**Database Backup Strategies**

One of the most important responsibilities of a DBA is the planning, testing and deployment of the backup plan.  The planning of the backup strategy, ironically, starts with the restore strategy that has been determined by the organization.   Two leading questions direct the restore strategy: What is the maximum amount of data loss that can be tolerated, and what is the accepted time frame for the database to be restored in case of a disaster.  The first question involves taking multiple transaction logs backups and the second question involves the use of high availability solutions.  To answer these question, you must ask the why, what, when, where and who for the strategy.

**Why back-up?**

Backups are taken in order to be able to restore the database to its original state just before the disaster and protect against the following reasons:

·         hardware failure

·         malicious damage

·         user-error

·         database corruption

**What is a backup?**

Simply put, it's an image of the database at the time of the full backup.

**There are several different kinds of backups:**

·         Full backup - a complete, page-by-page copy of an entire database including all the data, all the structures and all other objects stored within the database.

·         Differential backups -  a backup of a database that have been modified in any way since the last full backup.

·         Transaction Log - a backup of all the committed transactions currently stored in the transaction log

·         Copy-only backup -  a special-use backup that is independent of the regular sequence of SQL Server backups.

·         File backup - a backup of one or more database files or filegroups

·         Partial backup - contains data from only some of the filegroups in a database

**When to do backups?**

Again, this is a company decision that must be determined by asking the following questions: What is the maximum amount of data loss that can be tolerated, and what is the accepted time frame for the database to be restored in case of a disaster.  The first question will address the use of transaction logs and their frequency, and the second question will involve some type of high availability solution.

**Where should backups be placed?**

Ideally, the backups should be placed on their own drives onto separate servers from the production server, and then using a third party application, copies made to a remote server on a separate site. What should not be tolerated is having backups on the same drive and server as the data and log files.  If the server crashes, we lose all hopes or restoring critical data!

**What needs backing up?**

All user defined databases and system database should be backed up.

**Who should perform backups?**

The person in charge of the backup planning and executing will most likely be the Database Administrator (DBA).  He will coordinate with the upper management, direct and provide valuable advice on the best practices for restoring the database; however, note that on a production server, most of the backups will be automated by using the SQL Agent and jobs.

**Backup Retention Period**

The amount of backups retained is a question determined by the business needs. Most likely, it may involve a month of backups onsite and three months of backups offsite.  But again, this depends upon the organization needs.

**Performing Backups**

The following method illustrates the use of  T-SQL to backup database because it provides a greater and granular control of the process.  However, you can use the SSMS console.

**Full Database backup**

Backup the whole database, which includes parts of transaction log which is needed to recover the database using full backup.

Create a test database

Use master

go

Create database Sales

go

use sales

go

Create table Products

(ProductID int IDENTITY (1,1) Primary Key,

ProductName varchar (100),

Brand varchar (100))

go

insert into Products values ('Bike','Genesis')

insert into Products values ('Hat','Nike')

insert into Products values ('Shoe','Payless')

insert into Products values ('Phone','Apple')

insert into Products values ('Book','Green')

insert into Products values ('Cup','Large')

select \* from Products

BACKUP DATABASE [sales]

TO  DISK = N'c:\fullbackups\sales.bak'

WITH NOINIT,

NAME = N'sales-Full Database Backup',

COMPRESSION,

STATS = 10

GO

declare @backupSetId as int

select @backupSetId = position

from msdb..backupset

where database\_name=N'sales'

and backup\_set\_id=(select max(backup\_set\_id)

from msdb..backupset where database\_name=N'sales' )

if @backupSetId is null

begin

raiserror(N'Verify failed. Backup information for database ''sales'' not found.', 16, 1)

end

RESTORE VERIFYONLY

FROM

DISK = N'c:\fullbackups\sales.bak'

WITH  FILE = @backupSetId

GO

Differential backup

The database must have a full back up in order to take a differential backup; it only backups the changes since last full backup.

BACKUP DATABASE [sales]

TO  DISK = N'c:\fullbackups\sales.bak'

WITH  DIFFERENTIAL ,

NOINIT,

NAME = N'sales-Differential Database Backup',

COMPRESSION,

STATS = 10

GO

declare @backupSetId as int

select @backupSetId = position

from msdb..backupset

where database\_name=N'sales'

and backup\_set\_id=(select max(backup\_set\_id)

from msdb..backupset where database\_name=N'sales' )

if @backupSetId is null

begin

raiserror(N'Verify failed. Backup information for database ''sales'' not found.', 16, 1)

end

RESTORE VERIFYONLY

FROM  DISK = N'c:\fullbackups\sales.bak'

WITH  FILE = @backupSetId

GO

Transaction Log backup

You must backup the transaction log, if SQL Server database uses either FULL or BULK-LOGGED recovery model otherwise transaction log is going to full. Backing up the transaction log truncates the log and user should be able to restore the database to a specific point in time.

BACKUP LOG [sales]

TO  DISK = N'c:\fullbackups\sales.bak'

WITH

NAME = N'sales-Transaction Log  Backup',

COMPRESSION,

STATS = 10

GO

declare @backupSetId as int

select @backupSetId = position

from msdb..backupset

where database\_name=N'sales' and backup\_set\_id=(select max(backup\_set\_id)

from msdb..backupset where database\_name=N'sales' )

if @backupSetId is null

begin

raiserror(N'Verify failed. Backup information for database ''sales'' not found.', 16, 1)

end

RESTORE VERIFYONLY

FROM  DISK = N'c:\fullbackups\sales.bak'

WITH  FILE = @backupSetId

GO

**Recovery models**

There are three recovery models that can be set on each user database which determines the types of backups you’ll use. You set the recovery model via the GUI or use the ALTER DATABASE command:

·         SELECT name, recovery\_model\_desc FROM sys.databases --ß find the recovery model

·         ALTER DATABASE SALES SET RECOVERY FULL

·         ALTER DATABASE SALES SET RECOVERY SIMPLE

·         ALTER DATABASE SALES SET RECOVERY BULK\_LOGGED

Simple

When in this mode, the transaction are removed automatically at each checkpoint within the database and no log backups are possible. Recovery to a point in time is not possible and you could lose substantial amounts of data under simple recovery.  Not advised for production databases that are critical.

Bulk-logged

This recovery model reduces the size of the transaction log by minimally logging some operations such as bulk inserts. The problem is, recovery to a point in time is not possible with this recovery model because any minimally logged operations cannot be restored. This means that bulk-logged has all the overhead of Full Recovery, but effectively works like Simple Recovery. Unless you have specific needs or are willing to perform lots of extra backup operations, it’s not recommended to use bulk-logged recovery.

Full

In full recovery you must run a log backup on a regular basis in order to reclaim log space. Recovery to a point in time is fully supported. For any production system that has data that is vital to the business, full recovery should be used.

Backup History  
The following commands provides information as to the history of the backups in the MSDB database.

Use msdb

go

SELECT \* FROM dbo.backupfile  -- Contains one row for each data or log file that is backed up

SELECT \* FROM dbo.backupmediafamily  -- Contains one row for each media family

SELECT \* FROM dbo.backupmediaset -- Contains one row for each backup media set

SELECT \* FROM dbo.backupset  -- Contains a row for each backup set

SELECT \* FROM dbo.backupfilegroup -- Contains one row for each filegroup in a database at the time of backup

Backup File Information  
It is possible to look at a backup file itself to get information about the backup. The header of the backup stores data like when the backup was created, what type of backup it is, the user who took the backup and all other sorts of information. The basic commands available are:

RESTORE LABELONLY FROM DISK = N'c:\fullbackups\sales.bak'  
RESTORE HEADERONLY FROM DISK = N'c:\fullbackups\sales.bak'  
RESTORE FILELISTONLY FROM DISK = N'c:\fullbackups\sales.bak'

Basic Backup Process  
The following  backup schedule is one that I have used on a production server for critical data bases.

·         Sunday Night: Full Backup with database consistency check (DBCC CHECKDB)

Monday-Saturday: Differential Backup each day  
Midnight-11:45PM: Log Backup every 30 minutes  
What is important to understand is that you frequently test your backups with restores, at least once a month, to ensure that your backups are valid and restorable.  Having backups is useless unless they have been tested well.  When backing up database, starting with SQL version 2008 and up, you have the ability to compress the databases backup, which saves time to backup and restore.  Note that the CPU resource will consume about 20% additional resource, thus, the backup must be done in the off hours.

Example of backup plan using different types of backups

12:00 am - Create an empty database (20 gigs)

12:15 am - Full backup

12:30 am - Insert 100,000 rows (1 gig)

12:45 am - Differential backup (size1 gig after diff)

1:00 am - Insert 100,000 rows (1 gig)

1:15 am - Differential backup (size 2 gigs after diff)

1:30 am - Insert 100,000 rows (1 gig)

1:15 am - Differential backup (size 3 gigs after diff)

1:30 am - Insert 100,000 rows (1 gig)

1:45 am - Transactional log (size 4 gigs after t-log)

2:00 am - Insert 100,000 rows (1 gig)

2:15 am - Transactional log (size 5 gigs after t-log)

Restore the database process

12:15 am - Full backup

1:15 am - Differential backup (size 3 gigs after diff)

1:45 am - Transactional log (size 4 gigs after t-log)

2:15 am - Transactional log (size 5 gigs after t-log)