Liquibase

WHITE PAPER

5 Database Security Tips to Avoid Million Dollar Mistakes

Executive Summary

How are security violations in SQL scripts prevented at your company?

If your company is like most, that critical responsibility falls to the DBA. DBAs are typically the last and only safeguard against security violations that can put your company at risk. It's been that way for decades. And relying solely on the DBA to protect your data has been a reasonably successful security methodology—until recently.

The IT landscape has undergone significant changes in recent years. The development of new software applications has exploded. Countless new application releases occur every single day. And like other IT staff members, DBAs have struggled to keep pace with the ever-increasing demands upon their time and talents. Less time is available for the manual performance of routine chores. That means, of course, that harried DBAs are more likely than ever to rush through the completion of routine tasks. That also means that less time is spent in reviewing database change requests for security issues.

The ultimate result is the diminishment of database security. The incessant pressure to do more in less time is increasing the likelihood of the creation of inadvertent security holes—by a permission grant, for example, or a stored procedure. It's a burgeoning threat that will only grow stronger as the workloads of IT staff continue to escalate.

There is a simple solution to the problem; a solution that progressive companies are implementing on a wide scale. The solution is to simply automate the performance of routine and repetitive database deployment chores. Automation relieves DBAs from the incessant pressure of performing never-ending deployment chores, freeing them to focus more on tasks and problems that truly require human intervention. Most importantly, automation eliminates the critical mistakes that are the unavoidable by-products of forcing time-pressured DBAs to perform routine, repetitive tasks. And as a bonus, automation can perform routine deployment tasks with levels of speed and efficiency that no human could hope to match.

When automating database changes, our experience in working with thousands of DBAs has shown the importance of implementing certain security practices. This paper will list the most important of those security practices, and discuss the importance of each.

Are You Relying Upon TSA-Style Database Security?

It seems that news stories detailing some embarrassing failure of the Transportation Safety Administration are released on a near-daily basis. Sometimes the news is about a test scenario that TSA screeners failed, revealing dangerous flaws in the system. Sometimes the news is a little more serious, detailing a real failure that occurred during the screening process that potentially placed a planeload of passengers at risk.

In virtually every such story that's reported, a common culprit is blamed: harried agents that are overworked in trying to process the long lines of disgruntled passengers. The agents' workload isn't the only source of problems, of course. Sometimes simple incompetence is to blame. And upon occasion, even nefarious intent may factor into the equation. But the problem of too few agents trying to do too much has certainly primed the pump for this seemingly unending flow of news stories.

The TSA's problem is analogous to the database security risk facing most companies. Most DBAs these days are harried, overworked people. Most are also very competent, highly skilled professionals. But they are simply being asked to do too much.

It's a fairly new problem. For decades we've relied upon DBAs to perform the routine, repetitive tasks that are inherent with maintaining a database. We've also relied upon DBAs to assure that inadvertent errors did not occur during the process of performing database deployment tasks. In many cases just one person, the DBA, bore the burdens and responsibilities of assuring that routine deployment tasks were completed on time, and without the commission of errors resulting from unintended actions. In essence, one person served as a de-facto firewall against potential disaster. And though reliance upon a single employee seems a flimsy defense against a potential plethora of problems, it's a system that, for decades, has been reasonably successful.

Oopsie!

Even the most innocent of mistakes made in the process of implementing database change can have devastating consequences. Consider the following two high-profile cases…

1. Where'd It Go!?

A DBA at start-up Gliffy was making a few "routine" changes when he accidentally deleted the entire production database. They were able to restore the data, but it took a few days, and obviously resulted in a considerable PR black eye. And if that DBA was harried, hurried, and stressed before the accident…

2. OK, We're Garnishing Your Paycheck for the Next 4,000 Years

This is probably the granddaddy of glitches. Knight Capital Group's computer system errantly executed a series of stock trades that cost the company $440 million in less than one hour. Nearly bankrupted the company. Only a last-minute infusion of cash from a group of investors kept the company solvent.

But no more. Rare is the DBA that's not pressed to do more in less time than ever before. The results are predictable: mistakes that threaten database security are on the rise.

The Right People Doing the Wrong Things

The past reliance upon the DBA serving as the primary security shield for the database focused upon placing the right people in the job. The reasoning seemed logical if simplistic: Carefully vet the people granted DBA-level authority, and then rely upon them to always wield their power appropriately. Rely upon the 'right' people to do the right thing.

But in practice, that naive philosophy is flawed. It's built upon a foundation of assuming (hoping!) that humans can provide the error-free reliability of machines. Even the most wonderfully talented and dedicated of DBAs are capable of making basic mistakes, such as:

- Applying an intentional change to the wrong database, or conversely, applying an unintentional change to the intended database
- Simple errors of omission
- Failure to complete critical tasks in time

Human failings have always been a potential point of weakness in database security. But that security threat has magnified as the demands upon DBAs have grown. The more intense the workplace pressure, the greater the likelihood that the right people will do the wrong things.

Practices for Enhancing DB Security with Automation Practices for Enhancing DB Security with Automation

There's a simple solution for avoiding the security risks inherent with relying upon DBAs to perform all database deployment chores: automation. Automation shifts the burden of performing these repetitive chores from humans to machines. It's certainly a sensible change. Humans aren't particularly adept at performing repetitive chores without introducing errors, but machines are. Relieving DBAs from these time-consuming chores also frees them to spend more time focusing upon tasks that truly require the not-so-machinelike benefits of human creative thought.

Most importantly, automation eliminates the critical mistakes that are becoming more and more commonplace, and that threatens database security at most companies. And it's certainly a nice bonus that automation can perform these tasks faster and far more efficiently.

When automating database changes, adherence to the following five security practices can help you maximize the security benefits of automation.

1. Use Role Accounts

You may have configured your database server to rely on users' specific accounts, perhaps via a single sign-on solution. That's great for the security people that oversee the entire company, but it can create a management nightmare for DBAs. Relying upon individual accounts requires DBAs to elevate individual accounts permissions.

Instead of allowing elevated permissions for user accounts, rely on role accounts. Don't grant SYSDBA to anyone, for example. There is very little that a single person should need to do that requires SYSDBA instead of some other narrower permission set. Yet keeping track of who has what permission is very difficult. This challenge becomes even more complex if people switch departments or projects with some frequency. Relying on role accounts will allow you to better limit access. And, when you have concerns too many people are using the account, you can simply change a single password to lock it down again.

2. Be Alert for GRANT Statements In All SQL Scripts

All SQL to be executed should always be checked for GRANT statements. Consider, for example, SQL scripts forwarded from the development team for an upcoming application release. There is simply no reason that such scripts should include a GRANT statement. Though intentions may be good (i.e. some developer is trying to be "helpful"), it's also possible that someone is simply trying to help themselves. (Of course, you should also keep a watchful automated eye peeled for potentially damaging words like DROP and TRUNCATE while you're at it!)

3. Audit Your GRANTs

You should regularly review your GRANTs and permission sets. The older the application, and the more transactions the application has, the more important this becomes. GRANTs tend to accumulate like wire hangers and disposable chopsticks. As a database ages, the likelihood that GRANTs will be applied increases. Many of the accumulated GRANTs may be orphaned, so to speak, with the original reason for the GRANT now moot and possibly even forgotten. Regularly auditing GRANTs helps you to control GRANT sprawl before it becomes unwieldy. If you're already afflicted with GRANT sprawl, beware: you might have GRANTs that you do not need and that pose a significant security vulnerability. Regular audits and enforcement will also help to resolve this problem.

4. Independent DB Account for Each Application

No two applications are the same. Even though they may have access to the same database schema and objects, you should provide each with a different DB account. Doing so will allow you to see who accessed what and when. This information is invaluable in ensuring that others haven't accessed these accounts and used them improperly. Maintaining independent DBs will also enable the ability to trace back to determine how a user might have acquired an application login. And as a fringe benefit, you'll have the ability to treat these accounts like role accounts and disable them if necessary.

5. Regularly Audit a Subset of Your Changes

Ideally, you would be able to validate each and every one of your changes (Pro Tip: Liquibase Enterprise does that out of the box). But for some teams, auditing every single change is simply too difficult and time consuming. Instead, select a few of your most recent changes and validate them against your corporate, technical, and regulatory standards.

You won't catch everything that might have slipped by the DBAs. But by implementing random auditing you can expect to find and resolve issues, and use them as teachable moments for your team.

Head in Sand?

The increasing do-more-with-less pressure faced by most IT departments these days has yielded a number of undesirable side effects. One such is the growing "I'm too busy…" attitude. More and more, certain critical tasks are remaining undone, just because "I'm too busy." The failure to consistently and thoroughly review database changes is one result ofmthis attitude.

This isn't news to many IT executives. They know that a bit of a laissez-faire attitude has crept into the approach to database change deployment. But what to do? Many choose to just simply ignore the problem. Perhaps disaster has been avoided to date, and there's always the hope that such good luck will continue.

The better approach is to automate the deployment process for database changes. It's an approach that eliminates much of the potential for human error. It helps restrict database access to those that truly have a legitimate need. It provides an additional barrier to bad DB code putting your business at risk. And while relieving the burdens upon DBAs, automation can simultaneously leverage the invaluable tribal knowledge and expertise that DBAs have to offer.

Rare is the overworked DBA that will dispute that the age of automating database change deployment has truly arrived.

Automate With Ease

Automating database change can unburden your DBAs and significantly enhance database security. And implementing automation can be surprisingly easy. Liquibase Enterprise provides a simple and intuitive means of validating any and all database changes using custom rules. Datical also provides the important benefit of forecasting the impact of your planned changes, and enables the rehearsing of DB changes prior to pushing them to high-risk production environments.

To learn more about Liquibase and our approach to automating database changes, please visit **www.liquibase.com**.

www.liquibase.com | info@liquibase.com