
**14th ANNUAL USERS CONFERENCE**

January 25 – 27, 2016 | Bonita Springs, FL

# Redundancy Options

Presented By:

Chris Williams

PERENNIAL SOFTWARE 20TH ANNIVERSARY

Makers of SedonaOffice and AlarmBiller®

# Table of Contents

# Redundancy Overview

This topic describes the benefits of backing up SQL Server databases, basic backup and restore terms, and introduces backup and restore strategies for SQL Server and security considerations for SQL Server backup and restore.

The SQL Server backup and restore component provides an essential safeguard for protecting critical data stored in your SQL Server databases. To minimize the risk of catastrophic data loss, you need to back up your databases to preserve modifications to your data on a regular basis. A well-planned backup and restore strategy helps protect databases against data loss caused by a variety of failures. Test your strategy by restoring a set of backups and then recovering your database to prepare you to respond effectively to a disaster.

## Redundancy Benefits

- Backing up your SQL Server databases, running test restores procedures on your backups, and storing copies of backups in a safe, off-site location protects you from potentially catastrophic data loss.
- With valid backups of a database, you can recover your data from many failures, such as:
    - Media failure.
    - User errors, for example, dropping a table by mistake.
    - Hardware failures, for example, a damaged disk drive or permanent loss of a server.
    - Natural disasters. By using SQL Server Backup to the SedonaBackup storage service, you can create an off-site backup in a different region than your on-premises location, to use in the event of a natural disaster affecting your on-premises location.
- Additionally, backups of a database are useful for routine administrative purposes, such as copying a database from one server to another, setting up ‘AlwaysOn Availability Groups’ and archiving.

## Introduction to Backup and Restore Strategies

Backing up and restoring data must be customized to a particular environment and must work with the available resources. Therefore, a reliable use of backup and restore for recovery requires a backup and restore strategy. A well-designed backup and restore strategy maximizes data availability and minimizes data loss, while considering your particular business requirements.

*Important: Place the database and backups on separate devices. Otherwise, if the device containing the database fails, your backups will be unavailable. Placing the data and backups on separate devices also enhances the I/O performance for both writing backups and the production use of the database.*

A backup and restore strategy contains a backup portion and a restore portion. The backup part of the strategy defines the type and frequency of backups, the nature and speed of the hardware that is required for them, how backups are to be tested, and where and how backup media is to be stored (including security considerations). The restore part of the strategy defines who is responsible for performing restores and how restores should be performed to meet your goals for availability of the database and for minimizing data loss. We recommend that you document your backup and restore procedures and keep a copy of the documentation in your run book.

Designing an effective backup and restore strategy requires careful planning, implementation, and testing. Testing is required. You do not have a backup strategy until you have successfully restored backups in all the combinations that are included in your restore strategy. You must consider a variety of factors. These include the following:

- The production goals of your organization for the databases, especially the requirements for availability and protection of data from loss.
- The nature of each of your databases: its size, its usage patterns, the nature of its content, the requirements for its data, and so on.
- Constraints on resources, such as: hardware, personnel, space for storing backup media, the physical security of the stored media, and so on.

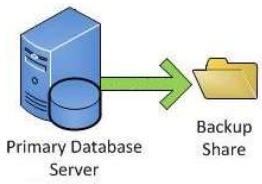
## Recovery Models

Too simply we will classify backups into four distinct types, each having unique properties and complexities: Cold Backups, Warm Backups, Hot Backups and SedonaBackup.

With each of the models, we always recommend local backups be moved off-site at regular intervals to protect your data. Backups made onto your local SQL Server are not good enough. Moving your backups to another server or internet cloud storage facility in also not enough. You must move regular backups to a safe location, preferably outside of your building, on media that can be encrypted and stored for long intervals (DVD, External Hard Drives, Flash Drives, etc.)

## Cold Backup

This is the simplest and most common method, where regular scheduled (or manual) backups of your data are created (and stored on a separate device). When needed, another SQL Server of the same version or greater can be used to bring the data back on-line.




Pros:

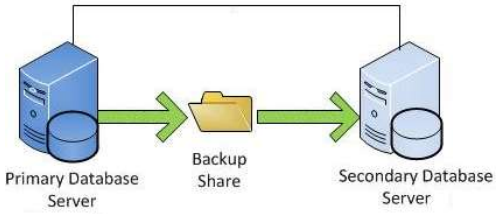
- Low cost, only one server is needed at any given time
- Easy to setup and maintain

Cons:

- Guaranteed data loss back to the last good Backup
- Time to recover is measured by the interval required to bring another SQL Server on-line (hours, possibly days)

Warm Backup

This method requires two (2) SQL servers, one primary and one secondary. The primary server moves regular scheduled backs to the secondary where they are stored until needed. If the primary server becomes unavailable, then the last held backups are manually restored on the secondary server and business resumes.




Pros:

- Relatively easy to maintain
- Short time to recovery (typically measured in minutes or hours)
- Data can be restore onto the secondary server to test your backups

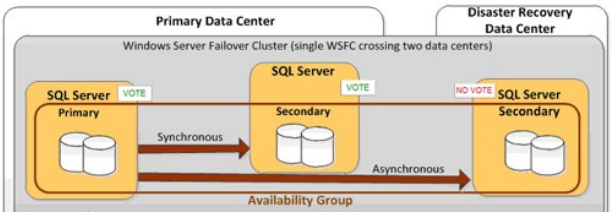
Cons:

- Cost, requires a minimum of two (2) servers with similar hardware specs and SQL installations.
- Recovery requires intervention by IT staff to manually restore last backups and redirect workstations.

## Hot Backup

Beginning with SQL 2012, Microsoft introduced 'AlwaysOn Availability Groups'. This method is by far the most robust of all the on-site recovery models, allowing for near instant recovery in the event of a failed SQL Server. This is also the only fully supported method of running SedonaOffice in a mirrored server configuration.

AlwaysOn Failover Clustering Instances support multi-subnet clustering across geographically dispersed sites, allowing you to deploy so-called Stretch Clusters. This architecture offers both a high level of availability as well as the basis for an advanced disaster recovery solution. SQL Server 2012's (and up) AlwaysOn FCI feature also improves failure detection and provides more flexibility for configuring an optimal failover policy.




Pros:

- Near real time recovery
- Secondary server can be used for reporting (SQL Enterprise and up)
- Supports automatic page repair for protection against page corruption
- Supports encryption and compression, which provide a secure, high performing transport.
- Up to 9 SQL Servers can be added to any single Availability Group

Cons:

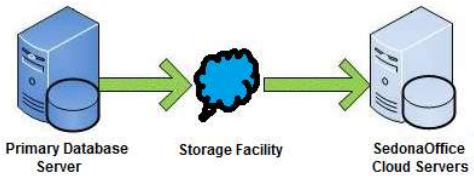
- Cost, requires SQL 2012 (or greater) installed on two (2) identical servers.
- Complex setup requiring knowledge of Microsoft Clustering Services
- High IT maintenance

*NOTE: Fail-over Rights for Microsoft SQL Server: For any Operating System Environment (OSE) in which you are running instances of SQL Server you may use up to the same number of "passive fail-over" Running Instances in a separate OSE on ANY Server for temporary support. You may run the passive fail-over instances on a server other than the licensed server. A passive SQL Server instance is one that is not serving SQL Server data to clients or running active SQL Server workloads.*

*If you can live with the passive server being your active server for the next 90 days, you need no further licensing. However, if you have another fail-over event within 90 days you may not reassign or move the licenses to a fail-over server, thus rendering your fail over rights unusable during the 90 days after the last fail over event.*

## SedonaBackup

With SedonaBackup, a complete backup of your production SedonaOffice database(s) is transmitted via a secured SSL connection to a password protected vault on our cloud based backup services located in a tier-1 access controlled data centers. SedonaBackup is unique from all the others in that when called upon, we will host your data and application on our servers until your on-site SQL server is restored.




Pros:

- Designed specifically for SedonaOffice
- Backup is verified once a month
- Scheduled daily, weekly or monthly backups (your choice)
- Secure Off-site storage
- No additional storage or equipment required
- No additional charge while using Perennial Software servers during an event
- Can be used from any internet connected workstation with Microsoft Internet Explorer.

Cons:

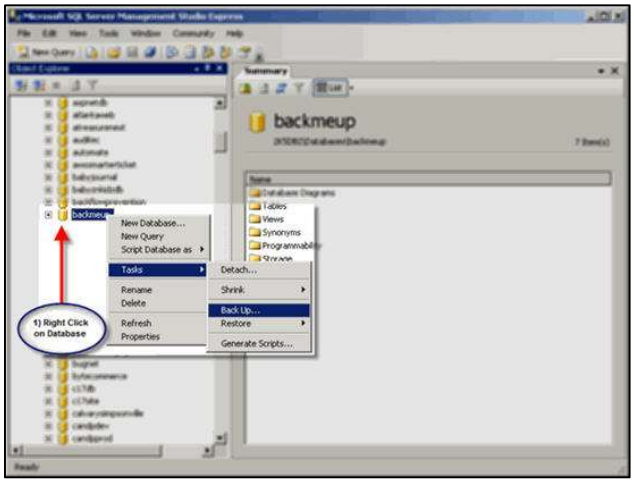
- Times to backup or restore limited by internet speeds.
- Data loss possible between last received backup and time of failure.

# Backups How-To

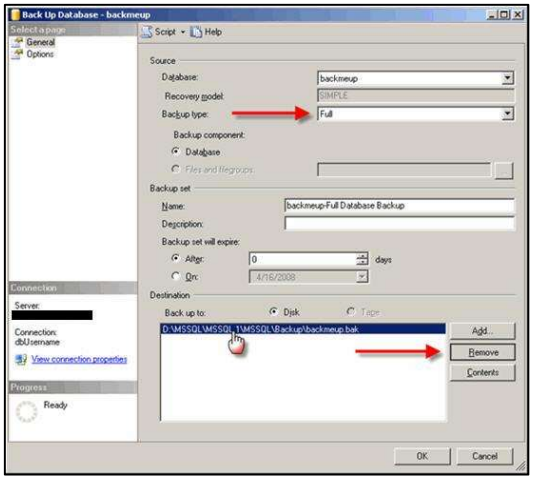
## Creating a Basic Manual Backup

Step 1: You will first need to connect to your database server, using SQL Server Management Studio Express. Please see our article "How to connect to a database using SQL Server Management Studio Express" if you require assistance with this.

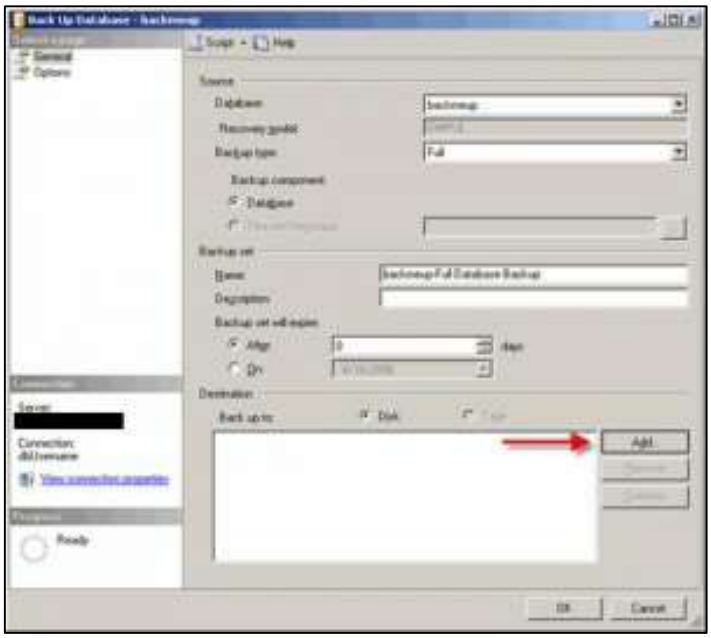
- For this example, we will back up the database named "backmeup" to our "C:\backup" folder.
- Once connected to your database server, you will need to browse to your database in the left window pane of Management Studio.
- Right Click on your database, and under "Tasks" choose "Back Up…"



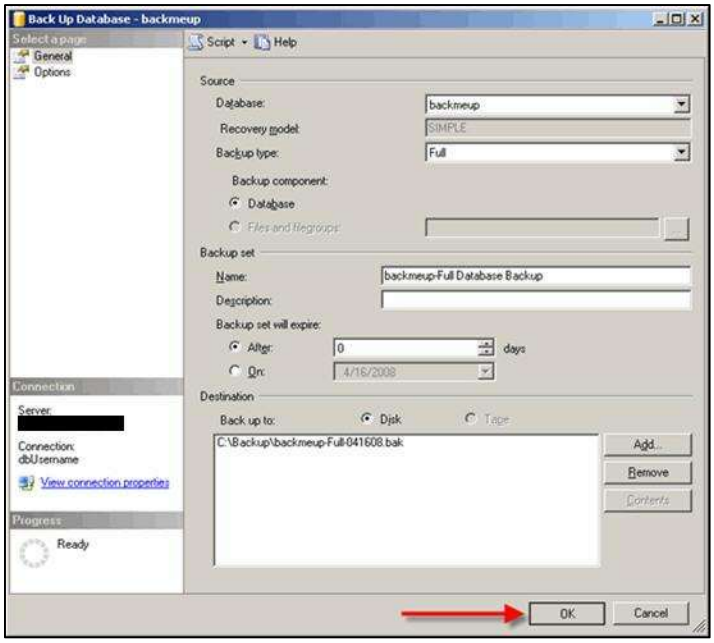

Step 2: A new window will open where we will configure where to save the database backup file. You will want to ensure that the "Backup type" is set to "Full" to get a Full backup of your database. Now, highlight the Destination file that is already in the list by clicking on it, and Click "Remove".



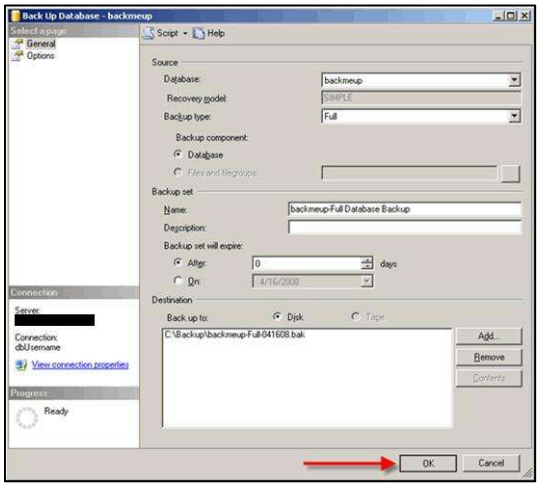
Step 3: Now, Click “Add…” to specify where to save the backup file.



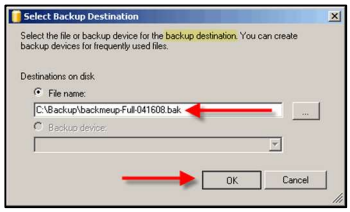
Step 4: A window will pop up, asking you where to save the backup file. Enter a location of your choice. For this example, we will save it to “C:\Backup\backmeup-Full-041608.bak“. The can also be a network location if you run SQL 2008 and up: “\\servername\backmeup-Full-041608.bak”



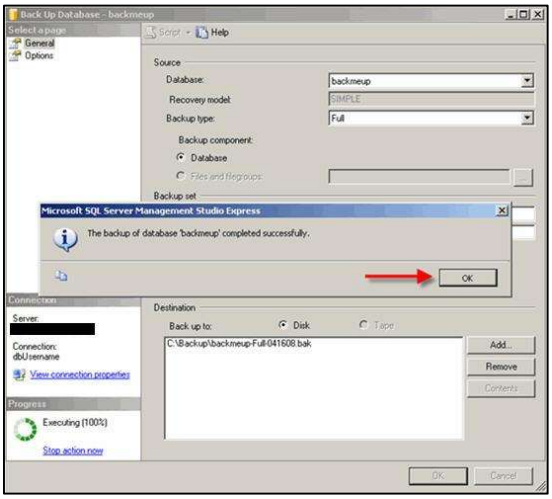
Step 5: Now, we have to make sure that the backup you are creating overwrites all existing backup sets, as appending it to an existing set can cause conflicts when attempting to perform a restore. On the left-hand side of the window, click on 'Options', and then click on 'Overwrite all existing backup sets'.




Step 6: Once this option is in place, all that is left to do is to run the backup! Click "OK" to begin the database backup.




Step 7: If the database backed up successfully, you should receive a message as pictured below.

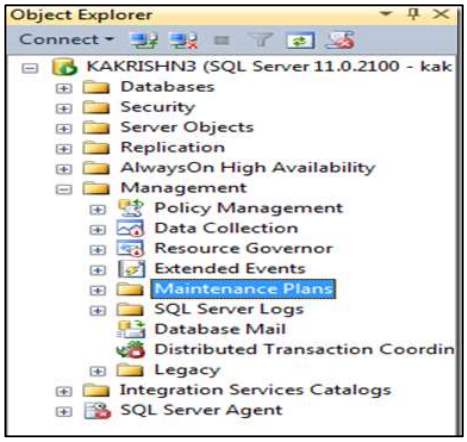


## How to create backups using database maintenance plan

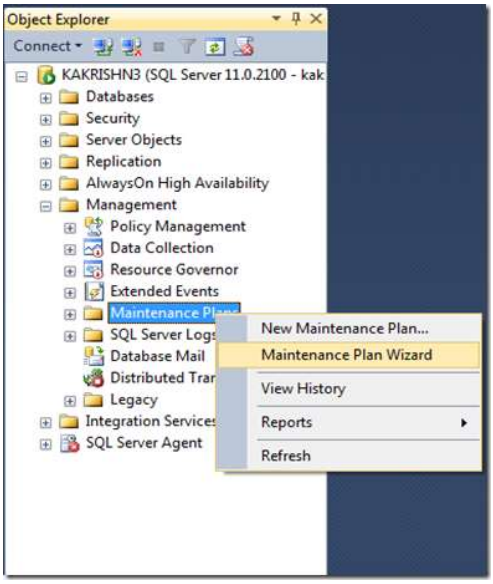
You can create a database maintenance plan to automate the SQL Server database backups. SQL Server backup maintenance plan can be scheduled to back up the databases automatically or executed manually.

Follow the steps below to create a database backup maintenance plan and schedule it to execute automatically

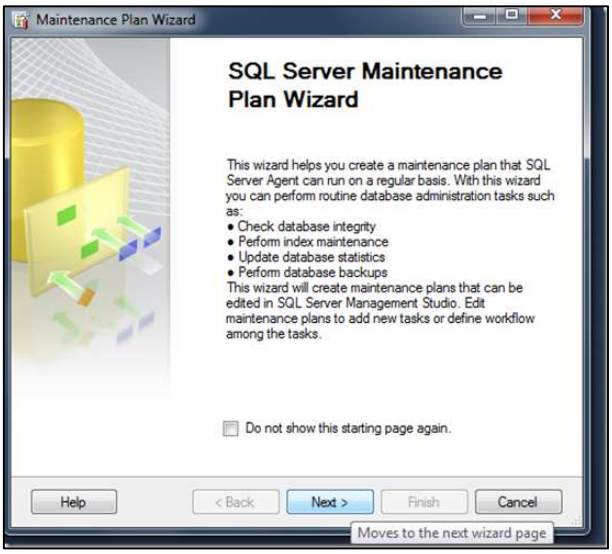
Step 1: Open SQL Server Management Studio, expand the Management node, and then expand the Management Plans node.



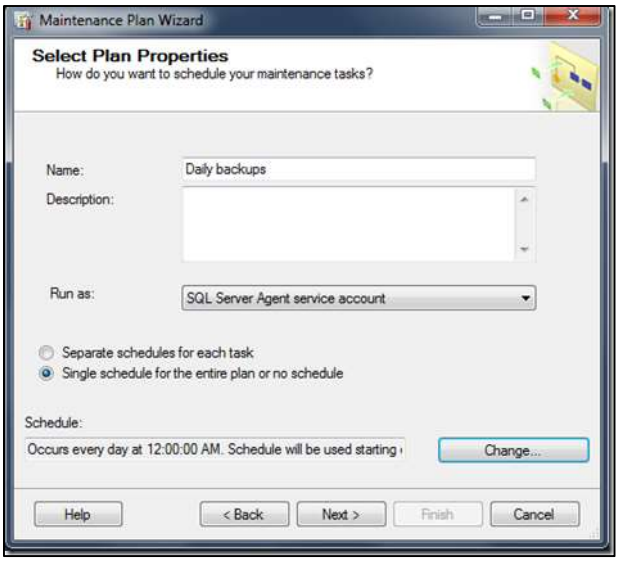

Step 2: Right-click Maintenance Plans, click Maintenance Plan Wizard.



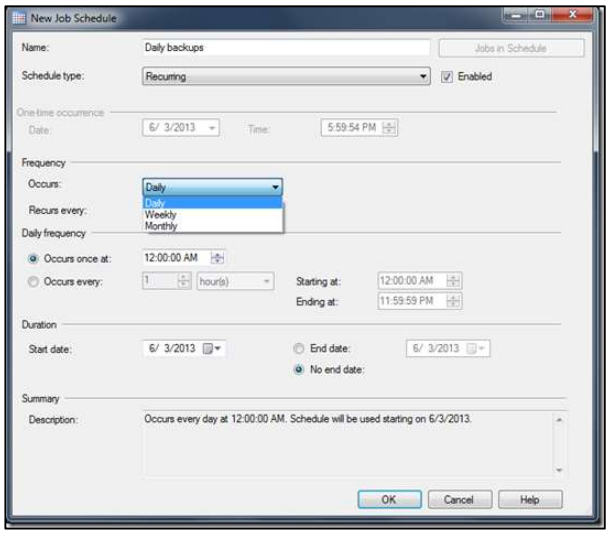
Step 3: SQL Server Maintenance Plan Wizard window will appear. Click next.




Step 4: Then type a name for this database backup plan.



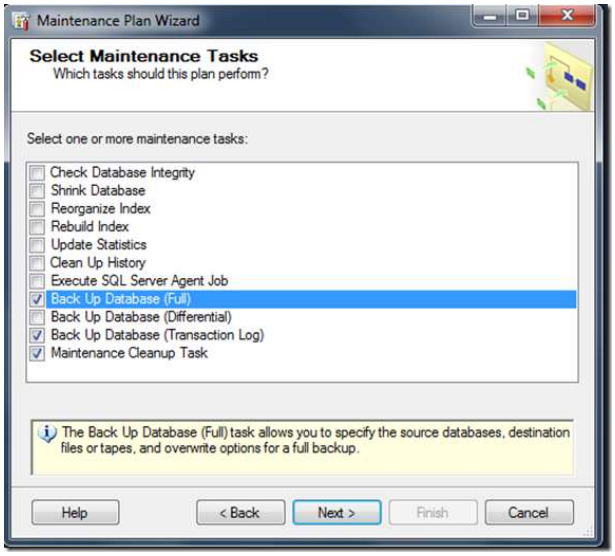
Step 5: Select schedule according to your need.



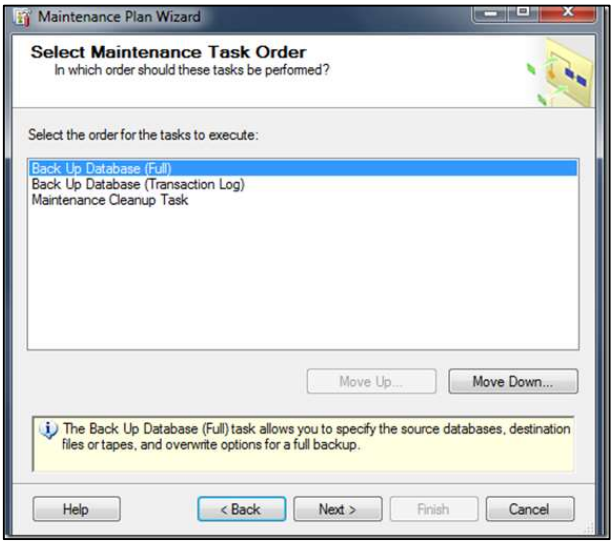

*Scheduling Consideration: No matter if you choose one or multiple schedules for your tasks you will want to do these operations during off peak hours if at all possible. You can run some tasks in parallel or overlap, while others should not be. If you do, you may find your server becoming too slow.*

*If you choose to have separate schedules for each task then you will need to take an educated guess as to how long each task will take and set the schedule accordingly. The tricky part is that the same task doesn't always take the same amount of time to run each time. So, you will want to watch it over time to see if it needs to be adjusted to accommodate the longer durations. You can get these durations by going to the View History on the jobs.*

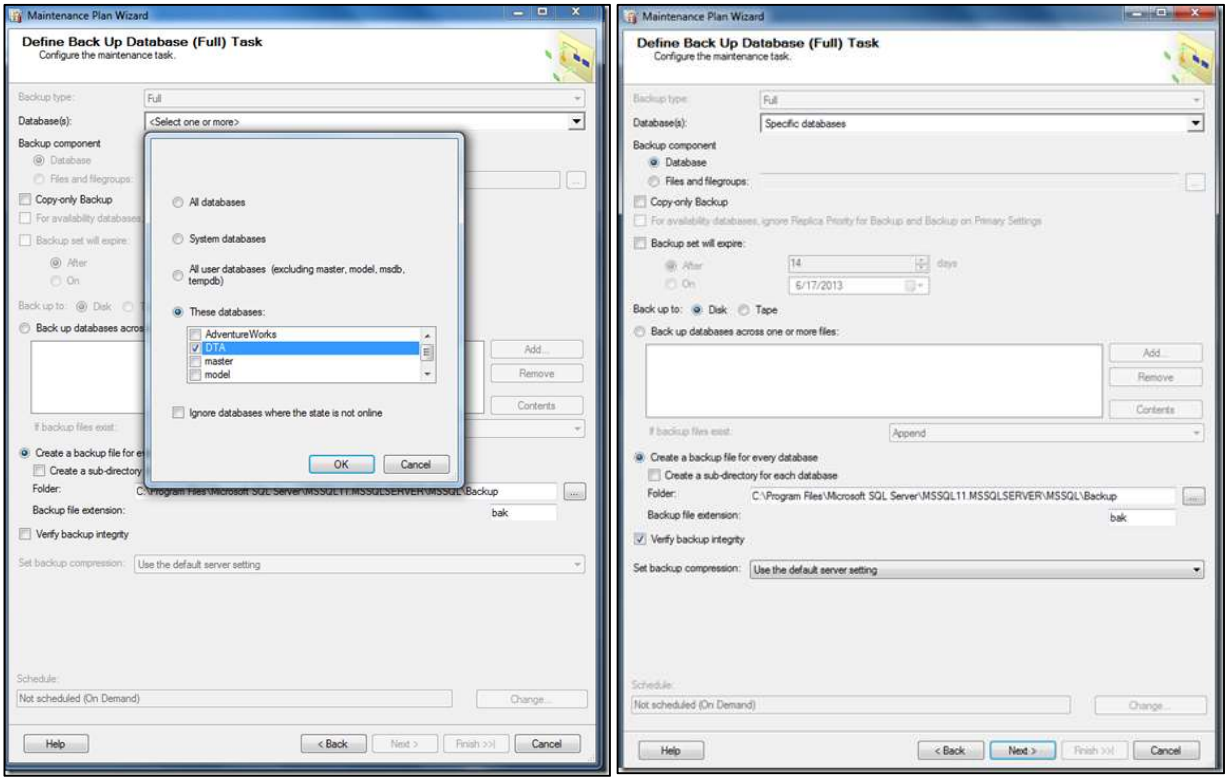
Step 6: Select the maintenance tasks, which you wanted to plan. (Transaction Log backups are not required for 'Simple' recovery mode databases)



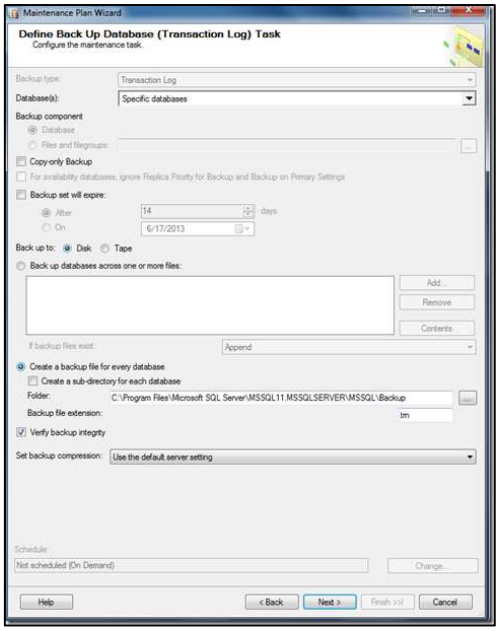

Step 7: Select the order of your plan. Click Next.



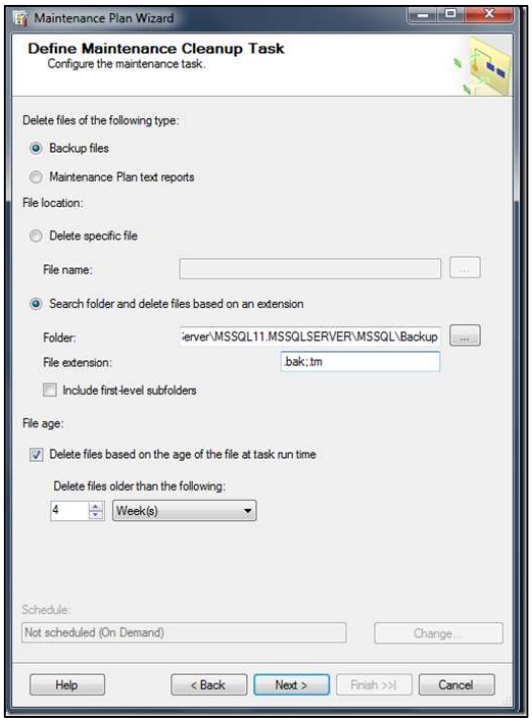
Step 8: On the Define Backup Database (Full) Task dialog box, specify information about the full backup. Specify database where this plan has to be applied. Press ok. Then Click Next.



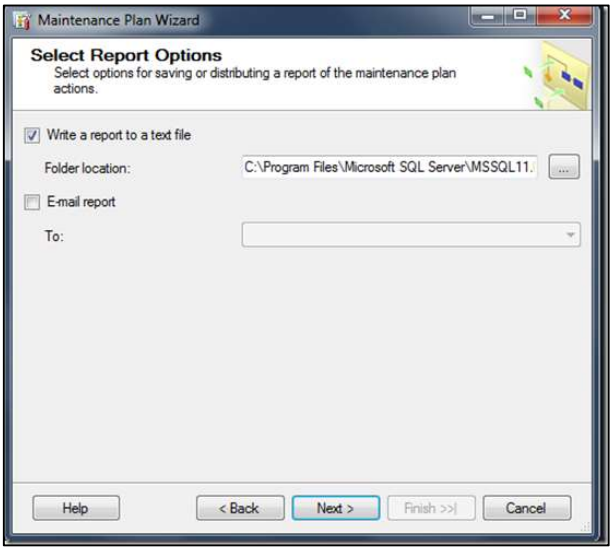

Step 9: Choose the backup destination



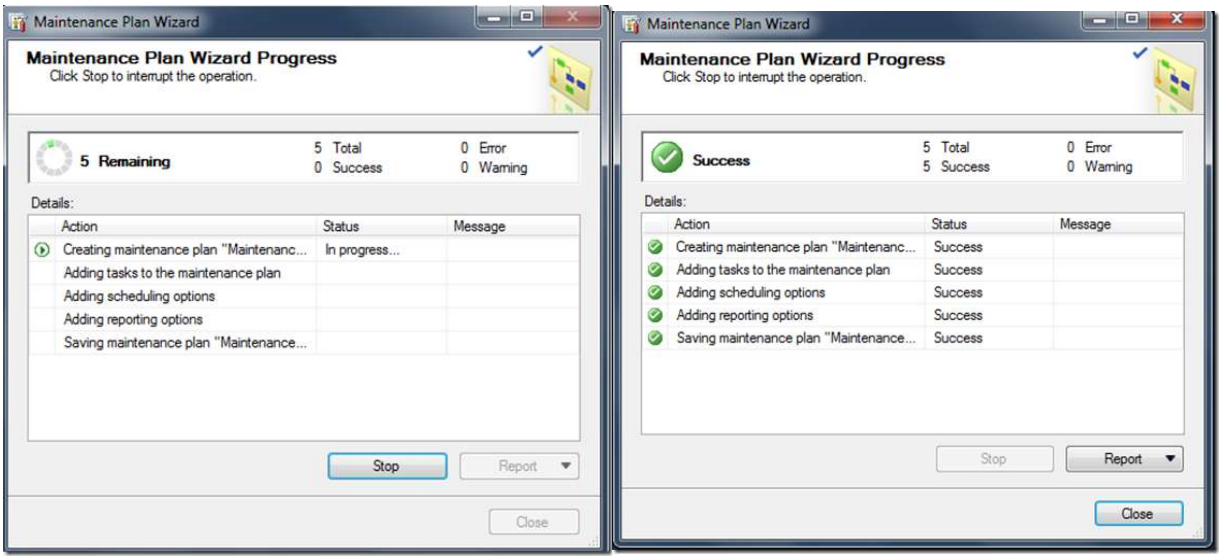
Step 10: On the Define Maintenance Cleanup Task dialog box, configure the cleanup tasks. Specify the folder name where you take backups. Then specify the backup folders extension. Click Next.




Step 11: On the Select Report Options dialog box, select whether to write the report to text file or send the report through email. Click Next.



"Plan creation in progress and finished screens"

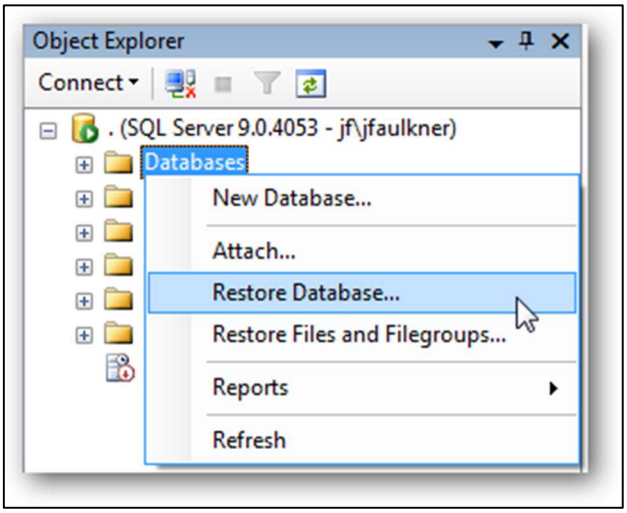


## Restoration

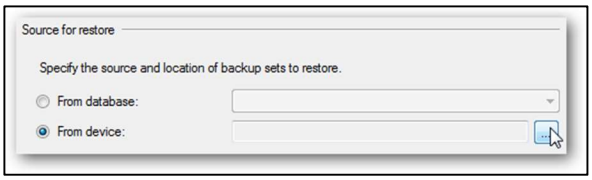
Congratulations, you have created your backups and jotted down your recovery plans, now you need to test your backup. Database restoration is very easy if you follow this simple guide:

Open SQL Server Management Studio and login to the SQL Server you want to restore the database to. It is best to either login as a Windows Administrator or as the SQL 'sa' user.

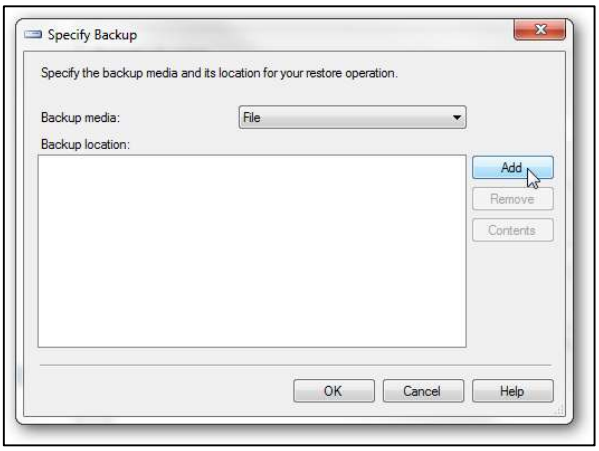

Once logged in, right click on the Databases folder and select 'Restore Database'.



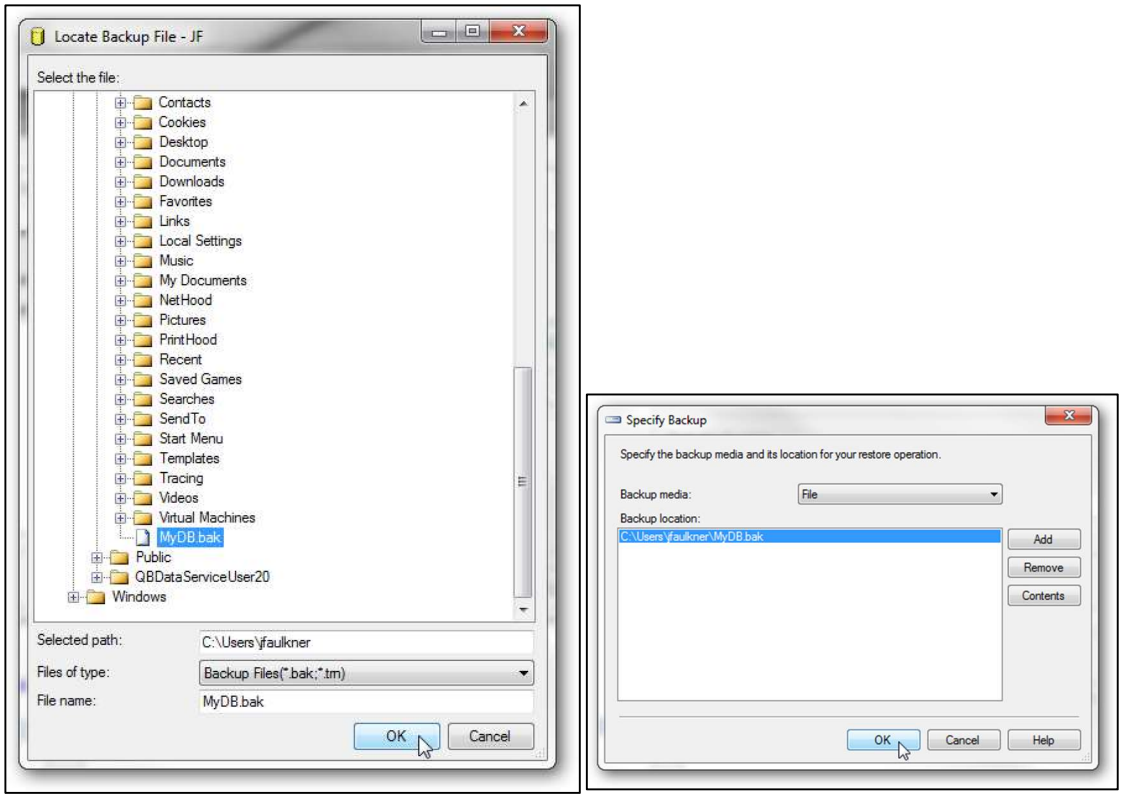

Click the ellipses button next to 'From device' under the 'Source for restore' section.



Set 'File' as the backup media and then click 'Add'.




Browse to the SQL backup (BAK) file you want to restore.

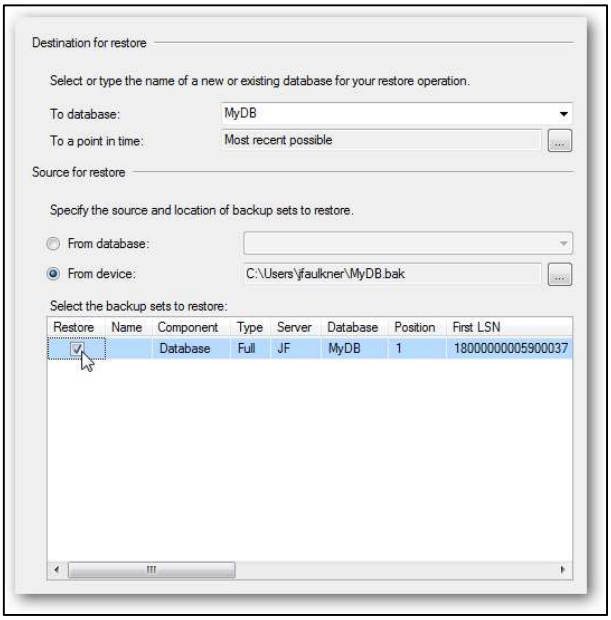



In the Restore Database dialog, type or select the name of the database you want this backup restored to.

If you select an existing database, it will be replaced with the data from the backup.

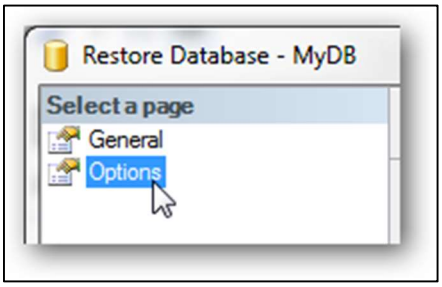
If you type a database name which does not currently exist in your SQL Server installation, it will be created.

Next, select the restore point you want to use. Since a SQL backup file can hold multiple backups you may see more than one restore point listed.




At this point, enough information has been entered for the database to be restored. However, SQL backup files store information about where data files are copied so if there are any file system problems such as a the destination directory not existing or conflicting data file names an error will occur. These problems are common when restoring a backup created on a different SQL Server installation.

To review and change the file system settings, click the Options page on the left in the Restore Database dialog.

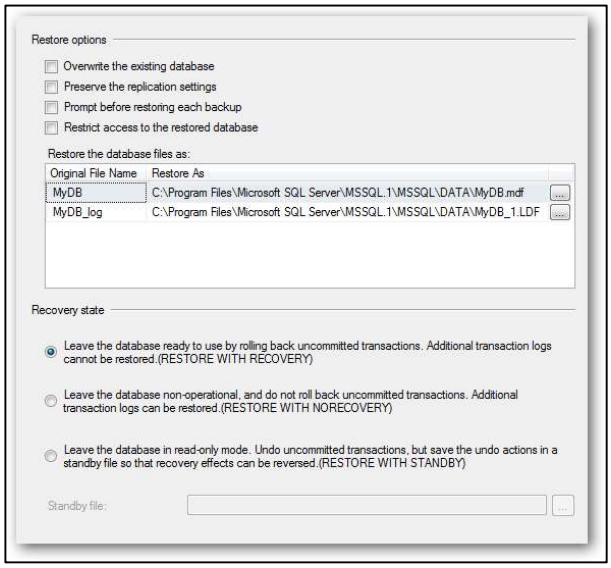



On the options page, you will want to make sure the 'Restore As' column points to valid folder locations (you can change them as needed). The files do not have to exist, however the folder path must exist. If the respective files do exist, SQL Server follows a simple set of rules:

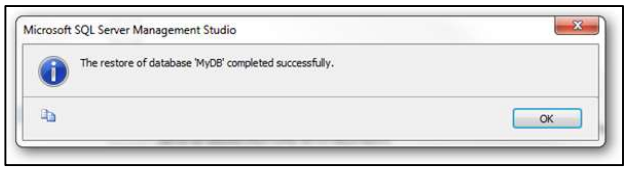
If the 'To database' (from the General page) matches the restore database backup (i.e. restoring to matching databases), the respective files will be overwritten as part of the restore.

If the 'To database' does not match the restore database backup (i.e. restoring to a different database), the 'Overwrite the existing database' will need to be checked for the restore process to complete. Use this function with caution as you can potentially restore database backup information on top of data files from a completely different database.

Generally, you can tell the databases differ based on the 'Original File Name' which is the internal name SQL Server uses to reference the respective files (SedonaOffice will always use 'security_data and security_log)




Once your restore options are set, click Ok.



That’s it, you’re done. If you are restoring to the original SQL Server, no further steps are required. But, when you restore a Microsoft SQL Server database on a different machine, you cannot access the database until you fix the permission.

The problem is that the user in the database is an "orphan". This means that there is no login id or password associated with the user. This is true even if there is a login id that matches the user, since there is a GUID (called a SID in Microsoft-speak) that has to match as well.

This used to be a pain to fix, but currently (SQL Server 2000, SP3 and up) there is a stored procedure that does the heavy lifting.

All of these instructions should be done as a database admin, with the restored database selected. Open a new Query window and execute the following against the restored database:

```
EXEC sp_change_users_login 'Auto_Fix', 'SedonaUser'

GO

EXEC sp_change_users_login 'Auto_Fix', 'SedonaReports'

GO
```

If successful, your users should be back in order and the database accessible.

# Q and A

### Can I restore a backup onto a different version of SQL Server? What snags might I hit?

You can restore to a different version of SQL Server, but you can only restore upwards. In other words, you can restore from 2000 to 2005 or from 2005 to 2008R2 or from 2008 to 2012, but you can never restore in the reverse direction. Each version of SQL Server makes modifications to the binary of the database and its storage. Microsoft doesn't go back in time and rewrite the previous versions to support these changes.

When restoring a database to a newer version of SQL Server, it is possible to hit incompatibilities within the database. The safest approach is to run Microsoft's Upgrade Advisor (a free tool available for each version of SQL Server) on the database you wish to migrate, in order to ensure that it's ready, and then take a backup and restore it to the new instance (don't do it the other way round i.e. attempt the restore then run the advisor).

After the restore, the database will be running in the compatibility mode that it had prior to the upgrade. This means that it will be set to support the functionality of the version of SQL Server from which you migrated.

Finally, not all upgrades are possible. Microsoft only allows you to jump forward a maximum of two versions. For example, you cannot restore a SQL Server 2000 database backup to a 2012 instance. You would first need to restore it to 2008, update the compatibility level, perform a backup, and then restore that backup to 2012.

### Why can't I just backup SQL Server's data files with my Windows backup tool? I don't need up-to-the-minute backups.

SQL Server is not like a word processing application. It manages its own files internally in order to guarantee the ACID (Atomic, Consistent, Isolated, Durable) properties of its databases. In a nutshell, to ensure that a transaction completes successfully, SQL Server maintains very tight access control over its files and it modifies theses files as it sees fit.

If you simply copy the data file, ignoring the locks and ignoring the transactions that may be currently in progress, it means that when you attempt to attach that database later you will have a database file that is in an inconsistent state. It will generate errors.

Only if you are in a situation where the database is completely unchanging, ever, could you reliably copy the file and attach it later. However, if there is even a remote possibility of a transaction being open on the system when you copy the file, you're extremely likely to have an unsuccessful backup. The only safe way to ensure no transactions are running when you run your Windows backup tool would be to take the database offline first.

It's is much safer and easier to use SQL Server backups in order to ensure that you have a safe copy of your database that respects the ACID properties of the transactions within it. It is much easier to do a normal database backup.

### If I leave a backup on a network share, surely nobody can read it?

Unless you encrypt the backup file directly, then yes, that is a readable file. If someone got access to the network share, he or she could read the backup file directly using a text editor, or simply copy the file and run a restore on another instance of SQL Server.

It's even possible to extract the schema code or the data from the backup file without restoring it. If you have SQL Data Compare, then the /Export command line parameter will allow you to read all the data out of the backup, in CSV format, by comparing the backup to a blank database, without any password being required. Likewise, a similar process with SQL Compare will allow you to create a compete script to recreate the database schema.

You must take steps to prevent unauthorized access of a backup file. First off, make sure the network location where you're storing backups is only available to a select group of people. Next, only keep the backups there that you actually need. Don't keep extra copies of a backup locally unless you have to.

### My Database is on a SAN. I'm told that the SAN backups are sufficient. Is this true?

It might be true. The critical point is that your SAN must support transactions within the SQL Server system. If so, then it will be are aware of the fact that SQL Server databases have transactions and that these transactions mean that the data in the data files may be incomplete because the process of writing the transaction isn't complete at the point that the backup is taken. SQL Server native backups, of course, always consider this!

Data Domain from EMC, a combination of software and a SAN, certainly offer transactional support, as do other vendors, but check your SAN documentation. You're looking for phrases like 'transaction consistency' or 'transaction aware' or similar. If you don't see them, then I strongly suggest you test a restore of a database before you assume that the SAN backups alone will support your backup requirements. Even if you do see them, don't automatically assume you don't need to take SQL Server database backups. If you need to support point-in-time restores, for example, then you will need to be taking database and log backups.

A SAN backup solution that does support SQL Server will hook into the SQL Server VDI interface and quiesce the database prior to taking the backup. If you were to run such a backup and then look in the SQL error log, there will be messages stating that the IO was frozen on the database.

If you are relying on SAN backup, you're still going to have to check for database consistency, either by running DBCC checks on the live database, or by restoring a database from a SAN backup and running the checks on that. Otherwise, you may just be backing up a corrupted database.

## Components and Concepts

Back up [verb]: Copies the data or log records from a SQL Server database or its transaction log to a backup device, such as a disk, to create a data backup or log backup.

Backup [noun]: A copy of data that can be used to restore and recover the data after a failure. Backups of a database can also be used to restore a copy the database to a new location.

Backup Device: A disk or tape device to which SQL Server backups are written and from which they can be restored. SQL Server backups can also be written to a Windows Azure Blob storage service, and URL format is used to specify the destination and the name of the backup file.. For more information, see SQL Server Backup and Restore with Windows Azure Blob Storage Service.

Backup Media: One or more tapes or disk files to which one or more backup have been written.

Data Backup: A backup of data in a complete database (a database backup), a partial database (a partial backup), or a set of data files or filegroups (a file backup).

Database Backup: A backup of a database. Full database backups represent the whole database at the time the backup finished. Differential database backups contain only changes made to the database since its most recent full database backup.

Differential Backup: A data backup that is based on the latest full backup of a complete or partial database or a set of data files or filegroups (the differential base) and that contains only the data that has changed since that base.

Full Backup: A data backup that contains all the data in a specific database or set of filegroups or files, and also enough log to allow for recovering that data.

Log backup: A backup of transaction logs that includes all log records that were not backed up in a previous log backup. (full recovery model)

Recover: To return a database to a stable and consistent state.

Recovery: A phase of database startup or of a restore with recovery that brings the database into a transaction-consistent state.

Recovery Model: A database property that controls transaction log maintenance on a database. Three recovery models exist: simple, full, and bulk-logged. The recovery model of database determines its backup and restore requirements.

Restore: A multi-phase process that copies all the data and log pages from a specified SQL Server backup to a specified database, and then rolls forward all the transactions that are logged in the backup by applying logged changes to bring the data forward in time.