tasks performed by the *Loader*. The *Loader* is responsible for refreshing the DW databases according to the methodology presented in [18, 19], briefly explained in section II. It also checks if all data has been loaded and completes all loading tasks that did not finish with success. The *Query Executor* will redirect querying to the database that is available for the DW end users and applications. To demonstrate our methodology, in Figure 6 we present a data schema of a real-world commercial sales enterprise DW, having four dimensional tables (`Time`, `Customers`, `Products` and `Promotions`, representing business descriptor entities) and one fact table (`Sales`, storing business measures aggregated from transactions). In what concerns the fact table, `S_SaleID` and `S_LineNumber` are primary key columns, `S_TimeID`, `S_CustomerID`, `S_ProductID` and `S_PromotionID` foreign keys for referencing the dimensional tables. Columns `S_ShipToCost`, `S_SalesAmount`, `S_Quantity` and `S_Profit` are additive factual attributes, while `S_SalesMean` and `S_Tax` are descriptive attributes.

Figure 7 shows the *24/7 RTDW Tool*, along with the related objects and people that interact with it. `DB1` and `DB2` represent the duplicated database schemas, with a temporary replica table of the `Sales` fact table, named `SalesTmp`. Our methodology requires the ETL tool to produce flat files for updating each table of the DW. This requirement should be easy to fulfill, since all ETL tools we know are able to do this [7, 14, 15, 22]. The *24/7 RTDW Tool Loader* will then use those flat files for updating the tables in the DW, using bulk loading whenever possible (in order to achieve the highest magnitude of speed) or SQL INSERTs (for instance, Oracle and MySQL allow using high speed bulk loaders, *SQL*Loader* and *Load Data Infile*, respectively, for appending new data). The following subsections explain the *24/7 RTDW Tool Database* and each of the tools' components.

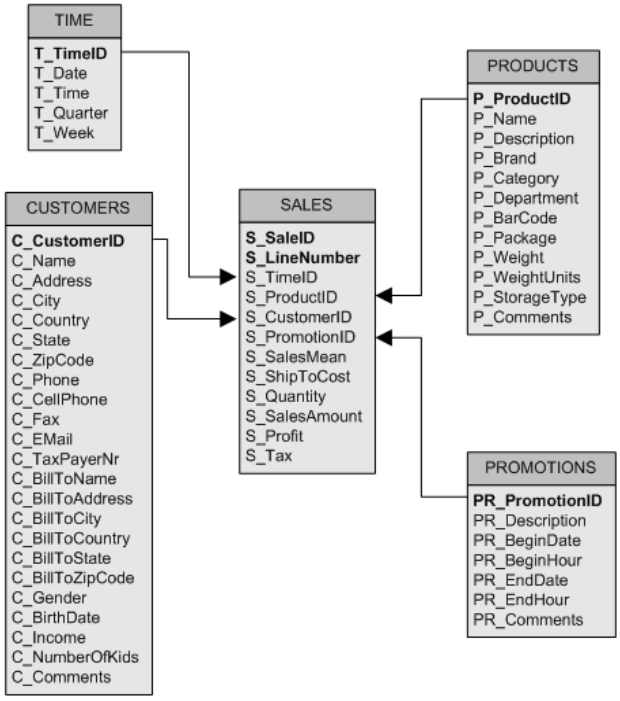


Figure 6. Real-world commercial sales DW schema.

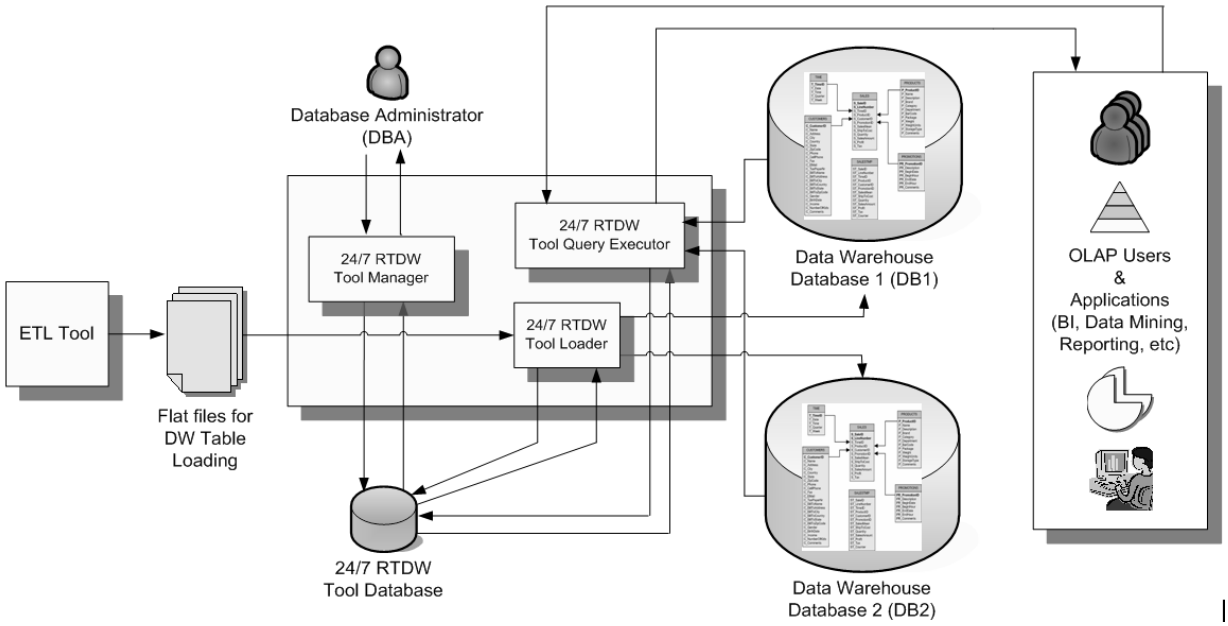


Figure 7. The 24/7 RTDW Tool middleware scenario (interaction and data flow).

A. *The Real-Time DW Tool Database*

The 24/7 RTDW Tool Database represents the data structures for managing our tool's functionality. Its schema is shown in Figure 8. The `DB` table is for defining the access to each DW database to be managed, defining its DBMS (Oracle, MySQL, SQL Server, etc), the ODBC driver name to access the database, DBMS database service name, DBA username and password. There are also fields concerning database reoptimization and OLAP usage:

`RB_ReoptimizingDB`, which indicates the number of the database that is currently being reoptimized (1 or 2), or a zero value if the database isn't reoptimizing;

`RB_ReoptType` and `RB_ReoptLimit` which indicates how the database will be reoptimized (after loading every *N* transactions, after every *N* seconds, or every day at *N* o'clock); and

`RB_CurrentDB` that indicates which is the currently available database for OLAP querying (1 or 2).

Tables `DB_Indexes` and `DB_MatViews` will store the names and stored SQL scripts for all the indexes and materialized views to be rebuilt, respectively, belonging to each database in `DB`, whenever that database's reoptimization procedure is executed. The `DB_Tables` table is for defining each table in each database. Each table has a unique identifier, `T_ID`, and is characterized by its name, `T_TableName`, and type, `T_TableType` (where `D` means dimensional and `F` means it is a fact table). It also has two flag fields, `T_LoadingDB` and `T_QueryingDB` which will indicate, in real-time, if the table is being currently updated by the *24/7 RTDW Tool Loader* or queried by the *Query Executor*. These flags are required for checking routines, such as if they are being used for OLAP or by database reoptimization procedures, for example.

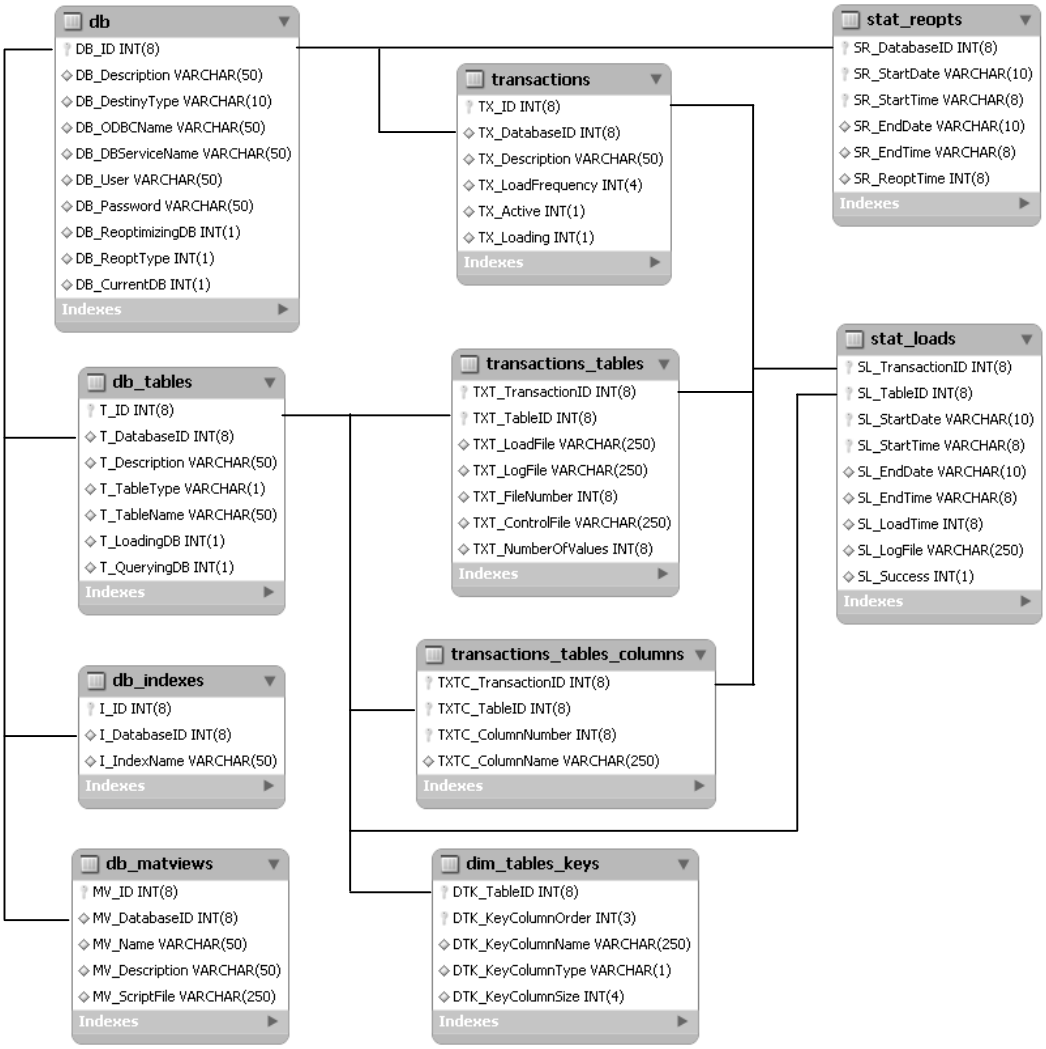


Figure 8. The 24/7 real-time data warehouse tool database.

Tables Transactions and Transactions_Tables play a main role in our loading methodology, acting as master and detail tables for defining a set of tables, which are all interconnected for a given transaction. For example, a typical star schema has a central fact table referenced with all dimensional tables belonging to it, similar to the schema in Figure 6. Each sales transaction is cross referenced with its foreign key tables (Customers, Products, Promotions and Time). Before a sales transaction is loaded, all foreign dimensional key values need to be checked to verify if they already exist in the database, ensuring referential integrity is not compromised. Therefore, a record in Transactions is created to define a transaction, giving a unique identifier value in TX_ID and the timespan between each load using TX_LoadFrequency (in seconds). In table Transactions_Tables we define which tables belong to each transaction defined in Transactions. In the former table, we indicate the physical address where the ETL flat file will be for loading, TXT_LoadFile, as well as the physical address of the log file for reporting loading success status, TXT_LogFile, and the control file for bulk loading, TXT_ControlFile, (the CTL file is used by Oracle bulk loader, *SQL*Loader*, for instance, comprising the data column and format instructions for the bulk loader), along with the number of fields to update per row, TXT_NumberOfValues. Field TXT_FileNumber acts as a unique counter for each complete flat file upload, for load checking purposes.

Table Dim_Tables_Keys contains a definition of which columns compose each dimensional tables's (defined in DB_Tables) primary key. Dim_Tables_Keys references each dimensional table in DB_Tables, containing all the existing primary key values. This is done for checking each dimensional table data upload done by the *24/7 RTDW Tool Loader*, where an SQL UPDATE is executed on the DW dimensional table if the dimensional primary key value already exists in a reference table, otherwise an SQL INSERT is done. In practice, using the schema from Figure 6, when the *Loader* receives an update for dimensional table Customers where S_CustomerID equals to 1, it checks for the value 1 in the reference table by DTK_TableID corresponding to the Customers table defined by TXT_TableID. If it is found, the customer already exists in Customers, and therefore executes an UPDATE, otherwise the record is INSERTed. On the other hand, before loading a Sales fact table flat file, a cross check is done for each dimensional table to see if the referring foreign key value exists, using Dim_Tables_Keys. If it does exist, this means the dimensional record to which the fact record refers to does not exist yet in the DW, and therefore is set aside to be subsequently loaded. This guarantees that referential integrity is assured.

Tables Stat_Loads and Stat_Reopts are included for measuring the time spent in each complete transactional data load and database reoptimization, respectively, allowing to monitor the tool's performance for optimization purposes. For instance, Stat_Loads table is useful for detecting if overlapping data loading threads are occurring for a certain table/transaction. If any value of SL_LoadTime is greater than TX_LoadInterval for a certain table/transaction, at least one data load procedure for that table was still executing while another had already started. Monitoring this, TX_LoadInterval is easy to tune for optimizing load performance, by what we have shown in [24], meaning that TX_LoadInterval's value should be increased for that table/transaction. The time spent in database reoptimization, SR_ReoptTime, can also aid the DBA to decide how often to reoptimize each database, given in DB_ReoptType and DB_ReoptLimit.

Table Transactions_Tables_Columns is used for defining all fields that belong to each table managed by the *24/7 RTDW Tool*. The tool's database is managed by the DBA, through the *24/7 RTDW Tool Manager*.

Supposing we use the tool for the schema shown in Figure 6, for loading transactions every 30 seconds, the values for the main tables in the *24/7 RTDW Tool Database* would be similar to what is shown in Figure 9.

B. The 24/7 Real-Time Data Warehouse Manager

This component allows the DBA to manage the tool's database and monitor data loading executions, according to what we have previously described. It also allows the DBA to have an option for building the original DW schema duplicates and setting up all the values, when the tool is to be used for the first time. This component is also responsible for reoptimizing the databases.

C. The 24/7 Real-Time Data Warehouse Loader

This component is responsible for executing the DW refreshment procedures, loading new data into its OLAP databases. All actions are executing according to what has been explained in the previous subsection A.

D. The 24/7 Real-Time Data Warehouse Query Executor

This component handles the queries issued by DW end users and OLAP tools, selecting which replicated database to use. It simply redirects the requested queries, according to the tool's database referring which is the available database for querying, and supplies the results to the query origin. It will also mark the flag field, T_QueryingDB, for each table, when it initiates or finishes querying.

IV. EXPERIMENTAL EVALUATION

To evaluate our tool, we used a real-world sales DW, based on the schema in Figure 6. The size of the sales DW is more than 2GB, corresponding to one year of commercial data, as shown in Table I. To build it, we used Oracle 11g DBMS on a 2.8 GHz Pentium IV CPU, with 1 GByte RAM, 7200 rpm hard disk.

Table DB

DB_ID	DB_Description	DB_DestinyType	DB_ODBCName	DB_DBServiceName	DB_User	DB_Password	DB_ReoptimizingDB	DB_ReoptType	DB_CurrentDB
1	Sales Data Warehouse	Oracle	Oraclellg_24x7RTDW	dwh1	system	daniel	0	1	1

Table DB_Tables

T_ID	T_DatabaseID	T_Description	T_TableType	T_TableName	T_LoadingDB	T_QueryingDB
1	1	Customers Table	D	Customers	0	0
2	1	Products Table	D	Products	0	0
3	1	Promotions Table	D	Promotions	0	0
4	1	Times Table	D	Times	0	0
5	1	Sales Table	F	Sales	0	0

Table DB_Indexes

I_ID	I_DatabaseID	I_IndexName
1	1	PK_Times
2	1	PK_Products
3	1	PK_Customers
4	1	PK_Promotions
5	1	PK_Sales
6	1	ProductsSales
7	1	TimesSales
8	1	CustomersSales
9	1	PromotionsSales
10	1	SMS
11	1	SCI
12	1	SPI
13	1	STI
14	1	SPRI

Table Dim_Tables_Keys

DTK_TableID	DTK_KeyColumnName	DTK_KeyColumnType	DTK_KeyColumnSize	DTK_KeyColumnOrder
1	C_CustomerID	I	7	1
2	P_ProductID	I	6	1
3	PR_PromotionID	I	8	1
4	T_TimeID	I	9	1

Table Transactions

TX_ID	TX_DatabaseID	TX_Description	TX_LoadFrequency	TX_Active	TX_Loading
1	1	Sales Transaction Set	30	1	0

Table Transactions_Tables

TXT_TransactionID	TXT_TableID	TXT_LoadFile	TXT_LogFile	TXT_FileNumber	TXT_ControlFile	TXT_NumberOfValues
1	1	Customers.dat	Customers.log	0	Customers.ctl	(NULL)
1	2	Products.dat	Products.log	0	Products.ctl	(NULL)
1	3	Promotions.dat	Promotions.log	0	Promotions.ctl	(NULL)
1	4	Times.dat	Times.log	0	Times.ctl	(NULL)
1	5	Sales.dat	Sales.log	0	Sales.ctl	(NULL)

Figure 9. The 24/7 Real-Time DW Tool Database main tables for the Sales DW (loading every 30 seconds).

To obtain results for decision making, a set of 12 OLAP queries was selected. These queries represent a sample of typical decision making information, such as customer product and promotion sales daily, monthly, quarterly and annually values. This set of queries represents a typical real-world workload, with a diversity of table joins, filtering, grouping and sorting actions, against the union of historical and recent business data.

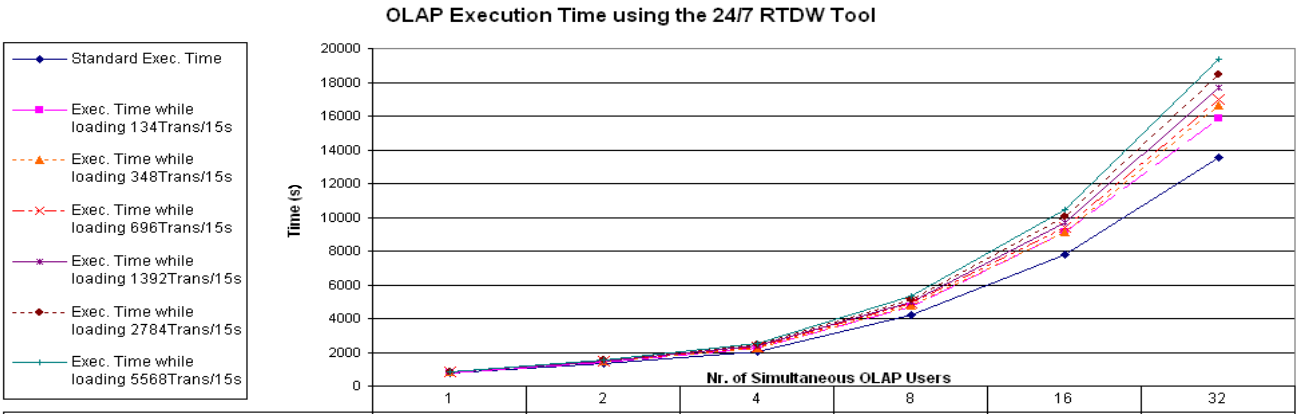
TABLE I. Dimensional features of the Sales Data Warehouse

	Times	Customers	Products	Promotions	Sales
Number of Rows	8 760	250 000	50 000	89 812	31M
Storage Size (MB)	0,12	90	7	10	1 927

We have measured the feasibility and performance of the OLAP query workload execution in scenarios with 1, 2, 4, 8, 16 and 32 concurrent users, while simultaneously updating the database with six different amounts of transactions on each loading procedure. The Oracle SQL*Loader was used every 15 seconds for inserting batches of 174, 348, 696, 1392, 2784, and 5568 transactions, each time. Inserting this amount of transactions every 15 seconds corresponds to a volume of 1, 2, 4, 8, 16 and 32 million transactions for the DW, on a 24-hour schedule.

Figure 10 shows experimental evaluation results, based on the OLAP workload execution time for each of the scenarios. As shown, the increase of execution time ranged from a minimal overhead of 4,55%, representing an increase of 35 seconds (from 764 to 799) for the scenario with only 1 OLAP user executing the workload while inserting 134 transactions every 15 seconds, to a maximum overhead of 42,82%, representing an increase of 5809 seconds (from 13566 to 19375) for the scenario with 32 users simultaneously executing the OLAP workload while inserting 5568 transactions every 15 seconds. This is the price to pay to build an effective 24/7 RTDW that better answers the needs of decision makers for obtaining near real-time continuous data integration and ongoing OLAP availability using our proposal, for the tested sales DW scenarios.

We also tested our solution using the 1GB and 10GB scales of the TPC-H benchmark [28], which represents a decision support DW, using the same software and hardware as for the Sales DW. A workload composed of all TPC-H queries using its fact table (queries 1, 3, 5, 6, 7, 8, 9, 10, 12, 14, 15, 17, 18, 19 and 20 of the benchmark) was used. The standard full query workload execution time measured for comparison, *i.e.*, without performing continuous data load, is 812 and 8155 seconds, for the 1GB and 10GB TPC-H database, respectively. Several rates of data loading were tested (loading intervals of 10, 30 and 60 seconds between each batch of load transactions), as well as different amounts of data for each loading batch (sets of 60, 600 and 6000 transactions for each load). The results for each tested scenario are shown in Figures 11 and 12.



	1	2	4	8	16	32
Standard Exec. Time	764	1336	2050	4206	7807	13566
Exec. Time while loading 134Trans/15s	799	1427	2243	4729	9091	15887
Exec. Time while loading 348Trans/15s	808	1441	2256	4790	9147	16599
Exec. Time while loading 696Trans/15s	824	1465	2299	4879	9378	17002
Exec. Time while loading 1392Trans/15s	840	1493	2362	5002	9671	17662
Exec. Time while loading 2784Trans/15s	850	1524	2422	5154	10016	18481
Exec. Time while loading 5568Trans/15s	858	1562	2493	5313	10443	19375

Figure 10. Query OLAP execution times using the 24/7 RTDW Tool with the real-world sales DW.

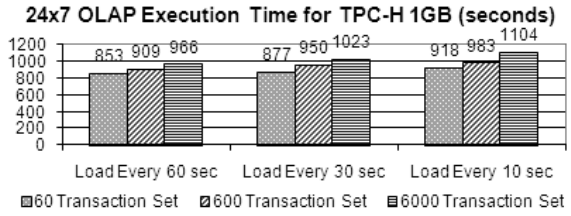


Figure 11. Query OLAP execution times using the 24/7 RTDW Tool with the 1GB TPC-H database.

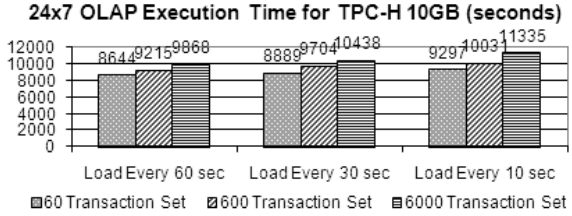


Figure 12. Query OLAP execution times using the 24/7 RTDW Tool with the 10GB TPC-H database.

As seen, the query execution time overheads range from 5% (853/812 seconds) to 36% (1104/812 seconds), for the 1GB database, and from 6% (8644/8155 seconds) to 39% (11335/8155 seconds) for the 10GB database. These are the workload response time overhead costs for implementing our proposed 24/7 RTDW, for the tested TPC-H scenarios.

V. Related Work

From the early days, research in data warehousing has mostly dealt with maintaining the DW in its traditional periodical update setup [4, 25]. Related literature presents tools and algorithms to load data in an off-line fashion. In a different line of research, data streams [5, 10, 17] are an alternative solution. However, research in data streams has focused on topics concerning the front-end, such as on-the-fly query computation without a systematic treatment of the back-end issues of a DW [11]. Many solutions for enabling RTDW have been proposed in the last two decades, both by IT business corporations and the research community. Much recent work dedicated to RTDW is focused on conceptual ETL modelling. A large number of papers and technical reports have been published concerning optimization of conceptual and logical ETL scenarios and workflows [1, 21, 24]. However, they lack the presentation of concrete specific ETL algorithms, along with their consequent OLTP and OLAP performance issues. DW updating processes are usually composed by a labor-intensive ETL workflow. To deal with this workflow, specialized tools are available in the market [7, 14, 15, 20, 22].

Nevertheless, not all transactional information needs to be immediately dealt with in real-time decision making requirements. We can define which groups of data are more important to include rapidly in the DW and other groups of data which can be updated in latter time. More recently, in [9], the authors present an interesting architecture on how to define the types of update and time priorities (immediate, at specific time intervals or only on DW offline updates) and respective synchronization for each group of transactional data items. This has also been explained and discussed in other recent publications [23]. In [23] the authors propose using SQL INSERT-like loading instructions with bulk load speed, taking advantage of in-memory databases as the data structures for data integration in the DW.

As mentioned before, DW is the core back-end for Business Intelligence (BI) [13]. The work presented in [6] refers to both query execution and data integration in analytical environments, discussing a comparison between techniques for loading and querying data simultaneously. A report published by Forrester Research [13] presents a

survey on ETL tools that look at enterprise DW as information-as-a-service means, considers BEA Systems, IBM and Oracle the leading companies in this field. The most recent solutions are based on the micro-batch load of small sets of data at the highest frequency possible, without jeopardising OLAP performance.

The approach presented in this paper is similar to Aster Data [27] in the sense of database duplication. Aster Data is the first commercial DW with 24/7 availability. However, their solution does not allow live data loading in the DW. In spite of all the published research and commercial tools available, as we mentioned earlier, none provides a concrete solution or, at least, a way of using their proposals or products for enabling RTDW on a 24/7 schedule, that is our paper's main contribution.

VI. Conclusions and Future Work

Data warehousing are no longer seen as a task that is physically separated from operational database systems and performed at specific time slots only, but that is further integrated into online systems operations. Indeed, data warehousing is more and more done in real-time. In this context, we have proposed a solution for the field area of 24/7 RTDW, maintaining the database online and available for OLAP on a 24/7 schedule while continuously integrating new data, and minimizing the impact in OLAP execution. This is achieved using data structure replication and mirroring, with our tool used as middleware between the usually employed ETL tool and the databases. The proposed schemas also act as fault tolerant mechanisms, potentially increasing DW quality and disaster recovery procedures by supplying an additional form of data replication.

Our proposal allows harvesting the benefits of seamless batch and real-time integration, without disabling any ETL procedure already in use by the enterprise DW. It takes advantage of the fastest data insertion methods in databases, just using frequent fast bulk loading or SQL-like INSERTs from small dimensioned flat files. This allows us to state that the apparent cost for enabling 24/7 RTDW according to our methodology is relatively small and acceptable on behalf of its users. The hardware costs involved in double storage data seem to be the most significant negative aspect of our solution, but in our opinion, these costs seem acceptable when comparing with the gains of enabling 24/7 RTDW.

As future work, we intend to test our ideas in various business contexts, namely in broader data volumes and update frequencies, e-business, etc. The drawbacks pointed out in our previous work [18] also need to be dealt with. Another aspect that can improve data loading performance is to use in-memory databases such as Oracle TimesTen [16] and IBM SolidDB [8], which can be explored in our methods for storing the temporary fact table's data. We also intend to extend the tool's capabilities towards data extraction and transformation, for developing an open source ETL suite.

References

[1] Agrawal, H., Chafle, G., Goyal, S., Mittal, S., Mukherjea, S.: *"An Enhanced Extract-Transform-Load System for Migrating Data in Telecom Billing"*, Int. Conf. on Data Engineering (ICDE), 2008.

[2] Baer, H.: *"On-Time Data Warehousing with Oracle Database 10g – Information at the Speed of your Business"*, Oracle Corporation, Oracle White Paper, 2004.

[3] Bruckner, R. M., List, B., Schiefer, J.: *"Striving Towards Near Real-Time Data Integration for Data Warehouses"*, Int. Conf. Data W. and Knowledge Discovery (DAWAK), 2002.

[4] Bruckner, R. M., Tjoa, A. M.: *"Capturing Delays and Valid Times in Data Warehouses – Towards Timely Consistent Analyses"*, Journal of Intelligent Information Systems, 2002.

[5] Golab, L. et al.: *"Stream Warehousing with DataDepot"*, Int. Conference on Management of Data (SIGMOD), 2009.

[6] Graefe, G.: *"Fast Loads and Fast Queries"*, International Conf. on Data Warehousing and Knowledge Discovery (DAWAK), 2009.

[7] *IBM Corporation*: *IBM* DW Manager & Data Integration - InfoSphere, www.ibm.com

[8] *IBM Corporation*: IBM SolidDB Memory DB, www.ibm.com

[9] Italiano, I. C., Ferreira, J. E.: *"Synchronization Options for Data Warehouse Designs"*, IEEE Computer Magazine, 2006.

[10] Jain, N., Mishra, S., Srinivasan, A., Gehrke, J., Widom, J., Balakrishnan, H., Çetintemel, U., Cherniack, M., Tibbetts, R., Zdonik, S. B.: *"Towards a Streaming SQL Standard"*, Int. Conf. Very Large Data Bases (VLDB), 1(2): 1379-1390, 2008.

[11] Karakasidis, A., Vassiliadis, P., Pitoura, E.: *"ETL Queues for Active Data Warehousing"*, ACM SIGMOD Int. Workshop Information Quality in Information Systems (IQIS), 2005.

[12] Kimball, R., Caserta, J.: *The Data Warehouse ETL Toolkit*, Wiley Computer Pub., 2004.

[13] Kobielus, J.: *"The Forrester Wave: Enterprise Data Warehousing Platforms"*, Forrester Research, Q1, 2009.

[14] *Microsoft Corporation*: Microsoft SQL Server & Microsoft SQL Server Information Integration Services, www.microsoft.com

[15] *Oracle Corporation*: Oracle Database Server & Oracle Data Integrator Enterprise Edition, www.oracle.com

[16] *Oracle Corporation*: Oracle TimesTen In-Memory Database, www.oracle.com

[17] Polyzotis, N., Skiadopoulos, S., Vassiliadis, P., Simitsis, A., Frantzell, N.: *"Meshing Streaming Updates with Persistent Data in an Active Data Warehouse"*, IEEE Transactions on Knowledge and Data Engineering, 20(7): 976-991, 2008.

[18] Santos, R. J., Bernardino, J.: *"Real-Time Data Warehouse Loading Methodology"*, International Database Engineering and Applications Symposium (IDEAS), 2008.

[19] Santos, R. J., Bernardino, J.: *"Optimizing Data Warehouse Loading Procedures for Enabling Useful-Time Data Warehousing"*, Int. Database Eng. App. Symp. (IDEAS), 2008.

[20] *SAS Institute, Inc*: SAS Enterprise ETL Server & SAS ETL Studio, www.sas.com

[21] Simitsis, A., Vassiliadis, P.: *"A method for the mapping of conceptual designs to logical blueprints for ETL processes"*, Decision Support Systems (45) 22–40, 2008.

[22] Talend, Open Source ETL Solution, www.talend.com

[23] Thomsen, C., Pedersen, T. B., Lehner, W.: *"RiTE: Providing On-Demand Data for Right-Time Data Warehousing"*, International Conference on Data Engineering (ICDE), 2008.

[24] Vassiliadis, P., et al.: *"A generic and customizable framework for the design of ETL scenarios"*, Information Systems, 30(7), 2005.

[25] Yang, J., Widom, J.: *"Incremental Computation and Maintenance of Temporal Aggregates"*, I. Conf. Data Engineering (ICDE), 2001.

[26] N. Yuhanna, and M. Gilpin, *"The Forrester Wave: Information-As-A-Service"*, Forrester Research Inc, Q1 2008.

[27] Aster Data, *Aster Data nCluster: "Always On" Availability*, Aster Data Systems, 2009.

[28] Transaction Processing Council, Decision Support Benchmark TPC-H, www.tpc.org/tpch/