2013 International Conference on Parallel and Distributed Computing, Applications and Technologies

Real Time Delta Extraction Based on Triggers to Support Data Warehousing

Carlos Roberto Valêncio, Matheus Henrique Marioto, Geraldo Francisco Donega Zafalon, José Márcio Machado, Julio César Momente
Departamento de Ciências de Computação e Estatística
São Paulo State University
São José do Rio Preto, São Paulo, Brazil
valencio@ibilce.unesp.br, matheus.comp@gmail.com, zafalon@gmail.com, jmarcio@ibilce.unesp.br, juliocesar.momente@gmail.com

Abstract— **Nowadays large corporations require integrated data from diverse sources, leading to the use of data warehouse architectures for this purpose. To bypass problems related to the use of computational resources to process large volumes of data, an ETL (Extract, Transform and Load) technique with zero latency can be used, that works by constantly processing small data loads. Among the extraction techniques of the zero latency ETL are the use of logs, triggers, materialized views and timestamps. This paper proposes a structure capable of performing this task by means of triggers and a tool developed for the automatic generation of the SQL (Structured Query Language) code to create these trigger, besides showing its performance and comparing it to other techniques. Said method is relevant for the extraction of portions of selected information as it permits to combine conventional and real time ETL techniques.**

Keywords- data warehouse; ETL (Extract, Transform and Load); CDC (Change Data Capture); trigger; automatic code generation

I. INTRODUCTION

One of the leading challenges for large corporations is to obtain and analyze large volumes of data coming from various sources [1], [2]. To create support structures for this exigency, different architectures are presented, having the data warehouse [3], [4] model as their main approach.

ETL tools are used to feed data to the data warehouse. These tools integrate data in three stages: extraction, which must be applied to various heterogeneous sources; transformation, used to guarantee standard, quality and reliable data; and the storage of data in a common final repository [5].

Even with all of today's computational support, there is still a new challenge: perform the previously described task in the shortest possible time, so that offered data may be more precise and recent for decision making [6]. In this context the zero latency ETL [7] technique is applied, where its first step, the data extraction, can be based on triggers, snapshots, logs and materialized views [8].

This work presents the architecture of a zero latency extraction tool of information from a relational database, with a future application in a zero latency ETL tool. The chosen strategy to capture altered data is based on the use of triggers [9], since research in this field is not conclusive about its scalability and time overhead.

II. MOTIVATION

To obtain and analyze large volumes of data coming from various sources is not a trivial task as, normally, the data is heterogeneous both to logical and structural levels, and also because large corporations need to extract continually updated data for their major decision-making. In the scientific field, this is also a challenge as the tools already proposed are ineffective in this aspect due to the exigency that they only be applied in the periods in which data sources are not being used. This type of approach is described as an offline tool execution; the data warehouse will generally be out of date during the day as these tools are usually applied at night [10].

On the other hand, there is a zero latency extraction approach that differs from the offline extraction as it is active during the same period in which the source system is used. With this, there will be a concurrence in the use of computational resources. The extraction process should therefore be perfected so that its execution will not overload the source system.

Retail chains are a good example where zero latency integration is necessary. Sales information has a direct impact on stock management and products price, and the use of this information in real time can maximize profits and be a strategical benefit against other companies.

III. CONTRIBUTION

This work focuses on the data extraction step of the zero latency ETL process, while the transformation and storage steps are not addressed because they are outside of its scope. The contributions of this work are: a proposition of a triggers based structure for the capture and storage of altered data; a method to automatically generate these triggers code and a study about the scalability of the solution.

Firstly, it makes possible the extraction of only specific parts of the delta to combine online and offline extraction techniques, which balances the user needs and the system performance. Secondly, the proposed tool also eliminates the difficulty of coding large blocks of trigger creation scripts, once such codes are created automatically. Finally, it presents a detailed study about the scalability of the proposed solution, since both related works and commercial tools do not expose conclusive data about time overhead.

978-1-4799-2419-6/13 $31.00 © 2013 IEEE
DOI 10.1109/PDCAT.2013.52

IV. Related Work

Various works in the area of data extraction for ETL tools have been published, especially about the extraction of only altered data, also defined as "Change Data Capture" or CDC. With this method it is possible to obtain a significant performance gain due to a reduction of the mass of data to be worked [8].

In the [6] work is proposed a data load distribution technique that processes data using zero latency ETL techniques in parallel with conventional ETL, where the extraction is based on Oracle CDC capabilities. The [8] work presents a framework for the extraction of altered data using logs generated by its own Database Management System (DBMS) to minimize structure alterations in the database.

The [11] work uses data aggregation and joining techniques to determine any alterations that were done, which eliminates changes on database or system structures but necessitates high computational resources to obtain these alterations. The [9] work presents a test case of extraction based on triggers applied to drugs data warehouse, and the [12] work compares several data extraction techniques. However, scalability tests are not fully explored to demonstrate the applicability of the trigger technique.

As exposed, many works intend to prove the applicability of triggers in the extraction stage, although any of them provides significant data concerning automation or scalability.

V. Architecture of the Zero Latency Extraction Tool

In a zero latency ETL tool, the CDC step is added to the extraction, since it must also be constantly active, storing altered data, so that other steps can process only that information instead of all the information stored in the base. As this capture process happens paralleling the use of the system, it is called online extraction, represented in Fig.1.

One of the problems of the zero latency ETL tools is the processing time as each step depends on previous steps. Therefore, if an extraction step takes a long time, the whole process is affected.

To perform this task, it is necessary to create a structure that is capable of automatically mapping the alterations made on the data stored in the source repositories, making it then possible to take the transformation and loading steps only for the mapped data. Furthermore, the structure must be able of efficiently mapping only those changes that are interesting to the data warehouse, as not all the alterations need to be integrated in the final repository.

The next items present the execution flow of the extraction structure in the database, the format of the script generated by the tool together with the necessary treatment for composite primary keys, besides discussions about other uses of this structure.

A. Execution Flow along the Extraction Step

As from the moment that the script generated by the tool is executed on the database, all the structure that is needed to capture altered data is created. With this created structure, each command executed in one of the chosen tables will activate the following flow of actions within the DBMS:

- The SQL command of the user ends its execution;
- If the executed operation (insertion, edition or deletion) is one of the operations that must be reproduced in the data warehouse, a trigger is activated. Otherwise, the extraction does not happen;
- After a trigger is activated, a function referent to the table that is undergoing the operation is executed. This function stores the altered tuple information and concludes the extraction.

Only the tuples that need to be migrated to the data warehouse are captured and identified, so it will be possible to recover altered data in a more direct and precise manner than with the offline integration strategy. Another advantage of this structure is that both the SQL command of the user and the extraction operation are kept within the same transaction, therefore guaranteeing that there will be no loss of data in this step. Fig. 2 shows the flow of executions that happen during an online extraction step.

B. Format of the Script Generated by the Tool

The proposed structure works using the DBMS PostgreSQL. For the storage of altered data, a new instance in the target database containing a single table is created. In this table, called log table, are stored the identifiers of the altered tuples. Said table has six columns:

- log id – an integer type field, having a sequential number, that stores the execution order of the operations;
- instance – a text type field, stores the name of the instance whose tuple was altered;
- table – a text type field, stores the name of the table whose tuple was altered;
- operation – stores the type of operation that was executed: INSERT, UPDATE or DELETE;
- pk column – a text type field, stores the name of the column that was used as a unique identifier in the table whose tuple was altered;
- tuple id – value of the unique identifier of the altered tuple.

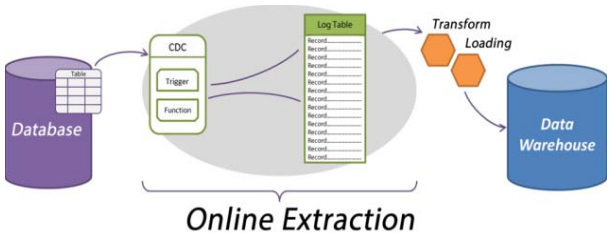

Figure 1. Model of an Online Extraction.

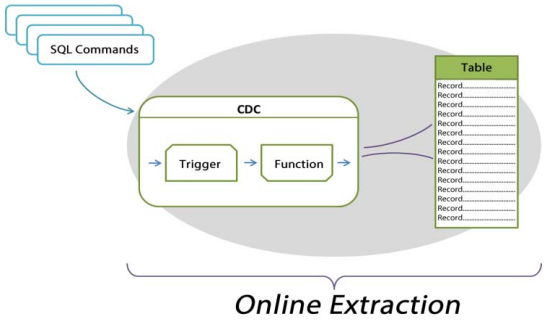

Figure 2. Execution flow along the extraction step.

Besides this new instance and table, a trigger and an extraction function are created for each table to be monitored. The triggers are created together with the monitored table and are responsible for the correct summoning of the extraction functions which, in turn, are created in the new instance generated by the script and are responsible for registering data in the log table, showed in Table I. In the following, the algorithm of the generated triggers is presented.

TABLE I. LOG TABLE

LOG ID	ID	PK	TAB	INST	OPER
1	1001	cust_id	customer	person	U
2	1357	prod_id	product	store	I
3	2910	cust_id	customer	person	D
4	3921	sale_id	sale	store	I
5	4709	sale_id	sale	store	I
6	9801	sale_id	sale	store	I

Algorithm of the generated triggers

```
CREATE TRIGGER cdc_table ON db.table AS
DECLARE @OPER, @INST, @TAB, @PK, @ID
Select @TAB=table of altered tuple
Select @PK= primary key of @TAB
Select @INST= instance in which @TAB belongs
Begin
   IF(@OPER = DELETE)
        Select @ID= ID from tuple before deletion;
   ELSE
        Select @ID= ID from tuple after insertion or update;
   ENDIF
   Insert into log table @ID, @PK, @TAB, @INST, @OPER;
END;
```

The structure of the capture – new instance, new table and extraction functions – are kept separate from the tables and database original instances, with only the triggers created together with the target table. Since the log table stores the identifiers of altered tuples in a generic form, it is then possible to access each target table listed in the log table in order to obtain the altered data using the primary key listed.

C. *Treatment of Composite Keys*

Another problem encountered during the drafting of this work was the existence, in some of the tables, of composite primary keys. In the table created to store the identifier of altered tuples, only a single field should have been available for the storage of the primary key value. Therefore, the only practicable alternative was the alteration of the source table by means of the creation of a new integer auto-increment type column. With that, it was possible to obtain a unique identification for each one of the tuples using only one value.

This new column uses the auto-increment type available in its own DBMS so that there is no impact on the source system. It is not necessary for this value to be passed on by the insertion commands as it is automatically calculated and, in the cases of alteration or removal, does not cause any interference.

D. *Other Applications for the Structure*

During the development of this work, other functions that could be added to the tool after some alterations were studied.

OLTP (On-Line Transaction Processing) systems constitute the base of information systems, and are used in the daily operations of most corporations. Once the structure is implemented in the database of these systems, it will capture and register the commands that were executed in the monitored tables of this database. In the same way, it is possible to also store in the logs table the identification of the user that executed each action, thus obtaining a log of user's actions, quite common in this type of system.

Another benefit that can be explored from the central role of the structure is the use of the changed data to generate safety backups. In this way, it will not be necessary for all the information in the database to be in the backup, only the altered information as from the moment the last backup was generated. Should the database need to be restored, it will suffice to use the initial backup and simultaneously apply each one of the action backups until the desired point.

Lastly, the capture of altered data can also be used to feed asynchronous replication tools between databases, as this can be done in fairly short intervals of time.

VI. EXPERIMENTAL RESULTS

As the most accepted way to prove the efficiency of the extraction step is to measure the execution time of that task, tests were done to compare the execution time of the transactions between a database that did not possess the extraction structure and a repository where the structure was created.

The equipment used to run the tests was a microcomputer having an Intel Core2 Duo P8600 (2.4 Ghz) processor, a DDR2 4 gigabyte main memory on a 400 gigabyte hard disc, SATA standard, transfer rate of 3.0 Gbit/s and 5,400 rpm (rotations per minute). The equipment had Microsoft

Windows Seven Ultimate system software, and the DBMS PostgreSQL 8.3. The tests were done in two stages.

In the first, a fictitious database containing a single table was used. Three operations were done in this table; the first operation only refers to the insertion of tuples; the second to the editing of those inserted tuples; and the third to exclude previously inserted tuples. Moreover, each operation was tested with different quantities of tuples, so it was possible to verify that the time spent on extraction had a linear behavior, that is, grew in proportion to the quantity of treated tuples. From these data, the linearity of the execution time for online capture was verified. Results are shown from Fig. 3 to Fig. 5.

In Fig. 6 the graph shows the quantity of tuples processed per millisecond when comparing the original database to the base modified to capture alterations, while Fig. 7 exhibits the mean percentage increase of tuples processing time for each command.

The second stage of the tests concentrated on the use of the tool on a real database, capturing data altered during the use of the system. All the commands – insertion, edition and deletion – were applied during test intervals. Again, execution times were obtained for determined quantities of commands, as can be seen in Fig 8.

In Fig. 9 the graph shows the percentage increase of processing time needed to process each quantity of tuples, and the average between the increases.

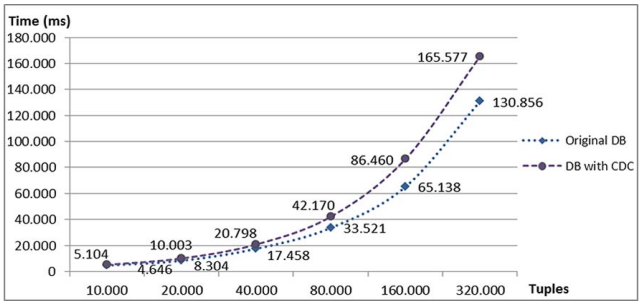

Figure 3. Execution Time of the INSERT Command.

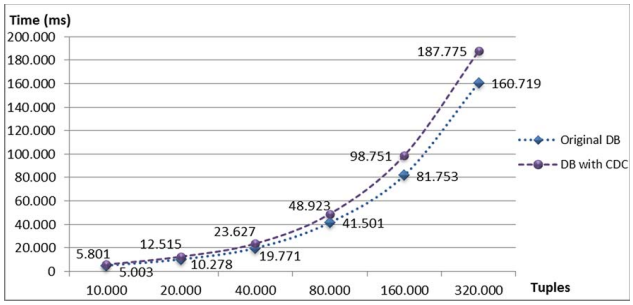

Figure 4. Execution Time of the UPDATE Command.

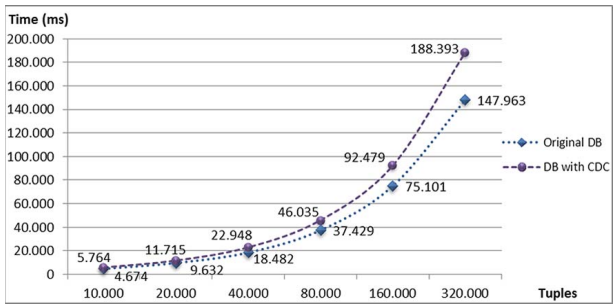

Figure 5. Execution Time of the DELETE Command.

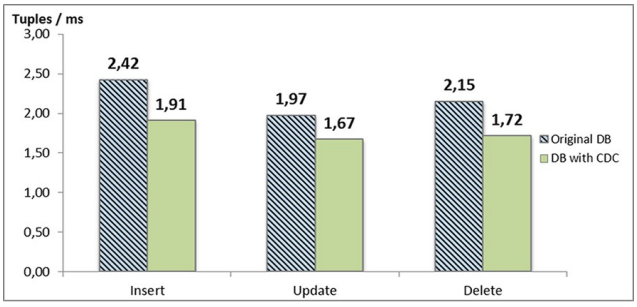

Figure 6. Tuples Processed per Millisecond.

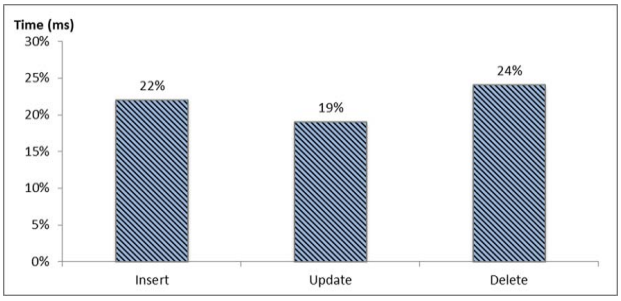

Figure 7. Increase of Processing Time.

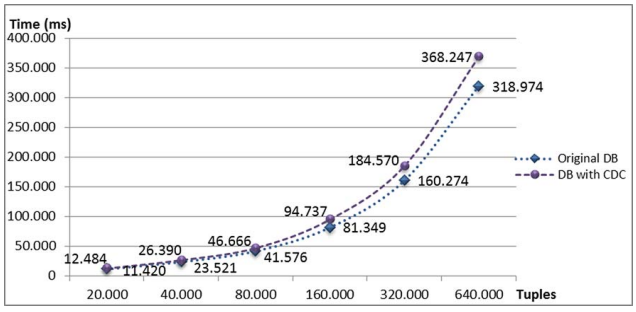

Figure 8. Real System Test.

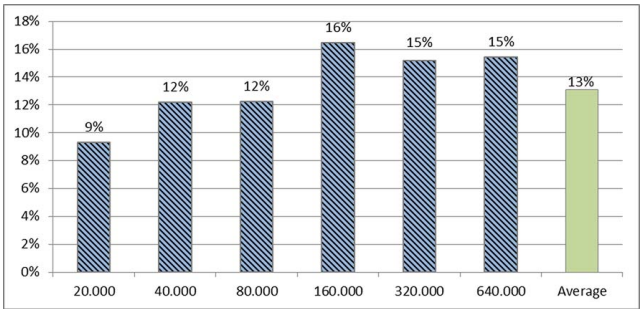

Figure 9. Percentage Increase of Processing Time.

VII. CONCLUSIONS

The tool presented in this work fulfills its objective that is to automatically create all the necessary structure to online extraction all altered data in a determined time interval, aiming to feed a zero latency ETL tool. However, results obtained from a test using fictitious data have shown that the generated overload time was higher than expected as, in literature an acceptable rate is between 5% and 13% [12]. Nevertheless, in this work tests using a real system have shown an acceptable average overload time of 13%. This discrepancy with the previous test may be attributed to the fact that, in real systems, the insertion, edition and deletion operations are used in parallel in different frequencies and quantities, and not separately as done in the first test.

Another important point is that the tool also enables the extraction of any subset of information held in the database, enabling the use of online and offline extraction techniques. The total overload time caused by this method will then be proportional to the portion of online captured data. Also, the use of triggers causes structural changes concerning only the database being monitored, and avoids overloads on other databases managed by the same DBMS.

In the case of using this structure for other purposes such as backups, data replication and maintenance of user's actions, its use becomes more attractive since the tool increases the number of offered services without decreasing its performance. An extension of the tool to other DBMSs, as for example Oracle or SQL Server, can be considered as a suggestion for future works. In the same way, the performance of the structure in these environments can also be studied.

REFERENCES

[1] A. Halevy, A. Rajaraman, and J. Ordille, "Data integration: the teenage years," in *32nd international conference on Very large data bases*, 2006, pp. 9–16.

[2] R. Hou, "Analysis and research on the difference between data warehouse and database," in *Proceedings of 2011 International Conference on Computer Science and Network Technology*, 2011, pp. 2636–2639.

[3] S. Saroop and M. Kumar, "Comparison of data warehouse design approaches from user requirement to conceptual model: a survey," in *2011 International Conference on Communication Systems and Network Technologies*, 2011, pp. 308–312.

[4] G. Liu and Z. Qi, "The application of data warehouse in decision support system," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, 2012, pp. 1373–1376.

[5] T. Jörg and S. Dessloch, "Near real-time data warehousing using state-of-the-art ETL tools," in in *Enabling Real-Time Business Intelligence. Third International Workshop, BIRTE 2009, Held at the 35th International Conference on Very Large Databases, VLDB 2009, Lyon, France, August 24, 2009, Revised Selected Papers*, Springer Berlin Heidelberg, 2010, pp. 100–117.

[6] M. Y. Javed and A. Nawaz, "Data load distribution by semi real time data warehouse," in *2010 Second International Conference on Computer and Network Technology*, 2010, pp. 556–560.

[7] T. M. Nguyen and A. M. Tjoa, "Zero-latency data warehousing (ZLDWH): the state-of-the-art and experimental implementation approaches", *Proc. 2006 International Conference on Research, Innovation and Vision for the Future*, pp. 167-176, 2006.

[8] J. Shi, Y. Bao, F. Leng, and G. Yu, "Study on log-based change data capture and handling mechanism in real-time data warehouse", *Proc. 2008 International Conference on Computer Science and Software Engineering*, vol. 4, pp. 478-481, 2008.

[9] H. Ping, "Research and design of the incremental updates of drug data warehouse", *Proc. 2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, 2010, no. Iccasm, pp. V4-628-V4-630.

[10] P. Vassiliadis and A. Simitsis, "Near real time ETL," *New Trends in Data Warehousing and Data Analysis*, vol. 3, Eds. Boston, MA: Springer US, 2009, p. 32.

[11] D. M. Tank, A. Ganatra, Y. P. Kosta, and C. K. Bhensdadia, "Speeding ETL processing in data warehouses using high-performance joins for Changed Data Capture (CDC) ", *Proc. 2010 International Conference on Advances in Recent Technologies in Communication and Computing*, no. Cdc, pp. 365-368, 2010.

[12] P. Ram and L. Do, "Extracting delta for incremental data warehouse maintenance", *Proc. 16th International Conference on Data Engineering (Cat. No.00CB37073)*, pp. 220-229, 2000.