IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⋇ Impact Factor 7.39 ⋇ Vol. 11, Issue 7, July 2022

DOI: 10.17148/IJARCCE.2022.11748

Study on Change Data Capture Techniques for Incremental Loading in Data Warehouse

Ramadas K Kamat[1], Dr. G S Mamatha[2]

Undergraduate Student, Department of Information Science and Engineering,

RV College of Engineering, Bengaluru, India1[1]

Professor and Associate Dean (PG Studies), Department of Information Science and Engineering,

RV College of Engineering, Bengaluru, India[2]

Abstract: Incremental loading is a preferred data warehouse (DW) refresh technique in the modern world for being more efficient than Full loading. A special case in full loading where the entire content at the target is dropped and reloaded for the purpose of refresh is known as Truncate and Load. Incremental loading involves refreshing only the changed contents from the data source into the data warehouse. The gathering of these changed contents is known as Change Data Capture (CDC). This paper presents a study on different CDC techniques used in practice to aid the incremental loading. An example scenario which uses the truncate and load approach is presented and the suitability of the CDC techniques to implement incremental loading is discussed. The implementation for conducting a CDC based on the delta rules for migrating from a naive truncate and load to an effective incremental load in the example scenario is presented in this paper.

Keywords: Truncate and Load, Incremental Load, Data Warehouse, Change Data Capture, Delta, Log, Snapshot, Timestamp, Metadata.

I. INTRODUCTION

Data Warehouse (DW) is a large collection of data gathered from multiple heterogeneous sources. The DW is used by analysts, executives, and other key decision makers to make an informed decision using the data [1]. The data loading is performed by Extract-Transform-Load (ETL) systems, which take the data from the source, transform it in the desired format and load into the warehouse. The ETL process responsible for populating the data into the DW for the first time is called initial loading. The changes at the source are reflected in the DW upon performing a refresh. There are two methods of performing the refresh namely Truncate and Load and Incremental load. The Truncate and Load or the Full Load, involves refreshing entire contents of Data Warehouse, it is more like running the initial load process again, so in order to reflect the changes happening at the source, the entirety of the existing warehouse content is modified. The older content may or may not be dropped based on the requirement of maintaining a history [2]. The Incremental Loading approach, however, proposes to refresh only the contents that have undergone change of any sort. The change can involve addition of new data, updating or even deletion of the existing data.

The Incremental strategy makes it possible to update the data warehouse more efficiently without reloading the entire content. It is clear that the incremental load can be more efficient than its counterpart truncate and load, but the latter does offer some advantages over the former. The full loading approach is simpler and easier to implement as compared to the incremental approach and that is what makes it preferable in normal and not so complex scenarios. However, when the size of both source and the target tables are large, the loading process becomes more and more time consuming and impractical [3]. Also, when the number of changes happening is relatively smaller than the size of the source, truncate and load start becoming inefficient. It can be inferred that the incremental loading is a better and more efficient choice in such scenarios. In real world applications, the size of the data is constantly increasing and so is the amount of data to be refreshed into the DW [4]. The changes from the source have to be periodically updated in the target and with increase in the size of the source data, performing full load will become time consuming with each passing day. Hence, to efficiently achieve the loading into target, incremental approach has to be undertaken.

The Incremental Loading involves implementation of Change Data Capture (CDC), which will bring the changed tuples that will be updated in the target tables [2]. The gathering of changed data is usually done with techniques which examine the state of the data before and after the source changes its state. The commonly used techniques include using a timestamp to track the last updated time, maintaining and comparing snapshots to identify the difference etc. The survey

of all such techniques have been carried out and has been presented in this paper. Additionally, a near real-time example scenario is discussed and the suitability of each of the CDC techniques is analyzed.
The following lines briefly describe the contents of this paper. In section 2, a literature survey on the work related to incremental loading has been carried out. The section 3 describes different ways to carry out change data capture. The section 4 describes the work carried out for implementing change detection. The paper ends with a conclusion section.

II. Related Work

A literature survey was carried out to understand the state-of-the art developments in the field of Data Warehousing and to understand the CDC techniques. Surajit and Umeshwar [1] in their description of Data Warehouse have provided an overview of architecture to be used in Online Transaction Analysis and Processing (OLAP). The paper also discusses the backend tools, database schema, front-end technologies among other things to be considered while effectively using the Data Warehouse model for OLAP applications. Janet and Jeffrey [5] in their paper on Incremental Loading of Object Databases discuss the usage of queries within the load data file to identify existing objects in Object-oriented and object-relational databases (OODB). Thomas Jörg and Stefan Dessloch [2] have presented an approach for automated derivation of incremental load jobs based on equational reasoning. They have also reviewed different change data capture techniques to identify the changes taking place in the source tables and presented the idea of Incremental loading [6]. Andreas Behrenda and Thomas Jörg [7] have identified limitations in the already existing incremental approaches and presented ideas on automated maintenance of Data warehouses.

The Change Data Capture (CDC) must be carried out to identify data for incremental loading [8,9]. Some of the well-known techniques to accomplish change detection have been to maintain database snapshots [10,11], or using the transaction logs of the underlying Database Management System (DBMS) [12], or making use of timestamps to identify the rows that have changed from time to time [13]. There has also been works in the areas of capturing real time changes which is a key for OLTP applications. In [14,15], authors propose the idea of capturing real time change in source using a web service approach. The idea of capturing changes in the source using database triggers in real time has been proposed by the authors in their work in Data Warehousing [16,17]. Denny et al. [18] has implemented CDC in the source tables using Apache Spark and Hadoop Distributed File System (HDFS) aiming to increase the efficiency and reduce the computation time of ETL processes.

Biswas et al. [19] have carried out a survey of various ETL tools written in Python, Java and R. They have presented results based on throughput, execution time, transformation support and number of lines of code. They have gone further and proposed algorithms to identify rows inserted, updated and deleted at the source and also provided ways to process large dimension tables. Igor Meketerovic and Ljiljana Brkić have proposed an algorithm to generate a delta view of the underlying sources based on the foreign key relationships in the fact table [3]. Darshan et al. have discussed the ways in which ETL processes can be speeded up using high performance joins for Change Data Capture [4].

Vijayalakshmi and Minu [20] have undertaken a cloud-based approach to develop a system to load data into target tables. The recommended system is designed to provide more efficient data retrieval following a cloud based incremental approach. In [21], the authors have presented a comprehensive review on using parallel processing to reduce computation time in ETL loads. The approach taken here can be clubbed with existing incremental loading strategies to reduce computation time in generating the final result table. Also, with the development of ideas under the names of data lakes and data marts [22], storage of raw and unstructured data for various purposes like business intelligence and business analytics has been explored.

III. Change Detection Techniques

There are several definitions for Change Data Capture (CDC). It can be defined as a method to identify and deliver changes occurring at the source in the operational system [4,18,23]. The CDC phase will provide the necessary deltas which can be processed to develop an incremental loading strategy. In this section, an example scenario is presented, and different CDC techniques are analysed to best suit the problem.

Consider an application that uses data related to domains, subdomains mapped with their Company and presents relevant information on which provider were these subdomains last seen. The provider in this scenario may be the Content Delivery Network (CDN), Load Balancers or Independent servers. The data related to these subdomains is periodically obtained from various sources. These subdomains often change their provider locations or are sometimes found on multiple providers. The new data coming into the system gives a clear picture of the subdomain's location and relevant computation has to be made in order to present a final view to the users. The web related data is ever increasing, and the subdomains are transient in nature based on their provider location. An incremental loading strategy would be suitable in this scenario. In order to implement an incremental loading strategy, CDC must be achieved. The commonly used CDC techniques are discussed in terms of their suitability with the example scenario.

A. Database Log

The DBMSs maintain logs of transactions of most DML operations. These logs are often used for backups and recovery. The log files are tracked by the systems employing these techniques to identify changes taken place at the data source since the last refresh time. The CDC process can scrape the file to find the operations that classify as a change and obtain the deltas. Figure 1 shows a general view of log-based CDC.

This method is quite efficient as most DBMSs have an efficient way of writing and maintaining these logs [3,24]. However, different DBMSs may have different ways of maintaining the log and the entire codebase has to be modified in case of migrating to a different DBMS [15].

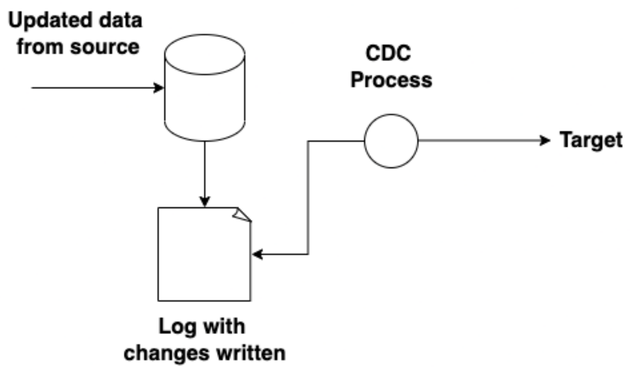

Fig. 1 Log based CDC technique

B. Transaction Log

A solution to the Database log's limitation can be a native transaction log. It can be in the form of a table, which stores the key of the rows being updated, deleted or inserted. The log can then be referred to keep track of the changes at the data source. This approach is DBMS agnostic and hence can be implemented with any of the providers. The log-based approaches are the only methods which are able to completely identify inserts, updates and deletes [2]. The metadata information such as number of rows added during each inserts, time of update can be stored to aid CDC. In real time OLAP applications however, this may account for a large number of write operations and may not be as efficient as the DBMS logs.

C. Annotations

This method involves having additional attributes in the source table to identify that particular entry when it undergoes a change. It can be seen as having a field say, "isModified" along with other necessary fields to understand if the row has been updated or not. A similar identifier can be used in cases of inserts and deletes. This method provides a way to query tables directly to capture the changes.

In the example scenario however, the migration of a subdomain can only be identified, when a new entry at the source table is made (the table is loaded in append only fashion). So to identify a change in such a scenario cannot be directly accomplished by annotating the rows.

D. Timestamps

This method requires having a timestamp field to identify the last updated time of the source data. This column is often referred to as the audit column [2]. Upon update, the timestamp can be modified to the current timestamp. The changed rows can be identified on the basis of their timestamp by comparing them with the time of last update.

This method requires modification of the schema to accommodate a timestamp field is one of its disadvantages [3]. The rows that have been deleted also won't be captured as the entire entry would have been removed [24]. Something similar to the annotation method has to be done to handle deletes using timestamps, by only marking the rows as inactive to prevent interference. In the example scenario, the timestamps of subdomains are obtained from multiple sources, which

DOI: 10.17148/IJARCCE.2022.11748

leaves direct change detection inaccurate. Additional timestamps with write time can however be used to identify the subdomain migration from their previously known location.

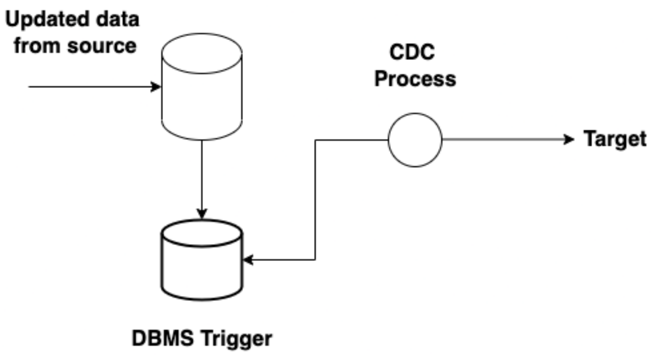

Fig. 2 Trigger based CDC technique

E. Database Triggers

Using database triggers to capture changes at the source is one of the popular CDC techniques. This technique involves setting up of procedures, which are triggered when a create, update or delete (CRUD) operation takes place [18,19,21]. Figure 2 shows a general view of database trigger-based CDC. The database trigger can write the data into a separate file which can be processed to identify the rows changed. The triggers are usually implemented by the DBMS. The trigger-based approach can come as an additional overhead when there are a large number of incoming rows at regular intervals. In the example scenario, when the amount of changes is large the time taken to identify them using triggers significantly increases and also burdens the underlying DBMS [4].

F. Snapshot Differential

A snapshot of a data source is taken and is compared with the next state or the changed state of the source. The difference between these is essentially the delta and can be used for further analysis. The snapshots have to be refreshed regularly to keep them up to date. There are chances of high latency when the source or snapshot is remote [10]. The snapshot can be refreshed by transferring the source table to snapshot or can be refreshed immediately when the changes are taking place, known as ASAP refresh. There are more efficient refresh algorithms in window refresh [11]. Direct implementations of snapshot differential pose problems of differentiating between inserted rows and updated rows and the need to refresh the snapshot every time requiring large volumes of data transfer are some of the drawbacks [2]. In the example scenario, a snapshot differential approach can be used to capture the subdomains changing on a daily basis by storing the snapshot of the source and then comparing with the changed version.

IV. Implementation Of Change Detection

In the example scenario described in section 3, let us consider the existing approach to handle the data coming into the system to update the user centric view is based on the idea of Truncate and Load i.e. the table, will be referred to as result table in the paper going forward, is computed from the scratch every single time, new data enters the system. This approach is simple but clearly inefficient because as the size of the source tables containing the data increases continuously every passing day, the time required to compute the result table also increases significantly. It is observed that not all the subdomain data is refreshed every time new data enters, and the updated information is also such that most subdomains are found on the same provider as they were seen during the previous iteration. Hence, it is more efficient to only identify the subdomains that have changed their locations and only involve them for the backend computation.

The implementation of CDC in the source data makes way for processing only the time deltas that will be obtained on a daily basis and will make way for the process moving away from the existing Truncate and Load approach to a more efficient Incremental approach. In order to implement Change Detection a combination of timestamp, metadata and Snapshot approach was applied. The metadata table is used to keep track of the count of rows being written into the source tables, assign a unique identifier to each iteration and keep the last updated timestamp. The Metadata table provides the necessary information from the previous iteration to use in the incremental process. The time attribute associated with the source data cannot be directly used to obtain time deltas due to the following reasons:

- The timestamp associated with each entry in the source tables originates from different sources and hence are inconsistent.
- The timestamp cannot be used to calculate the time deltas as the backend process could not have run on a particular day due to unforeseen circumstances or might have run multiple times a day leaving time attribute less useful to identify new sets of subdomains entering the system.

The identifier from the Metadata table provides the subdomain data specific to a particular iteration helping to overcome the second problem. The time deltas are identified for every iteration using the metadata identifier to distinguish between the subdomains of different iterations. The criteria for what constitute the deltas a.k.a. the delta rules in this use case are as follows:

- The subdomain must not be present in the source table during the previous iteration.
- The subdomain, if present in the source table during the previous iteration, then it must have undergone a change in the provider.

The batch of data coming from the data source is attached with a Universally Unique Identifier (UUID). The UUID will be stored in the metadata table, along with the count of the rows in the batch and the time of last update. The changes in the source tables are tracked on a daily basis by maintaining a snapshot of its previous increment. The snapshot is then compared with the updated version of the source to obtain the set of rows that have undergone a change. These rows are validated against the delta criteria using a series of join operations to obtain the final set of deltas. The deltas obtained can be used for further backend processing as needed or can be loaded to the DW using the ETL processes.

V. Conclusion

The paper presents a survey on different Change Data Capture (CDC) techniques used in Data Warehousing for Incremental loading. The different CDC techniques, their advantages and limitations are discussed with the perspective of handling a real time scenario of a system handling subdomain data. The implementation of a combination of snapshot differential technique with metadata and timestamp is presented in the perspective of migrating the loading process from Truncate and load to Incremental load has been discussed in this paper. The idea of incremental loading replacing full loading strategies is gaining more traction as the amount of data being collected and processed is increasing each passing day. The optimization of the CDC techniques to suit real life use cases is needed to improve incremental loading.

ACKNOWLEDGMENT

The authors of this paper express their gratitude to Mr. Nikhil Parakh and his colleagues of Akamai Technologies for their time and guidance.

References

[1]. Chaudhuri, Surajit and Umeshwar Dayal, "An overview of data warehousing and OLAP technology." ACM Sigmod Record, Vol. 26, Issue 1, 1997, pp: 65-74.
[2]. Jörg Thomas, and Stefan Dessloch. "Formalizing ETL jobs for incremental loading of data warehouses." Datenbanksysteme in Business, Technologie und Web (BTW)–13. Fachtagung des GI-Fachbereichs" Datenbanken und Informationssysteme"(DBIS), 2009.
[3]. Mekterović, Igor, and Ljiljana Brkić, "Delta view generation for incremental loading of large dimensions in a data warehouse." In Proceedings of the 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE, 2015, pp: 1417-1422.
[4]. Tank, Darshan M., Amit Ganatra, Y. P. Kosta, and C. K. Bhensdadia, "Speeding ETL processing in data warehouses using high-performance joins for Changed Data Capture (CDC)." In Proceedings of the International Conference on Advances in Recent Technologies in Communication and Computing, IEEE, 2010, pp: 365-368.
[5]. Wiener, J., and J. Naughton, "Incremental loading of object databases", Stanford InfoLab, 1996.
[6]. Jörg T, Dessloch S, "Towards generating ETL processes for incremental loading", In Proceedings of the 2008 international symposium on database engineering applications (IDEAS'08), ACM, 2008, pp: 101–110.
[7]. Behrend, Andreas, and Thomas Jörg, "Optimized incremental ETL jobs for maintaining data warehouses.", In Proceedings of the Fourteenth International Database Engineering & Applications Symposium, 2010, pp: 216-224.
[8]. Bokade MB, Dhande SS, Vyavahare HR, "Framework of change data capture and real time data warehouse", International journal of engineering research and technology, vol 2. ESRSA Publications, 2013.
[9]. Ankorion, I, "Change data capture efficient ETL for real-time bi", Information Management, Vol. 15, Issue 1, 2005, p.36.
[10]. Lindsay B, Haas L, Mohan C, Pirahesh H, Wilms P, "A snapshot differential refresh algorithm", In Proceedings of the ACM-SIGMOD conference, Vol. 15, Issue 2, 1986, pp: 53-60.
[11]. Labio W, Garcia-Molina H, "Efficient snapshot differential algorithms for data warehousing", In Proceedings of the 22nd international conference on very large data bases (VLDB'96), Morgan Kaufmann Publishers Inc, 1996, pp: 63–74.

DOI: 10.17148/IJARCCE.2022.11748

[12]. S. Kozielski and R. Wrembel, "New Trends in Data Warehousing and Data Analysis", Annals of Information Systems, Springer, Vol. 3, 2008, pp: 19-49.
[13]. M. Bokun and C. Taglienti, "Incremental Data Warehouse Updates," Information Management Magazine, 1998.
[14]. Liu Jun, Hu ChaoJu, Yuan HeJin, "Application of Web Services on The Real-time Data Warehouse Technology", Advances in Energy Engineering (ICAEE), 2010 International Conference, 2010, pp: 335 – 338.
[15]. Eccles, Mitchell J., David J. Evans, and Anthony J. Beaumont, "True real-time change data capture with web service database encapsulation.", 6th World Congress on Services. IEEE, 2010.
[16]. C. R. Valêncio, 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 Proceedings of The International Conference on Parallel and Distributed Computing, Applications and Technologies, 2013.
[17]. Sukarsa, I. Made, Ni Wayan Wisswani, and IK Gd Darma, "Change data capture on OLTP staging area for nearly real time data warehouse base on database trigger.", International Journal of Computer Applications, Vol. 52, Issue 11, 2012, pp: 32-37.
[18]. Denny, Atmaja, I. Putu Medagia, Ari Saptawijaya, and Siti Aminah, "Implementation of change data capture in ETL process for data warehouse using HDFS and apache spark.", In Proceedings of 2017 International Workshop on Big Data and Information Security (IWBIS), IEEE, 2017, pp. 49-55.
[19]. Biswas, Neepa, Anamitra Sarkar, and Kartick Chandra Mondal, "Efficient incremental loading in ETL processing for real-time data integration.", Innovations in Systems and Software Engineering, Vol. 16, Issue 1, 2020, pp: 53-61.
[20]. Vijayalakshmi, M., and R. I. Minu, "Incremental Load Processing on ETL System through Cloud.", In Proceedings of the International Conference for Advancement in Technology (ICONAT), IEEE, 2022, pp: 1-4.
[21]. Chakraborty, Sonali, "A Novel Approach of Using Materialized Queries for Retrieving Results from Data Warehouse." In Proceedings of the International Conference on Intelligent Vision and Computing, Springer, Cham, 2022, pp: 22-35.
[22]. Alkhayat, Zainab, and Kadhim BS Aljanabi, "Optimal data warehouse design with data marts and data cube aggregation." In Proceedings of 3rd International Scientific Conference of Alkafeel University, Vol. 2386, Issue 1, 2021, pp: 050012.
[23]. R. Kimball and J. Caserta, The Data Warehouse ETL Toolkit : Practical Techniques for Extracting,Cleaning,Conforming,and Delivering Data, ser. The Data Warehouse Toolkit. Wiley Publishing, Inc, 2004.
[24]. M. Bokun and C. Taglienti, "Incremental Data Warehouse Updates," Information Management Magazine, 1998.