Active Datawarehouse Loading by GUI Based ETL Procedure

V.Mallikarjuna Reddy[#], S.K.Jena[*], M.Nageswara Rao[#]

Computer Science, National Institute of Technology Rourkela, India

vmreddy89@gmail.com, s.k.jena@nitrkl.ac.in, chinninagesh@gmail.com.

Abstract: **Over the years, DataWarehousing has gone through a number of evolutions .from a relatively simple reporting database to sophisticated analytical applications such as analyzing customer lifetime values, market basket analyses, potentially defecting customers, fraud patterns, inventory churns, and so on. In all, though, these static sets of data could not give us the most current and recent changes necessary to *act* upon the results of Business Intelligence analyses. In this paper we present a methodology on how to adapt data warehouse schemas and loading the fresh data with the help of the GUI based ETL [12][13] (Extraction, Transformation and Loading) procedure for Active Data Warehouse. To accomplish this, we use techniques such as table structure replication and query predicate restrictions for selecting data, to enable continuously loading data in the data warehouse with minimum impact in query execution time.**

Keywords: **Real-time and active data warehousing, continuous data integration for data warehousing, data warehouse refreshment loading process, GUI based ETL.**

I. Introduction

A data warehouse (DW) provides information for analytical processing, decision making and data mining tools. A DW collects data from multiple heterogeneous operational source systems (OLTP– On-Line Transaction Processing) and stores summarized integrated business data in a central repository used by analytical applications (OLAP – On-Line Analytical Processing) with different user requirements.. To load the data into the Datawarehouses two types of techniques are used. Code based ETL and GUI based ETL. Here we are using the GUI based ETL technique to the continuous refreshments of the data in the DW. The ETL processes are responsible for identifying and extracting the relevant data from the OLTP source systems, customizing and integrating this data into a common format, cleaning the data and conforming it into an adequate integrated format for updating the data area of the DW and, finally, loading the final formatted data into its database.

Traditionally DataWarehouse stores the historical data. Here the data in the Datawarehouses are updated periodically typically in a monthly, weekly or daily. So, this type of Datawarehouses seems to be a Static Datawarehouses. Implying that its data is never up-to-date, for OLTP records saved between these updates are never recorded in the DataWarehouse. This implies that most recent operational records are not included in the DataWarehouse. So, for e-business, stock brokering, online telecommunications, for instance relevant information needs to be delivered as fast as possible to knowledge workers or decision systems who rely on it to react in a near real-time manner, according to the new and most recent data captured by an organization's information system. This makes supporting near RTDW a critical issue for such applications. This Data Warehouses are also called as Active DataWarehouses.

The main characteristics of Active Data warehousing environment.[1]

1. Active Data Warehousing means large volumes of data and, therefore, scalability becomes critical and absolutely required to provide the large amounts of detailed data needed to understand business events. Scalability also means the ability to support the concurrent queries described above. Therefore, the technology should provide the means to referee the contending actions and balance the conflicting needs of these various types of users.

2. The availability (and thus the reliability) is perhaps the most distinguishing characteristic of the technology to support both tactical and strategic queries. Traditional data warehouses do not usually have to be functional 24X7X365. As a result, the Active Data Warehouse must never go down, must never be inaccessible, or the business simply cannot operate.

3. Data Acquisition (ETL) must be performed much closer to the time a business event took place. Ideally, the acquisition mechanism will provide a continuous feed of new or changed data into the environment without blocking access to the very tables being updated.

Now a days the demand for fresh data in the DataWarehouse is increased. DataWarehouse refreshments are traditionally performed in an off-line fashion. This means that when this process is going on then the data in the DataWarehouse is not available to the OLAP users and applications to analyze the data. So, to overcome this RTDW (Active DW) are coming into picture. *Active* Data Warehousing [11] refers to a new trend where DWs are updated as frequently as possible, due to the high demands of users for fresh data.

So, to the continuous loading of the data into the Active DataWarehouse several issues needs to be taken considerations. (1) Operational OLTP systems are designed to meet well specified (short) response

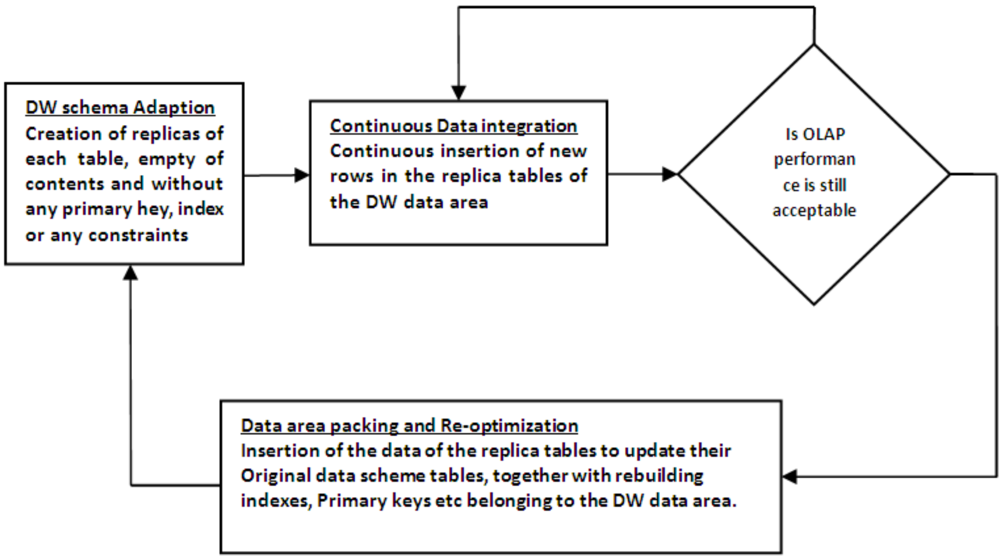

DW schema Adaption
Creation of replicas of each table, empty of contents and without any primary hey, index or any constraints
Continuous Data integration
Continuous insertion of new rows in the replica tables of the DW data area
Is OLAP performance is still acceptable
Data area packing and Re-optimization
Insertion of the data of the replica tables to update their Original data scheme tables, together with rebuilding indexes, Primary keys etc belonging to the DW data area.

Fig 1: Architecture for proposed ETL procedure

time requirements aiming for maximum system availability, which means that a RTDW scenario would have to cope with this in the overhead implied in those OLTP systems; (2) The tables existing in a data warehouse's database directly related with transactional records (commonly named as fact tables) are usually huge in size, and therefore, the addition of new data and consequent procedures such as index updating or referential integrity checks would certainly have impact in OLAP systems' performance and data availability. Our work is focused on the DW perspective, for that reason we present an efficient methodology for continuous data integration, performing the ETL loading process.

This paper presents a solution which enables efficient continuous data integration in data warehouses, while allowing OLAP execution simultaneously, with minimum decrease of performance. Here we are designing the target tables, required Transformations and loading procedures with the help of Informatica tool to meet the requirements of the Active DataWarehouse. i.e. we are designing the ETL procedures to the continuous loading of the data into the DataWarehouse. And at the same time this updated data is always available to the OLAP users and Applications.

To construct the ELT procedures we are using Ricardo Methodologies. In this methodology Based on the existing schema(s) of the DW's database, our methodology consists on creating a replica for each of its tables, empty of contents and without defining any type of indexes, primary keys or any other kind of restrictions or constraints. In this new schema, the replicated tables will receive and record the data from the staging area, which will be continuously loaded. The fact that they are initially created empty of contents and not having any sort of constraints allows to considerably minimizing the consumption of time and resources which are necessary in the procedures inherent to data integration. As the data integration is performed, the database's performance which includes the new most recent data deteriorates, due to the lack of usual data structures that could optimize it (such as indexes, for example), in the replicated tables. When the data warehouse's performance is considered as unacceptable by its users or administrators, the existing data within the replicated tables should serve for updating the original schema. After performing this update, the replicated tables are to be recreated empty of contents, regaining maximum performance. We also demonstrate how to adapt OLAP queries in the new schema, to take advantage of the most recent data which is integrated in real time.

II. Related Work

So far, research has mostly dealt with the problem of maintaining the warehouse in its traditional periodical update setup [5,9]. Related literature presents tools and algorithms to populate the warehouse in an off-line fashion. In a different line of research, data streams [1, 2, 7,8] could possibly appear as a potential solution. However, research in data streams has focused on topics concerning the front-end, such as on-the-fly computation of queries without a systematic treatment of the issues raised at the back-end of a data warehouse [10]. Much of the recent work dedicated to RTDW is also focused on conceptual ETL modelling [4, 5, 11,12] lacking the presentation of concrete specific extraction, transformation and loading algorithms along with their consequent OLTP and OLAP performance issues.

In [4] the authors used code based ETL to the continuous loading of the data in to the Active DataWarehouse. So, they are writing corresponding procedures and triggers for the tables of the Active DataWarehouse in SQL and PL/SQL. They are written the coding for these entire tables with the considerations of the Indexing of the tables and all the constraints. In [5], the authors describe the ARKTOS ETL tool, capable of modelling and executing practical ETL scenarios by providing explicit primitives for capturing common tasks (such as data cleaning, scheduling and data transformations) using a declarative language. ARKTOS offers graphical and

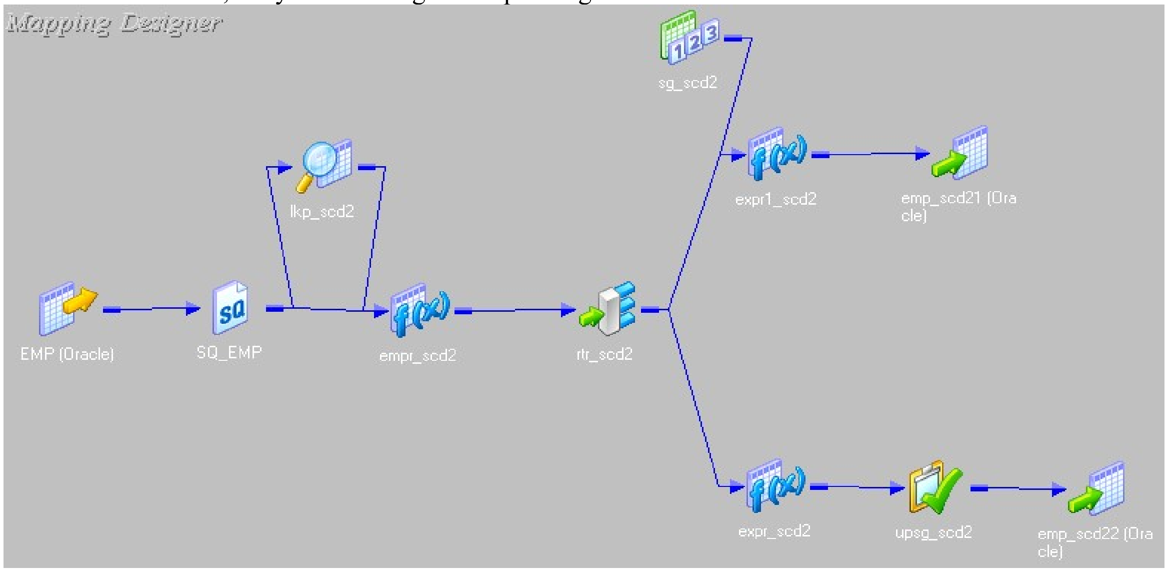


Fig.2. proposed ETL procedure mapping

Declarative features for defining DW transformations and tries to optimize the execution of complex sequences of transformation and cleansing tasks. Our techniques overcome some of the mentioned drawbacks and are presented in the next section.

III. Architecture For Proposed ETL Procedure

From the DW side, updating huge tables and related structures (such as indexes, materialized views and other integrated components) makes executing OLAP query workloads simultaneously with continuous data integration a very difficult task. Ricardo methodology minimizes the processing time and workload required for these update processes. It also facilitates the DW off-line update. Because the data already lies within the data area and all OLTP data extraction and/or transformation routines have been executed during the continuous data integration. Furthermore, the data structure of the replicated tables is exactly the same as the original DW schema. This minimizes the time window for packing the data area of the DW, since its update represents a one step process by resuming itself as a cut-and-paste action from the temporary tables to the original ones. Based on this architecture we are proposing the ETL procedures. The procedures and the description of the architecture are given in the section 4.

IV. Proposed ETL Procedures

To load the data in the Active Datawarehouse we are preparing the target replica tables and required transformations and loading procedure with the help of Informatica tool. The description is given in following sub sections.

A. Adapting the DataWarehouse Schema

Creation of an exact structural replica of all tables of the DataWarehouse that could eventually receive the new data. These tables (referred also as temporary tables) are to be created empty of contents, with no defined indexes, primary key, or constraints of any kind, including referential integrity. For each table, an extra attribute must be created, for storing a unique sequential identifier related to the insertion of each row within the temporary tables.

The main idea behind the creation of replica tables is, OLAP operations will take less time to execution. Because in the replica tables there is no indexing and no primary key constraints and this replica tables contains less amount of data. So, automatically an OLAP operation takes less time to execution. And OLAP must be changed to get the replica tables data. Because replica tables contains the fresh data. But in this paper we are concentrating on only the ETL procedures.

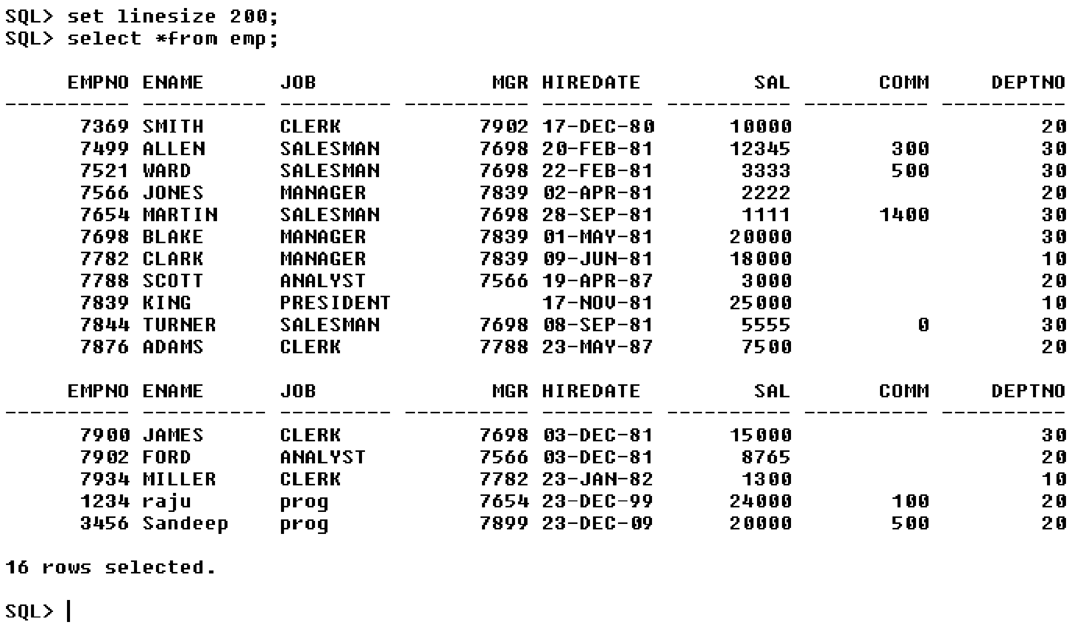
Here each replica table must contain one counter value. This counter value maintains the latest version of the record. And here this column is called as Surrogate key. The OLAP operations will take the latest version of the record based on this surrogate key.

B. ETL mappings

To refresh the data warehouse, once the ETL application has extracted and transformed the OLTP data into the correct format for loading the data area of the DW, it shall proceed immediately in inserting that record as a new row in the correspondent temporary table, filling the unique sequential identifier attribute with the auto incremented sequential number. This number should start at 0 for the first record to insert in the DW after executing the packing and Reoptimising technique, and then be auto incremented by one unit for each record insertion.

We are using Informatica tool for this ETL procedures. And this is GUI based ETL. We are taking Employee DataWarehouse as the example here. So, the corresponding mapping for this Employee DW is given below. And the explanation for this mapping is in the following sections.

In the above mapping [Fig 2] Emp is the source table. And emp_scd21, emp_scd22 are the replica tables of the Data Warehouse Schemas. And here emp_scd21 table gets the new records from the source data. Emp_scd22 will get the update records only. Here the following transformations are used in the proposed ETL procedure.



```
SQL> set linesize 200;
SQL> select *from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80      10000                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81      12345        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       3333        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2222                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1111       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81      20000                    30
      7782 CLARK      MANAGER         7839 09-JUN-81      18000                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81      25000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       5555          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       7500                    20

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7900 JAMES      CLERK           7698 03-DEC-81      15000                    30
      7902 FORD       ANALYST         7566 03-DEC-81       8765                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
      1234 raju       prog            7654 23-DEC-99      24000        100         20
      3456 Sandeep    prog            7899 23-DEC-09      20000        500         20

16 rows selected.

SQL> |
```

Fig.3 Input source data for the proposed ETL prcedure

1) Source Qualifier Transformation: This is of type an active transformation which allows us to read the data from OLTP Systems. Here the OLTP sources may be different databases or flat files (.txt, .doc). And this Source Qualifier Transformation will generate the corresponding SQL queries to get the data from different OLTP systems. And in this Transformation we can also change the default SQL queries as per our requirements. With this transformation, if the source data is coming from same Databases then directly we can join that data.

2) Lookup Transformation: We can use a Lookup transformation in a mapping to look up data in a flat file or a relational table, view, or synonym. Use multiple Lookup transformations in a mapping. Here in this ETL procedure we are using Lookup transformation to lookup the target Emp table. So, here the source records are checked with the target records which are already available in the target table. The source Emp records are checked with the target Emp records. If the record is not there in the target table then it is inserted in the emp_scd21 target table with the version number as 1. And if the source record is already available in the target table then it is treated as updated record and the corresponding version number is increased by one and it is inserted in emp_scd22.

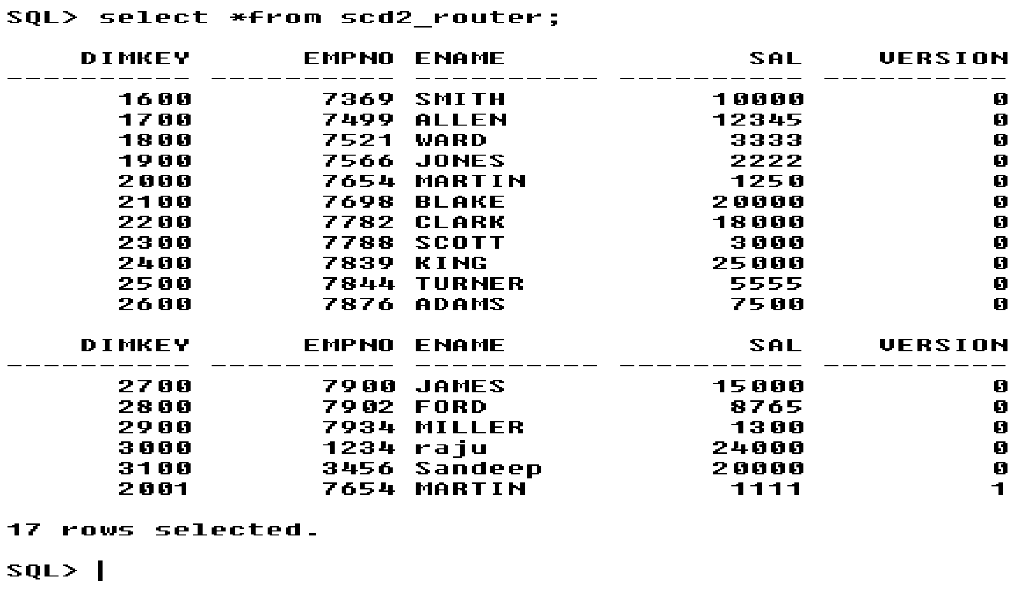
3) Expression Transformation: We can use the Expression transformation to calculate values in a single row before you write to the target. For example, you might need to adjust employee salaries, concatenate first and last names, or convert strings to numbers. Use the Expression transformation to perform any non-aggregate calculations. You can also use the Expression transformation to test conditional statements before you output the results to target tables or other transformations. Here we are using the expression transformation to create the two new fields. These are New and Update. These two attributes are helpful to find out whether the record is new or update with the help of Lookup transformation.

4) Router Transformation: A Router transformation is allow you to use a condition to test data. A Router transformation tests data for one or more conditions and gives you the option to route rows of data that do not meet any of the conditions to a default output group. So, here the Router Transformation takes the conditional attributes as New and Update which are generated in Expression transformation. So, the Router transformation checks the condition and if it is new it will route the source record to the emp_scd21 target table. And if it is Update then it will route the updated record to the emp_scd22 target table.

5) Sequence Generator Transformation: The Sequence Generator transformation generates numeric values. Use the Sequence Generator to create unique primary key values, replace missing primary keys, or cycle through a sequential range of numbers. Here we are using Sequence Generator Transformation to generate the version number of the source records. It will generate the version numbers from 1. Here if the source record is new then the version number for that record is 1. And if the record is updated record then the version number for that record is incremented it's old version number by one. And this acts as a surrogate key.

6) Update Strategy Transformation: This is of type Active Transformation which flags the records for insert, update, delete and reject data driven operations. Here if the source record is updated record then directly with the help of this transformation the corresponding source record is updated in the target table emp_scd22.

After creating this mapping we will create a session variable for this mapping. With the help of the session variable the mappings are executed and the source data is loaded into the target tables. And the session variable contains scheduling properties. In the scheduling properties we are giving the scheduling time as continuous. So, this session variable is executed continuously and the source data is loaded continuously in the Active DataWarehouse. So, the fresh data is also available to the OLAP users and applications in the Active DataWarehouse.


```
SQL> select *from scd2_router;

    DIMKEY      EMPNO ENAME             SAL    VERSION
---------- ---------- ---------- ---------- ----------
      1600       7369 SMITH           10000          0
      1700       7499 ALLEN           12345          0
      1800       7521 WARD             3333          0
      1900       7566 JONES            2222          0
      2000       7654 MARTIN           1250          0
      2100       7698 BLAKE           20000          0
      2200       7782 CLARK           18000          0
      2300       7788 SCOTT            3000          0
      2400       7839 KING            25000          0
      2500       7844 TURNER           5555          0
      2600       7876 ADAMS            7500          0

    DIMKEY      EMPNO ENAME             SAL    VERSION
---------- ---------- ---------- ---------- ----------
      2700       7900 JAMES           15000          0
      2800       7902 FORD             8765          0
      2900       7934 MILLER           1300          0
      3000       1234 raju            24000          0
      3100       3456 Sandeep         20000          0
      2001       7654 MARTIN           1111          1

17 rows selected.

SQL> |
```
Fig.4 The fresh data after change in the source data

V. Packing And Re-optimizing The Datawarehouse

Since the data is being integrated within tables that do not have access optimization of any kind that could speed up querying, such as indexes, for instance, it is obvious that its functionality is affected, implying a decrease of performance. Due to the size of physically occupied space, after a certain number of insertions the performance becomes too poor to consider as acceptable. To regain performance in the DW it is necessary to execute a pack routine which will update the original DW schema tables using the records in the temporary tables, and recreates these temporary tables empty of contents, along with rebuilding the original tables' indexes and materialized views, so that maximum processing speed is obtained once more.

For updating the original DW tables, the rows in the temporary tables should be aggregated according to the original tables' primary keys, maintaining the rows with highest unique counter attribute value for possible duplicate values in non-additive attributes, for they represent the most recent records. The time needed for executing these update procedures represents the only period of time in which the DW in unavailable to OLAP tools and end users, for they need to be executed exclusively. The appropriate moment for doing this may be determined by the Database Administrator, or automatically, taking under consideration parameters such as a determined number of records in the temporary tables, the amount of physically occupied space, or yet a predefined period of time. The determination of this moment should consist on the best possible compromise related to its frequency of execution and the amount of time it takes away from all user availability, which depends on the physical, logical and practical characteristics inherent to each specific DW implementation itself and is not object of discussion in this paper. This Packing and Re-optimizing will be done only the OLAP operations performance is not acceptable with the replica tables.

VI. Result

The source table [Fig 3] contains the Employee data. Suppose in the source table data there is a change in the salary of the employee Martin. So, these changes are captured by the target DataWarehouse with the help of the proposed ETL mapping. In this way if there is a continuous change in the source Database then all these changes are captured by the Target DataWarehouse with the zero latency. And here to get the fresh data continuously the session variable for that mapping is running continuously with the help of schedulers. And the target DataWarehouse identifies the fresh data with the version number. In this example Martin's salary is changed so the version number is increased to 1 [Fig.4].

VII. Conclusion And Future Work

This paper presents a GUI based ETL procedure to the continuous loading of the data in the Active DataWarehouse by enabling continuous data integration while minimizing impact in 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 real time data warehousing schemas.

And for designing the mappings for this Active Data Warehouse in code based ETL takes long time, execution of the this method will consume more time but in this GUI based ETL design takes very less time, for the execution of these mappings also takes less amount of time compared to code based ETL. In code based ETL we need to develop the procedures and triggers, this consumes the maximum time but we don't require it in GUI based ETL. So, automatically the performance of the OLAP queries and Applications is improved. And with the help of schedulers we can run the session variables continuously. So, the fresh data is loaded and is available for OLAP users and applications.

As future work we intend to develop an ETL tool which will integrate this methodology with extraction and transformation routines for the OLTP systems. There is also room for optimizing the query instructions used for our methods.

References

[1] Ricardo Jorge Santos and Jorge Bernardino," *Real-Time DataWarehouse Loading Methodology*", IDEAs'o8, ACM 978-1-60158-188-0/08/09,2008.

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

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

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

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

[6] R. M. Bruckner, and A. M. Tjoa, "*Capturing Delays and Valid Times in Data Warehouses – Towards Timely Consistent Analyses", Journal of Intelligent Information Systems (JIIS),* 19:2, pp. 169-190, 2002.

[8] Araque, F., Samos, " *Data warehouse refreshment maintaining temporal consistency"*, 5th International Conference on Enterprise Information Systems, ICEIS´03. Angers, France, 23-26, April 2003.

[9] Ramamritham, R. Sivasankaran, J. A. Stankovic, D. T. Towsley and M. Xiong, "Integrating Temporal, Real-Time, and Active Databases", ACM Sigmod Record, Vol. 25. No. 1, March 1996, pp. 8-12., 2002.

[10] Oracle Corp. Oracle9i™ Warehouse Builder User's Guide, Release 9.0.2. November 2001.

[11] Paton, N.W., Diaz, "*Active Database Systems*". ACM Computing Surveys (CSUR) 31(1), pp63-103, 1999.

[12] Karakasidis, A., Vassiliadis and P., Pitoura, *" ETL Queues for Active Data Ware- housing."*, Proceedings of the 2*nd* International Workshop on Information Quality in Information Systems (IQIS'2005), New York, USA, ACM Press 28-39,2005.

[13] Dr. R.S.Chhillar, BarjeshKochar, "*Extraction Transformation Loading-A Road to DataWarehouse*", 2nd National Conference Mathematical Techniques: Emerging Paradigms for Electronics and IT Industries, sep 26-28,2008.