Novateur Publication's
International Journal of Innovation in Engineering, Research and Technology [IJIERT]
ICITDCEME'15 Conference Proceedings
ISSN No - 2394-3696

REAL-TIME CHANGE DATA CAPTURE USING STAGING TABLES AND DELTA VIEW GENERATION FOR INCREMENTAL LOADING OF LARGE DIMENSIONS IN A DATA WAREHOUSE

Miss. Paurnima Ghugarkar
Department of Computer Engineering, VACOE, Ahemadnagar, India

Mr. Yogesh Borude
M.E.(Computer Science & IT), Pune, India Working in MNC, Charlotte, USA

Prof. Prabhudev Irabashetti
Assistant Professor Department of Computer Engineering, VACOE, Ahemadnagar, India

ABSTRACT
In the big data era, data become more important for Business Intelligence and Enterprise Data Analytics system operation. The load cycle of traditional data warehouse is fixed and longer, which can't timely response the rapid and real time data change. Real-time data warehouse technology as an extension of traditional data warehouse can capture the rapid data change and process the real-time data analysis to meet the requirement of Business Intelligence and Enterprise Data Analytics system. The real-time data access without processing delay is a challenging task to the real-time data warehouse. In this paper we discusses current CDC technologies and presents the theory about why they are unable to deliver changes in real-time. This paper also explain the approaches of dimension delta view generation of incremental loading of real-time data and staging table ETL framework to process the historical data and real-time data separately. Incremental load is the preferred approach in efficient ETL processes. Delta view and stage table framework for a dimension encompasses all its source table and produces a set of keys that should be incrementally processed. We have employed this approach in real world project and have noticed an effectiveness of real-time data ETL and reduction in the loading time of big dimension.

INTRODUCTION
Data transformations are often the most complex and, in terms of processing time, the most costly part of the extraction, transformation, and loading (ETL) process. Extraction is the operation of extracting data from a source system for further use in a data warehouse environment. This is the first step of the ETL process. After the extraction, this data can be transformed and loaded into the data warehouse. They can range from simple data conversions to extremely complex data scrubbing techniques.
The data acquisition process represents one of the most technically demanding and one of the most expensive parts of building a data warehouse. The data needed for the data warehouse are being extracted from their sources during an extract, transfer and load (ETL) process. Typical data warehouse architecture includes a data staging area where the extracted data are stored temporarily, and subsequently, upon cleaning and transformation phase, transferred into the data warehouse. Usually, the traditional data warehouse only supports the historical information query and analysis, which cannot
obtain the upto-date real-time data. After the first or initial loading, data warehouse must be periodically refreshed to keep up with source data updates.
Data warehouse loading is usually done periodically, by a background process. The update patterns (daily, weekly, etc.) for traditional data warehouses and data integration process result in outdated data to a greater or lesser extent. The naive approach to refreshing a data warehouse is referred to as full reloading. Possible refreshment scenario is to repeat the initial loading process using modified source data, compare the results with the current DW in order to determine changes that need to be done and finally perform the changes. This strategy is known as full DW reload. With the increasing size and complexity of DW, full reloading becomes inadequate, and in some cases inapplicable. More appropriate approach is a gradual change of DW in accordance with the changes that have occurred in the data sources since the last synchronization. Only the data that has changed since the previous reload needs to be transformed and processed. This approach is known as incremental reloading. Incremental reload work quicker and efficiently than full reload. The basic of incremental and real-time loading is that the changes of source data can be captured and later on propagated to the data warehouse. There is a number of known techniques for changed data capture.

Real time data warehouse will able to show the ETL working result in an exact time according to the transactional time on a number system. But ETL as the core of data warehouse cannot really work on real time. This happens because of the ETL need some time to process the data from various sources in a large amount, and has to go through some communication component. The delay time is needed by ETL to process this summary, which trigger the term Nearly Real Time Data Warehouse (NRTDWH).

To produce NRTDWH, ETL therefore can be implemented by applying Change Data Capture. CDC is used to know the changing on the data sources and then capture it to be given to the database destinations which need it. This ability made CDC able to capture data changing efficiently therefore NRTDWH will be easier to be implemented. Based on the above explanation, therefore the effort to create NRTDWH by CDC modeling becomes really important to be implemented.

PURPOSE

The purpose of the study is implementing the real time data warehousing considering current size of the data in all sectors is increased and will increase drastically in future. Increase in the size of the data and nature of rapid change in data increases problems of maintaining and loading real-time data and historical data. Loading and maintaining such a huge amount of data in data warehouse is challenging task. For doing this there are many ways one of them is delta view generation. This delta view generation does incremental loading of large dimension real-time and historical data by using ETL. So the purpose of this study is to analyze how it will work and give improved performance. Other way is to design a ETL framework on trigger based mechanism using stage area concept.

LITERATURE SURVEY

A lot of work has been done on the data warehouse architecture and gives different techniques for improving and better performance of architecture. Traditional data warehouse has static structure in schema and are not able to have dynamic structure. Due to periodic up gradation nature it can't allow for integration of data. To resolve this problem of integration solution is provided in integration of data warehouse during execution has been enabled with no effect on performance and minimize the delay on decision support database between the transaction of information and data updating. The researcher focuses on the loading processes and usage of data area for efficient integration. Data warehouse implements views that are maintained and updated are the materialized views.

In data warehouse incremental reloading is evidently related to incremental maintenance of materialized views as both areas cope with same issue – how to update physically integrated data under a given time constraint. The basic core idea of materialized view in incremental update assumes computing new updated content using materialized parent view (the outdated view needed to be refreshed) and its incremental update, often referred to as delta view. Change data capture (CDC) is a set of software design pattern used to determine (and track) the data that has changed so that action can be taken using the changed data. Also, Change data capture (CDC) is an approach to data integration that is based on the identification, capture and delivery of the changes made to enterprise data sources.

This concept of capturing information about changed tuples is valuable later in the extract phase of incremental ETL process. There are some tool for automatic creation of all necessary structures for altered data capture and subsequent extraction. They track changes by storing changed tuples identifiers (primary keys) in a special CDC tables and later on use stored identifiers for fast retrieval of tuples relevant for the next ETL cycle. Complementary to their work we addressed the issue, and proposed the solution, for finding identifiers of tuples responsible for refreshment of a particular dimension table. Since the contents of denormalized dimensions are refreshed based on the contents of numerous source tables, finding such identifiers can be a challenging task.

EXISTING SYSTEM

CURRENT CHANGE DATA CAPTURE APPROACHES

There are three widely accepted approaches to accomplish CDC. This section discusses the strengths and weaknesses of each approach. Following is a discussion about how each approach is incapable of delivering changes in real-time.

A. LOG SCANNING

Database management systems often keep logs for disaster recovery and roll-back purposes. Log-based CDC involves scanning database logs, deducing from them any changes made to the data. Log-based CDC is the least invasive approach to CDC as it does not require any modification to applications using the database, nor does it require any additional processing by the DBMS. Figure 1 show how log-based CDC works. As new data arrives in the database, the log is updated and the CDC tool will poll this log to work out the changes.

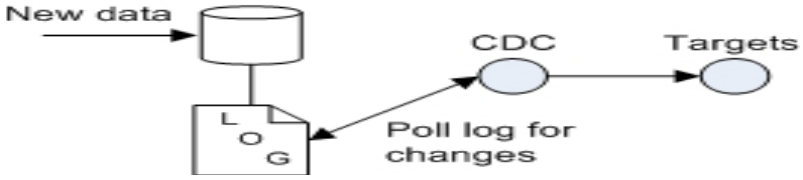

Figure 1. Log-based CDC: As new data arrives in the database, the log is updated and the CDC tool will poll the log to work out the changes.

The major pitfall of this approach is not every DBMS or data store is capable of producing a log. Additionally, the DBMSs that create logs have their own log format, meaning a different algorithm is needed to parse logs produced by different DBMSs. Although, this issue can be circumvented by purchasing an off the shelf product capable of parsing different logs. A potential cause for error in log-based CDC arises when logs are either purged or archived off-line. The logs can become large files in a short space of time, so they are either moved to off-line storage, or are purged to keep the file size to a minimum. If purging or archiving takes place before the CDC scan, then inevitably changes will be missed.

B. DATABASE TRIGGERS

A database trigger is a piece of user written code that will automatically execute when a given set of events have proceeded. Such events include inserting, updating and deleting records in a table. CDC triggers are set up to fire on such events. Figure 2 shows a trigger based CDC approach.

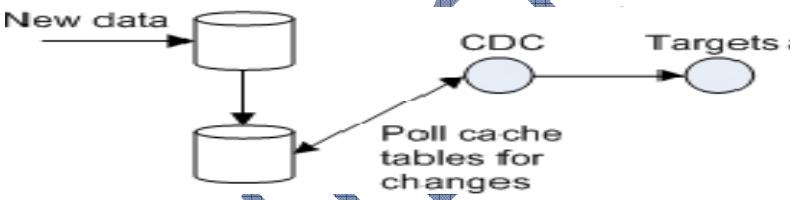

Figure 2. Trigger based CDC: when new data arrives, a trigger will fire, Sending the changed data to a cache area

When change data arrives, a trigger will fire, sending the changed data to a cache area. The CDC tool will then poll this cache to retrieve the changes. Using triggers means database application code does not need to be modified and all of the work for CDC is done by the DBMS. A DBMS dealing with a large volume of transactions will be burdened by also having to process the triggers, and this ultimately impacts the system's performance.

C. TIMESTAMPS

Timestamps can be applied to records in order to record the time when a record has been inserted or updated. It is not possible to capture record deletions with timestamps, because the timestamps get deleted with the record. Timestamps take relatively little time to set up, and they are maintained by the DBMS. Figure 3 shows the timestamps approach to CDC.

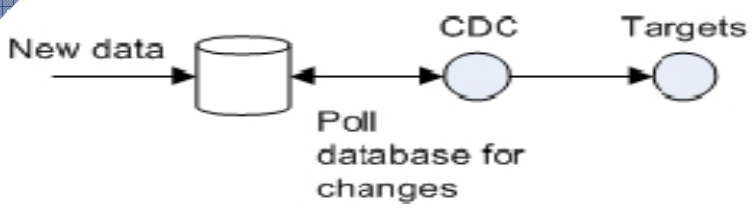

Figure 3. Timestamps CDC: new data is time stamped. The CDC tool polls the database, and extracts records that have been added or modified sincethe last scan.

Change data will be time stamped when the DBMS records it. The CDC tool will poll the database, and look for records that have been added or modified since the last scan. The timestamps should be indexed in order to improve

the performance of the CDC tool. However, indexing the timestamps increases the overhead of inserting and updating records, because the indexes will need to be regenerated by the DBMS.

D. THE COST OF POLLING

The three CDC technologies described above all require some resource to be polled. When setting up a polling mechanism a time interval is specified to govern how often the resource is polled. For CDC this means the poll interval is the determining factor on the timeliness of changes being delivered to targets. Moreover, polling increases the latency between detecting and delivering changes. This affect prohibits a true real-time CDC solution. Additionally, the poll interval will be set to a value that is respectful to the resource being polled, in order to avoid a denial of service. In log-based CDC, if the log is polled too frequently, there is risk of locking the log file so the DBMS can't write to it, meaning the DBMSs resources are consumed to temporarily cache log data. In the timestamps CDC, the operational database is directly polled, meaning the DBMS will have to deal with the CDC workload as well as the normal transactional workload. High frequent polling will inevitably affect the transactional performance.

METHOD/PROPOSED SYSTEM

DELTA VIEW

When we know what records have changed in the source system, we can use that information to incrementally update fact and dimension tables in the data warehouse. Fact tables are also the ones that benefit most from the incremental loading style since they typically contain a huge number of records. On the other hand, dimension tables are usually relatively small and can be updated in a full reload manner. However, there are cases when dimension tables are also fairly large, both in terms of number of records and number of attributes. Such dimensions are sometimes referred to as "monster dimensions", "whose size exceeds 100 million rows".

Typically, those are customer or client dimensions in large enterprises. Such dimensions may take a long time to load, since they are both large in record count and have a considerable number of attributes that need to be calculated along the way (e.g. student's Grade Point Average, rank, etc.). We believe that, for such large dimensions, it is beneficial to construct the "delta view" – a view that will contain keys of records that need to be addressed in the ETL procedure. Unlike fact tables, delta view records are deduced not only from the base source table (for the dimensions), but also from many other tables that are being used in the dimension's ETL process. In our, real-world case, our dimension table is based on 23 relational tables. A change in any of those 23 tables can impact the content of the dimension.

DELTA VIEW CONSTRUCTION

The delta view construction idea is to, starting from the base table (e.g. Student), recursively traverse the graph and gather primary keys of the related tables. The related tables are divided in two sets:

• REFD – a set of tables referenced by the current table
(e.g. REFD (Student) = {City},
REFD (City) = {State})
• REFING – a set of tables referencing the current table
(e.g. REFING (Student) = {YearEnroll, StudAcdmYear},
REFING (City) = {CityName, StudAcdmYear,Student})

Finally, the delta view delta(t) for the dimension table t is given with:

$$delta(t) = cdc(t) \cup \bigcup_{r \in REFD(t)} refdcdc(t,r) \cup \bigcup_{r \in REFING(t)} refingcdc(t,r) \quad (1)$$

where:

$$cdc(t) = \pi_{PK(t)}\, t' \quad (2)$$

$$refdcdc(t,r) = \pi_{PK(t)}\, t \triangleright\!\triangleleft delta(r) \quad (3)$$

$$refingcdc(t,r) = \begin{cases} \pi_{PK(t)}\, delta(r), & if\ PK(t) \subseteq PK(r) \\ \pi_{PK(t)}\, delta(r) \triangleright\!\triangleleft (r \cup \sigma_{op='del'} r'), & \text{otherwise} \end{cases} \quad (4)$$

CHANGE DATA CAPTURE FRAMEWORK

One of the goals of real-time CDC is to propagate the changes through the system without interruption. Moreover, an ideal CDC system will automatically allow changes to flow from the source system to interested target systems. To attain this flow, it is imperative that the CDC tool knows where to deliver the changed data to. One way of doing

this is to hard code the target destinations into the CDC tool. A more flexible system will freely allow targets to be added and removed when required. Obviously there can be multiple target destinations for the changed data, and it is likely that each destination will require different changed data. An intelligent CDC system will know what data to deliver to each of the interested target systems. This feature is seldom implemented. Therefore, we propose a framework for a CDC system based on web service technology. Our framework will provide the foundation for a CDC architecture that can automatically allow targets to subscribe to changes and also allow the changes to be filtered to tailor each targets data requirements. The framework shown in Figure 4 depicts how publishers and subscribers interact within our framework.

Figure 4. The interaction between publishers and subscribers

The publisher is a web service with two operations, Subscribe() and Notify(). Interested targets will consume the Publisher web service and register by calling Subscribe(), passing in an XQuery as the parameter. When changes arrive at the publisher, the Notify() operation will automatically send the data to the targets, applying the XQuery to the changed data, therefore filtering the data, so that the target subscriber gets only the changed data it is subscribed to.

CONCLUSION

In this paper we have proposed an approach for delta view construction that enables incremental loading of dimension tables. This paper also explained the ETL stage framework for real time data. These both methods are suitable for large, complicated (both in terms of attribute and row count) and real time dimensions, because they can have significant impact on the ETL process. By applying algorithmic approach of delta view generation and staging mechanism to the real-time and historical data we can achieve reduction in incremental loading times. Hence the data warehousing can be called as nearly real time data warehousing (NRTDWH).

REFERENCES

[1] W. Eckerson and C. White, "Evaluating ETL and Data Integration Platforms," 2003.
[2] T. Palpanas, R. Sidle, R. Cochrane and H. Pirahesh, "Incremental Maintenance for Non-Distributive Aggregate Functions," in Proceedings of the 28th VLDB Conference, Hong Kong, 2002.
[3] H. Gupta and I. S. Mumick, "Incremental Maintenance of Aggregate and Outer join Expressions," Information Systems, pp. 435-464, 2006.
[4] W. Liang, H. Wang and M. Orlowska, "Materialized view selection under the maintenance time constraint," Data & Knowledge Engineering, vol. 37, pp. 203-216, 2001.
[5] R. Prabhu and D. Lyman, "Extracting delta for incremental data warehouse maintenance," in IEEE 16th International Conference on Data Engineering (ICDE'00), San Diego, CA, USA, 2000
[6] C. R. Valencio, M. H. Marioto, G. F. Donega Zafalon, J. M. Machado and J. C. Momente, "Real Time Delta Extraction Based on Triggers to Support Data Warehousing," in International Conference on Parallel and Distributed Computing, Applications and Technologies, 2013.
[7] X. Zhang, W. Sun, W. Wang, Y. Feng and B. Shi , "Generating Incremental ETL Processes Automatically," in Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06), 2006.
[8] T. Jorg and S. Desloch, "Towards Generating ETL Processes for Incremental Loading," in IDEAS08, Coimbra, Portugal, 2008.
[9] R. Rocha, F. Cardoso and M. Souza, "Performance tests in data warehousing ETLM process for detection of changes in data origin," in Proceedings of Data Warehousing and Knowledge Discovery, DaWaK 2003, LNCS 2737, 2003..