International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 4 Issue 5, May 2015

Near Real-Time Data Warehousing Using ETL(Extract, Transform and Load) Tools.

Mr. Narendra Gajanan Mane

ASM INSTITUTE OF MANAGEMENT & COMPUTER STUDIES (IMCOST), THANE, MUMBAI

University Of Mumbai

Abstract--

Business time is increasingly moving towards real time – in order to grow their competitive advantage, organizations should seek to capture and respond to business events faster and more rigorously than ever.
A data warehouse provides information for analytical processing, decision making and data mining tools. As the concept of real-time enterprise evolves the synchronism between transactional data and data warehouses, statically implemented, has been redefined.
Data warehouses are normally refreshed after a particular time period, most often on a daily or weekly basis. Thus, there is some delay between a business transaction and its appearance in the data warehouse. The most recent data is trapped in the operational sources where it is unavailable for analysis. For timely decision making, today's enterprise consumers require fresher data.

Near real-time data warehousing deals with this specific obstacle simply by shortening the data warehouse refreshment time periods and hence, providing resource information to the data warehouse together with data latency (The time it takes for data packets to be stored or retrieved). One particular end result is often that may data warehouse refreshment still cannot be executed in long time only. Specifically, the original data could possibly be modified as well for you to data warehouse refreshment.
Data warehousing is really a popular approach to materialized data integration. Important data scattered all over several heterogeneous alternatives will be integrated in to any main database system known as data warehouse. Data integration is done in three steps: Important data is taken out from various sources, later transformed and cleansed, and ultimately loaded into the data warehouse. Dedicated systems referred to as Extract-Transform-Load (ETL) resources are created to assist these kinds of information integration measures.
A point such as below this analysis is: The actual anomalies which may happen below most of these situations leading to an inconsistent condition about on the facts data warehouse and a few methods to avoid refreshment anomalies.

Key words-- Near real-time data warehousing, On time data capture, Extract-Transform-Load (ETL), incremental loading of data warehouses, Changed Data Capture (CDC)

I. INTRODUCTION

Data warehousing has rapidly developed into a unique and popular business application class. Beginning builders associated with data warehouses currently consider their systems to be key components of their IT strategy and architecture. Numerous examples can be cited with hugely productive data warehouses produced and deployed for businesses of most styles and different types. Hardware and software distributors get swiftly produced products and services which particularly concentrate on the data warehousing market place. This paper will certainly add crucial methods adjoining the data warehousing techniques.

Data warehouses applications are designed to answer exactly the types of questions that users would like to pose against real-time data. They are able to analyze vast quantities of data over time, to determine what the best offer is for customer or to identify potentially fraudulent, illegal, or suspicious activity.

- Today’s integration project teams face the daunting challenge that, while data volumes are exponentially growing, the need for timely and accurate business intelligence is also constantly increasing.
- Batches for data warehouse loads used to be scheduled daily to weekly but today’s businesses demand information that is as fresh as possible.
- One of the most important keys to success is the use of a data warehouse as the foundation for a Business Intelligence solution.
- The data in the warehouse is closely synchronized with data from source databases in a real-time process.
- The value of this real time business data decreases as it gets older, latency of data integration is essential for the business value of the data warehouse.
- Conventional “Extract, Transform, Load” (ETL) tools closely intermix data transformation rules with integration process procedures, requiring the development of both data transformations and data flow.
- Airlines and government agencies need to be able to analyze the most current information when trying to detect suspicious groups of passengers or potentially illegal activity. Fast-paced changes in the financial markets may make the personalized suggestions on a stockbroker's website obsolete by the time they are viewed.

The need for fresh new data with data warehouses has become a substantial desideratum. Data warehouse refreshment (integration associated with new data) is traditionally done in the off-line manner. Because of this even though processes for updating the data area are executed, OLAP users and programs are not able to access any data. This specific list of activities usually takes place in a pre-specified loading time window, to prevent overloading the operational OLTP source systems with the extra workload of this workflow. Even now, consumers are driving for greater amounts of freshness, since more and more enterprises operate in a business time schedule of 24x7. Active Data Warehousing is the term for a brand new tendency in which DWs usually are updated as frequently as possible, because of the higher demands associated with users for fresh files. This idea is also termed as Real-Time Data Warehousing.

The particular ETL market has already made efforts to interact with these fresh requirements. The major ETL suppliers formerly shipped “real time” ETL solutions using their traditional platforms. In practice, such solutions require software packages that allow the application of light-weight changes on-the-fly so as to limit the time needed for the creation of specific reports. Generally, the delay between the moment a transaction occurs at the operational site and the time the change is propagated to the target site is a few minutes, usually, five to fifteen. Such a response should be characterized more as “near real time” reaction, rather than “real time”, despite how appealing and promising can the latter be in business terms.

II. Related Work

So far, research features generally handled the situation regarding the problem of maintaining the warehouse with its classic regular revise setup. Similar books reveal methods and also algorithms for you to populate your warehouse within an off-line fashion. In a different line of research, data streams could possibly appear as a potential solution. Nonetheless, research in data streams focuses on topics concerning the front-end, including on-the-fly computation regarding inquiries without a systematic treatment of the issues raised at the back-end of a data warehouse. A lot of your recent work dedicated to RTDW can also be focused on conceptual ETL modelling, lacking the presentation of concrete specific extraction, transformation and loading algorithms with their resultant OLTP and also OLAP performance problems.

It is necessary for you to track unanswered questions sent to the sources, identify source modifications that will occurred concurrently to query evaluation, create compensating questions, or perform local compensation of previous query results. For example, the algorithms are designed for a message-oriented data exchange with the source systems. State-of-the-art ETL tools, however, allow for the implementation and execution of rather simple data flows only. The underlying model is most often a directed, acyclic chart where the edges point out the flow of data and the nodes represents a variety of transformation operators provided by the ETL tool.

III. ARCHITECTURE OF NEAR REALTIME DATA WAREHOUSE

The architecture of near real-time data warehouse contains log data files which reference is taken by change data capture (CDC) module to find new records inserted or alterations. These fresh information or alterations are placed in data queue then on this data queue treatment is applied which recognizes the important data. Based on this particular recognition 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.

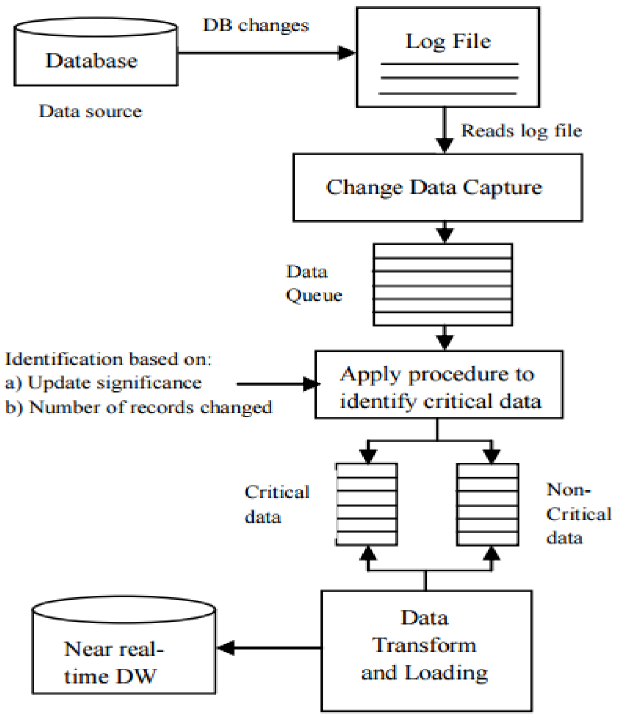

Figure 1: Near Real-Time DW Architecture

IV. Properties of Operational Data Sources

Incremental loading is the preferred approach to data warehouse refreshment because it generally reduces the amount of data that has to be extracted, transformed, and loaded by the ETL system. ETL jobs for incremental loading require access to source data that has been changed since the previous loading cycle. For this purpose, so called Change Data Capture (CDC) mechanisms at the sources can be exploited, if available. Additionally, ETL jobs for incremental loading potentially require access to the overall data content of the operational sources.
Change data capture is a procedure to acquire 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 methods:
- Synchronous: Altered data is captured instantly. This is carried out by using triggers on OLTP tables. This type of capturing mode effects OLTP as it is a part of transaction processing.
- Asynchronous: Altered data is grabbed after SQL statement performs its functioning. Then after committing the operation data is written into log files. These log files are read to capture

altered data. It has no impact on OLTP.

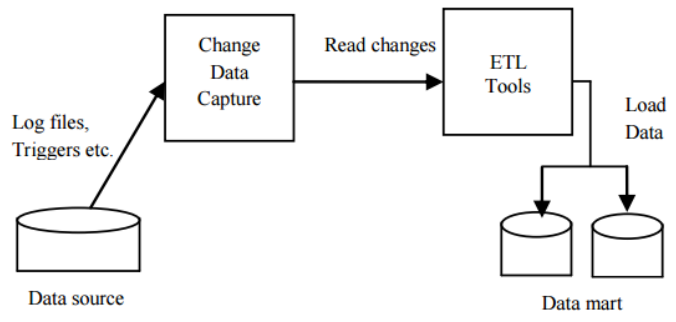

Figure 2: Implementation of CDC along with ETL tools

Some advantages of Change data Capture are:
- Completeness: CDC can capture all outcomes of insert, update and delete operations. Also it can capture change in data value, if any, after alteration.
- Performance: Asynchronous change data capture has minimum effect on the source database.
- Cost: Change data capture minimizes overhead cost because it simplifies the extraction process as it collects only change data from the database.
- Makes ETL process effective.
- Data warehouse process and functional systems are entirely decoupled from each other. This means any functional failure or recovery in either of the systems will not impact 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.

Snapshot sources: Legacy and custom applications often lack a general purpose query interface but allow for dumping data into the file system. The resulting files provide a *snapshot* of the source's state at the time of data extraction. Change data can be inferred by comparing successive snapshots. This approach is referred to as *snapshot differential.*

Logged sources: There are operational sources that maintain a *change log* that can be queried or inspected, so changes of interest can be retrieved. Several implementation approaches for log-based CDC exist: If the operational source provides active database capabilities such as triggers, change data can be written to dedicated log tables. Using triggers, change data may be logged as part of the original transaction that introduced the changes. Alternatively, triggers can be specified to be deferred causing change data to be written in a separate transaction.

Using database specific utilities, changes of interest can be extracted from the transaction log. The idea of log scraping is to parse archive log files. Log sniffing, in contrast, polls the active log file and captures changes on the fly. While these techniques have little impact on the source database, they involve some latency between the original transaction and the changes being captured. Obviously, this latency is higher for the log scraping approach.

Time-stamped sources: Operational source systems often maintain timestamp columns to indicate the time tuples have been created or updated, i.e. whenever a tuple is changed it receives a fresh timestamp. Such timestamp columns are referred to as *audit columns* [4]. Audit columns may serve as the selection criteria to extract just those tuples that have been changed since the last loading cycle. Note that deletions remain undetected though.

V. Making Change Propagation

In this section we recommend "anomaly-proof" change propagation strategies that work effectively in spite of a change data mismatch and can be carried out using ETL tools.

Convergence: For each series of source modifications and each series of incremental loads, after all modifications have been grabbed and no other modifications occurred in the meanwhile, a final incremental load leads to the same data warehouse state as a full reload would do. However, the data warehouse may go through intermediary states that would not appear, if it was completely reloaded in each loading cycle.

Weak Consistency: Convergence contains and for each data warehouse state reached after incremental loading, there are appropriate source states such that full reloading led to this state of the data warehouse.

Consistency: For each series of source alterations and each series of loading cycles, incremental loading leads to the same data warehouse state as full reloading would do.

Lockable sources: Operational sources may offer mechanism to lock their data to prevent it from being modified. For instance, database table locks or file locks may be used for this purpose.

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. This DBMS feature invokes a prewritten routine each time a specific set of conditions are met, such as the addition or updating of a record in the database. In concept, the code that the trigger launches can be written to write a record of the transaction to a database table and the ETL tool can then poll those tables on a periodic basis. 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. This approach can lose track of deletions since the entire record, including the time stamps, are gone.

- *Log based CDC:* Database maintains log files to minimize the loss of data in case of an uncontrolled shutdown. 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.

VI. IDENTIFYING CRITICAL DATA

For identifying critical data two aspects are considered which are Update importance, and Number of records changed.

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.

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.

Algorithm to identify critical data

```
1. Begin
2. Calculate update importance for each record inserted
   //consider the example of bank.
3. (if TX_AMT>=50000)
            Update_ importance = 1
4. else
5.          Update_ importance = 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. (if Record_changed>=20%)
            No_of_Records_changed = 1
8. else
9.          No_of_Records_changed = 0
10. If(Update_ importance =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.
```

VII. OPERATIONAL ISSUES

Traditionally, ETL processes deal with the following generic categories of problems

1. *Large volumes of data*: The operational data are extremely huge, as well as incur considerable data supervision troubles in all of the about three stages of development of the ETL procedure.
2. *Data quality*: The data are not always clean and have to be cleansed.

• *Evolution of data store:* The further advancement on the resources as well as the data warehouse may at in the end lead to daily maintenance operations.

• *Performance issues:* The whole process has to take place within specific time period and it is necessary to decrease its execution time. Actually, the ETL process frequently refreshes the data warehouse during idle or low-load, periods of its operation; e.g., every night. Any failures of the process must also be accomplished within the specific time period.

Additionally, in each individual phase of an ETL process multiple issues should be taken into consideration.

Extraction: The extraction conceptually is the easiest step, looking at the identification of the subset of source data that should be submitted to the ETL workflow for further processing.
In practice, this task is not easy mainly due to the following two factors:

(a) The source must suffer minimum overhead during the extraction, since other administrative activities also take place during that period.
(b) Both for technical and political reasons, administrators are quite reluctant to accept major interventions to their system's configuration.

There are four policies for the extraction of data from a data source.

1. Processing of the whole data source in each execution of the ETL process; however, this policy is usually not practically used due to the volumes of data that have to be processed.
2. Use of triggers at the source side: Typically, though, this method is not practical due to above mentioned requirement regarding the minimum overhead at the source site, the intervention to the source's configuration and possibly, the non applicability of this solution in case the source is of legacy technology.

The two realistic policies in practices are:

1. The consideration of only the newly changed - inserted, deleted or updated – operational records (e.g. by using appropriate timestamps at the source sites)
2. The parsing of the log files of the system in order to find the modified source records.

Transformation & Cleaning: After their extraction from the sources, the data are transported into an intermediate storage area, where they are transformed and cleansed. That area is frequently called Data Staging Area, DSA.

The transformation and cleaning tasks includes the core functionality of an ETL process. Depending on the application, different problems may exist and various kinds of transformations may be required.
The problems can be categorized as follows:

1. *Schema-level problems*: Naming and structural conflicts, including granularity differences.

2. *Record-level problems*: Duplicated or contradicting records, and consistency problems.

3. *Value-level problems*: Several low-level technical problems such as different value representations or different interpretation of the values.

Loading: There are two broad categories of solutions for the loading of data:
1. Bulk loading through a DBMS-specific utility.
2. Inserting data as a sequence of rows.

Below mentioned are 3 main performance issues faced during loading data back to the data warehouse.

1. The overheads of the parsing of the insert statements, the maintenance of logs and rollback-segments (or, the risks of their deactivation in the case of failures.)

2. The possibility of efficiently discriminating records that are to be inserted for the first time, from records that act as updates to previously loaded data. DBMS's typically support some declarative way to deal with this problem (e.g., the MERGE command.)

3. The existence of indexes, materialized views or both, defined over the warehouse relations. Every update to these relations automatically incurs the overhead of maintaining the indexes and the materialized views.

VIII. Problems In Real time data warehousing

Real-time data warehousing is usually answer transactional questions with analytic data structures. Following are some points which describe the problems.

- The issue is how do you properly cleanse the data, integrate it and align it across the enterprise, and keep track of changing attributes all on a real time basis. In most cases, you have to create separate, parallel structures that support real time loading and querying, along with cleansed, aligned, detailed history to provide an analytic context.

- When you deal with real-time information, you are dealing with individual transactions rather than batches of transactions. There is no time in between to run multiple aggregations, data quality, cleansing routines, or even potentially some lookups. The information movement needs shift. Traditional batch techniques often do not apply to processing real-time information feeds.

Solutions of above problems:

ETL tools indeed provide a approach of communication between databases and applications, but pose significant challenges over time. Because creating this type of connectivity requires an comprehensive knowledge of each operational database or application, inter-connectivity can get complicated as it calls for implementing very invasive custom integration's.

Minimize tight coupling, interdependencies create the potential for big data because of which it shows unpredictable impacts when even the slightest changes are made. An enterprise service bus (ESB) provides API-based connectivity with real-time integration. Unlike traditional ETL tools used for data integration, an ESB isolates applications and databases from one another by providing a middle service layer. This abstraction layer reduces dependencies by decoupling systems and provides flexibility.

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 important in their decision making process and also there is small impact on the source database because only critical data is retrieved from source database frequently.

Near time period information deposition reduces the latency between business transaction at the operational sources and their look at the data warehouse. It facilitates the analysis of more modern information and so, timelier deciding.

The advantage of near real-time data warehousing over "true" real-time solutions is that it builds on the mature and proven ETL system and does not require a reimplementation of the ETL transformation logic on another platform.

It is likely that a lot of the challenges discussed in this paper will become less challenging over time, as database, ETL, OLAP, reporting, and alerting tool vendors begin to add features to their systems to make them work better with real-time data streams. In the meantime, it is important to make sure real-time warehousing systems are well planned and designed, and thoroughly tested under realistic data and user load conditions before they are deployed.

The benefits of data warehousing in real-time are becoming clearer every day. With the right tools, designs, advice, approaches, and in some cases tricks, real-time data warehousing is possible using today's technologies, and will only become easier in the future. In any case, the time to begin planning and prototyping is now.

References

[1] https://www.mulesoft.com/resources/esb/etl-tools-vs-esb
[2] http://en.wikipedia.org/wiki/Extract,_transform,_load
[3] http://dssresources.com/papers/features/langseth/langseth02082004.html
[4] http://gerardnico.com/wiki/dit/data_latency
[5]http://wwwlgis.informatik.uni-kl.de/cms/fileadmin/users/joerg/documents/joergBIRTE09.pdf
[6] Gupta, A., Jagadish, H. V., Mumick, I. S.: Data Integration using Self-Maintainable
Views. *EDBT, 1996, 140-144*
[7] JÄorg, T., Dessloch, S.: Towards generating ETL processes for incremental loading.*IDEAS, 2008, 101-110*
[8] JÄorg, T., Dessloch, S.: Formalizing ETL Jobs for Incremental Loading of DataWare-houses. *BTW, 2009, 327-346*
[9] Kimball, R., Caserta, J.: The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, *Conforming, and Delivering Data. John Wiley & Sons,2004*
[10] Labio, W., Garcia-Molina, H.: E±cient Snapshot Di®erential Algorithms for DataWarehousing. *VLDB, 1996, 63-74*
[11] Widom, J.: Research Problems in Data Warehousing. *CIKM, 1995, 25-30*
[12] Zhuge, Y., Garcia-Molina, H., Hammer, J., Widom, J.: View Maintenance in a Warehousing Environment. *SIGMOD Conference, 1995, 316-327*
[13] Zhuge, Y., Garcia-Molina, H., Wiener, J. L.: Consistency Algorithms for Multi-Source Warehouse View Maintenance. *Distributed and Parallel Databases, 1998, 6,7-40*
[14] 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
[15] 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
[16] Paul Raj Poonia, "Fundamentals of Data Warehousing", John Wiley & Sons, 2003.

[17] 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

[18]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
[19] 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
[20] 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
[21] 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
[22] Paul Raj Poonia, "Fundamentals of Data Warehousing", John Wiley & Sons, 2003.
[23] 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
[24] 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

BIOGRAPHY

Narendra Gajanan Mane
born in Sangli, Maharashtra India.
He is pursuing
MCA final year in ASM's Institute
of Management & Computer
Studies ,IMCOST, Thane.