Study on Log-Based Change Data Capture and Handling Mechanism in Real-Time Data Warehouse

JinGang Shi, YuBin Bao, FangLing Leng, Ge Yu
Department of Computer Science, Northeastern University, Shenyang, China
shijg5@163.com, {baoyubin, lengfangling, yuge}@mail.neu.edu.cn

Abstract

This paper proposes a framework of change data capture and data extraction, which captures changed data based on the log analysis and processes the captured data further to improve the quality of data. Then processed data are pushed to a data queue and the system processes the data queue using priority-based scheduling algorithm. Ultimately processed data are loaded to real-time data warehouse to support decision analysis. After analysis of a test case, this method can capture all changed data coming from the source data in time without changing the structure of the source system, and has a little impact on system performance to the source system. In addition, the real-time scheduling algorithm can effectively improve the data quality and data freshness of the real-time data warehouse to give a better data support for business's routine tactical decision.

1. Introduction

A real-time data warehouse (RTDWH) [1] is required because of the lack of real-time update in traditional data warehouse. A key technology in RTDWH is the capture and extraction of real-time data. This paper provides a general framework and specific change data capture method. This paper implements change data capture by using log-based online analysis. At the same time, the paper presents an import structure based on the priority scheduling algorithm to enhance data quality and data freshness of RTDWH effectively.

2. Contribution and Related Work

The literature [2] introduces a J2EE-based RTDWH architecture; The literature [3] introduces concept of QoS in real-time database; The literature [4] presents a zero-delay data warehouse framework. The literature [5] introduces data capture technologies; The literature [6] explains that change data capture mechanism including change capture agents, changes data services, data distribution, and other components.

These literatures introduce RTDWH structure and change data capture technology. This paper promotes new methods and framework by researching and improving these references.

3. The Framework of Data Capture

This paper mainly studies log-based change data capture methods, but the data capture framework will also support the trigger, data replication, and other capture methods. This paper mainly introduces log-based change data capture as an example.

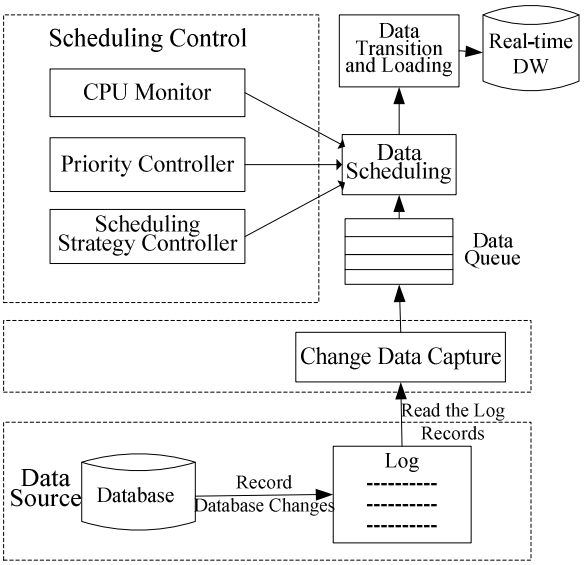

Figure 1. Framework of Log-based change data capture

Design structure of log-based change data capture method is shown in Figure 1. The structure includes log-based change data capture component, scheduling

978-0-7695-3336-0/08 $25.00 © 2008 IEEE
DOI 10.1109/CSSE.2008.926

controller and data loading and transform, and other components.

Scheduling component of tasks adopt a scheduling means of comprehensive FIFS (First in first scheduling) and the priority, to ensure data freshness and data quality in RTDWH. At the same time scheduling controller monitors utilization rate of CPU, using real-time feedback method to adjust scheduling strategy, to provide better data quality assurance.

Data transition and loading component transform and clean scheduled and processed data, and then connect to RTDWH through the loading tools. The paper focuses on data capture strategy of RTDWH, without elaborating that part.

4. Online Log Capture Data

In this paper, we use Oracle database as an example to study. Other databases are somewhat different in the analysis of specific aspects of the database log file. This will not affect the data capture process and system structure.

4.1. The Process of Change Data Capture

Analysis of logs and change data capture process are shown in Figure 2. The process contains the initialization of log, the establishment of data dictionary, loading the log file, log analysis and data collection, and several other processes. The pseudo-code of the process is shown in algorithm 1.

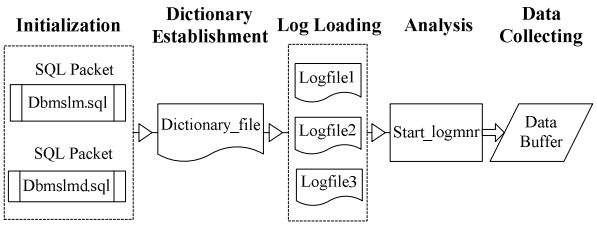

Figure 2. Online log change data capture

Algorithm 1. Change data capture

```
1 BEGIN
2   EXEC SQL EXECUTE
3    begin
4      dbms_logmnr_d.build(dic_filename=>'vdict.ora');
5      dbms_logmnr.add_logfile(LogFileName=>'redo01.log',
                    Options=>dbms_logmnr.new);
6      dbms_logmnr.add_logfile(LogFileName=>'redo02.log',
                    Options=>dbms_logmnr.addfile);
7      dbms_logmnr.add_logfile(LogFileName=>'redo03.log',
                    Options=>dbms_logmnr.addfile);
8      dbms_logmnr.start_logmnr(DictFileName=>'vdict.ora');
9    end;
10  END-EXEC;
11  EXEC sql select max(scn) into :scn_now
                    from v$logmnr_contents
                    where timestamp in (Specified Time);
12  EXEC sql select max(taskid) into :taskid from test.task;
13  EXEC sql select max(scn_old) into :scn_old from scn1;
14  if(scn_now == scn_old)
15    return;
16  else
17      Extract the analyzed dictionary information and store
          the incremental data;
18 END
```

In algorithm 1, the steps 2-10 set the data dictionary and load the log data for analyzing; the steps 11-17 analyze the log file, capture changed data information and export the incremental changed data to prepare for the following processes.

4.2. The Process of Change Data Capture

There may be some invalid intermediate data in the changed data. Therefore we need analyze the data to minimize invalid data and improve the quality of data captured. And we may reduce unnecessary data load for the following process by the following process.

In the specified time period, if different change operations occur on the same record (read operation ignored). We can analyze these change operations to remove invalid operations within them. Below we analyze the dual combination of data status (I, D and U representing insert, delete and update; X represents no change in status). Then they constitute a matrix of state SAM(shown in table 1).

Table 1. State Analysis Matrix

		First		
		I	U	D
Last	I	U	U	U
	U	U	U	U
	D	X	D	D

(I, U): Update after the insert, equivalent to the insertion of the updated data. SAM (I, U) = U.

(I, D): Delete after the insert, equivalent to no change in status. SAM (I, D) = X.

For limited length of this paper, all state combinations are no longer listed. And the final status matrix is shown in table 1.We process the incremental changed data from the log analysis and compare with status analysis matrix to minimize the size of data for the following processes and to enhance the quality of data captured. The final data are transmitted to the data queue for the data scheduling component.

5. Data Real-time Scheduling Strategy

Source database will continue to generate new data changes, but the importance of all the data is different. The important data have the priority to be imported to RTDWH. This produces a problem of real-time scheduling strategy to data generated from log analysis.

Based on the fuzzy feedback control real-time scheduling algorithm [8], we design real-time scheduling strategy for RTDWH. Figure 3 show the framework of real-time scheduling. The framework includes the scheduling and controller. The controller is composed of monitor and priority controller, and is the core of the algorithm. There are a total of five components in the framework of the controller.

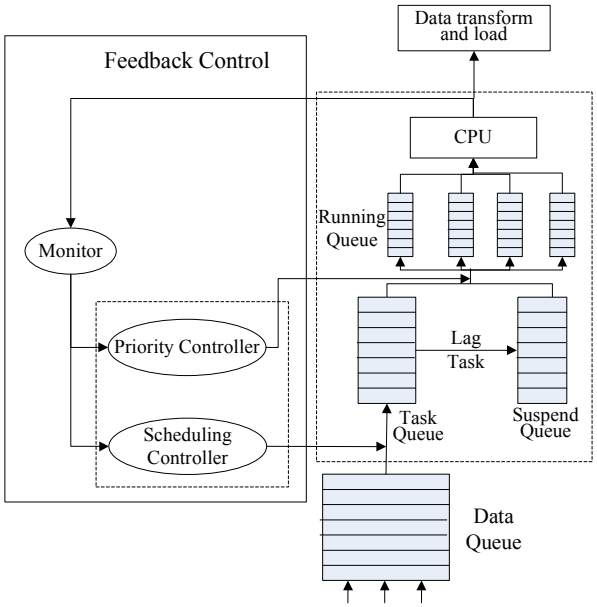

Figure 3. Real-time Scheduling Strategy

Data queue: It is composed of data that are imported by change data capture component and the time index is set on queue.

Scheduling controller: According to different scheduling strategy and CPU utilization ratio, it adjusts the specific scheduling strategy to control data access.

Suspend queue: It stores the data that can not be handled temporarily and sorts them by time.

Priority controller: According to the predefined priority, it distributes the data in the queue into various processing queue to ensure that important data are processed first.

Monitor: It monitors CPU utilization ratio and the task status, giving feedback and adjusting scheduling strategy controller. Then it ensure a more reasonable and efficient process of data.

Scheduling controller solves the task scheduling problems, while addressing priority scheduling problem for the important task. Real-time scheduling strategy algorithm is as follows:

Step 1: Polling data queue. If the data queue has a new task, it is ready to implement Step 2.

Step 2: Scheduling controller sets the priority of all tasks. According to the status of implementation of the current system, it loads tasks to task queue or puts some tasks to suspend queue to prepare for the following scheduling.

Step 3: Priority controller detects the status of task queue and classifies the tasks. According to the priority of tasks, it delivers tasks to a different running queue.

Step 4: To schedule and implement tasks, using the means of cycle judgment.

First of all it judges whether there is a major event in the tasks table. If so, implement the major event first. If not, determine whether there is a new task (a fresh data to first process), then jump to Step 2. If there are no new analysis tasks completed, process it according to the priority of data. Otherwise, after the current task is completed, it shifts uncompleted tasks within the running queue to suspend queue. Then jump to Step 1.

Step 5: If all the tasks in the queue are completed, it implements the unfinished suspend tasks. If it detects a new analysis task completed in the process, it will withdraw from the process and jump to Step 1.

By using real-time scheduling strategy, the system can process more important data better. And in step 4 scheduling strategy is improved, using the priority of new tasks. Thus it gives priority to achieving the freshest data, to ensure data quality and data freshness of RTDWH.

6. Test Case and Results Analysis

Through the test case, there is a good result about log-based data capture framework and method. Below we give brief analysis about the log capture and real-time scheduling strategy.

Table 2 Change Data Capture of RTDWH

Change data capture	Support real-time capture	Impact to source database
Record-based	no	All impact
Reproduce	(transaction)yes (snapshot)no	Performance impact
trigger	yes	Structure impact
DB snapshot	no	Performance impact
Log-based	yes	No impact
Refresh table	no	Performance impact

Table 2 describes the comparison between the log-based capture methods and other real-time data capture methods. As can be seen from the table, other methods have to change structure of the source database heavily,

or have more impact on the performance to the source database. But our method has small impact on the performance and will not change any of the source structure. Also it has a quick response to data changes. So it is a more ideal way to change data capture.

Table 3. The average success rate of different scheduling algorithm (%)

Scheduling algorithm	Important tasks	Ordinary tasks	Non-important tasks
Priority-based scheduling	93.01	65.82	66.35
EDF	55.56	70.64	72.26
LSF	53.05	71.08	71.65

Table 3 describes the comparison of performance between scheduling algorithm used by data queue. As seen from the table, using EDF-based and LSF-based scheduling, the average scheduling success rate is close to all tasks subset. In other words, the success rate is close between the important task and other tasks. However, in the priority-based real-time scheduling algorithm, the more important tasks are processed and the average scheduling success rate of the important task subset is far higher than other. It achieves the purpose that the important tasks are processed first.

When the load has a sudden increase or decrease, we use the priority-based scheduling, combined with feedback control module. Then the miss rate of deadline will be steady in the vicinity of the reference value after it has a big change in the initial stage. If do not use this method, the miss rate will be always high and far away from the reference value, and vary with the different loads.

Finally, this paper improves task scheduling methods of the feedback algorithm and uses the principle that the new task queue of the same rank is processed first. This enhances data quality and data freshness of RTDWH greatly.

7. Conclusions and Further Work

This paper proposes a log-based change data capture and data extraction framework in RTDWH. We elaborate the method of change data capture by using online log and researching data scheduling strategy of captured data. Then it successfully achieves data capture and extraction of RTDWH. Finally, through test case, it is a successful framework and method of data extraction of RTDWH. However, we only introduce the framework and method using Oracle database and need more research and exploration upon other database logs.

References

[1] Michael Haisten. Real Time Data Warehouse: The Next Stage in Data Warehouse Evolution. DM Review, 2003.

[2] Robert M Bruckner, Beate List, Josef Schiefer. Striving to-wards near real-time data integration for data warehouses. DaWaK 2002, 317-326.

[3] K. D. Kang, S. Son, J. Stankovic, T. Abdelzaher. A QoS-Sensitive Approach for Timeliness and Freshness Guarantees in Real-Time Databases. In the 14th Euromicro Conference on Real-Time Systems. 2002, 203-212.

[4] Tho Manh Nguyen, A. Min Tjoa. Zero-Latency Data Warehousing for Heterogeneous Data Sources and Continuous Data Streams. iiWAS 2003.

[5] Lin Ziyu, Yang Dongqing, Song Guojie. Study on change data capture in Real-time data warehouse. *Journal of Computer Research and Development,* 2007, 44: 447-451.

[6] Itamar Ankorion. Change data capture-efficient ETL for real-time BI[J], DM Review, 2005, 16(1):23-27.

[7] Jin Hong, Wang HongAn, Fu Yong, Wang Qiang, Wang Hui. A Fuzzy Feedback Control Real-Time Scheduling Algorithm. *Journal of Software*, 2004, 15(06):791-798.