QUEST SOFTWARE
Smart Systems Management

Automating DBA Processes for Microsoft® SQL Server®

Editor
Kevin Kline, Quest Software

Contributing Authors
Allen White, Scalability Experts
Brent Ozar, Quest Software
Rod Colledge, www.rodcolledge.com,
author of "SQL Server 2008 Administration in Action"

Round Table Contributors
Dan Jones, Program Manager, SQL Server, Microsoft
Thomas LaRock, Database Engineering Lead, investment management company
Charley Hanania, Production Product Owner, SQL Server, UBS
Jonathan Kehayias, SQL Server DBA, OSI Restaurant Partners Inc.

Code Contributors
Michelle Ufford, Senior Database Administrator, GoDaddy
Bart Duncan, Software Engineer, Microsoft
Paul Ibison, SQL Server MVP
Ola Hallengren, Database Administrator, Sweden
Glenn Berry, Database Architect, NewsGator
Chad Miller, Senior DBA Manager, Raymond James

WHITE PAPER

Quest Software World Headquarters
LEGAL Dept
5 Polaris Way
Aliso Viejo, CA 92656
www.quest.com
email: **legal@quest.com**

Trademarks

Updated—January 2010

Contents

Introduction

About this Document

At a recent PASS[1] Summit gathering, a group of well-known SQL Server experts had a free-ranging chat about best practices for automating database administration processes and activities. This white paper presents many of the elements of their discussion.

A Brief History of the Automation of Database Administration

Database administration is not a new profession; it has been around since mainframe times, when most automation was accomplished using JCL (IBM's Job Control Language). Mainframe IT environments required strong job management because they were batch processing systems that did not allow interactive user input.

As technology advanced into the client-server era, dynamic user interaction became the norm. But many applications and databases were small and simple enough to be managed with simple operating system jobs—cron jobs on Unix and Windows scheduler jobs on Windows. These automation technologies didn't support strong IF-THEN-ELSE conditional tests, but those features were rarely needed, and a few third-party applications met the needs of those few IT shops that required more robust features.

As we entered the 21st century, highly scalable database applications required increased sophistication in job management and automation. New techniques and technologies were needed by a growing number of IT shops to automate as many aspects of their operations as possible.

Specifically, two major aspects of applications launched during the Internet era forced new ways to automate processes:

- First, many applications were scaled up, not by utilizing bigger and more powerful servers, but by scaling out to more and more commodity servers. This meant that database administrators (DBAs) needed to be able to automate activities across many servers simultaneously to help ensure standardization and consistency.
- Second, the automation needs of many DBAs grew in sophistication and complexity. DBAs no longer defined automation as "start JobA at 1:00 a.m." and "start JobB at 2:00 a.m." Instead, DBAs began looking for a rich set of features in their automated process that take into account job completion time (e.g., if JobA finishes at 1:10 a.m., why not run Job B immediately instead of at 2:00 a.m.?), computational resources (e.g., if Job A consumes 100 percent of the system CPU, let's throttle it back but allow it to take longer to complete), and conditional situations (e.g., if Job A returns a status code of 1, start Job B; if it returns a status code of 2, start Job X; and if it returns a status code of 0, notify the DBA).

This white paper discusses many of the issues to think about concerning automation, including best practices when automating and recommendations from experienced professionals on how they best met their automation needs.

1 To find out more about PASS, the Professional Association for SQL Server, go to http://www.sqlpass.org.

Why Automate?

If you're not automating your database administration, why should you start now? There are three main reasons:

1. You can do more [computer] work with less [human] work
2. Automation provides standardized processes that are executed completely and consistently
3. Automation can ensure the timely performance of prescribed maintenance, which is a critical but often low-priority task

Let's examine each of these reasons for automation in greater detail.

Do More With Less

A good man is hard to find, and a good DBA is even tougher to track down. Whether the problem is an insufficient budget or a small local talent pool, organizations can't always hire enough staff to handle their SQL Server instances. In order to handle the growing number of instances in a shop without a matching growth in head count, IT organizations need their DBAs to learn to automate processes.

Not only do database administrators have to accomplish more in less time, but they have to accomplish it in very specific time windows. Business won't stop for blocking activity like index rebuilds; these processes must be automated in a way that enables them to happen during low-load maintenance windows to minimize end-user disruptions. The more tasks DBAs can automate and schedule for unattended after-hours execution, the less they have to be physically present during those time windows, and the more time they can spend with friends and family.

Ensure Tasks are Standardized, Consistent and Complete

SQL Server tasks can often be performed in a wide variety of ways: pointing and clicking in SQL Server Management Studio, banging out TRANSACT-SQL scripts or using third-party management tools. The more complex a task becomes, the more likely changes will be introduced during execution. Two DBAs may not perform the same task the same way twice, even when they use the same tools, and these differences introduce the possibilities of errors.

Automating tasks, whether with TRANSACT-SQL scripts, PowerShell scripts, or any other macros or scripts, means that the tasks will be performed consistently no matter who does them or when they're done. Senior DBAs can be assured that the more junior members of the team can accomplish tough tasks more reliably, and the junior DBAs can examine the scripts to learn more about what's going on under the hood.

Make Regular Maintenance Easier

Automation enables a DBA to minimize the repetitive tasks to keep the trains running on time and instead focus on providing additional value. Business managers want their database administrators to spend the least possible time doing regular maintenance work because it isn't seen as adding value to a business. Instead, they want the DBA to focus on enhancing performance, increasing capabilities and servicing end-user requests. Database administrators don't get raises for performing 500 backups per day; rather, they get fired for being unable to perform that one critical restore. In order to get those hundreds of backups done, the key is automation.

What to Automate?

Now let's examine some of the most common tasks for DBAs to automate:

- Backups, along with restore testing
- DBCC commands
- Index rebuilds
- Performance data collection

Backups and the Testing of Restores

Ideally, backups would happen as fast as possible—not as fast as humanly possible, but as fast as possible, period. Backups are the most obvious example of how automation pays off, because the faster and more frequently we can run our backups, the better our restore options become in case of a recovery situation, whether that be an unplanned emergency recovery or a planned recovery. For example, a DBA who automates transaction log backups every 15 minutes will be able to step back to a specific point in time and recover data to within 15 minutes of a failure, thereby saving the company money through tight recovery windows.

Beyond automating backups, though, automation enables the DBA to run regular fire drill restore tests. Recovery testing is one of those unglamorous tasks that does not merit time in most DBAs' busy schedules. On the other hand, when the DBA can automate restore testing just once for a production server by scripting out a scheduled weekly restore onto a sandbox server, then the restores can happen on an ongoing basis. That way, the DBA will know for certain that the backups are working, and that the database can be recovered when things go wrong. If the restored database is kept online after the test, it can come in handy for emergency restores of static configuration tables that rarely change or can be used for reporting and analysis.

Another important payoff of automated restore testing is that the DBA will be able to tell management ahead of time exactly how long a restore will take. Over time, databases grow and hardware performance changes, making restore time estimates an evolving process. If the DBA uses an automated restore sandbox that restores a different server's databases every day, then the DBA can easily check how long those jobs take, and report on estimated restore times to management. Then, before disaster strikes, managers will know how long outages might last and exactly what to expect.

Reuse It! Backup and DBCC Scripts from Ola Hallengren and Microsoft

There are several very good examples of database automation for backups and the next topic: database consistency checks (DBCC). We recommend you begin building your tool box with two sets of automation scripts:

- First, SQL Server enthusiast Ola Hallengren has a very useful set of scripts for backups and database consistency checks. Ola's homepage is located at http://ola.hallengren.com. Release notes located for his scripts are at http://ola.hallengren.com/Versions.html and documentation is at http://ola.hallengren.com/Documentation.html/. If you're simply looking for the script, you can find it at http://ola.hallengren.com/scripts/MaintenanceSolution.sql.
- Second, members of Microsoft's internal IT group, who support their own production databases, have made their automation scripts public. You can download them directly at http://download.microsoft.com/download/4/0/C/40CBAD9A-D990-450B-8785-F288CEBFB448/AITScripts.zip. However, there's very little documentation, so we recommend you read Kevin Kline's *SQL Server Magazine* article about the scripts; the article is available at http://www.sqlmag.com/Article/ArticleID/96463/sql_server_96463.html.

DBCC Commands

Is every database free of errors? Are you sure? One of the most painful elements of database recovery is discovering that the database that took so long to back up is corrupt. Corrupt databases cannot be recovered. So it behooves DBAs to first ensure that the database is free from corruption, and then perform the database backup.

The only way to be positive that your database is free from corruption and errors is to perform time-intensive, hardware-intensive database consistency checks, and then remediate any errors. Unfortunately, like index rebuilds, these operations usually require that the database be "quesced"—that is, all users have to be kicked out of the database. Because of that requirement, these operations just aren't practical for daytime use. So unless the DBA wants to work every weekend or risk an ugly surprise when the database becomes unusable or unrecoverable due to corruption, the DBCC checks must be automated.

Refer to the sidebar just prior to this section for links to sample scripts that automate DBCC tasks.

Index Rebuilds

Index maintenance is very low-hanging fruit: Microsoft studies have shown properly defragmented indexes can give a 13 – 460 percent performance benefit. Index maintenance doesn't require application changes, development time, compatibility testing, or any other risks.

The downside is that users with SQL Server Standard Edition may be reluctant to perform index maintenance because those tasks require locks on the database tables. During index rebuild operations, end-user queries can be forced to wait while maintenance takes place. As a result, index maintenance usually needs to take place after business hours; therefore, the DBA has to automate it.

Reuse It! Michelle Ufford's Index Management Scripts

Michelle Ufford (known in the Twitter world as @SQLFool) has some excellent scripts that are perfect for automating your index defragmentation, cleanup and rebuilds. Michelle's blog is located at http://sqlfool.com. You can find the latest edition (at the time of this writing) of her index defragmentation script at http://sqlfool.com/2009/06/index-defrag-script-v30. Michelle's index cleanup scripts are located at http://sqlfool.com/2009/01/index-clean-up-scripts.

And don't forget: maintaining indexes is more than defragmentation. You should also regularly check to ensure that index statistics are fresh, a topic also covered by Michelle Ufford's scripts.

Performance Data Collection

When an application starts to perform poorly, the database server is usually one of the immediate suspects. Therefore, DBAs need to be in a position where they can quickly (and confidently) confirm or refute that suspicion. That requires keeping a history of database server performance data so that the DBA can quantitatively recognize "normal" performance patterns and compare those benchmarks to situations where performance is problematic. In other words, how can you properly identify abnormal performance if you don't have a very good idea what normal performance looks like in the first place?

Another benefit of collecting performance data is accurately predicting impending performance bottlenecks, as well as acceptable slowdowns due to natural business cycles. For example, if a benchmark test shows that the current hardware platform is capable of servicing a maximum of 600 batches/second, and collected baseline data shows an upwards trend in these figures, then proactive plans can be put in place to manage this issue before it actually happens. On the other hand, if the normal workload is 50 batches/second—except during the last week of the quarter where extra report processing causes a temporary spike of 750 batches/second—the DBA can simply properly manage expectations that the system will be slower until the reporting finishes.

While many DBAs collect performance metrics manually only when poor performances forces them to do so, it goes without saying that the best-run SQL Server environments automatically collect performance data. There are a number of data sources to be collected: Windows PerfMon counters, trace data, dynamic management views (DMV), disk space usage, etc. But manually collecting all of this data for each database server in your environment usually does not make a lot of sense. Recognizing the importance of this task, Microsoft introduced in SQL Server 2008 the Management Data Warehouse (MDW), which automates the collection of performance data and simplifies reporting. MDW is discussed in more detail below.

Reuse it! Performance Queries by MVP Glenn Berry

When you are not using the MDW or a third-party tool to automate the collection of performance data, you might consider looking at the excellent performance queries compiled by Glenn Berry in his on-going blog posts on performance. His latest post, at the time of this writing, is located at http://glennberrysqlperformance.spaces.live.com/blog/cns!45041418ECCAA960!2229.entry.

Configuration and Security

Organizations that rely on PCs often develop a standard operating environment (SOE) for the purpose of reducing support costs and the time required to deploy new PC instances. Similarly, having a standard database configuration and security settings enables DBAs to avoid many of the common performance and security issues that arise through inconsistent configuration settings.

DBAs need to ensure that database servers are not only installed and configured using consistent settings, but that automation is used to ensure they *remain* correctly configured. SQL Server 2008 introduces a number of tools to assist in this regard. One is the ability to install SQL Server using a configuration file. Another—and arguably the most powerful—new DBA feature is Policy Based Management, which enables DBAs to either prevent changes that deviate from the desired state, or, at the very least, be alerted to such changes. Both of these tools are discussed below.

For example, transactional replication can be difficult to configure consistently, especially in an environment where the replication infrastructure might need to be rebuilt.

Reuse It! Powershell Script Examples by Paul Ibison and Chad Miller

For a very good example of automating and scripting transactional replication using PowerShell, look at the article, *Using PowerShell to Script out Replication on SQL Server 2005/SQL Server 2008*, by Paul Ibison. It is located at http://www.replicationanswers.com/Powershell.asp.

Chad Miller has a variety of excellent security-related PowerShell scripts. His blog is located at http://chadwickmiller.spaces.live.com/default.aspx. Chad has provided a sort of online documentation for these PowerShell scripts at http://www.sqlservercentral.com/articles/powershell/64316, while the reporting component is described at http://www.sqlservercentral.com/articles/powershell/64350.

If you're strictly interested in security, take a look at the Web site http://sqlsecurity.com. It has many useful articles and scripts.

How Do We Automate?

Using Windows Scheduler or SQL Server Agent

Most DBAs use one of two methods to automate SQL Server processes: SQL Server Agent or Windows Scheduler. The most common method is to use Transact-SQL batches controlled by SQL Server Agent. One drawback of using Windows Scheduler for automation is that it has minimal logging and alerting built in. SQL Server Agent allows for use of either Database Mail or SQLmail to alert personnel that something has gone wrong, but DBAs using Scheduler jobs are forced to build their own mail solution.

Automating processes with Transact-SQL carries the advantage that scripts can work across multiple versions of SQL Server. As long as the syntax is correct, the same scripts can be used for automation with v7, v2000, v2005, v2008 and beyond. And while permissions, security policies and company standards may prevent other tools from being used, Transact-SQL is always available to the database administrator.

In addition, the better DBAs become with Transact-SQL, the more efficient and powerful their queries become, and the better job they can do when troubleshooting queries from developers, report writers, or BI users. Knowing how to accomplish a task in Transact-SQL can get the DBA out of a jam with a difficult stored procedure or function. Even if the DBA prefers to use other scripting methods, it may be useful to understand how to accomplish the task in Transact-SQL first before moving on to other languages.

Reuse it! Allen White's PowerShell Scripts for SQL Server

Allen White, a SQL Server MVP and consultant, has written a great white paper on PowerShell for SQL Server that is available on Microsoft TechNet at http://msdn.microsoft.com/en-us/library/dd938892.aspx. Allen's blog has many more tips; it is located at http://sqlblog.com/blogs/allen_white/default.aspx. For DBAs who prefer PowerShell to Transact-SQL, Allen's blog also provides PowerShell code.

Alternative Approaches

Some DBAs enjoy using other approaches and languages for their automation. For instance, some DBAs—particularly those with a strong developer background—choose to use SQLCLR routines instead of Transact-SQL routines. Because the CLR supports many different .NET languages, developers-turned-DBAs are often able to write powerful automation scripts very quickly because of their strong familiarity with their favorite .Net language.

DBAs can also integrate the SMO, RMO and AMO namespaces into their CLR and PowerShell scripts, as well as WMI (for Windows Server capabilities), to further leverage the capabilities of SQL Server in a way that's not directly supported by Transact-SQL.

Try It! Free Scripting Tools

Don't know PowerShell or WMI? There are some great free tools that can help. Quest Software's free, community-supported PowerGUI tool is available at http://www.powergui.org. Scriptomatic, a well-known WMI scripting tool from Microsoft, is available at http://www.microsoft.com/downloads/details.aspx?DisplayLang=en&FamilyID=09dfc342-648b-4119-b7eb-783b0f7d1178.

The examples referenced earlier in the white paper provide an excellent variety of templates that you can customize and reuse in your own environment.

Automation Innovations in SQL Server 2008

SQL Server 2008 introduced several new features that make automation of administrative processes even easier, including Policy-based Management (PBM), Central Management Server (CMS) and Management Data Warehouse (MDW). These are all described in further detail below.

Policy-based Management (PBM)

A pivotal new feature of SQL Server 2008 is Policy-based Management (PBM). PBM enables a DBA to define a policy that a database must adhere to and how the server must react when it is out of compliance with the policy. PBM can be used to automate how servers respond to a variety of different conditions.

Read It! Great Resources for Policy-based Management

Policy-based Management is a large topic unto itself. Learn more about it by starting with the SQL Server Books Online entries on PBM located at http://technet.microsoft.com/en-us/library/bb510667.aspx. Another great resource is http://www.sqlcrunch.com, which has an entire section devoted to PBM content located at http://www.sqlcrunch.com/PolicyBasedManagement/tabid/91/Default.aspx.

In addition, you might want to do some further reading on the team PBM blog located at http://blogs.msdn.com/sqlpbm. You can also read Tom LaRock and Brent Ozar's webcasts on PBM; part 1 of 3 is located at http://thomaslarock.com/2009/08/policy-based-management-podcast-part-1.

Finally, Bart Duncan has an excellent blog post describing how very complex server policies can be configured using PBM; it is located at http://blogs.msdn.com/bartd/archive/2008/09/11/defining-complex-server-health-policies-in-sql-2008.aspx.

Central Management Server (CMS)

The new Central Management Server (CMS) feature is useful for fast-changing shops where a lot of servers are added and removed. A CMS is just a central repository that holds a list of managed servers. Imagine a shop with two DBAs who have their own desktops (plus maybe laptops), and where each SQL Server has its own list of registered servers. Trying to keep the list of registered servers in sync each time a new instance of SQL Server comes online could be a nightmare. With a CMS, however, the list of registered servers is stored on the central SQL Server. When the DBA opens SQL Server Management Studio, he or she points at the CMS, and SSMS grabs the list of registered servers from there. It is simple—and it comes in very handy. In fact, it is practically a prerequisite for a good PBM deployment.

CMS has a couple drawbacks. First, the CMS server list is just a list of server names; nothing more, nothing less. Authentication is not saved at all. When you connect to any server in the CMS list, your Windows authentication is used. You can't save an override list of logins, like an SA login for a specific server in the DMZ. Second, the Central Management Server doesn't work across DMZs or in multiple domain forests (unless trusts are preconfigured). You could set up multiple CMSs, one in each domain, to get around this limitation, but that's not exactly ideal.

Management Data Warehouse (MDW)

Finally, SQL Server 2008 now ships with the Management Data Warehouse (MDW). The MDW automatically collects a large number of performance metrics for SQL Server 2008 instances, stores them in a central repository and provides a variety of reports on that data. While MDW does not automate any reactions to values returned by the collected performance metrics, it does automate the collection of metrics from a variety of sources, including Windows PerfMon counters, DMVs, dynamic management functions, system tables, and the like.

The MDW might sound like the panacea for all of your performance data collection needs, but it does have a few drawbacks: it collects data only from SQL Server 2008 or later; the repository can be located only on a SQL Server 2008 Enterprise Edition or later server; the amount of data collected can be quite large (in gigabytes per day per monitored instance); it is difficult to clean up customer collections; and the MDW provides very limited insight into what the metrics actually mean. However, if you don't have any other means of collecting performance metrics across your enterprise, MDW can be a good choice. And it does provide a limited amount of analysis on the metrics it collects, particularly in the areas of disk usage, locking and blocking, and query and index performance.

Read It! The MDW White Paper by Kalen Delaney

If you spend any time troubleshooting performance problems and your organization uses the MDW, then you'll need to know how to interpret all of the details returned by the reports and how to remediate the problems you might find. Get all of the details about MDW from Microsoft's MSDN white paper, located at http://msdn.microsoft.com/en-us/library/dd939169.aspx.

Summary

No matter how big or small your SQL Server environment, you can benefit from automating important database administration tasks. You'll be able to do more work and enjoy more personal time; standardize processes for greater consistency and fewer errors; and ensure that preventative maintenance runs at appropriate times without your personal intervention.

You will want to ensure the following tasks are automated and, for good measure, we've provided a good online resource for scripts that you can automate and find more information:

- **Backups and DBCC**: If you haven't already developed your own, grab the scripts used by Microsoft's internal production database management team (http://www.sqlmag.com/Article/ArticleID/96463/sql_server_96463.html) or those developed by Ola Hallengren (http://ola.hallengren.com).
- **Index defragmentation and rebuilds**: These crucial administrative tasks are necessary to maintain performance over the long term. Be sure to review Michelle Ufford's work at http://sqlfool.com.
- **Performance data collection**: You have to know the normal performance behavior of your SQL Servers in order to quantitatively identify "abnormal" performance. Glenn Berry's scripts and queries are a good place to start; see http://glennberrysqlperformance.spaces.live.com/blog.
- **Configuration and security**: Security and server configuration should be standardized across groups of like SQL Servers. Consider using PowerShell scripts, such as those described by Allen White at http://msdn.microsoft.com/en-us/library/dd938892.aspx and Chad Miller at http://www.sqlservercentral.com/articles/powershell/64316.

Don't forget the useful and free scripting tools, PowerGUI (http://www.powergui.org) and Scriptomatic (http://www.microsoft.com/downloads/details.aspx?DisplayLang=en&FamilyID=09dfc342-648b-4119-b7eb-783b0f7d1178).

And finally, you should avail yourself of great multi-purpose Web sites like http://www.SQLServerPedia.com and http://www.SQLcrunch.com. They're continually posting new information, important findings, and relevant new tips and tricks.

About Quest Software, Inc.

Now more than ever, organizations need to work smart and improve efficiency. Quest Software creates and supports smart systems management products—helping our customers solve everyday IT challenges faster and easier. Visit **www.quest.com** for more information.

Contacting Quest Software

PHONE 800.306.9329 (United States and Canada)
If you are located outside North America, you can find your local office information on our Web site.

E-MAIL **sales@quest.com**

MAIL Quest Software, Inc.
World Headquarters
5 Polaris Way
Aliso Viejo, CA 92656
USA

WEB SITE **www.quest.com**

Contacting Quest Support

Quest Support is available to customers who have a trial version of a Quest product or who have purchased a commercial version and have a valid maintenance contract.

Quest Support provides around-the-clock coverage with SupportLink, our Web self-service. Visit SupportLink at **https://support.quest.com**.

SupportLink gives users of Quest Software products the ability to:

- Search Quest's online Knowledgebase
- Download the latest releases, documentation, and patches for Quest products
- Log support cases
- Manage existing support cases

View the Global Support Guide for a detailed explanation of support programs, online services, contact information, and policies and procedures.

QUEST SOFTWARE
Smart Systems Management

5 Polaris Way, Aliso Viejo, CA 92656 | PHONE 800.306.9329 | WEB www.quest.com | E-MAIL sales@quest.com
If you are located outside North America, you can find your local office information on our Web site.

WPD-AutomatDBAProc4MSSQLSrv-US-AG-20100114