ISSN No. 0976-5697

Volume 1, No. 3, Sept-Oct 2010

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

DIDW: An Approach for Dynamic Updating in Intelligent Data Warehouse

N. Badal* and A. K. Agarwal
Department of Computer Science & Engineering,
Kamla Nehru Institute of Technology,
Sultanpur, KNIT, U.P, INDIA.
n_badal@hotmail.com, abhay.knit08@gmail

Abstract: Transfer of information from source database into data warehouse i.e. data updating is one of the most important issue for discussion and research. Organizations that are going through it are solving by deploying dynamic and intelligent data warehouse in real-time manner. Overall strategy in dynamic and intelligent data warehouse relies on updates that are made to source database. These updates in source database are then stored in some temporary storage device through which they are transferred to data warehouse. This paper proposes a Dynamic and Intelligent Data Warehouse (DIDW) architecture with its implementation, using multi real-time data cache. Data cache stores updated data retrieved by update data translator from expandable horizontally partition database. Thus it is analyzed and suggested that using this DIDW architecture in real-time manner helps to the organizations.

Keywords: DIDW, UDT, EHPD, RTDC

I. INTRODUCTION

Now a day there is cut-throat competition among the various organizations. Information which is nothing but meaningful data plays a vital role for such organizations. Those organizations will have an added advantage over the others that are able to access meaningful data in a timely and efficient manner. Sharing of data among the various departments of an organization and moving of data to business partners [7] is equally important.

Data warehouse is subject oriented, non-volatile, integrated and time-variant collection of large data set. Extraction Transformation and Load (ETL) tool is used to replenish data warehouse from source databases termed as informational data. This informational data is valuable for Decision Support System (DSS). Operational data on the other hand is detailed, non redundant and updateable [1]. Operational data requires a dynamic and intelligent data warehouse for which the necessity for data warehouse is to handle this data in real time manner. Real-time data warehouse is the combination of the real-time behavior and the data warehouse [6]. Many researchers proposed Capture-Conversion-Flow process to update data in real time data warehouse [9].

The related work is being given in the next section of this paper followed by the warehouse problems. The DIDW architecture with its implementation is being proposed in the subsequent sections. Analysis and outcomes are given in the last of this paper.

II. RELATED WORK

Much have been talked and lots of research been done on data updating in real-time data warehouse. Many researchers had proposed various architectures about the same. Few of which being are discussed here. Joseph H. H. et.al. in [4], used an auxiliary store between source database and data warehouse so that, updated data in source database doesn't allowed to move directly to data warehouse. By doing so it prevented the data warehouse not to go offline each time source database is updated. When any query is fired for updated data, the response of it is given by the auxiliary storage, making the system fast.

In one of the other works Youchan Zhu et.al. in [10], proposed feasible real-time data warehouse architecture based on SOA. In this work various changed data was captured by the data capture web service. The update strategy was based on web service. Multi-level cache was used for real-time data storage which play role to give response to real-time query. As various cache were connected in series to one another, system will crash if any of the cache gets unstable or crashes, this was the biggest drawback of the proposed system.

III. DATA WAREHOUSE UPDATE PROBLEM

The basic approach in performing updates to a data warehouse is to collect all the updated information from the source database at some pre-calculated time, most preferable will be during the least active period of the day, transfers the information to the data warehouse, and view updates immediately. Not only this method is disruptive, but also the information stored in the data warehouse system will always be somewhat out of date with the data stored in the source database [5]. The more conventional approach used in performing updates to data warehouses has been to gather the necessary information from the source database and immediately replenish the data warehouse with updates. Once this data has been replenished, the user's views are updated and the work of the data warehouse carries on [11]. Regardless of the method chosen to perform the data warehouse update function, there is bound to be some impact on the data warehouse users while the updates are being executed. As source database changes at a very rapid rate and data warehouse as a repository for huge amount of diverse information, the correctness of these query response provided by the warehouse is highly dependent on the changes made to the source database.

There are three traditional methods to obtain the updated data required by the data warehouse.

- One of the simplest methods is the modification of the source database applications to provide the updated data set of interest to the data warehouse.
- Obtaining access, and extracting the required information, from the source database log or a specially constructed 'delta' file [3]. The advantage to this implementation is that the updates necessary for the data warehouse can be gathered without any impact to the source database.
- Maintaining a snapshot of the source database and comparing it with the current contents of the source database. Any changes that have been made in the source database require the changed data to be extracted and forwarded to the data warehouse. The archived snapshot of the source database is then updated to show the data warehouse's current view of the source database [3, 5].The advantage to this method is, the work of providing updated information to the data warehouse can be done with no impact to the activities of the source database. The snapshot data then can be forwarded to the data warehouse [8].

IV. DIDW ARCHITECTURE

Figure1 illustrate the multi-cache and a switch based data warehouse architecture, having mainly 'n' Expandable Horizontally Partitioned Database (EHPD), 'n' Update Data Translator (UDT), 'n' multi Real-Time Data Cache (RTDC), a Switch, a large Cache, Real-Time Data Integrator (RDI), Data Warehouse, ETL, OLAP server and various applications.

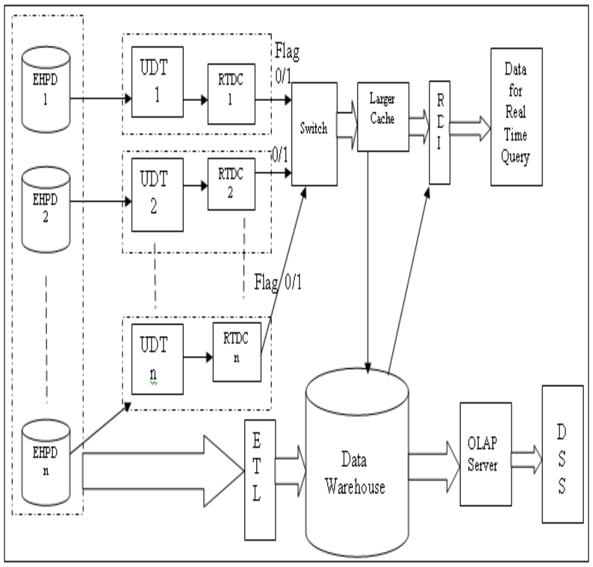

Figure 1: DIDW Architecture

In proposed architecture for DIDW there are 'n' expandable horizontally partitioned database. Each of these expandable horizontally partitioned database store equal number of data set in it. Collectively it is called source database. The stored data set in expandable horizontally partitioned database goes to data warehouse through ETL. An OLAP server is connected to data warehouse .The function of OLAP server is analytical process of data in a data warehouse which is used in various applications such as decision support system.

When new data set comes they enter in random fashion to various expandable horizontally partitioned databases. The 'n' UDT are connected one to one with 'n' EHPD. The individual update data translator extracts the update data set if available from respective EHPD. The tasks of UDT are update data extraction, data cleaning, data transformation, and loading [2]. The 'n' RTDC are connected one to one with 'n' update data translators. The data set that was extracted by individual update data translator moved and stored in respective RTDC. Each RTDC generates the flag with value of either 0 or 1. Flag value with '1' signifies that there are update data set in RTDC and flag value with '0' there is no update data set in RTDC.

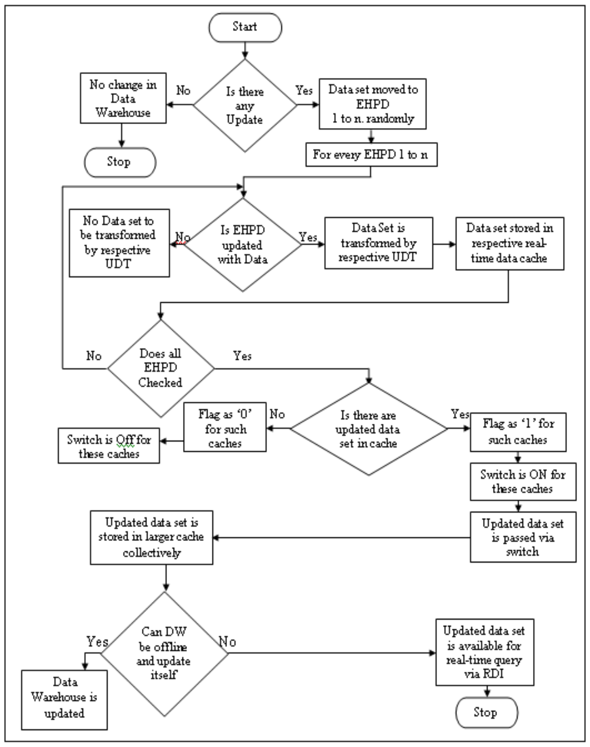
Start
Is there any Update
No
Yes
No change in Data Warehouse
Stop
Data set moved to EHPD 1 to n. randomly
For every EHPD 1 to n
Is EHPD updated with Data
No
Yes
No Data set to be transformed by respective UDT
Data Set is transformed by respective UDT
Data set stored in respective real-time data cache
Does all EHPD Checked
No
Yes
Is there are updated data set in cache
No
Yes
Flag as '0' for such caches
Switch is Off for these caches
Flag as '1' for such caches
Switch is ON for these caches
Updated data set is passed via switch
Updated data set is stored in larger cache collectively
Can DW be offline and update itself
Yes
No
Data Warehouse is updated
Updated data set is available for real-time query via RDI
Stop

Figure 2: Flow Chart for Dynamic Update

The flag with value '1' will be passed by the switch and flag with value '0' will be blocked thus updated data from various real-time data cache is passed to larger cache and each such real-time data cache is refreshed, this larger cache store all update data set collectively. Whenever data warehouse is offline it gets updated by data set in this cache. Otherwise the collective data set is available for real-time query via real-time data integrator. The data will remain in the larger cache and continuously will be added in it until data warehouse is updated by it or the cache is full by 90% at which data warehouse is forced to get itself updated. Once data warehouse gets updated the larger cache will be refreshed and emptied. The flowchart for procedure and working of architecture is shown in figure 2.

V. WORKING PROCESS OF DIDW ARCHITECTURE

Figure 3 illustrate step-by-step description of the working of the DIDW architecture process, starting from the entering of updated data set in various EHPD. It transitioned by UDT, moving to RTDC. Finaly, moved to the larger cache via a switch, which remain ON for those RTDC having flag value as '1' means updated data set in this RTDC.

```
Step 1: Process Starts
Step 2: Is there are new data set, and source database to be updated.
    2(a) If there's no update there will be no change in data warehouse.
    2(b) If there are updates then updated data set moved to any EHPD 1 to n randomly.
Step 3: Every EHPD 1 to n, will be checked to see, whether a EHPD is updated with data set or not.
    3(a) If No there will be no data set for UDT to be captured and transformed.
    3(b) If Yes, Data Set is captured and transformed by respective UDT.
Step 4: Data set that are transformed by UDT are moved and stored to respective real-time data cache.
Step 5: Now a condition will be applied to see whether all EHPD are checked or not.
    5(a) If No step 3 and step 4 will be repeated.
    5(b) If Yes process will flow to check Real-time data cache.
Step 6: Every Real-time data cache will be checked for updates depending on the updates.
    6(a) If No, Flag with value '0' will be generated means switch will be OFF for these caches.
    6(b) If Yes, Flag with value '1' will be generated means switch will be ON for these caches.
Step 7: Updated data set from real-time cache with flag value of '1' will be passed via switch and each such real-time data cache is refreshed and emptied.
Step 8: Updated data set is moved and stored in larger cache collectively.
Step 9: Can data warehouse goes offline and update itself with the data set stored in larger cache.
    9(a) If Yes data warehouse is updated.
    9(b) If No updated data set is available for real-time query via RDI.
Step10: If larger cache is full by 90% with the updated data set it will force the data warehouse to becomes offline and update itself via data set thus cache is refreshed and emptied.
Step 11: Process Stop.
```

Figure 3: Algorithm for Dynamic Updation in DIDW

VI. ANALYSIS AND OUTCOMES

The analysis and viewed outcomes of the proposed DIDW architecture are being illustrated here.

1. The reliability of system is high as set of various elements are connected in parallel, if in a set of elements any of it fails system doesn't crash.
2. The cache which store collective updated data set is large in size the query load of cache will not increase making system stable.
3. The architecture can be extended for distributed system.
4. The architecture uses 'n' EHPD in which updated data goes randomly. Those EHPD in which data was moved involve in further processing and others are ready to accept new updated data. Thus there is no delay between two updating process.

VII. CONCLUSION AND FUTURE SCOPE

The various methods of data warehouse update with there drawbacks are discussed in this paper. It is also discussed about the conventional methods that how to obtain the updated data set required by data warehouse. The proposed DIDW architecture in real-time manner allows the current data set to move to the data warehouse. In this architecture there is facility for data warehouse that when it can go offline it can fetch updated current data from larger cache. Thus, using this DIDW in real-time manner provides a good assistantship to the organizations.

VIII. REFERENCES

[1] A. Berson, Stephen J. Smith, "Data Warehousing, Data Mining, & OLAP" Fourth ed. Tata McGraw Hill Publication, Page 14-15.

[2] A. K. Pujari, "Data Mining Techniques" Fourteenth Impression 2008. Universities Press, Page-30.

[3] Hanson, Joseph H. An Alternative Data Warehouse Structure for Performing Updates.December 1996, UMI Press.

[4] Hanson Joseph H. Modeling a Faster Dara Warehouse, IEEE Journal, April 1997, pages 260-265.

[5] Inmon, W.H., Building the Data Warehouse. Second ed. Vol. 1. 1996, New York: Wiley Computer Publishing.pg. 401.

[6] J. Langseth. Real-Time Data Warehousing: Challenges and Solutions. DSSResources.COM, 004.

[7] John Vandermay. Considerations for Building a Real-time Data Warehouse. Data Mirror Corporation. <http:// www.dmreview.com/>.

[8] Labio, W.J. and H. Garcia-Molina, Efficient Snapshot Diferential Algorithms for Data Warehousing, Technical Report. 1996, Stanford Univ: Palo Alto.

[9] YANG Le. Design and Implementation of Real-time data extraction Mechanism in data warehousing [D].Beijing: Beijing University of Posts and Telecommunications .2007.03.

[10] Zhu Youchan, Lie An and Liu Shuangxi, Data Updating and Real-time Data Warehouse System, IEEE Conference, August 2008, pages 1295-1297.

[11] Zhuge, Y.H.G.-M., Jachim Hammer, Jennifer Widom, View Maintenance in a Warehousing Environment Technical Report. 1994. Stanford: Palo Alto.

AUTHORS

N. Badal is a Sr. Lecturer in the Department of Computer Science & Engineering at Kamla Nehru Institute of Technology (KNIT),Sultanpur (U.P), INDIA. He received B.E. (1997) from Bundelkhand Institute of Technology (BIET), Jhansi in Computer Science & Engineering, M.E. (2001) in Communication, Control and Networking from Madhav Institute of Technology and Science (MITS), Gwalior and PhD (2009) in Computer Science & Engineering from Motilal Nehru National Institute of Technology (MNNIT), Allahabad. He is Chartered Engineer (CE) from Institution of Engineers (IE), India. He is a Life Member of IE, IETE, ISTE and CSI-India. He has published about 30 papers in International/National Journals, conferences and seminars. His research interests are Distributed System, Parallel Processing, GIS, Data Warehouse & Data mining, Software engineering and Networking.

A. K. Agarwal is a Lecturer in Computer Science & Engineering Department at KNIT, Sultanpur. He received his B.Tech degree in 1999 from BIET Jhansi in Electronics & Instrumentation Engg and M.Tech in 2006 from Samrat Ashok Technology Institute (SATI), Vidisa in Information Technology.