DM Review

August 24, 2007

Database Access Auditing: Who Did What to Which Data When?

by Craig S. Mullins

As just about anyone in business these days knows there is a growing list of government regulations that organizations must understand and comply with. This increasing compliance pressure is particularly intense on data stored in corporate databases. Companies need to be ever more vigilant in the techniques used to protect their data, as well as to monitor and ensure that sufficient protection is in place. Such requirements are driving new and improved software methods and techniques.

One of these techniques is database auditing. At a high level, database auditing is basically a facility to track the use of database resources and authority. When auditing is enabled, each audited database operation produces an audit trail of information including information such as what database object was impacted, who performed the operation, and when. The comprehensive audit trail of database operations produced can be maintained over time to allow DBAs and auditors, as well as any authorized personnel to perform in-depth analysis of access and modification patterns against data in the DBMS.

Database auditing helps to answer questions like "Who accessed or changed data?" and "When was actually changed?" and "What was the old content prior to the change?" Your ability to answer such questions is very important for regulatory compliance. Sometimes it may be necessary to review certain audit data in greater detail to determine how, when, and who changed the data.

Why would you need to ask such questions? Consider, HIPAA the Health Insurance Portability and Accountability Act. This legislation contains language specifying that health care providers must protect individual's health care information even going so far as to state that the provider must be able to provide an individual a list of everyone who even so much as looked at their information. Think about that? Could you produce a list of everyone who looked at a specific row or set of rows in any database under your control?

Tracking who does what to which piece of data is important because there are many threats to the security of your data. External agents trying to compromise your security and access your company data are rightly viewed as a threat to security. But industry studies have shown that the majority of security threats are internal – within your organization. Indeed, some studies have shown that internal threats comprise 60% to 80% of all security threats. The most typical security threat comes from a disgruntled or malevolent current or ex-employee that has valid access to the DBMS. Auditing is crucial because you may need to find an unauthorized access emanating from an authorized user.

But keep in mind that auditing tracks what a particular user has done once access has been allowed. Auditing occurs post-activity; it does not do anything to prohibit access. Audit trails help promote data integrity by enabling the detection of security breaches, also referred to as intrusion detection. An audited system can serve as a deterrent against users tampering with data because it helps to identify infiltrators.

There are many situations where an audit trail is useful. Your company's business practices and security policies may dictate a comprehensive ability to trace every data change back to the initiating user. Perhaps government regulations (such as the Sarbanes-Oxley Act) require your organization to analyze data access and produce regular reports. You may be required to produce detailed reports on an ongoing basis, or perhaps you just need the ability to identify the root cause of data integrity problems on a case-by-case basis. Auditing is beneficial for all of these purposes.

A typical auditing facility permits auditing at different levels within the DBMS, for example, at the database, database object level, and user levels. One of the biggest problems with existing internal DBMS audit facilities is performance degradation. The audit trails that are produced must be detailed enough to capture before- and after-images of database changes. But capturing so much information, particularly in a busy system, can cause performance to suffer. Furthermore, this audit trail must be stored somewhere which is problematic when a massive number of changes occur. Therefore, a useful auditing facility must allow for the selective creation of audit records to minimize performance and storage problems.

There are several different names used for database auditing. You may have heard database auditing capabilities referred to as any of the following:

- Data Access Auditing
- Data Monitoring
- Data Activity Monitoring

Each of these is essentially the same thing: monitoring who did what to which piece of data when. In addition to database auditing, you may wish to include database authorization auditing, which is the process of reviewing who has been granted what level of database access authority. This typically is not an active process, but is useful for regularly reviewing all outstanding authorization to determine if it is still required. For example, database authorization auditing can help to identify ex-employees whose authorization has not yet been removed.

Database Access Auditing Techniques

There are several popular techniques that can be deployed to audit your database structures. Let's briefly discuss three of them and highlight their pros and cons.

The first technique is trace-based auditing. This technique is usually built directly into the native capabilities of the DBMS. Commands or parameters are set to turn on auditing and the DBMS begins to cut trace records when activity occurs against audited objects. Although each DBMS offers different auditing capabilities, some common items that can be audited by DBMS audit facilities include:

- login and logoff attempts (both successful and unsuccessful attempts)
- database server restarts
- commands issued by users with system administrator privileges
- attempted integrity violations (where changed or inserted data does not match a referential, unique, or check constraint)
- select, insert, update, and delete operations
- stored procedure executions
- unsuccessful attempts to access a database or a table (authorization failures)
- changes to system catalog tables
- row level operations

The problems with this technique include a high potential for performance degradation when audit tracing is enabled, a high probability that the database schema will need to be modified, and insufficient granularity of audit control, especially for reads.

Another technique is to scan and parse the database transaction logs. Every DBMS uses transaction logs to capture every database modification for recovery purposes. Software exists that interprets these logs and identifies what data was changed and by which users. The drawbacks to this technique include the fact that reads are not captured on the logs, there are ways to disable logging that will cause modifications to be lost, performance issues scanning volumes and volumes of log files looking for only specific information to audit and the difficulty of retaining logs over long periods for auditing when they were designed for short-term retention for database recovery.

Additionally, third party vendors offer products that scan the database logs to produce audit reports. The DBMS must create log files to assure recoverability. By scanning the log, which has to be produced anyway, the performance impact of capturing audit information can become a non-issue.

The third database access auditing technique is proactive monitoring of database operations at the server. This technique captures all SQL requests as they are made. It is important that all SQL access is audited, not just network calls, because not every SQL request goes over the network. Proactive audit monitoring does not require transaction logs, does not require database schema modification, and should be highly granular in terms of specifying what to audit.

The Questions That Must be Answerable

As you investigate the database access auditing requirements for your organization, you should compile a list of the types of questions that you want your solution to be able to answer. A good database access auditing solution should be able to provide answers to at least the following questions:

1. Who accessed the data?
2. At what date and time was the access?
3. What program or client software was used to access the data?
4. From what location was the request issued?
5. What SQL was issued to access the data?
6. Was the request successful; and if so, how many rows of data were retrieved?
7. If the request was a modification, what data was changed? (A before and after image of the change should be accessible)

Of course, there are numerous details behind each of these questions. A robust database access auditing solution should provide an independent mechanism for the long-term storage and access of audit details. The solution should offer the canned queries for the most common types of queries, but the audit information should be accessible using industry standard query tools to make it easier for auditors to customize queries as necessary.

Synopsis

Database auditing can be a crucial component of database security and compliance with government regulations. Be sure to study the auditing capabilities of your DBMS and to augment these capabilities with third party tools to bolster the auditability of your databases.

From DM Direct, August 24, 2007.