IDERA® | SQL Safe Backup

# TOP SQL SERVER BACKUP MISTAKES (AND HOW TO AVOID THEM)

BY GREG ROBIDOUX

# INTRODUCTION

Backing up SQL Server databases is one of the most important tasks DBAs perform in their SQL Server environments every day. The backup process is so important it usually requires well defined processes, checklists and multiple DBAs to perform and manage. Still, backup errors are among the most common, time-consuming and costly problems administrators face on a regular basis.

We've compiled a list of some of the most common backup mistakes and provided insight into how to avoid them. Check this list of mistakes against the processes you have in place and avoid that terrible 'oops' moment.

**The paper will cover the following ten areas:**

- Planning
- Protection
- Recovery
- Monitoring
- Testing
- Scheduling
- Knowledge
- Performance
- Management
- Tools

## PLANNING

### Not understanding recovery needs

The whole point of creating backups is to be able to retrieve data in case data is lost or there is a failure. But to setup the proper backup and recovery plan you need to know what needs to be recovered to determine if the recovery was successful or not.

You need to make sure you understand your recovery needs for each database and application that you support—otherwise your recovery may fall short.

### Not defining roles

In many environments, the DBA is responsible for the databases, but the network group, storage group or backup group is responsible for backups. While this is fine, the question remains: who is responsible to recover the databases? More importantly, if there are multiple groups involved, who takes the lead for recovery?

So, you should make sure that along with knowing who is doing the backups, you also know who is responsible for restores and all the steps required for a successful recovery. In addition, you must know who will perform each step. As a result, you will then know that the recovery will occur as quickly as possible rather than you scrambling to find out who will do what.

### Wrong backup plan and / or recovery model

Often you will find someone who is not expecting to lose any data, but their databases are in the Simple recovery model and they are only doing full backups. Another scenario is where the database is in the full recovery model and only full backups are being issued.

You should really take the time to understand the differences between recovery models and the different backup options as these two items go hand in hand.

### I only need to recover the databases

You need to know that there is more to recovering SQL Server then just the databases.

When setting up your database backup plan, you should be sure to remember to put into place backups for Reporting Services, Analysis Services, Integration Services, as well as items that may be outside of SQL Server, including third party tools, sqlcmd files, batch scripts, etc. Often people are only thinking about the databases so make sure you encompass the entire SQL Server environment.

## PROTECTION

### Not protecting physical backups

Since a full database backup contains all data in your database you should think about the value of the data or the consequences of it getting into the wrong hands. If you're storing credit card data or Social Security numbers or passwords, you will want to make sure this data is always protected when it is live and also in your backups. Once someone has a backup it is a very easy for them to do a restore to access the data.

Another reason why you want to protect the files is to make sure they exist when you need to restore. Files could easily be deleted either on purpose or by accident and then no longer exist for the restore.

You should also limit access to where backups are stored and either limit or not allow backups from being copied to other locations.

### Not encrypting backups

Another topic related to the physical protection is encrypting key data in the database in the event someone does gain access to your backups.

By default, data is not encrypted in the database.

However you can encrypt key data within the databases by using encryption  keys or transparent data encryption. You also have the ability to create encrypted backups with some third-party SQL Server backup tools. This will add another level of protection for your backups, just in case the backup falls into the wrong hands.

### Storing backups on same physical drives as database files

Typically, the default for creating backups is the same drive and path as the database files. So unless you make a conscious decision, your backups are probably stored right alongside your actual database files. If for some reason you lost the server or if the files on your drives were corrupt, you could potentially lose both your live data and your backups.

You can eliminate this risk by making sure you move your backups to other drives or to other servers or by simply creating the backups on different physical disks.

### Writing multiple backups to the same physical file

Another issue is when multiple backups are written to the same physical file. This approach could result in a single point of failure if the file becomes corrupt. In addition, it is hard to tell what is in the backup file by just looking at the file name and or size. So if there are multiple backups in one file the only way to determine what is in the file is to use the RESTORE commands.

A better way is to create one backup per file and use a consis tent naming convention. This way you can just look at the file name and tell what kind of backup it is, for what database and when it was created. Also, if a file is corrupt, it will only impact the one backup in that file.

## Allowing too many accounts to have access to create backups

As we discussed earlier, having your backups fall into the wrong hands could be a major issue. Another area to consider is whether several people have the ability to create their own database backups. If someone does not have access to the location where the backups are stored and if they can create their own backups, then they can easily take a copy of the entire database.

To avoid this from occuring, you should limit who has permission to create backups as well as monitor when backups are created, who is creating them and where they are being created. This can be done by using the system tables and also the Windows event log and the SQL Server error log.

## Not backing up system databases

Another thing that is often overlooked is backing up the system databases. It is unclear whether this is simply due to a lack of understanding of what is stored in these databases, or a case of not knowing how to use these for a restore.

The two main system databases that you should backup are master and msdb. The master database stores security information as well as metadata about the other databases. Msdb stores information about jobs, operators, and alerts etc. This is not very critical unless you are making changes to the model database. Since tempdb is recreated each time you restart SQL Server there is no need to backup tempdb. The other database that you could backup is the resource database, but this needs to be done outside of SQL Server. In addition, you could script out important information for these databases as another way to recover important objects, but you will not have the historical data.

# RECOVERY

## Not having a good recovery process

One big part of your backup process needs to include the recovery procedures.

You should have a rehearsed recovery plan, so when you need to recover any of your databases you know the exact steps to follow. Also, keep in mind that there are many other steps to this than just the database backups, so be sure to put together a plan ahead of time.

## Restoring from tape

You should keep the latest full set of backups on disk to ensure the fastest recovery of your databases. Disk restores in most cases are always going to be faster than going to tape.

As a general rule, you should use disks for recovery and use tapes for long term storage.

## Not having the most recent backups on disk

Another issue occurs when the most recent backups are not available on disk and therefore you need to search for the backups on tape.

As such, you should keep a full set of backup s, which includes the full, differential and transaction log backups on disk. Also it would be a good idea to keep two sets on disk if possible. Anything beyond that probably doesn't make sense to keep on disk because you generally would want to restore the most recent data that you can.

## Not having a priority order for recovery

We mentioned having a rehearsed plan for recovering your databases. In addition, you should set a priority order for the restore process.

A priority order allows you to tackle the most critical databases first. Also if there are databases that are dependent on other databases they can be done in an order that allows the applications to function properly. It is impossible to do everything at once, so it is a good idea to think through the order that you will approach things in the event of multiple failures.

## Not knowing how to recover for applications that use multiple databases

This may not be a common scenario for everyone, but there are many of you that have applications that rely on more than one database to function. This is why you really need to think through the recovery process for this type of recovery.

Do all databases need to be recovered to the same point in time? If you have databases that are replicated, how does the recover y process impact replication? Also, if you are using third party applications that use SQL Server, then what special steps do you need to take to restore the databases to get the applications working again? You should make sure you think about the complete recovery and not just one database.

## Not backing up the tail of the transaction log

One last thing to consider is backing up the tail of the transaction log. This allows you to get any additional transactions that have not been backed up yet, so you can restore as much data as possible with minimal data loss.

This is something you should make sure you have the necessary knowhow to do in case there is a failure. For the most part it is just like taking a transaction log backup. This may not be something you can always do, but you should be aware of the steps.

## MONITORING

### Not checking for backup failures

I often see backup failures for several reasons in my consulting work when reviewing SQL Server error logs. When this is brought to the client's attention; they are sometimes unaware that there were failures.

There are several ways that you can be alerted to backup issues – both automated and manual. The best approach is to automate the process, but either way this should be something you check for every day. You don't want to be caught trying to restore a backup that doesn't exist.

### Logging all successful backups

On the flip side of logging and monitoring failures is the issue of bloated error logs with all successful backups. This becomes a problem very quickly if you have several databases and are doing transaction log backups.

You can use Trace Flag 3226 to turn off successful logging to keep your error logs small.

### Not monitoring file size and backup time

Also with monitoring you should monitor how long backups take and also file sizes. This ensures you do not overlap with other processes and that you do not run out of space to create the backups.

You should keep an inventory of what is going on as this can be automated by using the backup system tables in msdb, which can help you avoid these types of issues.

## TESTING

### Not using restore verify option

The last thing you want is to discover that your backup is not good when trying to restore it. Although doing a restore verify does not guarantee that the restore will be successful it does at least give you some peace of mind that SQL Server was able to read the contents of the backup without issue.

Adding this step is easy when you use maintenance plans. But when you create your own scripts you should also include this step in the process as well. Lastly, you should monitor the output from this command to ensure the restore verify did not have any issues.

If it did have issues, you can always create another backup.

### Not testing backups

Another item to consider is testing the entire restore process.

You should randomly and periodically do complete restores of your databases to ensure there are no issues. This also gives you the ability to rehearse your entire recovery process, so you know what works and what doen't in the event of a failure.

## SCHEDULING

### Wrong backup schedule

The wrong backup schedule is another common problem. In a lot of cases where a full backup is done once a week, differentials are then done every other day of the week. This may be your only option if you do not have space, but space really shouldn't be an issue anymore. Another common scenario is to have the database in full recovery, doing full backups every day and then truncating the transaction log. If you are going to throw the data away you might as well just use the Simple recovery model.

Ideally, you should have a mix of full, differential and transaction log backups and try to avoid doing high I/O activities, such as full backups during peak usage times. Another issue is that full backups are run multiple times a day. While there may be a need for this, you should keep in mind that you could also use differential backups with the full backups to minimize the impact.

## KNOWLEDGE

### Not understanding options

Another big problem is people not being aware of the different options that are available using T-SQL commands and using SQL Server Management Studio.

You should take the time to read SQL Server Books Online, so you understand all of the options for backup and recovery. Also there are several free resources on the Internet that you can benefit from reading. You should not assume that you know everything without first reading up on what is available.

### Not understanding the restore process

We talked about recovery some already, but another area where you should take the time to learn is the restore process. While backups are important, recovery is even more important.

Make sure that you know as much as possible, so that when the time comes to restore a database, or databases, you are not learning at the same time.

### Relying on SSMS

Another common problem is relying on Management Studio. This is a great tool to do most of what you need to do, but there are some options that you cannot use with SSMS.

You should take the time to understand the T-SQL commands. You are going to have to know the T-SQL commands if you need to do restores or backups from a command line.

### Just downloading and implementing scripts without understanding

You should probably avoid just downloading scripts and implementing them without knowing exactly what they are doing.

Take the time to read through the script and also any documentation about the script. You want to make sure you are getting what you expect.

Without taking the time to understand the script, how would you know if it is working correctly?

### Not taking time to learn new features

Another common mistake is not taking the time to learn new features. You may think that backups are backups, which is true, but there are always new features for backup with each release of SQL Server. It is best to stay current as new versions of SQL Server are released and also read up on any changes when you apply service packs.

## PERFORMANCE

### Writing directly to tape

A lot of third–party backup tools have agents that allow you to write directly to tape. This was great in the old days when disk space was expensive and limited, but this should not be the case anymore.

You can write to disk first and then archive to tape as disk backups will generally always be faster.

### Not maximizing disk I/O

Another common problem that occurs is not maximizing disk I/O. Backups and restores are very I/O intensive since they have to read the entire database and then write the entire database.

By creating your backups on different physical disks you can reap the benefit of reading from one set of disks and writing to the other set of disks.

To further reduce disk I/O you can compress your backups, so you are writing out much less data. You can also write out your backups to multiple files over multiple disks to further increase I/O throughput.

## Backing up junk databases

OK, you have enough to do already, so why do more? Do you have a bunch of extra non-production databases on the server that you are backing up every night?

If they are test or temporary databases, you shouldn't waste cycles backing them up. If you have a process that backs up every database you could be wasting time by backing up these databases you don't need. You should take a look at your backup routines and not bother backing up databases that you do not need.

## Backing up Read Only databases

Along the same lines is the issue of backing up read–only databases over and over again. If these databases don't change every day, you shouldn't bother backing them up every day.

## Not creating compressed backups

When a SQL Server backup is created, it will by default basically write out a page for every page that is used in the database. There is a lot of wasted space when a backup is created since some of these pages may have little data.

There are several tools that allow you to create compressed backups. At the high end you can compress your backups by up to 95%, allowing you to run your backups much faster and save a lot of disk space. This also helps if you need to move the backups to other servers since the file is so much smaller.

## Backing up across the network

While you have the ability to create a backup anywhere on your network using UNC paths, but the downside is that you cannot control the network bandwidth and therefore backup times can vary from day to day.

It is a better idea to backup to local drives and then copy the files. This is another reason why backup compression is so useful. In some cases you may have a separate VLAN for your backup traffic where you can control the network bandwidth and lessen the impact.

## Running all backups at the same time

It is easy to setup backup schedules using SQL Agent, but you should try to do all backups at once. In most cases on one instance you are doing these serially, but if you are writing to a centralized backup location from multiple servers it is easy to use the same scheduled time. You should be aware of this and adjust your schedule accordingly.

This is where it is helpful to monitor your backups to see if they take any longer from day to day and then adjust your schedule as needed.

## MANAGEMENT

### Inconsistent backup plans

While it is hard enough to manage SQL Server, you will find that having inconsistent backup plans from server to server will make your life even harder.

Instead, you should try to classify your databases into a few groups and then apply the same backup plan across the board to these different groups of databases.

### Not including new databases in backup plan

It is not uncommon to hard code which databases to back up when you set up backup plans. You should make sure you make this flexible enough to add new databases, but also monitor your backups to make sure you are not missing any new databases.

### Non-centralized backup processing

If you have a lot of servers to manage you should really look at setting up a centralized backup process, so you can manage everything from one place. This can be done by creating your own processing or using third party tools that allow you do this right out of the box.

### Taking the DBA out of backup processing

While creating backups may seem like a routine thing, there are still lots of mistakes made every day.

For example, when recovering databases in SQL Server there is a lot more to it than just doing a file restore. The backup plan you have for your file servers probably won't work for your database servers, so make sure you don't take the DBA out of the backup and restore process.

### Inconsistent naming conventions

You should make sure you use a standard naming convention when creating your backups. SQL Server by default uses bak and trn, but you could also use dif for differentials and flg for filegroups, etc.

You should also include the server name, instance name, the database name and the date the backup was created. This allows you to just look at the file name and know exactly what is in the file.

### Using multiple applications or schedules to do backups

Another problem is when there are multiple backup applications backing up the same database. So maybe some backups are done natively and other backups are done using third party tools. If you are mixing the tool along with the scheduling, this make is difficult to do restores. This could become a real problem, so be sure to keep things consistent and do not backup the same database using different tools and or schedules.

## Writing backups to inconsistent locations

Another issue is writing backups to different locations. For example, full backups are written to the E drive, log backups are written to the F drive, etc. This could get confusing if the files are not kept together. Also, for someone who is notfamiliar with the server this makes it even more confusing. So be sure to try to keep a complete backup set of files together.

## Not removing old files and data

In most backup processing, you can delete older backups to manage disk space. Also, make sure you clean up older log files and purge some of the old history data from the backup and restore system tables.

## TOOLS

### Not knowing what tools are available

There are several tools that come with SQL Server that can be used for backup processing, such as SSMS, T-SQL, System Tables, Windows Event Viewer, SQL Server Error Logs, Alerts, and Notifications, etc.

You should take the time to understand what these tools can do for you, so you have all the information and tools at your fingertips.

### Being afraid of third party tools

For quite some time, Microsoft has made hooks into SQL Server for backups and most vendors use VDI the Virtual Device Interface to backup databases, so that things are done consistently. There are a lot of great management features these tools offer as well as some features that don’t even exist in SQL Server. You shouldn’t be afraid to research and implement these tools.

## CONCLUSION

Realizing a backup has failed after it's too late is one of the worst feelings any DBA could experience.

Backup mistakes are often very expensive, always time consuming and can even cost DBAs their jobs. I hope this list of common backup mistakes improves your backup processes and helps you avoid the disasters associated with backup mistakes.

## ABOUT THE AUTHOR

***GREG ROBIDOUX***

Greg Robidoux is the President and founder of Edgewood Solutions LLC, a Microsoft Gold Partner specializing in services and solutions for Microsoft SQL Server.

IDERA understands that IT doesn't run on the network – it runs on the data and databases that power your business. That's why we design our products with the database as the nucleus of your IT universe.

Our database lifecycle management solutions allow database and IT professionals to design, monitor and manage data systems with complete confidence, whether in the cloud or on-premises.

We offer a diverse portfolio of free tools and educational resources to help you do more with less while giving you the knowledge to deliver even more than you did yesterday.

**Whatever your need, IDERA has a solution.**

IDERA

https://www.idera.com/ContactSales