International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Review on Scheduling Algorithms for Data Warehousing

S. M. Ramesh[1], B. Gomathy[2]

[1]Deparment of ECE, Bannari Amman Institute of Technology, Sathyamangalam – 638401, India
[2]Deparment of CSE, Bannari Amman Institute of Technology, Sathyamangalam – 638401, India

***Abstract:** These instructions Poor performance can turn a successful data warehousing project into a failure. Consequently, several attempts have been made by various researchers to deal with the problem of scheduling the Extract-Transform-Load (ETL) process. In this paper present several approaches in the context of enhancing the data warehousing Extract, Transform and loading stages. To focus on enhancing the performance of extract and transform phases by proposing two algorithms that reduce the time needed in each phase through employing the hidden semantic information in the data. Also focus on the problem of scheduling the execution of the ETL activities, with the goal of minimizing ETL execution time. Explore and invest in this area by choosing three scheduling techniques for ETL. Finally, the experimentally show their behavior in terms of execution time in the sales domain to understand the impact of implementing any of them and choosing the one leading to maximum performance enhancement.*

Keywords: Data warehousing, Data loading schedules, ETL optimization, Scheduling algorithms, Round robin.

1. Introduction

Lately data warehousing (DW) has gained a lot of attention both from both the industry and research community communities. From the industrial perspective, building an information system for the huge data volumes in any industry requires lots of resources as time and money. Unless those resources add to the industry value, such systems are worthless. Thus, people require that information systems should be capable to provide extremely fast responses to different queries specially those queries that affect decision making. From the research perspective, researchers find that due to the increasing need and value of for efficient data warehouses, it is still a fruitful research direction where further improvements can be added, further investigation in data warehouses performance and techniques are still needed and present fruitful research directions.

In [1], the authors show how data warehousing systems address the issue of enabling managers to acquire and integrate information from different sources, and to efficiently query very large databases. The authors mention that during challenging times, good decision-making becomes critical. The best decisions are made when all the relevant data is taken into consideration. Today, the biggest challenge in any organization is to achieve better performance with least cost, and to make better decisions than competitors. That is why data warehouses are widely used within the largest and most complex businesses in the world. According to the authors in [2], a data warehouse is a collection of consistent, subject-oriented, integrated, time-variant, nonvolatile data along with processes on them, which are based on current and historical information that enable people to make decisions and predictions about the future.

The DW is suitable for direct querying and analysis, and it stands as a source for building logical data marts oriented to specific areas of an enter prize. Due to the importance of enhancing the data quality several vendors presented a number of quality measurement tools such as IBM Info Sphere Information Analyzer, Oracle Data Profiling, open source as Data Cleaner, and Microsoft Data Profiler, and others.

In this paper, to measure the quality of results and selected Microsoft data profiler tool due to its simplicity and richness of the analysis provided [3,4]. Generally, a data profile is a collection of aggregate statistics about the data in the different relations (tables) that might include: number of rows in a table, and/or Count count of distinct values in a column, number of “null NULL” or missing values in a column, the distribution of values along a column, as well as the strength of the functional dependency of one column on another (this will help in choosing the Primary primary Key key) on another. The statistics that a data profile provides gives the information that one needs in order to effectively minimize the quality issues that might occur from using heterogeneous source data. Inspired by the importance of building efficient data warehouses, in this work we investigate the use of domain semantics to enhance the extraction and transformation phases in the staging area. Focus on the “Sales” business area domain due to its simplicity and familiarity.

In Addition, loading is the process of populating the data into the data warehouse. As simple as this process may seem to be, it can’t be replaced, whether we are dealing with flat files, excel sheets, or relational databases, it all comes to delivering the data to its final repository to have one single version of truth enabling to make the right decision at the right time. Thus, enhancing the loading process and a crucial ingredient in the overall DW enhancement process.

Authors in [5] explored the scheduling phase from system memory utilization perspective as they expressed in their paper that since having many applications involving continuous data streams, data arrival is burst and data rate fluctuates over time. Systems that seek to give rapid or real-time query responses in such an environment must be prepared to deal gracefully with bursts in data arrival without compromising system performance. So their goal was to utilize resources during times of peak load by choosing the appropriate scheduling strategy which can have significant impact on the run time system memory usage as well as output latency. Besides, many researches were interested in the relationship between data-to-application flow and transferring data to traditional data warehouse while managing system resources. Others investigated the possibility to enhance the load process by not only focusing on the best scheduling technique but also analyzing and understanding the behavior and structure of its repository to deliver the data in the most efficient manner.

In this direction, authors in [6] mentioned that a simple low-cost shared-nothing architecture with horizontally fully-partitioned facts can be used to speedup response time of the data warehouse significantly and they concluded after experiments that, although it is not possible to guarantee linear speedup for all query patterns, workload-friendly placement can prevent very low speed up and provide near to linear speedup for most queries in Node Partitioned Data Warehouses.

The goal is to continue the effort towards an enhanced data warehousing performance through its final phase "loading". For motivated by the fact that in real life important information that is delivered late results in making inaccurate decisions. In this context, explore three scheduling techniques (First-In-First-Out (FIFO), Minimum Cost, and Round Robin (RR) based on time and records) for scheduling the ETL process. We experimentally show their behavior in terms of execution time with our sales data and discuss the impact of their implementation.

2. Related Work

Today, DW and on-line analytical processing (OLAP) are essential elements for decision support. In [7], the authors delved into the logical optimization of the ETL processes, modeling it as a state space search problem. They considered each ETL workflow as a state and presented the state space through a set of correct state transitions. Moreover, some algorithms were provided towards the minimization of the execution cost of the ETL workflow. In addition, there has been a lot of work to optimize the performance of relational data warehouses. Authors in [8] suggested combining three major techniques that can be used for this objective: enhanced index schemes (join indexes, bitmap indexes), materialized views, and data partitioning. Currently, the existing research prototypes or products use materialized views alone or indexes alone or combination of them, but none of the prototypes use all three techniques together. In the paper the authors showed by a systematic experimental evaluation that the combination of these three techniques reduces the query processing cost and the maintenance overhead significantly. On the other hand, authors in [9] focused on proposing an ontology-based ETL framework for covering schema integration as well as semantic integration.

In their approach, beside the schema based semantics in Common Warehouse Metamodel (CWM), they claim that semantic interoperability in ETL processes can be improved by means of an ontology-based foundation for better representation, and management of the underlying domain semantics. Several other work considered the ontology-based ETL including [10, 11]. The last phase in the ETL stage after extracting and transforming data is the load phase. The objective of the load phase is simple "load the data into the end target" which is usually the data warehouse. The rate of exporting data may vary from daily, weekly or monthly basis. Basically, the main challenge resides in delivering the right data at the right time. In this direction many researches were conducted to optimize the transfer process. Authors in [12] investigated the best loading method by presenting a methodology for achieving useful-time data warehousing by enabling continuous data integration, while minimizing impact of query execution on the user end of the DW. This is achieved by data structure replication and adapting query instructions in order to take advantage of the new schemas, while executing the best previously determined continuous data integration methods.

On the other hand, authors in [13] realized that with the increase in the updates, previously fast queries tend to slow down considerably. However, depending on the user requirements, the response time or the data quality can be improved by scheduling the queries and updates appropriately and thus they mentioned that if both criteria are to be considered simultaneously, in this faced with a so-called multi-objective optimization problem. Hence, they developed a scheduling approach that provides the optimal schedule with regard to the user requirements at any given point in time supported by their evaluation for the scheduling in an extensive experimental study. In addition, authors in [14] presented an approach for automating the derivation of incremental load jobs based on educational reasoning. They started by reviewing existing Change Data Capture techniques as well as loading facilities for data warehouse refreshment then based on them they provided transformation rules for the derivation of incremental load jobs. In the same context of data loading, authors in [15] considered the issue of bulk loading large data sets for the UB-Tree, a multidimensional clustering index which inherits all good properties of B-Tree. Especially in data warehousing, data mining and OLAP it is necessary to have efficient bulk loading techniques, because loading doesn’t occur continuously, but only from time to time with usually large data sets. The authors thus proposed two techniques, one for initial loading, which creates a new UB Tree, and one for incremental loading, which adds data to an existing UB-Tree. Both techniques try to optimize both the I/O and CPU cost.

Authors in [16] presented a GUI based ETL procedure for the continuous loading of the data in the active data warehouse. The main idea is to enable continuous data integration along with minimizing the impact on query execution at the user end. Despite the researches published and how each try to contribute in the enhancement of the loading process in the DW, for must not ignore the fact that and are not dealing with just one type of data, so authors in [17] proposed a new technique called selective load to handle big dimensions in distributed data warehouses by maintaining nearly linear speed up in query execution time.

Finally, authors in [18] dealt with the problem of scheduling off-line ETL scenarios, aiming at improving both the execution time and memory consumption without allowing data losses. In doing so, they assessed a set of scheduling policies for the execution of ETL work flows like minimum cost and other techniques and analyzed their behavior with different input sizes.

3. Problem Definition

Sales and marketing are considered slippery territory as they deal with people (customers), and people often do allow their emotions to drive their decision-making process. However, this characteristic does not hinder the ability to rationally define measure, analyze, and improve this important business domain. Nevertheless, it is true that most organizations' failure is usually due to the lack of confidence in the information gathered about production/people, along with failure to acquire the right data at the right time. Hence, in order to overcome this situation, an organization should define the major factors affecting its sales and profit such as lack of proper strategy, consuming the time of salespeople with unnecessary tasks, and defining the right products that will return high ROI.

Consequently, in this work we are interested in finding a way to assist the sales industry enhancing their repository that basically contains all necessary data about their productivity. However, due to the fact that there is a wide spectrum of possible research directions including: enhancing the performance of the staging area, enhancing data analysis, focusing on design issues, and studying Business Intelligence (BI) along with many others, we decided to start exploring the extraction and transformation processes of raw data acquired from the operational level from heterogeneous data sources and how it could be improved to finally load high quality data into the DW as well as the best approach to load it. In addition, in order to capture the added value and to benefit from enhancing the data warehouse staging area, several contributing dimensions need to be considered.

3.1 Human factor

Prior knowledge about the type of data is very important. Given the fact that there are a lot of changes going on the percentage of understanding of the raw operational data to be extracted is one of the factors impacting in the quality of the extraction process.

3.2 General ontologies

Those Ontologies represent general rules and guidelines for the ETL phases. Those rules are usually based on the business domain considered. Hence, gathered the most commonly used extraction ontologies, refined, and tailored them to our problem domain (i.e. the sales domain) to eliminate non-relevant data and achieve higher quality of extracted data.

3.3 Results from statistics and quality measures

Realizing that every process costs time, and in order to improve the human factor; for used a data quality tool to provide us with a better understanding of the data in hand in terms of the inner relationship within data per table and outer relationship among different tables. Those tools generally enable us to evaluate the distribution of data and determine if there exists any pattern in the data. Such characteristics in the data help us to determine if there any candidate primary keys that require us to apply architecture design change, or to change the size of data.

In general, the advantage of the tool used is cumulative as in the transformation stage the results reflect an improved version of data quality after extraction. Integrating those tools and dimensions are managed in this work to propose an algorithm to efficiently perform each of the extract and transform phases

4. Scheduling Algorithms

Following the approaches proposed to optimize the ETL process, and more specifically the "load" phase of this stage, the decided to focus on 3 scheduling techniques where each represents a different perspective of data processing. They are "First-in-First-Out"(FIFO), Minimum Cost (MC), and Round Robin (RR). We will first introduce each one of them and then explain how they were mapped on our data. Those techniques were used at different stages and in the following section we will show the what-if scenarios results on our test data.

4.1 First in first out

First In First Out (FIFO) is one of the very primitive algorithms that simply takes the data as soon as it comes and transfers it to the destination regardless of any priorities. The input to the algorithm is simply all tables required for the DW, and the output is their successful transfer. the implemented algorithm proceeds as follow: First, all queries of those tables (Tnq) are added to one list (AL.FIFO.Tnq) where each query represents the selection of all columns of the table(T), then all tables names (Tn) are added to the same list. For each query in the list "AL.FIFO.Tnq" a connection to the Database holding the table was created and then we started measuring the difference between the start time (S.T) and end time (E.T) for processing the query "Table Total Execution Time" (ET). At the end we added all those ET together to have the total time Ttot to load all data using FIFO technique over different stages.

4.1.1 Algorithm 1: FIFO

Input: Database Tables at source or after Extraction or Transformation Phase
Output: Data loaded to DW without waiting if queue is idle

```
for (each Tnq) do
/* add Tnq to AL.FIFO.Tnq */
end
for each T do
/* add Tn to AL.FIFO.Tnq */
```

```
end
for each AL.FIFO.Tnq do
/* create connection to the Database holding the table */
S.T= System.nanoTime (); /* the above formula represents the Start time of processing a Query */
/* Process Query */
E.T= System.nanoTime (); /* the above formula represents the end time of processing a Query */
ET= E.T - S.T; /* calculate the total execution time of Table to be loaded to the DW */
Ttot += ET; /* total time to transfer all tables */
Return Ttot
End
```

4.2 Minimum cost

The Minimum Cost (MC) scheduling is the second proposed algorithm to reduce the time needed for the execution of the loading phase. Similar to the FIFO algorithm, MC takes as input the data from any stage of Extract/Transform or at sources and as output the successful transfer of data but based on those with maximum volume first. Initially, after specify the list of Tables (T) required we add all their names (Tn) to a list (AL.Tn).

Afterwards, the process each table to retrieve its size (Ts) to add it beside (Tn) and its query (Tnq) to one list (AL.MC.Tnqs). Then, we take this list and re-sort it in a descending order (AL.MC.Tnqs.Desc) based on the size. Finally, once the list is ready, we create another connection to each table in this list and start measuring the difference between its start time (S.T) and its end time (E.T) to get our total table execution time ET and their summation leads us to the total time Ttot needed for MC algorithm to finish its job.

4.2.1 Algorithm 1: MC

Input: Database Tables at source or after Extraction or Transformation Phase

Output: Data loaded to data warehouse by maximum size first for (each AL.Tn) do

```
/* create connection to the Database holding the current table in the list */
/* Retrieve table size Ts */
/* add Tn,Tnq and Ts to AL.MC.Tnqs */
end
for (each AL.MC.Tnqs) do
/* Re-Order AL.MC.Tnqs by maximum size and then add to AL.MC.Tnqs.Desc */
end
For (each T Î AL.MC.Tnqs.Desc) do
/* create connection to the database holding the table */S.T= System.nanoTime (); /* the above formula represents the start time of processing a query */
/* Process Query */
E.T= System.nanoTime ();
/* the above formula represents the end time of processing a query */
ET= E.T - S.T;
/* calculate the total execution time of Table to be loaded to the DW */
Ttot+=ET;
/* total time to transfer all tables */
return Ttot
End
```

4.3 Round robin

For third technique is Round Robin (RR) which is implemented in two versions rather than the traditional one to analyze their behaviors. So, instead of implementing the traditional Round Robin based on assigning time slices in equal portions for every table. Also implemented another version based on fixed threshold number of records to get a new perspective about what if having to wait for processing a complete set of records regardless of their size as the rotation
factor.

4.3.1 Time based round robin (TRR)

In the first version of RR is started with setting rotations based on time, thus as the pervious algorithms the input is the data from any stage of Extract/Transform or at sources along with the time slice. The algorithm starts by creating connections to all tables to be loaded and at the same time setting their status initially to false (i.e. idle status) until they get processed.

Thus, when the table status (S) changes to True we will set the current time (C.T) value to be the start time (S.T) of the table. Then check if the table was fully processed or not by comparing an incremental count of table records (C.Tr) with its total size (Ts). If there is still unprocessed records we check if this table was partially processed before to avoid miss-capturing of table actual start time by verifying the status of the indicator (ind) assigned to this table which initially is set to 0 (i.e. table was never processed).

Afterwards, as long as rotation turn isn’t reached (C.T is less than sum of S.T and TRR) and the table is not fully processed (C.Tr is not equal to Ts), for process the records using the table query (Tnq) while adjusting table C.T value. Once the table gets fully processed we capture the table end time (E.T) and calculate the difference to get the total table execution time (ET). At the end we add all those (ET) together to have the total time Ttot to load the tables.

(a) Algorithm 3: Time Based Round Robin

Input: Database Tables at source or after Extraction or Transformation Phase besides specifying the Round Robin Time Limit

Output: Data loaded to data warehouse based on time rotations

```
/* create connections to all tables to be loaded */
/* Set the status of all Tables.Processed to "False" */
/* set all tables indicators to 0 */
while (S != True) do
/* set C.T to current system time */
/* set S.T to current system time */
if (C.Tr != Ts) then
if (ind==0) then
S.T=System.NanoTime();
/* indicator is set to 1 */
end
while (C.T < (S.T + TRR)) and (C.Tr != Ts) do
```

```
/* process table query (Tnq) till TRR is reached */
/* set C.T to current system time */
end
end
if (C.Tr == Ts) then
E.T=System.NanoTime(); ET= E.T - S.T
/* set S to true */
end
End
```

/* add the summation of all Tables ET to get Ttot*/

4.3.2 Record limit based round robin

So as with prior techniques we take as input the data coming from any stage of Extract/Transform or at sources along with the Round Robin records limit (LRR) for rotation. First, create a connection to all the tables to be transferred then as long as didn't finish processing all the data for set the Round Robin status (S) to false then we start capturing the start time (S.T) of processing a table and change its status to true (T).

While the Round Robin Limit (LRR) is not reached and haven't finished processing the whole table size (Ts), the query referring to all table's data (Tnq) get executed. Then, when we finish loading all the table we capture its end time (E.T) then calculate the table total execution time (ET) and add it to a list of all tables total execution time (Al.ET).

Finally when all tables are loaded adjust in this algorithm end round robin status (S) to "True" and from (AL.ET) get in Total Time (Ttot) of Round Robin based on records limit technique.

(a) Algorithm 4: Records Limit Round Robin
Input: same as with previous algorithms besides specifying the Round Robin Records Limit
Output: Data loaded to data warehouse based on record limit.

```
/* create connections to all tables to be loaded */
while (S == False) do
if (T ==notIdle) then
S.T=System.NanoTime();
/* set T active */
end
while (LRR isReached = False and Ts isReached = False) do
/* process table query (Tnq) till LRR is reached */
if (Ts isReached = True) then
E.T=System.NanoTime()
ET= E.T - S.T
/* add ET to Al:ET */
end
end
if (all Ts isReached = True) then
/* set S to True since all tables have been processed */
end
End
```

/* add the summation of all Al.ET to get Ttot */

5. Scheduling Experiments

In this section, we discuss the experimental results of our proposed algorithms. The used data was from Adventure Works Database [http://msftdbprodsamples.codeplex.com/] to simulate the loading phase to a sales DW. Our objective is to evaluate data transfer using different techniques (FIFO, MC, RR time and record rotation).

The data included in our test is coming from data at their sources after extraction and transform phases as we wanted to capture the time needed to transfer data from each stage and which technique is the most suitable in case there is a decision required. For choosing FIFO, FIFO turns to be a typical solution if we went random with just a simple knowledge about the data in hand which sometime might be the case with the need for fast response for critical inquiries.

Table 1: Scheduling Techniques by Minutes for Data Loaded at Different Stages

Scheduling Techniques	Data Source db	Extracted db	Transformed db
FIFO	0.47	0.34	0.34
MC	0.42	0.38	0.39
25K Records	2.08	2.86	1.27
50K Records	2.90	2.33	1.94
75K Records	2.98	2.38	1.82
100K Records	2.71	1.22	1.07
15 sec	0.43	0.73	0.52
30 sec	0.42	0.32	0.39
45 sec	0.41	0.32	0.38
1 min	0.45	0.37	0.33

Table 2: Records Limit Based Round Robin Statistics for Data Loaded at Different Stages

Records Limit Based Round Robin Statistics	Data Source db	Extracted db	Transformed db
25K Records	2.36	2.36	1.97
50K Records	2.40	2.33	1.74
75K Records	2.38	2.28	1.62
100K Records	2.01	1.92	1.37

Table 3: Time Based Round Robin Statistics for Data Loaded at Different Stages

Time Based Round Robin Statistics	Data Source db	Extracted db	Transformed db
15 sec	0.45	0.73	0.56
30 sec	0.42	0.34	0.33
45 sec	0.41	0.32	0.38
1 min	0.40	0.37	0.32

As for MC, this one targets large data sets first which requires having sufficient memory allocation. For the Round Robin, we tried to look not only at the traditional time rotation but also what if used specified number of records as limit. All experiments have been conducted on a Core i7 with 2.5 GHz and 16 GB main memory. As for the Database size over different stages that is 14 GB for data at sources, 6.3 GB at extract and 6.89 GB at Transform. From those experiments, noticed as shown in Table.1 that when comparing all scheduling techniques that FIFO has slightly better performance than MC followed by Time Based Round Robin, while Record Limit Based Round Robin behaves the worst. However, when increasing the record limit as shown in Table.2, the performance improves which can be taken into consideration for scenarios where there is a need to quickly load part of the data set into a data mart. On the other hand, after testing several Time Based Round Robin as shown in Table.3, in concluded that it behaves best with smaller data set (as with Extracted Data Set).

6. Conclusions

In a typical DW environment, data is extracted periodically from the applications that support business processes and copied to special dedicated machines. There it can be validated, reformatted, reorganized, summarized, restructured, and supplemented with data from other sources which will lead to having a DW acting as the main source of information for future analysis, report generation, and presentation through ad-hoc reports, portals, and dashboards. In this paper, for tried to analyze and evaluate different scheduling techniques namely, FIFO (Random), MC (Maximum Size First), RR (based on time), and finally a new approach for RR which that is based on rotating on fixed number of records regardless of their size.

References

[1] El-Gamal, N.: Data warehouse conceptual modeling approaches. In: Proceedings of the 37th International Conference on Computers and Industrial Engineering, Alexandria, Egypt (October 2007) 231-242
[2] Inmon, W.H.: Building the data warehouse. Wiley Publishing,Inc., Wellesley, MA,USA (1992)
[3] Server, M.S.: Data profiling task and viewer. http://technet.microsoft.com/en-us/library/bb895310.aspx (2013)
[4] Knight, B., Veerman, E., Dickinson, G., Hinson, D., Herbold, D.: Professional Microsoft SQL Server 2008 Integration Services. Wiley Publishing, Inc. (2008)
[5] Bateni, Mohammad Hossein and Golab, Lukasz and Hajiaghayi, Mohammad Taghi and Karloff, Howard. Scheduling to Minimize Staleness and Stretch in Real-time Data Warehouses. In Proceedings of the twenty-first Annual Symposium on Parallelism in Algorithms and Architectures, 2009.
[6] Furtado P. Experimental Evidence on Partitioning in Parallel Data Warehouses. In Proceedings of the 7th ACM International Workshop on Data Warehousing and Olap, 2004.
[7] Simitsis, A., Vassiliadis, P., Sellis, T.: Optimizing etl processes in data warehouses.In: Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on. (2005) 564-575.
[8] Ladjel, B., Michel, S., Herv, L., Mukesh, M.: Bringing together partitioning, materialized views and indexes to optimize performance of relational data warehouses.In Kambayashi, Y., Mohania, M., W, W., eds.: Data Warehousing and Knowledge Discovery. Volume 3181 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2004) 15-25.
[9] Thi, A.D.H., Nguyen, B.T.: A semantic approach towards cwm-based etl processes.In: Proceedings of I-SEMANTICS 08, Graz, Austria (September 2008) 58-66.
[10] Skoutas, D., Simitsis, A., Sellis, T.: Journal on data semantics xiii. Springer-Verlag, Berlin, Heidelberg (2009) 120-146.
[11] Skoutas, D., Simitsis, A.: Ontology-based conceptual design of etl processes for both structured and semi-structured data. International Journal on Semantic Web and Information Systems (IJSWIS) 3(4) (2007) 1-24.
[12] Santos, Ricardo Jorge and Bernardino, Jorge. Optimizing Data Warehouse Loading Procedures for Enabling Useful-time Data Warehousing. In Proceedings of the 2009 International Database Engineering and Applications Symposium, 2009.
[13] Thiele, Maik and Bader, Andreas and Lehner, Wolfgang. Multi-Objective Scheduling for Real-Time Data Warehouses. In Business, Technologie Und Web (BTW), 2009.
[14] Thomas Jorg and Stefan Dessloch. Formalizing ETL Jobs for Incremental Loading of Data Warehouses. In Datenbanksysteme in Business, Technologie Und Web (BTW), 13.Münster, Germany, 2009.
[15] Fenk, Robert and Kawakami, Akihiko and Markl, Volker and Bayer, Rudolf and Osaki, Shunji. Bulk Loading a Data Warehouse Built Upon a UB-Tree. In Proceedings of the 2000 International Symposium on Database Engineering and Applications, 2000.
[16] Mallikharjuna Reddy V and Sanjay K. Jena. Active Datawarehouse Loading by Tool Based ETL Procedure. In Proceedings of the 2010 International Conference on Information and Knowledge Engineering, july 12-15, Las Vegas Nevada, USA, 2010.
[17] Costa, Marco and Madeira, Henrique. Handling Big Dimensions in Distributed Data Warehouses Using the DWS Technique. In Proceedings of the 7th ACM International Workshop on Data Warehousing and Olap, 2004.
[18] Anastasios Karagiannis, Panos Vassiliadis, Alkis Simitsis. Macro Level Scheduling of ETL Workflows. In 9th International Workshop on Quality in Databases (QDB 2011), in Conjunction with VLDB, 2011.