USING DATABASE AUDIT FOR ANALYZING ON HISTORICAL DATA

Adnan Hodžić
International Burch University
Bosnia and Herzegovina
adnan.hodzic@ibu.edu.ba

Adem Karadag
Turkey
nuhadem@gmail.com

Abstract: *Database auditing is one of the biggest issues in data security. Absence of information auditing drives the business applications to the lost trail of business procedures. To cope with auditing and in order to track operations and the actors of those operations in time, we need historical data or temporary database. Legitimate and exchange times are two important time-stamps in temporary database. In this paper, we show the methods to handle database auditing in business exchange operations, accurate times, and performers of the operations. These strategies are separated in two sets; utilizing relational databases, and utilizing semi-structured information.*

Keywords: *Database Audit, Historical Data*

Introduction

It is very crucial that a company no matter how big is it maintains the security of its information. Since there are many stealing of valuable data such as customers' credit card data, designs and maybe source codes, the data should be protected all the time. Keeping safe your data is protecting its confidentiality, integrity, and availability. To ensure the data security, there should be a security plan. Authentication and administration can facilitate the security at a point (Mullins & Craig, 2002). However, there is a need to keep log files and check them separately from the database. Thus database audit was introduced to inspect the trail maintenance. Data servers help to create a database audit policy to protect the database safe. In this way, user entries can be controlled. There will be some techniques showing how to make database auditing depend on historical data. This paper divided into 4 parts. In part 2 and 3, there is literature review of historical data and auditing. The outline of auditing of database was described in some ways. We used a relational database (Grad, 2013) to represent the row, column based and log-file auditing strategies.

Database Auditing

Database auditing includes inspecting a database to control and view the actions of database users. In this way, auditor can see the manipulations, corruptions or glitches on the data. Database audit also refers to a professional database auditing resolution-giving chance to track and inspect of any database activity involving accessing, login, protection breaches, user activities, insert-delete-change the data. Recently, to supply accurate data auditing a framework has been introduced in respect to data retention strategies. (Lu & Miklau, 2009) Under retention restriction a formula applied to audit data in the protected history. In this way database audit would be more accurate.

It is important to detect changes that are deviates from standard. To differentiate the normal behaviors on the data and have better results in audit, data mining techniques are generally applied. This method can only detect the static actions of the user. This disadvantage can be affected by tracking all activities of user in an data audit system. As a result, anomaly detection method was introduced to model the normal behavior of the user. (Park & Lee, 2008) In this way normal behaviors can be easily differentiated from suspicious ones.

To teach database security and auditing and make the students have better understanding about it, hands –on lab studies are set (Luebbers, Grimmer, & Jarke, 2003) In these studies various database scenario are set to integrate theories of database protection into practices.

Historical Data

Historical data is the information outlining activity, conditions and trends in a company's past database. Historical data is often archived, and may be held in non-volatile, secondary storage. Historical data can be useful in helping to predict the future of a company and a market, as when conducting predictive analyses.

Table 1: Operational Student Table Referenced By Student-History Table For Row-Based Auditing

Student Number	Name	Birth	Adress	Registration Date	Fee
445	Zeynep	10.10.1988	Ankara	15.01.2008	2400
822	Mahmut	12.09.1990	Istanbul	01.09.2010	2600
544	Ayşe	15.05.1991	Istanbul	01.09.2011	2600

It is very significant to detect who made the changes like insertion of a new data, data manipulation or deletion on the database. In this way, a good data audit can be retrieved. The time and the user is important issue to analyze the modification of data. When was the action happened can be answered by valid and transaction times. In a study it is mentioned that valid and transaction times should assure no data loss. (Bhargava & Gadia, 1993)

Arranging Historical Data For Auditing On Relational Database

There are some ways to design historical data in a relational database (Margaret Rouse, 2015) like separated tables for recording past data and transaction log files. The idea of arranging separated tables for each relational database table is easy way to track to changes for each item. With both strategies there is no change on the original data tables. There are 3 ways that we represent here to supply historical data for auditing database. They are auditing on a row level, column level and log-table.

Database Audit on a Row Level

Our original relational tables stay same but we create a separate table for each table to apply data audit. Operational "Student" table as shown in Table 1 supplies the current data of each student for operations. There are 2 kinds of data type in this table;

static and operational data. Static data stays same or rarely change like Student Number, Registration date or Name. Historical or operational data continuously can be updated like address of the student. Static query, which is always used, already stays same to call the data from "Student". Table 2 is an auditing table that includes all students' data in the operational table. Two time intervenes needed for valid times. We need to know the beginning and ending time to sustain the life cycle of the data. Besides the valid time, we acquire to have operation type to diminish the complexity of comparison among histories of the same data and the user to make him responsible from the action.

History of "Student" table is shown in Table 2. It can be seen from history table that Ali Oz has been a student since 01.09.2005. The user Mustafa updated his fee 2 times by increasing by $100 each and updated address by changing it from Istanbul to Adana. Ali has finished the school and his record deleted from the Student table by Semih. Ahmet moved from Hatay to Ankara on 23.09.2008 and his record terminated on January 2009 by Mustafa. Zeynep's fee was increased by $100 by Mustafa. Finally, Semih added two new students Mahmut and Ayşe to the Student table.

Table 2: Operational Student Table Referenced By Student-History Table For Row-Based Auditing

Student Number	Name	Birth	Address	Regist. Date	Fee	Begin	End	Op	User
966	Ali	21.04.1986	Istanbul	01.09.2005	2300	01.09.2005	01.09.2007	I	Mustafa
966	Ali	21.04.1987	Adana	01.09.2005	2300	01.09.2007		U	Mustafa
966	Ali	21.04.1988	Adana	01.09.2005	2450	01.09.2007	23.06.2008	D	Semih
855	Ahmet	11.05.1986	Hatay	01.09.2006	2350	21.09.2007	01.09.2008	I	Semih
855	Ahmet	11.05.1986	Ankara	01.09.2006	2350	23.09.2008	15.01.2009	D	Mustafa
445	Zeynep	10.10.1988	Ankara	15.01.2008	2300	15.01.2008	15.06.2010	I	Mustafa
445	Zeynep	10.10.1988	Ankara	15.01.2008	2400	15.01.2008		U	Mustafa
822	Mahmut	12.09.1990	Istanbul	01.09.2010	2600	01.09.2010		I	Semih
544	Ayşe	15.05.1991	Istanbul	01.09.2011	2600	01.09.2011		I	Semih

Operational table and audit table records are identical. Data is repeated in different rows but this is kept for the sake of historical query.

Database audit on a row level has some advantages and drawbacks. It is easier to apply auditing. When the user wants to insert, update or delete something from the operation table, the program can simply copy the all value in the record into the historical table. Besides, the end column should be updated with the operation. This operation can be achieved by the database as used in (Yang, 2009) article. Drawbacks can be mentioned that redundancy makes the system complicated. Also, calling historical data is needed to the comparison between operational table and auditing table by using recursive query.

```
SELECT S1.fee, MINS, MAXS, S1.USER, OPERATION FROM Student_HISTORY_R S1,
( SELECT S2.fee, MIN(S2.begin) MINS, MAX(S2.end) MAXS
FROM Student_HISTORY_R S2
WHERE Student Number =  966 GROUP BY fee) S3 WHERE S1.fee = S3.fee
```

Database Audit on Column Level

Column level audit is not including redundant data as seen in the row level audit. This historical table does not contain static data like birth date and registration date. The auditing table just sustains the changed data except primary key like student number. This is required to save the data in the operational table. Student history in Table 3 keeps just the changed data and it is less redundant than the Table 2. The student number 966 Ali moved from Istanbul to Adana on 01.09.2007 got raised fee from 2300 to 2450 on

01.09.2007. Selecting not-null value on a particular auditing column in SELECT statement would display only the actual change. For example,

```
SELECT fee, begin, end, USER, OPERATION FROM Student_HISTORY_C
WHERE Student Number = 966 AND fee IS NOT NULL
```

The query displays the auditing of Ali's fee. Comparing with row-based auditing on the same query, the SELECT statement is much less complex.

Each record in column-based auditing table cannot contain more than one value of historical data because of the uncertainty of end time of each auditing data.

Table 3: Student_History_C Table Using Column-Based Auditing

Student Number	Address	Fee	Begin	End	Operation	User
966	Istanbul		01.09.2005	01.09.2007	I	Mustafa
966	Adana		01.09.2007		U	Mustafa
966		2450	01.09.2007	23.06.2008	D	Semih
855	Hatay		21.09.2007	01.09.2008	I	Semih
855	Ankara		23.09.2008	15.01.2009	D	Mustafa
445	Ankara	2300	15.01.2008	15.06.2010	I	Mustafa
445		2400	15.01.2008		U	Mustafa
822	Istanbul	2600	01.09.2010		I	Semih
544	Istanbul	2600	01.09.2011		I	Semih

Since it is less complicated column level audit is faster. Less disk space is used also. However, many NULL values would cause other issues when writing queries

Auditing on Log Table

A log table that tracks changes to a system are also referred audit as it gives a bunch of information like user, data, time of execution that can be used to audit a system. Relational Database Management Systems (RDBMS)'s like audit option like in DB2 (IBM Knowledge Center, 2015), SQL (Stankovic, 2016) and ORACLE Servers (Stackowiak, Bales, & Greenwald, 2004) and facilitate database administrators to sustain an audit trail (Logging, Auditing, and Monitoring the Directory) and saved it in a log file. However, log tables are not keeping the finished time to program. To prevent this, there may be two ways.

Column Based Log Audit Tables for Operation Logs

We need to isolate auditing log data from the operational data. To do this, we make additional table for each auditing column. For instance, if ADDRESS and FEE columns in the STUDENT table are auditing columns, we make ADDRESS and FEE tables for auditing purposes as appeared in Table 4 and Table 5. There are some advantages about this way. First, it decreases the amount of auditing data and it makes it easier to analyze the tables. However, the number of independent tables may increase.

Table 4: Audit Log Table For Address

PK	**Student Number**	**Adress**	**Begin**	**End**	**OP**	**User**
1	966	Istanbul	01.09.2005	01.09.2007	I	Mustafa
2	966	Adana	01.09.2007		U	Mustafa
3	855	Hatay	21.09.2007	01.09.2008	I	Semih
4	855		23.09.2008	15.01.2009	D	Mustafa
5	822	Istanbul	01.09.2010		I	Semih
6	544	Istanbul	01.09.2011		I	Semih

Table 5: Audit Log Table For Fee

PK	**Student Number**	**Fee**	**Begin**	**End**	**Op**	**User**
1	966	2300	01.09.2007		U	Mustafa
2	966	2450	01.09.2007	23.06.2008	D	Semih
3	445	2300	15.01.2008	15.06.2010	I	Mustafa
4	445	2400	15.01.2008		U	Mustafa
5	822	2600	01.09.2010		I	Semih
6	544	2600	01.09.2011		I	Semih

One Log Audit Table for Operation Logs

To join audit data into one spot, we coordinate each auditing column from all operational tables into one single auditing log table. The audit log table makes out of name of table and column, Student ID of the record in the operational table, changed value, begin time, operation that causes the change and name of user who controls this data.

Case of single audit log table of the database containing Student and Faculty tables is appeared in the Table 6. All changes made on the tables is built into the single audit log table. A solitary insertion of Student number 966 into Student table makes the insertion into audit log table two times; one log record for ADDRESS and another for Fee if Student table has two auditing columns. Upgrading on an auditing trait will embed an auditing record into the log table. You can see same action like in insertion; deletion of a record will be logged twice into audit log table if there should be an occurrence of two auditing columns, for example, deletion of Student 966 in Table 6.

Table 6: One Audit Log Table For Every Table; Student And Faculty In Database

PK	**Student Number**	**Table**	**Column**	**Value**	**Begin**	**End**	**Op**	**User**
1	966	Student	Address	Adana	01.09.2007		I	Mustafa
2	966	Student	Fee	2450		23.06.2008	D	Mustafa
4	855	Student	Address	Hatay	21.09.2007	01.09.2008	I	Semih
5	855	Student	Fee	2350	23.09.2008	15.01.2009	D	Semih
6	445	Student	Address	Ankara	15.01.2008	15.06.2010	I	Mustafa
7	445	Student	Fee	2300	15.01.2008		I	Mustafa
8	445	Student	Fee	2400	01.09.2010	13.04.2011	U	Semih
9	822	Student	Address	Istanbul	01.09.2010		I	Semih

10	822	Student	Fee	2600	01.09.2010		I	Mustafa
11	544	Student	Address	Istanbul	01.09.2011		I	Mustafa
12	544	Student	Fee	2600	01.09.2011		I	Semih
13	221	Faculty	Manager	108	01.01.2012		I	Semih
14	103	Faculty	Manager	120	21.06.2013		U	Mustafa

Audit log table is expansive if there are numerous auditing columns from various tables. Separating the data in columns and having a solitary audit log table for every subsystem are suggested. Both methodologies require additional handling for each operation at the databases, particularly, the auditing data. Of course, database motors have as of now controlled log tables. With this additional handling, the general framework will be slowed down.

Conclusion

Operation tables and auditing tables should be apart from each other. In this way database engine could be much faster in running the auditing query when we compare a table includes both operational and auditing data. Overhead of checking which partition will be used against the query is added to execution time. Also, database administrator would manage the database management system easier.

There are many options for auditing database. Some solutions are appropriate for relational databases. On the other hand, marketing databases are mostly using semi-structured databases.

Database auditing is one of the crucial issue for a company to maintain its' not only security-related concerns but also performance and reliability. Monitoring and recording of selected user database actions determine the future of the company's business. Overall, security and reliability of the data can be sustained by a good database auditing method

References

- Bhargava, G., & Gadia, S. K. (1993). Relational Database Systems with Zero Information Loss. *5* (1), 76- 87.
- Grad, B. (2013). Relational Database Management Systems: The Business Explosion. *IEEE Annals of the History of Computing archive , 35* (2), 8-9.
- IBM Knowledge Center. (2015, January 15). Retrieved April 30, 2016, from ibm.com: http://www.ibm.com/support/knowledgecenter/#!/SSEPGG_8.2.0/welcome.html
- Lu, W., & Miklau, G. (2009). Auditing a Database Under Retention Restrictions. *IEEE Inter. Conf. on Data Eng.* (pp. 42-53). ICDE.
- Luebbers, D., Grimmer, U., & Jarke, M. (2003). ystematic Development of Data Mining- Based Data Quality Tools. *proc. of the 29th VLDB Conference*, (pp. 548 - 559). Berlin.
- Margaret Rouse. (2015). *Relational database management systems (RDBMS)*. Retrieved April 5, 2016, from TechTaregt: http://searchsqlserver.techtarget.com/definition/relational-database-management-system

- Mullins, & Craig. (2002). *Database administration: the complete guide to practices and procedures.* Addison-Wesley.
- Park, N. H., & Lee, W. S. (2008). Anomaly Detection over Clustering Multi-dimensional Transactional Audit Streams. *IEEE International Workshop on Semantic Computing and Applications* (pp. 78-80). IWSCE.
- Stackowiak, R., Bales, D., & Greenwald, R. (2004, August 26). *Oracle Docs*. Retrieved April 30, 2016, from docs.oracle.com: http://download.oracle.com/docs/cd/B14099_19/idmanage.1012/b14082/logging.htm#i126963
- Stankovic, I. (2016, April 5). *SQL Server Audit (Database Engine)*. Retrieved April 30, 2016, from msdn.microsoft.com: https://msdn.microsoft.com/en- us/en%20us/library/cc280386.aspx
- Yang, L. (2009). Teaching Database Security and Auditing. *SIGCSE*, (pp. 241-245).