International Journal of Advanced Scientific Technologies ,Engineering and Management Sciences (IJASTEMS-ISSN: 2454-356X) *Volume.3,Special Issue.1,March.2017*

ENHANCING PERFORMANCE FOR DATA WARE WAREHOUSE STAGING AREA

Dr.M.SREEDEVI, ASSISTANT PROFESSOR, DEPT.OF.COMPUTER SCIENCE,S.V.UNIERSITY,TIRUPATI

Abstract -- Reduction in the performance can turn a successful data warehousing project into a failure. Many attempts have been made by various researchers to deal with the problem of scheduling the Extract Transform-Load (ETL) process. This paper therefore deals with the different in the context of improving the data warehousing in transform stage. We focus on improving the performance of transform phase next to the analysis stage. We focus on the problem of scheduling the execution of the transform activities, with the goal of reducing the execution time. We represent here three scheduling techniques for improving the performance of the data warehousing projects.

KEYWORDS -- Data Warehouse, ETL, Scheduling techniques, ETL optimization

I. INTRODUCTION

Lately data warehousing (DW) has gained a lot of attention from both the industry and research 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 this direction, we are mentioning 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 speedup
and provide near to linear speedup for most queries in Node Partitioned Data Warehouses. Our goal is to continue the effort towards an enhanced data warehousing performance through its final phase "loading". We are motivated by the fact that in real life important information that is delivered late results in making inaccurate decisions. In this context, we 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.

II. SCHEDULING ALGORITHMS

Following the approaches proposed to optimize the ETL process, and more specifically the "load" phase of this stage, we 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.

2.1 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. Our implemented algorithm proceeds as follow: First, all queries of those tables (Tnq) are added to osne 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" (E_T). At the end we added all those E_T together to have the total time T_{tot} to load all data using FIFO technique over different stages.

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
```

2.2 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 we specify the list of Tables (T) required we add all their names (Tn) to a list (AL.Tn). Afterwards, we 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 E_T and their summation leads us to the total time T_{tot} needed for MC algorithm to finish its job.

2.3 ROUND ROBIN

Our third technique is Round Robin (RR) which we implemented in two version 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. We 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.

2.3.1 TIME BASED ROUND ROBIN (TRR)

In the first version of RR we 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 we 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), we will 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 (E_T).At the end we add all those (E_T) together to have the total time T_{tot} to load the tables.

Algorithm 2: MC

```
Input: Database Tables at source or after Extraction or
Transformation Phase Output: Data loaded to
datawarehouse 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 */ fnd
For (each T ∈ 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
```

Algorithm 3: Time Based Round Robin

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

Output: Data loaded to datawarehouse 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*/*

2.3.2 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, we create a connection to all the tables to be transferred then as long as we didn't finish processing all the data we set our 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 we 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 (E_T) and add it to a list of all tables total execution time (Al.E_T). Finally when all tables are loaded we adjust our algorithm end round robin status (S) to "True" and from (AL.E_T) we get our Total Time (T_{tot}) of Round Robin based on records limit technique.

Algorithm 4: Records Limit Round Robin

```
Input: same as with previous algorithms beside specifying the Round Robin Records Limit

Output: Data loaded to datawarehouse 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 */*

III. SCHEDULING EXPERIMENTS

In this section, we discuss the experimental results of our proposed algorithms. The used data was from AdventureWorks 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. 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 we used specified number of records as our 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: 14 GB for data at sources, 6.3 GB at extract and 6.89 GB at Transform. From those experiments, we noticed as shown in figure 4.a and 4.b 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 figure 5.a and figure 5.b, 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 figure 6.a and 6.b, we concluded that it behaves best with smaller data set (as with Extracted Data Set).

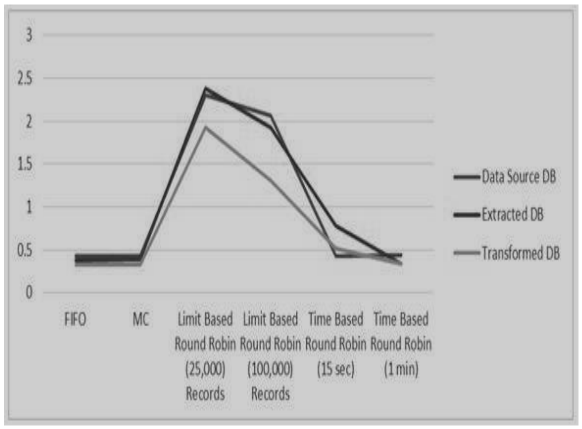

Figure 4.a Scheduling Techniques by Minutes for Data Loaded at Different Stages

FIFO	0.427844281	0.374048844	0.324229626
MC	0.432603715	0.388010876	0.329426132
Limit Based Round Robin (25,000) Records	2.306548228	2.386927216	1.927858971
Limit Based Round Robin (50,000) Records	2.490471779	2.333232646	1.794040655
Limit Based Round Robin (75,000) Records	2.398116768	2.238604187	1.682960375
Limit Based Round Robin (100,000) Records	2.071130743	1.922173568	1.307447341
Time Based Round Robin (15 sec)	0.423583402	0.77385139	0.512963346
Time Based Round Robin (30 sec)	0.432563809	0.332146974	0.369322797
Time Based Round Robin (45 sec)	0.421170103	0.382187158	0.338832748
Time Based Round Robin (1 min)	0.435068669	0.33775838	0.333256225

Figure 4.b Scheduling Techniques Statistics by Minutes for Data Loaded at Different Stages

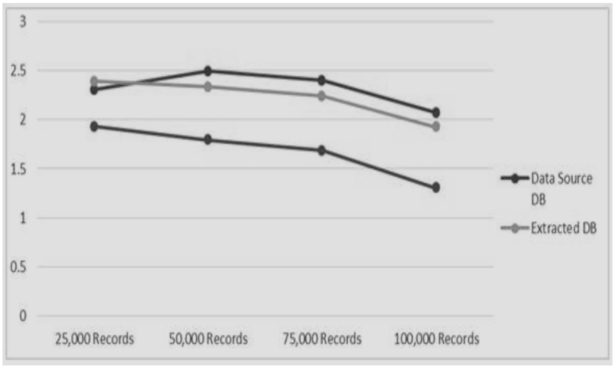

Figure 5.a Records Limit Based Round Robin for Data Loaded at Different Stages

FIFO	0.427844281	0.374048844	0.324229626
MC	0.432603715	0.388010876	0.319426132
Limit Based Round Robin (25,000) Records	2.306548228	2.386927216	1.927858971
Limit Based Round Robin (50,000) Records	2.490471779	2.333232646	1.794040655
Limit Based Round Robin (75,000) Records	2.398116768	2.238604187	1.682960375
Limit Based Round Robin (100,000) Records	2.071130743	1.922173568	1.307447341
Time Based Round Robin (15 sec)	0.423583402	0.77385139	0.512963346
Time Based Round Robin (30 sec)	0.432563809	0.332146974	0.369322797
Time Based Round Robin (45 sec)	0.421170103	0.382187158	0.338832748
Time Based Round Robin (1min)	0.435068669	0.33775838	0.333256225

Figure 5.b Records Limit Based Round Robin Statistics for Data Loaded at Different Stages

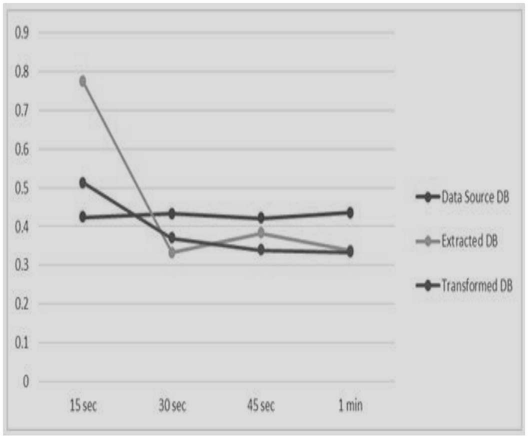

Figure 6.a Time Based Round Robin for Data Loaded at Different Stages

Time Based Round Robin	Data Source DB	Extracted DB	Transformed DB
15 sec	0.423583402	0.77385139	0.512963346
30 sec	0.432563809	0.332146974	0.369322797
45 sec	0.421170103	0.382187158	0.338832748
1 min	0.435068669	0.33775838	0.333256225

Figure 6.b Time Based Round Robin Statistics for Data Loaded at Different Stages

IV. CONCLUSION AND FUTURE WORK

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, we introduced a new approach to enhance performance using semantics for Extraction and Transformation which reside in the staging area just before the final loading phase. We proposed a semantics-based algorithm for each phase and presented the statistical results of applying those algorithms on the "Sales" schema. The data profiling tool results show that incorporating semantics in the staging area has an obvious impact on the quality of the resulting data that is presented to the loading phase. We also think that the ontology's presented in this work are tailored for the sales case study; however, different businesses imply different ontology's that need to be considered. As for loading data continuous attempts to select the best and most convenient approach to data transfer will vary depending on the data in hand as well as available resources such as CPU and main memory beside the urgency factor. In this work we 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. For future work, we would like to consider Minimum Memory (MM) and other algorithms. In addition implementing those algorithms in a distributed environment is also a possible direction for future work.

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.

Author:

Dr.M.Sreedevi,
Assistant Professor,
Dept.of.Computer Science
S.V.University,
Tirupati – 517 502
Andhra Pradesh,
India .

Mail-ID: msreedevi_svu2007@yahoo.com