

# Windows® IT Pro

## Best Practices for Backup and Restore in SQL™ Server 2005

By Javier Loria

Edited with permission from *SQL Server Magazine*.  
Copyright © 2008 Penton Media, Inc.  
All rights reserved.



Third-party information brought to you courtesy of Dell.

THIS WHITE PAPER IS FOR INFORMATIONAL PURPOSES ONLY, AND MAY CONTAIN TYPOGRAPHICAL ERRORS AND TECHNICAL INACCURACIES. THE CONTENT IS PROVIDED AS IS, WITHOUT EXPRESS OR IMPLIED WARRANTIES OF ANY KIND.

# Best Practices for Backup and Restore in SQL Server 2005

## → Contents

Introduction.....	1
Audience: Who should read this.....	1
Introduction to the Data Recovery Plan.....	2
Threat Analysis (What can happen, and what you can afford to lose).....	2
Source.....	2
Business Requirements.....	2
Operational Requirements.....	2
Backup Objectives.....	3
Backup Strategies (What you are willing to pay).....	3
Backup Types.....	3
Full Backup.....	3
Differential Backup.....	4
Log Backup.....	4
Backup Targets.....	4
Database Backup.....	4
File Backup.....	5
Filegroup Backup.....	5
Partial Backup.....	5
Backup Strategies.....	5
Full Backup—Simple Mode.....	5
Log & Full Backup.....	6
Full & Log Backup.....	6
Full – Differential – Log.....	6
Full – File/Filegroup – Log Backup.....	7
Service Level Agreement.....	7
Operational Best Practices.....	8
Performance.....	8
Security.....	8
Other Best Practices.....	10
Summary of Backup Best Practices.....	11



## Introduction

In an ideal world, hard drives and other hardware never fail, software is never defective, users do not make mistakes, and hackers are never successful. However, we live in a less than perfect world and we should plan and prepare to handle adverse events.

In our database environment, that plan is called the Data Recovery Plan (DRP). The DRP is the set of actions that database administrators (DBAs) take to handle adverse events that may affect the availability of their database environments. Microsoft SQL Server 2005 has a variety of different technologies that can help DBAs build a DRP. These technologies include: Failover Cluster, Database Mirroring, Database Replication and Log Shipping. In spite of the advantages these technologies might offer, DBAs should always plan backups and restores as the last line of defense of the DRP.

In this whitepaper, we will discuss some of the best practices in Backup and Restore operations when using SQL Server 2005. In this whitepaper, we will cover:

- What types of Backup and Restore options SQL Server offers
- How to combine Backup and Restore types in a DRP
- Best practices you should follow in your backup and restore strategy

### Audience: Who should read this

This whitepaper is intended for DBAs involved in the design, plan and execution of the backup and restore operations in the Microsoft SQL Server 2005 platform. This paper assumes some knowledge of SQL Server commands and a basic foundation in administering the Microsoft® Windows® platform.

## Introduction to the Data Recovery Plan

The first best practice for Backups and Restores in SQL Server 2005 is to **have a DRP**. DRPs are like insurance plans for your database. When you buy an insurance policy, you have to consider two main factors: how much can I afford to lose in case of an adverse event; and how much am I willing to pay to protect from it, even when nothing wrong happens. How much you can afford to lose is the Threat Analysis. You should **begin designing your plan with a threat analysis**. How much you are willing to pay is the backup and restore strategy. Obviously these two components are tightly related and influence each other.

## Threat Analysis (What can happen, and what you can afford to lose)

Following the insurance paradigm, the first factor to consider in your DRP is what coverage your insurance policy must have. You should plan and design a strategy according to the events you want to take preventive actions about.

### Source

The threats that menace your data can be classified according to the source: Hardware, Software, People and Environment.

- **Hardware:** Hardware is always the first source of threats to consider in a database environment. Some events you should consider in your plan:
  - Hard drive failure.
  - Other I/O component failures.
  - CPU, memory and video malfunction.
  - Total server collapse.
- **Software:** Some of the software vulnerabilities you should consider:
  - Risks related to software upgrades or patches. A faulty software component may inadvertently delete or damage information in the database.
  - Software bugs may inadvertently damage data.
  - Malware (Virus, Trojans) may also delete or damage data.
- **People:** Some of the people-related vulnerabilities you should consider:
  - Users may accidentally delete or incorrectly update information in your database, even when they do not have administrative rights.
  - Database administrators may intentionally or unintentionally delete or wrongly update data in the database. For example a delete without a WHERE statement.
  - Database administrators may intentionally or unintentionally drop a database or database objects.

- A non-user (hacker) may gain access to your database server and delete or destroy data or database objects.
- **Environment:** The last source of events, which too often database administrators forget, is environment vulnerabilities. Your database DRP should also consider:
  - Fire and Flood in the data center.
  - Earthquake, Tornado.
  - Theft and Civil disorder.

**Make sure your plan handles threats from all sources: Hardware, Software, People and Environment.**

### Business Requirements

After considering all the possible events that may affect your database, it is time to analyze the Business Requirements. Some of the elements you should consider:

- 1) **Value:** The first question you should answer when designing your plan, is how much is your data worth? How much money the company will lose if you lose one day of transactions? How much money if you lose one hour of information? Answers to these questions are important because they also tell you how much you should invest in protecting the database.
- 2) **Volatility:** The second element to consider is how volatile is the database? Some databases are read only and change only sporadically, while others handle millions of transactions per minute. Volatile databases often require frequent backups.
- 3) **Size:** the third element to take into account is the size of the database. Obviously, large databases take longer to back up and restore than small databases. Very Large Databases (VLDB) may take weeks to restore in case of a total server collapse.
- 4) **Usage:** the final factor to consider is database usage in order to minimize the impact of the backup procedures. For example: if the database is used only 8x5 (eight hours, five days a week) you can adopt a backup plan to be executed when users are not working. Such a plan will take more time to execute but will take less time to restore in the eventuality of a failure. If, however, the database is heavily used 24x7 (twenty-four hours, seven days a week) you may need to adopt a backup plan that have the lowest impact to the system. Such a plan will take less time to execute but will take more time to restore in the eventuality of a failure.

**You should consider the value, volatility, size and usage of your database when designing your data recovery plan.**

### Operational Requirements

As any other system, the DRP should also take into account operational requirements. **The most relevant operational requirements of data recovery plans are**

**security, performance and manageability. Make sure your plan considers these requirements.**

The most relevant requirement to consider is security. Security is the most sensitive operational requirement to contemplate, because backups are more exposed to physical attacks than the databases they hold. Consider the following examples<sup>1</sup>:

- 1) Someone takes the backup and makes a copy. This is a confidentiality threat.
- 2) Someone overwrites the content of the backup. This is an integrity threat.
- 3) Someone steals the tapes. This is an availability threat.

We will discuss later in this white paper some of the measures you may take to reduce these threats.

### Backup Objectives

There are three main objectives of performing backups:

- 1) To recover lost data: This is the most obvious objective of a backup/restore plan. In case of an event, you will be able to recover the data that is lost.
- 2) To minimize the data loss: Sometimes it is inevitable to lose some data. Your plan should also try to minimize the amount of data that is lost. For example, if you back up databases daily, your system may lose up to 24 hours of data. You will reduce this number if you perform other backups during the day.
- 3) To minimize the downtime cost: Another factor to consider is how difficult and lengthy it will be to restore the system, and how to reduce the downtime cost. If you need to restore one Full and forty-five transaction log backups, the downtime cost is probably higher than restoring one Full, one differential, and three transaction log backups.

### Backup Strategies (What you are willing to pay)

After performing the threat analysis, the DBA should create a plan for handling backup and restores. To create such strategy you need to understand the different backup types and backup options available.

## Backup Types

SQL Server 2005 offers three basic types of backups: Database (Full), Differential, and Log backups.

### Full Backup

A full backup makes a copy of all used database pages in the database, meaning it copies all the information (Data and indexes) of the database, and excludes all empty data pages. Full backups also duplicate the part of the transaction log that includes all open transactions and transactions that occurred during the backup process. This allows the restore process to achieve transactional integrity. The following code is an example of Full Database Backup command:

```
BACKUP DATABASE AdventureWorks  
TO DISK = N'C:\Backup\AdventureWorks.bak'
```

You may also create backups using SQL Server Management Studio (SSMS). To create a database backup using SSMS (Figure 1), follow these steps:

- 1) Open SSMS
- 2) Connect to the SQL Server instance that holds the database.
- 3) Navigate to the Server\Databases folder.
- 4) Right click the database you want to backup.
- 5) From the shortcut menu, select Tasks, Backup.
- 6) In the Database Backup dialog box, select the type of backup you want the server to perform, the backup destination path and the backup options.
- 7) Select OK to backup the database or press the script button if you want to generate a script to generate a Backup with the selected options.

<sup>1</sup> This example was authored by the Application Consulting & Engineering (ACE) Team (<http://blogs.msdn.com/threatmodeling/>). The ACE Team created the Threat Analysis and Modeling Tool, that can be downloaded from <http://msdn2.microsoft.com/en-us/security/aa570413.aspx>

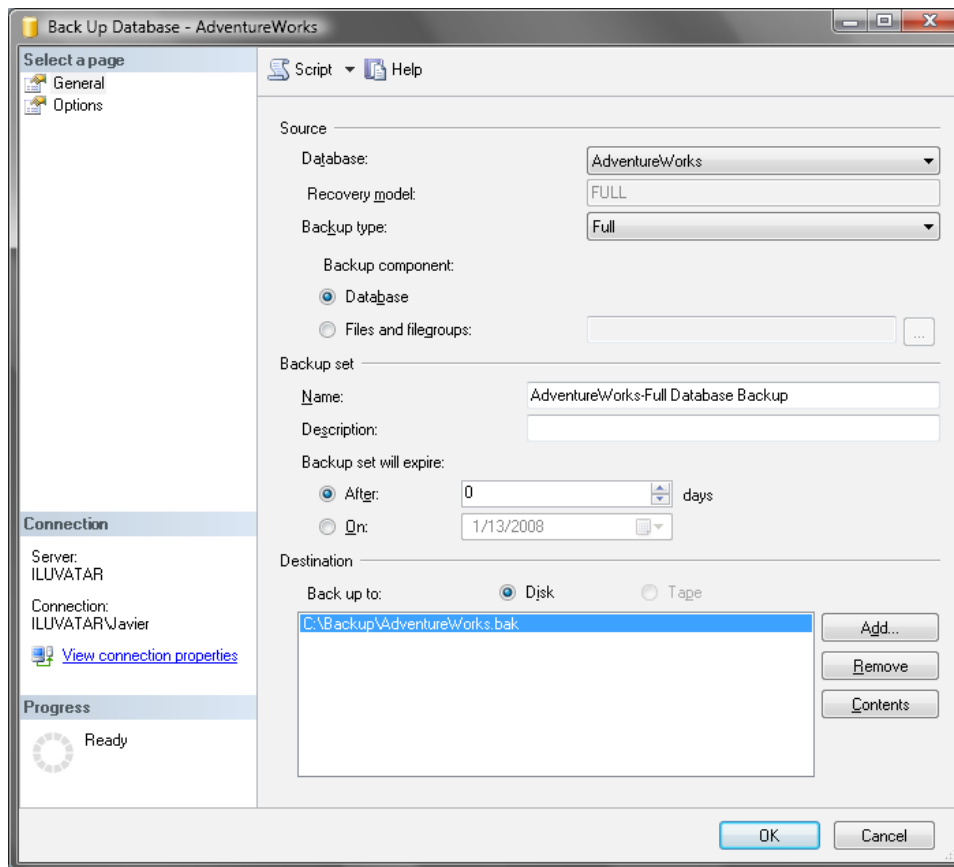


Figure 1 Backup Database Window in SSMS 2005

## Differential Backup

Differential backups are designed to reduce the amount of time it takes to perform full backups. Instead of copying all used pages, the differential backup copies only pages that have changed since the last full backup. Similar to the Full Backup, differential backups also copy a part of the transaction log to maintain the transactional integrity when the backup is restored. The following code is an example of how to create a Differential Backup.

```
BACKUP DATABASE AdventureWorks
TO DISK = N'C:\Backup\AdventureWorks.bak'
WITH DIFFERENTIAL
```

## Log Backup

The only type of backup that does not actually copy database pages is the Log backup. The Log backup copies the transaction log of the database. After it copies the log, the backup truncates the part of the log that is not required by active transactions or transactional replication. The syntax to back up the Log is:

```
BACKUP LOG AdventureWorks
TO DISK = N'C:\Backup\AdventureWorks.bak'
```

## Backup Targets

Besides the backup type, SQL Server allows four different backup targets: Database, Filegroup, File and Partial. These targets only apply to the Full and Differential Backup types; they do not apply to the log backup.

### Database Backup

A database backup makes a copy of the entire database. If you are using a Full database backup it includes all used database pages in the database; if you are using a Differential Database backup, it includes all changed database pages of the database that have changed since the last full backup. Some examples of how to perform a Full and Differential Database backup are:

```
-- Full Database Backup
BACKUP DATABASE AdventureWorks
TO DISK = N'C:\Backup\AdventureWorks.bak'
-- Differential Database Backup
BACKUP DATABASE AdventureWorks
TO DISK = N'C:\Backup\AdventureWorks.bak'
WITH DIFFERENTIAL
```

## File Backup

File backups do not copy the database; instead the file backup aim is to back up only one or more files that are part of the database.

Databases in SQL server 2005 have three different types of files: Primary, Secondary, or Log Files. Primary and Secondary files hold database pages; in contrast, Log files hold the transaction log. File backups can make copies of pages from Primary and Secondary Files.

This type of backup can be used to speed up the restoration process, when the device that stores one of the files fails but the rest of the devices are working correctly. In this case, there is no need to restore the full database; only the affected file. The following SQL code performs Full and Differential File backups:

```
-- Full File Backup
BACKUP DATABASE AdventureWorks
FILE='AdventureWorks_Data'
TO DISK = N'C:\Backup\AdventureWorks.bak'

-- Differential File Backup
BACKUP DATABASE AdventureWorks
FILE='AdventureWorks_Data'
TO DISK = N'C:\Backup\AdventureWorks.bak'
WITH DIFFERENTIAL
```

One important element to remember when using File Backups is that they should either be read only (the filegroup they belong to) or combined with Log Backups to restore transactional integrity of the database. File backups do not automatically back up the transaction log.

## Filegroup Backup

You can group Primary and Secondary files in two different types of Filegroups: Primary and Secondary filegroups. The main difference between primary and secondary filegroups is that the Primary filegroup holds the Primary file, where system tables are stored.

Similar to File backups, Filegroup backups do not copy the entire database; they copy only pages from files that belong to the filegroup. Some examples of how to perform Filegroup backups are:

```
-- Full Filegroup Backup
BACKUP DATABASE AdventureWorks
FILEGROUP='PRIMARY'
TO DISK = N'C:\Backup\AdventureWorks.bak'

-- Differential Filegroup Backup
BACKUP DATABASE AdventureWorks
FILEGROUP='PRIMARY'
TO DISK = N'C:\Backup\AdventureWorks.bak'
WITH DIFFERENTIAL
```

## Partial Backup

Partial backup is similar to File and Filegroup backups; they do not back up the entire database. Partial backups always back up the Primary Filegroup and every Read and Write Filegroup that is part of the database. Besides the fact that partial backups simplify the backup statement to back up only read and write filegroups, they also offer the advantage of allowing this type of backup when using a simple recovery model in the database. To perform a partial backup, use the Read\_Write\_Filegroups option of the Backup command as the following example does:

```
-- Full Partial Backup
BACKUP DATABASE AdventureWorks
Read_Write_Filegroups
TO DISK = N'C:\Backup\AdventureWorks.bak'

-- Differential Partial Backup
BACKUP DATABASE AdventureWorks
Read_Write_Filegroups
TO DISK = N'C:\Backup\AdventureWorks.bak'
WITH DIFFERENTIAL
```

## Backup Strategies

**Combine different backup types to create a backup strategy.**

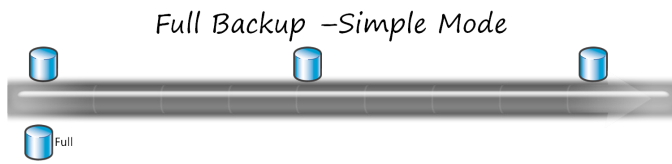
### Full Backup—Simple Mode

When you follow this strategy, the DBA configures the database recovery model to simple, and performs periodic Full Database backups. This is the simplest backup strategy, and the simplest to restore. But it is also the one that takes the longest to back up and is not feasible in most scenarios. You can use this strategy in the following cases:

- Development Databases: Databases used in developing or testing environments, if you can afford to lose daily work, otherwise consider a Full and Log Backup strategy.
- Stage Databases: Databases used in Data Warehouse to store intermediate data extracted from source systems.
- Read Only Databases: Databases that have only read only information, for example subscriber databases that store information for reporting purposes.
- System Databases: master, msdb and distribution. **Do not back up tempdb. And unless you plan to change its contents, do not back up the model database.** Backing up Master also backs up the ResourceDB.

Because the transaction log is never backed up and it is never truncated; the recovery mode should be changed to simple mode to avoid the continuing growth of the database. **Use the Full Backup simple mode strategy only on development, read only, stage or system databases.** For example, you can schedule daily backups every night or every weekend of all system databases.

A diagram of Full-Backup and Simple Mode strategy:<sup>2</sup>



### Log & Full Backup

This strategy is similar to the previous one; however, instead of changing the recovery model of the database, you back up the log just before the full backup. This approach has the advantage of capturing not only the data, but also the changes that occurred since the last backup. The benefits of a Full & Log backup strategy are:

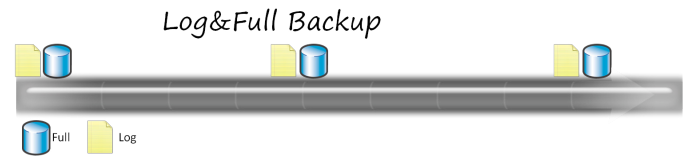
- 1) If Database files and Log are physically separated and the database hard drive fails, you may back up the Log and not lose any data.
- 2) In case of a software failure, human error (end user or DBA) or security breach; the DBA can first backup the transaction log first and then restore the database to a point in time, reducing the amount of loosed data. Leaving the full recovery model on, allows you to backup the transaction log first and restore the database and the part of the transactions up to the last known good reliable state.
- 3) If the database has multiple files and one fails, you can first backup the tail of the transaction log (that include the transactions that occurred since the last log backup), and using the last full backup restore only the failed file and the transactional log to restore transactional integrity.
- 4) The transactional log backups can be used in forensic analysis.

There is some additional space involved in having the transaction log backup on a daily basis; however, in most databases this is not a relevant factor to consider.

**Schedule the Log backup before the full backup.** This approach will speed up the recovery process. If you schedule the Log backup before the full backup, you will only need to restore the last full backup and one transaction log to recover the database. Scheduling after the full backup will demand the restoration of the last full backup and two transaction logs backups.

**Use the Log & Full backup strategy in 8x5 databases where you can afford to lose up to one day of work.**

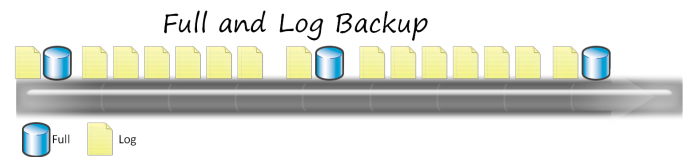
A diagram of a Log & Full backup strategy:



### Full & Log Backup

In this scenario, the DBA schedules one full backup and multiple Log backups, between full backups. This plan has the advantage of reducing the amount of lost information in case of total server collapse, because you can schedule log backups more regularly than you can schedule full backups.

A diagram of a Full & Log backup strategy:



**Consider the use of the Full & Log backup strategy in 8x5 databases, and non-volatile 24x7 databases.** For example, you may schedule nightly Full Database backups and Log backups every four hours.

### Full – Differential – Log

Sometimes the information stored in a database is so valuable that you want to back up the information very often. However, relying on backup logs is not feasible because, in case of failure, you will have to restore every log backup since the last full backup. In this case, the Full – Differential and Log strategy may help.

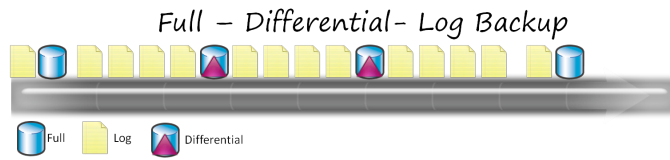
When using Full, Differential, and Log Backups the DBA schedules a Full Backup occasionally, differential backups often, and Logs very often. In this case, occasionally means as frequently as needed, to minimize the impact of a full backup, but enough to avoid differential backups to grow too much; often means as frequently as needed, to minimize the impact of restoring too many log backups; and very often means as frequently as needed to minimize data loss exposure. Some implementations of this strategy may be:

- 1) Full backup every month, differential backups every night, and log backups every two hours.

<sup>2</sup> The scale on the Backup strategies diagrams is deliberately abstracted. Some databases will require weekly full database backups; more often business should choose daily backups; and finally others will require multiple daily full backups. Take into account the business requirements: value, volatility, size and usage, when defining your backup plan.

2) Full backup every week, differential backups every 6 hours, and logs every fifteen minutes.

A diagram of a Full – Differential – Log backup strategy:

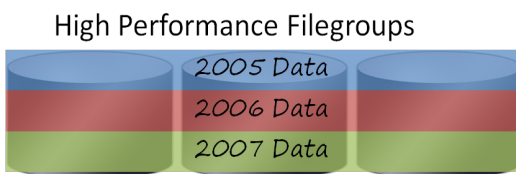


**Use the Full, Differential and Log strategy in volatile 24x7 databases, when the value of your data requires very frequent log backups, and in non-partitioned Data marts.**

### Full – File/Filegroup – Log Backup

In some databases the impact of a full backup is so large that you want to avoid it as much as possible; at the same time your database is most likely large enough to consider the use of multiple file and filegroups. In that scenario, most likely you don't want to back up the complete database at once, you only want to back up parts of it.

To create a backup strategy when using Filegroups, you have to be aware of the two main patterns that DBAs often use to create filegroups. The first pattern is used to maximize performance and the other to maximize availability. In the first approach, to maximize performance, the DBA assigns every Filegroup to one or more files in each hard drive or I/O subsystem. Using this schema the performance is increased because the server can use all the devices in parallel. The following diagram may help how understand how filegroups are assigned when their purpose is to maximize performance.



On the contrary, to maximize availability, Filegroups are assigned to one or more files in a single hard drive or I/O subsystem. The benefit of using this approach is that if one of the hard drives fails, you will only have to restore that Filegroup. The following diagram may help you understand how filegroups are assigned when their purpose is to maximize availability.

### High Availability Filegroups

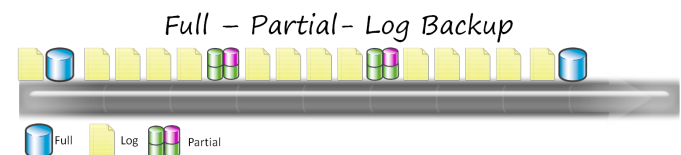


Backup filegroups partitioned by I/O System (High availability), is simple because you only have to back up one filegroup every time. However, when the filegroups are partitioned across I/O systems (to increase performance) you have to back up every file of each filegroup that resides in the same hard drive I/O System. Finally you may consider flagging a Filegroup as read only to eliminate or reduce the frequency of the backups of that file-group. The following code marks a Filegroup as read only:

```
ALTER DATABASE AdventureWorks
    MODIFY FILEGROUP AdventureWorks2001 READONLY
```

**Consider the use of a Full - File/Filegroup – Log Backup strategy in Very Large Databases (VLDB) or partitioned Data marts/Data warehouses.** For example, you may back up a Full Database once a year, a Filegroup (assuming that the filegroups are designed with High availability in mind) backup every day, and Log backups every hour.

A diagram of a Full – Partial – Log backup strategy:



### Service Level Agreement

Based on the analysis of the business and operational requirements, the possible threats, and the options that SQL Server 2005 provides to protect the data; the DBA should write a Service Level Agreement (SLA). The SLA is an agreement between customers and users on one side and service providers on the other side. The SLA is written like a contract and describes a set of operational goals that the IT Department promises to achieve. The following example is the part of an SLA<sup>3</sup> regarding disaster recovery and backup and restores plans:

- In case of the following incidents that disrupt the database service, the IT department will resolve the problem under the following conditions:
  - In case of I/O system failure or human error, the database service will be available within 4 hours of the first report of the incident, and no more than 15 minutes of data will be lost.

<sup>3</sup> A short SLA template can be found in [http://www.nextslm.org/internalsla\\_short.html](http://www.nextslm.org/internalsla_short.html).

- In case of a server breakdown due to software or hardware issues, the database service will be available within 12 hours of the first report of the incident and no more than 15 minutes of data will be lost.
- In case of data center breakdown due to fire, flood, earthquake or other environment or civil disasters, the database service will be available within 24 four hours and no more than 4 hours of data will be lost.

To set unambiguous expectations of users, **write an SLA that describes the response time the IT Department will provide in case of each type of event.**

## Operational Best Practices

Beside the backup and restore strategy, the DBA should also consider different options and practices to handle backup and restore activities. The following guidelines may help you take some decisions about backup and restore procedures:

### Performance

- Schedule Backup Operations When Database Activity Is Low: Database operation may have an important impact on the server; avoid backup operation when the activity of the server is high, especially full database backups.
- Back up first to disk, whenever possible. Consider using a File Server to store the backups. Backing up to disk will greatly increase the performance of the backup process and free the resources of SQL Server. Using file backups also simplifies the restoration process. When backing up to a remote share, use UNC paths and do not use mapped drives.
- When using a File server to store backups, consider using a private LAN trunk to avoid general network congestion. A private trunk requires that both the File and the Database Servers have an additional network card connected to a private LAN. This practice also increases the security of the backup process.
- Back up to tape or other devices for long run storage. Tapes and other media types offer an economical option to store the information for the long run.
- Do not use the same physical disk that hold the database files or Log files for backup purposes. Using the same physical disk not only affects the performance, but also may reduce the recoverability of the plan.

### Security

The following measures will help you reduce or handle security threats:

### Confidentially

- **Do not use the Password option of the Backup statement:** The password option of the backup statement provides weak protection and does not prevent reading of the data. The password option is also deprecated and

will be removed in future releases of SQL Server.

- **When using the file system for backups, grant file and folder access to the SQL Service account and DBA only.** The SQL Service account will require Read and Write access to the folder or shared folder and the DBA will required Full Control and should be the owner of the folder. Restricting access to the backup files will prevent users from copying the files and restore them in a server where they have administrative rights.
- **Consider establishing an Auditing Policy on the Backup Folder.** Enabling file access auditing to the Backup Folder will monitor user access to the backups.
- **After the database is back up in a file or files, compress and encrypt the files before moving the contents to tape backups or other forms of long term storage.** Encrypting the backup files will help you protect the confidentiality of the information if somebody gains physical access to the media.
- **Physically secure backup media.** Do not rely only on encryption to secure the media, make sure that the media is always physically secure to avoid someone to copy the media, and use brute force attacks to decrypt the file and eventually restore the database in another server.
- **Sanitize the backup media before disposal.** Do not throw tape media in the trash, unless you are absolutely certain that the data has been erased. Deleting the files or overwriting the files may not be enough. Consider the use of specialized software or magnetic methods to erase the information.

### Integrity

- **Use the CHECKSUM option of the Backup command.** This option, new in SQL Server 2005, essentially performs two types of verifications on database backups. The first one is that it stores a CHECKSUM of the backup that can help you verify if the backup is corrupt. The second verification only works if the PAGE\_VERIFY option of the database is in CHECKSUM or TORN\_PAGE\_DETECTION mode. If one these modes are enabled, the BACKUP statement will verify that the pages are reliable before they are backed up. The default of the PAGE\_VERIFY option is CHECKSUM; however the CHECKSUM option of the Backup command is not enabled by default. The following exemplifies the CHECKSUM option of the backup command:

```
BACKUP DATABASE AdventureWorks
    TO DISK = N'C:\Backup\AdventureWorks.bak'
WITH CHECKSUM
```

- **Test Backup files periodically using the RESTORE VERIFYONLY command.** The following code is an example of how to use RESTORE VERIFYONLY command:

```
RESTORE VERIFYONLY
    FROM DISK = N'C:\Backup\AdventureWorks.bak'
WITH CHECKSUM
```

## Availability

- **Use the Mirror option of the Backup command to backup to a local hard drive and a backup server simultaneously.** Having a local copy of the backup increases the response time of the restoration process and having a remote copy makes the copy available in case of a server collapse. The following command is an example on how to use the MIRROR clause of the BACKUP command:

```
BACKUP DATABASE AdventureWorks
TO DISK='C:\Backup\AdventureWorks.bak'
MIRROR TO DISK='\\ILUVATAR\Backup\AdventureWorks.bak'
WITH FORMAT
```

- **Perform trial restorations frequently.** Frequently you should verify that the backup system is properly copying the database and that the backup hardware and process is working correctly. Trial restorations should use test server and should never be performed in a production environment.
- **Perform Fire Drills periodically.** A fire drill is a procedure that tests if the backup and restore plan will help the IT department to be compliant with the SLA. In a fire drill not only the backup is restored to verify the integrity, the whole recovery plan is executed to verify readiness in case of an event. A fire drill will assume that the production server or data center is down and

the recovery team will test their ability to configure an alternate environment ready for production.

## Manageability

- **Use SQL Server Agent to automate the Backup process.** Automating the backup process will facilitate the backup process and reduce the likelihood of human errors. The SQL Server agent will help DBAs automate administrative tasks, which include the backup process.

To create a Job to backup a database use the following procedure:

- 1) Open SSMS
- 2) Connect to the SQL Server instance that holds the database.
- 3) Navigate to the Server\Databases folder.
- 4) Right click the database you want to backup.
- 5) From the shortcut menu, select Tasks, Backup.
- 6) In the Database Backup dialog box, select the type of backup you want the server to perform, the backup destination path and the backup options.
- 7) Press Ctrl+Shift+M or select Script to Job option in the Backup Database window to create a Job.
- 8) Select Schedules in the Select a Page listbox.
- 9) Press new to create a schedule for the backup.
- 10) Configure the backup schedule (Figure 2) and select Ok to confirm your selection.

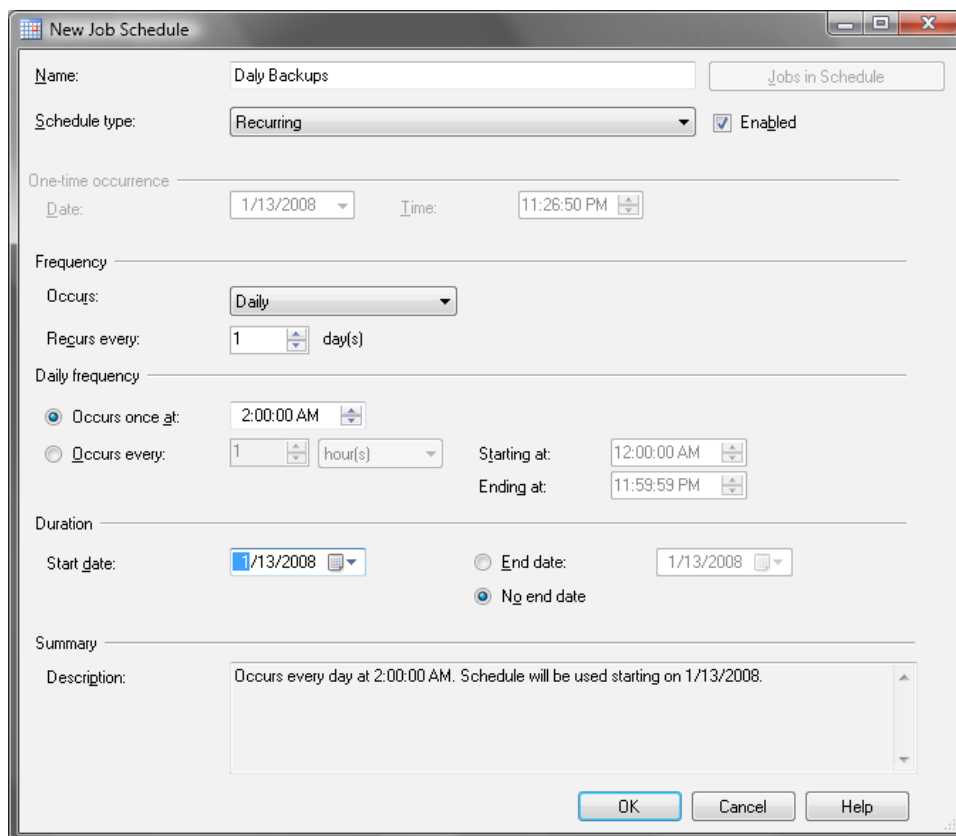


Figure 2 New Schedule Window

11) Select Ok to create the Job.

- **Document the backup strategy:** Documenting the backup strategy will help the restoration process.
- **Define a backup media/tape rotation policy.** A rotation policy guarantees that there will be multiple copies of backups to recover from and protects the media from excessive wear and damage. You should always keep at least three different medias and at least one of them should be an off-site copy.

The two most common strategies for tape rotation are Grandfather – Father – Son (GFS) and Tower of Hanoi.

The GFS approach uses three levels of tape rotation. The first level contains daily backups, the second level weekly backups and the third level contains monthly or four week period backups. Using this method in a 24x7 database will require twenty two tapes per year, assuming that each backup will use one tape. Six of the tapes will be used in for daily backups (Sons), three tapes for weekly backups (fathers) and thirteen tapes will be used as monthly backups (Grandfathers). Monthly backup will also be used for long term archiving. The following diagram will help you understand the GFS tape rotation policy:

Week	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	Son 1	Son 2	Son 3	Son 4	Son 5	Son 6	Father 1
2	Son 1	Son 2	Son 3	Son 4	Son 5	Son 6	Father 2
3	Son 1	Son 2	Son 3	Son 4	Son 5	Son 6	Father 3
4	Son 1	Son 2	Son 3	Son 4	Son 5	Son 6	Grandfather

The Tower of Hanoi is a more complex system but uses fewer tapes than the GFS strategy. It will require 18 tapes per year to back up a database that uses one tape per backup. In the Tower of Hanoi approach there are 5 levels of tape rotation: Level A is used every 2 days; Level B every 4 days, Level C every 8 days, Level D every 16 days, and Level E every 32 days. Finally, a Level F is also every 32 days, but it is used only for long-term archiving. The following diagram will help you understand the Tower of Hanoi tape rotation policy:

Week	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	A	B	A	C	A	B	A
2	D	A	B	A	C	A	B
3	A	E	A	B	A	C	A
4	B	A	D	A	B	A	C
5	A	B	A	F			

Either approach can be adapted to handle intra-day backups.

- **Create a backup media log:** A backup media log keeps records of who handles backup media. Keeping logs may be helpful in case of an event that requires restoring the data and may be used for legal purposes.

## Other Best Practices

- **Before restoring any database, consider the need of backing up the existing log file.** When the database files and the log are in different devices and the database device fails; or if the database file is corrupted or damaged you may be able to backup the log and recover the transactions up to the moment of failure. Backing up the log file may also be useful for forensic purposes in the case of security breaches. To back up a database that has the database files damaged, use the NO\_TRUNCATE or the COPY\_ONLY and CONTINUE\_AFTER\_ERROR options of the BACKUP command. The following code is an example of how to back up an orphan log.

```
BACKUP LOG Test
    TO DISK = 'C:\Backup\AdventureWorksLog.bak'
WITH COPY_ONLY, CONTINUE_AFTER_ERROR
```

- **Before performing any High Risk operation add a Log Mark or create a Database snapshot.** Adding a Log Mark or a Database snapshot will facilitate the recovery process in the event of failure or need to recover.

To add a Log Mark use the WITH MARK option of the BEGIN TRANSACTION

Command. An example of the code can be found below:

```
BEGIN TRANSACTION UpgradeVersion
    WITH MARK 'Upgrade to Version 2.1 '
GO
    -- Add Transactional code here
COMMIT TRANSACTION UpgradeVersion
```

To restore a database up to the mark, use the STOPATMARK option of the RESTORE LOG command. The following code will restore the database up using the previously created log mark.

```
RESTORE DATABASE AdventureWorks
FROM DISK = N'C:\Backup\AdventureWorks.bak'
WITH FILE = 1
, NORECOVERY

GO
RESTORE LOG AdventureWorks
FROM DISK = N'C:\Backup\AdventureWorks.bak'
WITH FILE = 2
, STOPATMARK='UpgradeVersion'
GO
```

You can also use database snapshots when you want to have the ability to recover in case of a high risk operation. Database snapshots contrary to Log Marks can be used to recover databases that use the simple recovery model. On the other hand, database snapshots are only available in the Enterprise Edition of SQL Server 2005. The following code creates a database snapshot of the AdventureWorks database:

```
CREATE DATABASE AdventureWorks_PreviousVersion
ON ( NAME = AdventureWorks_Data
, FILENAME = 'C:\Backup\AdventureWorks_
PreviousVersion.ss' )
AS SNAPSHOT OF AdventureWorks;
GO
```

The following code will restore the database from a previously created snapshot:

```
RESTORE DATABASE AdventureWorks
FROM DATABASE_SNAPSHOT = 'AdventureWorks_
PreviousVersion'
```

- **Only DBAS should be able to restore and backup databases:** Limit access to Backup and Restore operations, because either one can hinder your backup plan.
- **Use the COPY\_ONLY option for unscheduled backups.** The COPY\_ONLY option of the BACKUP command identifies backups that should not affect the normal sequence of backups. This command is useful when you want to backup databases outside your DRP. If you do not use this option you may disrupt the plan, because the backup may not be available in case of a disaster. When you run a full backup with the COPY\_ONLY, the command will not affect following differential backups that will not use this backup as the starting point. When you run a Log backup with the COPY\_ONLY option, the Log is not truncated and will be available for the next Log backup. An example of how to use the COPY\_ONLY option:

```
BACKUP DATABASE AdventureWorks
TO DISK = 'C:\Backup\AdventureWorks.bak'
WITH COPY_ONLY
```

## Summary of Backup Best Practices

The following guidelines are a summary of the best practices included in this whitepaper:

- Have a Data Recovery Plan
  - Begin designing your plan with a threat analysis
  - Make sure your plan covers threats from all sources: Hardware, Software, People and Environment.
  - You should consider the value, volatility, size and usage of your database when designing your data recovery plan.
  - The most relevant operational requirements of data recovery plans are: security, performance, scalability and manageability. Make sure your plan considers these requirements.
- Backup Strategies
  - Combine different backup types to create a backup strategy.
  - Do not backup tempdb.
  - Unless you plan to change its contents, do not back up the model database.
  - Use the Full Backup simple mode strategy only in development, read only, stage or system databases.
  - Schedule the Log backup before the Full backup
  - Use the Log & Full backup strategy in 8x5 databases where you can afford to lose up to one day of work.
  - Consider the use of the Full & Log backup strategy in 8x5 databases, and non-volatile 24x7 databases.
  - Use the Full, Differential and Log strategy in volatile 24x7 databases, when the value or your data requires very frequent log backups, and in non-partitioned Data marts.
  - Consider the use of a Full - File/Filegroup – Log Backup strategy in Very Large Databases (VLDB) or partitioned Data marts/Data warehouses.
- Write a Service Level Agreement that describes the response time the IT Department will provide in case of each type of event.
- Performance
  - Schedule Backup Operations When Database Activity Is Low.
  - Back up first to disk, whenever possible. Consider using a File Server to store the backups.
  - When using a File server to store backups, consider using a private LAN trunk to avoid general network congestion.
  - Back up to tape or other devices for long run storage.
  - Do not use the same physical disk that hold the database files or Log files for backup purposes.
- Security
  - Do not use the Password option of the Backup statement.
  - When using the file system for backups, grant file

- and folder access to the SQL Service account and DBA only
- Consider establishing an Auditing Policy on the Backup Folder
- After the database is back up in a file or files, compress and encrypt the files before moving the contents to tape backups or other long term storage.
- Physically secure backup media.
- Sanitize the backup media before disposal.
- Use the CHECKSUM option of the Backup command
- Test Backup files periodically using the RESTORE VERIFYONLY command.
- Use the Mirror option of the Backup command to backup to a local hard drive and a backup server simultaneously.
- Perform trial restorations frequently.
- Perform Fire Drills periodically.
- Manageability
  - Use SQL Server Agent to automate the Backup process.
  - Document the backup strategy.
  - Define a backup media/tape rotation policy.
  - Create a backup media log.
- Other Best Practices
  - Before restoring any database, consider the need of backing up the existing log file
  - Before performing any High Risk operation add a Log Mark or create a Database snapshot
  - Only DBAS should be able to restore and backup databases.
  - Use the COPY\_ONLY option for unscheduled backups.

For more information, contact Dell. Information in this document is subject to change without notice.

Microsoft, SQL Server, and Windows are registered trademarks of Microsoft Corporation. Other trademarks and trade names may be used in this document to refer to either the entities claiming the marks and names or their products. Dell disclaims proprietary interest in the marks and names of others.

**Javier Loria** is a Solid Quality Mentors Mentor based in Costa Rica. He began his professional career in 1992, as a software developer and system engineer. His career evolved rapidly to the training side, especially in the XML and OLAP world, training customers at different locations in Latin America. He is a software architect, and Business Intelligence Architect assisting our customers in Latin America. Javier became SQL Server MVP in 2001. Javier is an MCT, MCSE, MCSD, MCDBA, and MCAD. Javier was an active member responsible for writing the Microsoft SQL Server 2005 courseware. He is the coauthor of several books including *MOC2782: Designing Microsoft® SQL Server™ 2005 Databases*, *MCTS Training Kit 70-431: Implementing and Maintaining SQL Server 2005* and *Microsoft SQL Server 2005 Database Essentials Step by Step*.