Refreshing Datawarehouse in Near Real-Time

Tanvi Jain
Centre for Development of Advanced Computing
Noida, India

Rajasree S
Centre for Development of Advanced Computing
Noida, India

Shivani Saluja
Centre for Development of Advanced Computing
Noida, India

ABSTRACT
Data warehousing technology has made a huge impact in the world of business; it helps to turn data into information that helps analysts to make strategic decisions. Currently most data warehouse approaches employ static refresh mechanisms. But for various business requirements this is not an appropriate solution. Some critical data need to be refreshed in real time. We propose an approach to identify critical data by considering two factors, namely: a) impact from one update, b) number of records affected. The identified critical data will be stored in the temporary tables, these temporary tables will be refreshed in real time and remaining data will be refreshed in conventional way.

Keywords
Near Real-time data warehouse, Change Data Capture (CDC), Extract, Transform and Load (ETL)

1. INTRODUCTION
Data warehouse is a database that is managed separately from organization's operational database. It is used in an organization to collect data from one or more data sources into a central data location, known as the Data Warehouse, and later it is used to report those data, in user specified format, to business users in the organization. A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process. Some of the applications were data warehousing can be used are: decision support, trend analysis, financial forecasting, insurance fraud analysis, call record analysis, logistics and inventory management, agriculture.

ETL in data warehousing stands for extracting data from source systems, transforming the data according to the business rules and loading to the target data warehouse. ETL plays a significant role in data warehouse construction. To perform ETL many tools are available such as Informatica, Pentaho etc.

Data warehouse can be refreshed in many different ways. The mechanism of refreshing the warehouse is very important in deciding the reliability of warehouse's content. Traditionally data warehouse were refreshed at specific intervals, e.g., every night, weekly, or monthly. Due to demand for fresh data in data warehouse we need to shift from traditional data warehouse to real time data warehouse.

Some of the applications for real time data are, 1) most recent information is required to detect suspicious group of passengers in airlines, 2) in banking systems real time data is required for certain critical areas of operations such as auditing systems and anti money laundering, 3) to detect ATM frauds also we need real time data.

The objective of the paper is creation of near-real time data warehouse for applications which needs real time data by developing a decision support system that aims for identifying critical data and refreshing this critical data in near-real time.

2. MOTIVATION FOR NEAR REAL-TIME DATA WAREHOUSE
Data warehouse can be refreshed either in traditional way (i.e. at specific intervals) or it can be refreshed in real time (i.e. after every new record warehouse is refreshed). Both techniques have its cons and pros. The disadvantage of traditional methodology is that the data content is not updated; therefore wrong decisions are likely to be made. One more disadvantage is vastness of data. The disadvantage of real time refreshing methodology is that it affects source OLTP systems as it puts additional workload.

Therefore, we need near real time data warehouse where some data is refreshed in real time and remaining data to be refreshed in traditional way.

By refreshing data warehouse using this near real time methodology, neither there will be overload on the source system because only critical data will be extracted from the source system frequently nor strategic decisions would be made using old data.

3. ARCHITECTURE OF NEAR REAL-TIME DATA WAREHOUSE
The architecture of near real-time data warehouse includes log files which are read by change data capture (CDC) module to find new records inserted or modifications. These new records or modifications are placed in data queue then on this data queue procedure is applied which identifies the critical data. Based on this identification two separate queues are maintained, one for critical data and another for non-critical data. Next the data before loading into data warehouse are cleansed and transformed according to the business rules.

For identifying critical data aspects are considered, 1) update significance, 2) number of records affected.

The architecture of near real-time data warehouse is shown in Figure 1.

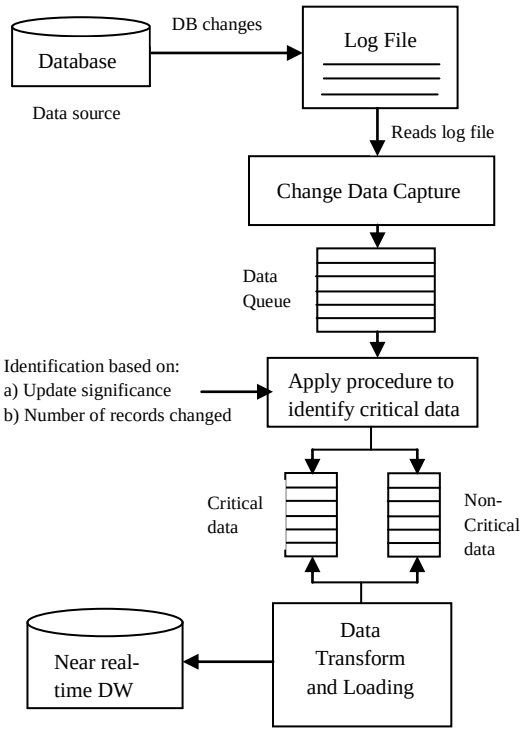

Figure 1: Near Real-Time DW Architecture

4. CHANGE DATA CAPTURE (CDC)

Change data capture is a mechanism to extract changes from source database, i.e., it identifies data that has been added to, updated in or removed from source systems.

Change data capture can capture modified data in two modes:

- Synchronous: Modified data is captured immediately. This is performed by using triggers on OLTP tables. This type of capturing mode effects OLTP as it is a part of transaction processing.
- Asynchronous: Modified data is captured after SQL statement performs its operation. Then after committing the operation data is written into log files. These log files are read to capture modified data. It has no effect on OLTP.

Some advantages of Change data Capture are:

- Completeness: CDC can capture all effects of insert, update and delete operations. Also it can capture change in data value, if any, after modification.
- Performance: Asynchronous change data capture has minimal impact on the source database.
- Cost: Change data capture reduces overhead cost because it simplifies the extraction process as it extracts only change data from the database.
- Makes ETL process efficient.
- Data warehouse process and operational systems are completely decoupled from each other. This means any operational failure or recovery in either of the systems will not affect one another.

Change data capture (CDC) must be integrated with ETL tools so that ETL process can be made efficient. Implementing Change data capture with ETL tools makes ETL process fast.

In design structure of CDC along ETL tools (as shown in Figure 2) ETL tools read changed data from CDC then performs all cleaning and transformations on data and finally loads it in the warehouse.

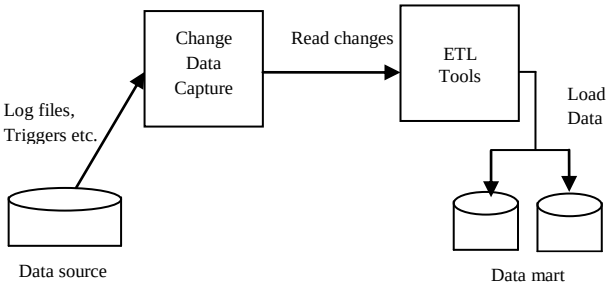

Figure 2: Implementation of CDC along with ETL tools

There are various ways to implement CDC. Following are some of the techniques to capture change data:

- Triggers: a trigger is a special kind of stored procedure that is invoked whenever an attempt is made to modify the data in the table against which this trigger is written. Disadvantage of using triggers for CDC is that it puts overhead burden on the source database.
- Date & Timestamps: Each table in source database contains a field named as 'Last modified'. It stores the date and time when the data was lost modified. CDC reads this last modified field and captures the most recent modified data. Disadvantage of using this CDC approach is that if program fails to write into 'Last Modified' field then some modified data can be missed.
- Log – based CDC: Database maintains log files to minimize the loss of data in case of an uncontrolled shutdown. Speeds up the diagnostic process. Log files contain detail of all the transactions which has taken place on OLTP tables. Only log-based CDC is one approach through which all changes can be captured and updated into data warehouse and also log-based technique has minimal impact on source database.

Due to advantages of log-based technique we will use this as CDC technique in our paper.

Table 1 show the comparison between various CDC techniques and also shows that log-based is better technique.

Table 1. Comparison between various CDC techniques

CDC Technique	Supports near real-time update	Impact on source database
Trigger	Yes	Adds overhead on source database
Date & timestamp	No	Impacts performance of source database
Log- based CDC	Yes	Minimal impact on source database

5. IDENTIFYING CRITICAL DATA

For identifying critical data two aspects are considered: (A) Update significance, (B) Number of records changed. To illustrate our methodology, we will use a simple operational

database for banking application (see Figure 3). For simplicity all other tables are not shown.

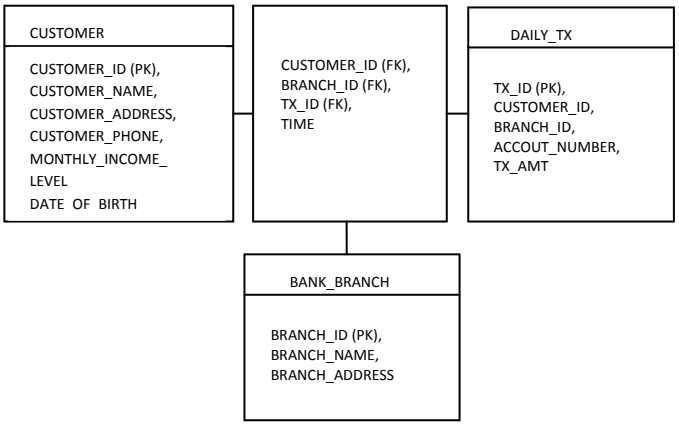

Figure 3: A sample of operational database

5.1 Update significance

Update significance measure is used to find the impact of a particular update from an operational database. Whenever a new record is inserted into database its update significance is calculated. If a record has significant impact then that is entitled as critical data and is inserted into data queue of critical data. Whereas if the record do not have significant impact then it is inserted into data queue named as non-critical queue.

Consider an e.g. if a customer makes a transaction of more than Rs.50, 000 he must be rated as high risk customer (anti money launderer). Such update has a significant impact; analysis of this information in real time is valuable in decision making. Therefore, such record would be inserted into critical data queue and will be refreshed immediately.

Defining the criteria to decide whether an update is significant or not is difficult task because each user have different requirement for the sensitivity of data If the user has a specific focus on the analysis, they can define a specific data and percentage, and if the data reaches that percentage or higher compared with the data they pre-defined, the update is significant. For example, in our example we have set the values as 50, 000 i.e. whenever TX_AMT is greater than or equal to 50, 000 it is identified as significant record.

Table 2. Daily Transaction Table

TRANSACTION_ID	CUSTOMER_ID	TX_DATE	TX_AMT
44	23	30-DEC-11	6000
45	26	30-DEC-11	1230
46	58	30-DEC-11	450
47	67	30-DEC-11	475
48	34	30-DEC-11	6578
49	57	30-DEC-11	100
50	12	30-DEC-11	100
51	1	30-DEC-11	200
52	24	30-DEC-11	54550

In table 2, we give a very simple example of the daily transactions in the bank. When there is a new update retrieved, we calculate the update impact. If it is not significant it will not be refreshed in the real time. If the update impact is significant, that record will be pushed to a temporary table, this table will be refreshed in the real time. If a transactions amount is greater than or equal to Rs.50, 000 that will be moved to a temporary table which will be updated in the real time.

5.2 Number of records changed

The number of records changed measure is used to calculate how much data have not been updated. Along with update significance measure, explained before, we will use this measure because there are situations when impact from one update is not significant but major portion of data is not updated. Although these new insertions do not have significant impact but due to major portion of old data decision making will be effected. These records whose update significance is not more is inserted into waiting queue.

For example, if there are 12 records in the DW, and there are 10 updates in the waiting queue and have negligible impact (see table 3), then there are around 10 out of 12 records that are outdated in the DW. So it is desirable to update the DW in near real time in such a situation.

Table 3. Waiting queue

TRANSACTION_ID	CUSTOMER_ID	TX_DATE	TX_AMT
54	102	31-DEC-11	4000
55	79	31-DEC-11	230
56	56	31-DEC-11	650
57	78	31-DEC-11	75
58	79	31-DEC-11	570
59	80	31-DEC-11	567
60	34	31-DEC-11	670
61	81	31-DEC-11	890
62	82	31-DEC-11	550

To calculate number of records changed, ratio of records in update queue to total number of records in DW is taken.

$$\text{Records\_changed} = \frac{Curr\_count}{Total\_count} \times 100$$

where,

Curr_count: number of records in waiting queue

Total_count: total number of records in DW

According to the business requirement different percentages can be set for this criterion. For example, a user requirement could be that if number of records changed exceeds 45% data need to be updated in real time.

5.3 Algorithm

Based on the above explanation of two measures, combination of both will be considered to determine when to refresh the DW. These two measures will be checked on each tables of the operational database to be inserted in the DW.

Algorithm for identifying critical data may be similar to:

Algorithm: Identify critical data

1. Begin
2. Calculate update significance for each record inserted

// considering example of anti money laundering

3. Update_significance = 1 (if TX_AMT>=50000)
4. else
5. Update_significane = 0
6. Calculate number of records changed for records in waiting queue

// Taking threshold value for number of records changed to be 20%. This is defined by the user according to their requirement

7. No_of_Records_changed = 1

   (if Record_changed>=20%)
8. else
9. No_of_Records_changed = 0
10. If(Update_significance=1 or No_of_Records_changed=1) then
11. Insert into temporary_daily_transaction table which is to be updated in near real time
12. End.

In the above algorithm whenever any of the variable, i.e., either Update_significance or No_of_Record_changed is set to 1 the records will be inserted into the temporary table.

6. METHODLOGY FOR LOADING DATA

ETL is very time consuming process. Our focus is to reduce the time window of ETL process. Each step in ETL, i.e., extraction, transformation, and loading are dependent on each other. In our proposed architecture some data has to be processed in real time fashion and remaining at scheduled time, i.e. , either daily at night, weekly or monthly.

The critical data which has been identified using above defined factors is retrieved in the temporary tables. These temporary tables will be refreshed in the real time and other remaining data will be refreshed at the scheduled time.

These temporary tables are to be created empty of contents, and exact structural replica of all the tables of the data warehouse that could eventually receive new data. The modified schema for supporting RTDW based on our proposed approach is shown in Figure 4.

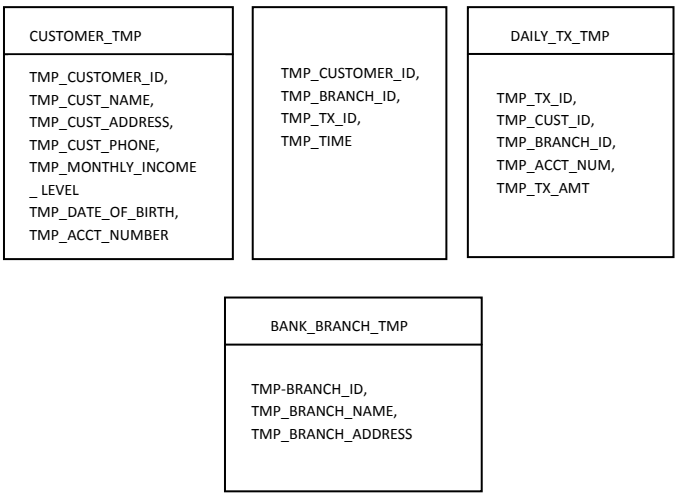

Figure 4: A sample of modified data warehouse schema

Different mappings should be designed for these temporary tables and tables with remaining data. Set of mappings for temporary tables containing critical data should be run frequently and set of mappings for tables containing non-critical data should be run at the scheduled time, i.e. once a week, or daily.

7. ADVANTAGES OF NEAR REAL-TIME DATA WAREHOUSE

Following are the advantages of implementing near real-time data warehouse:

- A data warehouse that does not appear to always be up to date tends to lose the confidence of the user community. Lost user confidence can lead to the failure of a BI effort.
- Provides real-time information to support operational business intelligence.
- Designed for minimal impact on source systems
- Lower the risk of bulk process failures
- Easy and cost-effective implementation

8. CONCLUSION

In this paper, critical data is identified which is refreshed in the real time. By this the freshness of the data warehouse is maintained. Therefore, analyst gets updated data which is critical in their decision making process and also there is minimal impact on the source database because only critical data is extracted from source database frequently.

9. ACKNOWLEDGMENTS

I would like to acknowledge my family and my friends for their support throughout the course of this project.

10. REFERENCES

[1] Youchan Zhu, Lei An, Shuangxi Liu, "Data Updating and Query in Real-time Data Warehouse System", *International Conference on Computer Science and Software Engineering*, 2008, pp. 1295-1297

[2] Li Chen and Wenny Rahayu,David Taniar, "Towards Near Real-Time Data Warehousing", 24th IEEE *International Conference on Advanced Information Networking and Applications*, 2010, pp. 1150-1157

[3] JinGang Shi, YuBin Bao, FangLing Leng, Ge Yu, "Study on Log-Based Change Data Capture and Handling Mechanism in Real-Time Data Warehouse", *International Conference on Computer Science and Software Engineering*, 2008, pp. 478-481

[4] Dr.Muhammad Younus Javed, Asim Nawaz, "Data Load Distribution by Semi Real Time Data Warehouse", *Second International Conference on Computer and Network Technology, 2010*, pp. 556-560

[5] Paul Raj Poonia, "Fundamentals of Data Warehousing", John Wiley & Sons, 2003.

[6] Lukasz Golab, Theodore Johnson, and Vladislav Shkapenyuk, "Scheduling Updates in a Real-Time Stream Warehouse", *IEEE International Conference on Data Engineering, 2009*, pp. 1207-1210

[7] Kamber and Han, "Data Mining Concepts and Techniques", Hartcourt India P. Ltd., 2001

[8] Xiaoliang Li, Fang Deng, Wensheng Li, "The Research and Application of an ETL Model Based on Task", *The 1st International Conference on Information Science and Engineering*, 2009, pp. 1006-1010

[9] Li Jian, Xu Bihua, "ETL Tool Research and Implementation Based on Drilling Data Warehouse", *Seventh International Conference on Fuzzy Systems and Knowledge Discovery*, 2010, pp. 2567-2569

[10] Michael J. Donahoo, Gregory D. Speegle,"SQL practical guide for developers", Elsevier Inc., 2005

[11] Darshan M. Tank, Amit Ganatra, Y P Kosta, C K. Bhensdadia, "Speeding ETL Processing in Data Warehouses Using High-Performance Joins For Changed Data Capture (CDC)", *International Conference on Advances in Recent Technologies in Communication and Computing*, 2010, pp. 365-368

[12] Ricardo Jorge Santos, Jorge Bernardino, "Real-Time Data Warehouse Loading Methodology", *IDEAS'08*, Coimbra, Portugal Editor: Bipin C. DESAI, September 10–12, 2008, pp. 49-58

[13]Oracle Xi Reference Manual

[14] Sam Anahony, "Data Warehousing in the real world: A practical guide for building decision support systems", John Wiley, 2004

[15] W. H. Inmon, "Building the operational data store", 2nd Ed., John Wiley, 1999.

11. AUTHORS PROFILE

Tanvi Jain, done B.Tech (Computer Science) in 2010 with 76.4% from M.D.U. (Rohtak), Currently pursuing M.Tech (Computer Science) from Centre for Development of Advanced Computing, Noida, I.P. University and doing a project on "Refreshing Data warehouse in Near Real Time".

Rajasree S. , presently serving as Associate Professor, School of IT, Center for Development of Advanced Computing Noida, a premier R&D organization under the Government of India has a total of 18 years of experience in industry and academia. Her industrial experience includes an enriching stint in IT Department of Banking Industry, Private Sector practice as IT implementer in e- Commerce and Research and Development experience in Health Informatics sectors including managing Turnkey project in Hospital Management Information Services. A qualified Project Management Professional (PMP), she has a varied experience in Project Management. Her academic experience is primarily at Post Graduate level in Technology and Computer Applications. Her primary areas of interest are Software Engineering, Software Quality and Network Security.

Shivani Saluja, done B.Tech (Computer Science) in 2010 with 78% from UPTU, Currently pursuing M.Tech (Computer Science) from Centre for Development of Advanced Computing, Noida, I.P. University and doing a project on "Refreshing Data warehouse in Near Real Time".