**These are the different types of backup SQL Server allows:**

1. Full **Backup**. ...
2. Differential **Backup**. ...
3. Transaction Log **Backup**. ...
4. Tail-Log **Backup**. ...
5. File and Filegroup **Backup**. ...
6. Partial **Backup**. ...
7. Copy-Only **Backup**.

### Full Backup

Full database backup takes a copy of the entire database (basically copies all of the pages) including the part of the transaction log file so that the full database can be recovered after a full database backup is restored. Backup can be taken on a backup device which could be a local or network disk or local tape drive. A full database backup represents the database at the time the backup operation finished and allows restoring to the point-in-time when the backup was taken. This is the simplest of backup types and often needed prior to the other backup types. With full database backup, you can restore the complete database in one step and as it includes the transaction log with it, it recovers the database to the time when the backup operation finished minus any uncommitted transactions (uncommitted transactions are rolled back if you are doing recovery with restore, more on recovery in the next article of this series).

USE AdventureWorks2012;

GO

BACKUP DATABASE AdventureWorks2012

TO     DISK = 'D:\backup\AdventureWorks2012.Bak'

          WITH FORMAT, --Specifies whether the media header should be written on the volumes used for this backup operation, overwriting any existing media header and backup sets.

          COMPRESSION, --In SQL Server 2008 Enterprise edition and later versions only, specifies whether compression is performed on this backup, overriding the server-level default.

          MEDIANAME = 'AdventureWorks2012Backups',

          NAME = 'Full database backup of AdventureWorks2012';

GO

### Differential Backup

Differential database backup includes only extents (collections of eight physically contiguous pages) which were changed since the last full database backup. SQL Server looks into the Differential Changed Map (DCM) page to identify all the extents which were changed since the last full database backup and captures only them; hence, often differential database backup is very fast compared to taking a full database backup.

Differential Changed Map (DCM) is a bitmap page that contains a bit for every extent and tracks the extents that have changed since the last full database backup. When you take full database backup, the bit for each extent is reset to 0 and then if there is any change in the extent after that, the bit is changed to 1. During differential backup, SQL Server identifies all the extents for which the bit value is 1 by looking at these DCM pages and hence the length of time that a differential database backup runs is proportional to the number of extents modified (value set to 1) since the last full database backup and not the overall size of the database.

The last full database backup (it’s mandatory to have a full database backup at least once to start with differential database backups), upon which the differential backup is based, is called the base of the differential backup. Generally, the older a differential backup base is, the larger a new differential backup will be, as the difference between the current database and a specific differential backup base increases. The longer the time between differential database backup and its base, the larger the differential backup would be in size - sometimes approaching the size of the base itself. This means a large differential database backup loses the advantages of a faster and smaller backup if the time between a differential database backup and its base grows significantly longer. This also increases the time to restore a database and hence it’s recommended to take a full database backup at set intervals to establish a new differential base. For example, you might consider taking a full database backup on every Sunday followed by a regular series of differential database backups from every Monday through Saturday.

Differential backups are cumulative in nature, which means if you take a full database backup on Sunday followed by differential backups on Monday, Tuesday and Wednesday, and if you want to restore till Wednesday, then you must restore your latest full database backup first followed by restoring the most recent differential backup (i.e. differential database backup taken on Wednesday) only.

USE AdventureWorks2012;

GO

BACKUP DATABASE AdventureWorks2012

TO

          DISK = 'D:\backup\AdventureWorks2012Differential.Bak'

          WITH FORMAT, --Specifies whether the media header should be written on the volumes used for this backup operation, overwriting any existing media header and backup sets.

          COMPRESSION, --In SQL Server 2008 Enterprise edition and later versions only, specifies whether compression is performed on this backup, overriding the server-level default.

          DIFFERENTIAL, --Specifies that backup should consist only of the portions of the database changed since the last full backup

          MEDIANAME = 'AdventureWorks2012Backups',

          NAME = 'Differential database backup of AdventureWorks2012';

GO

### Transaction Log Backup

First of all, transaction log backup is supported only with either Full recovery model or Bulk-logged recovery model and like differential database backup, you must have taken a full database backup as its base. Transaction log backup captures all the transaction log records that have been written after the last full database backup or last transaction log backup.

It’s recommended to take transaction log backups frequently enough to minimize the data loss exposure and to truncate the transaction log so that it does not grow significantly long. Please note, transaction log does not get truncated (even after checkpoint operation) for committed transactions unless you take the transaction log backup. There are certain other cases as well when transaction log will not get truncated, for example if you have setup AlwaysOn, Database mirroring, Transactional replication, Log shipping, Change Data Capture and they are not working correctly. This will cause a delay to the transaction log truncation.

Unlike differential database backup, which captures only the last changed values when a row was changed repetitively, transaction log backup captures all the changes with each repetition.

USE AdventureWorks2012;

GO

BACKUP LOG AdventureWorks2012

TO

          DISK = 'D:\backup\AdventureWorks2012Log.Bak'

GO

### Tail-Log Backup

Before I discuss this type of backup, let me first talk about its need. Assume you have a backup process already in place to take transaction log backup at every one hour. Now consider, what will happen if data files for your database get corrupted at 8:45 AM. With the current process in place, you have taken the last transaction log backup at 8 AM and hence you can restore the database till 8 AM but what about changes made after 8 AM and up to 8:45 AM. This is where Tail-log back might be help for you.

If your database gets corrupted (database is offline, damaged, or missing data files) but the transaction log file is still available, you can take Tail-log backup, which captures any log records that have not yet been backed up (the tail of the log) to prevent data loss and to keep the log chain intact. Backup of Tail-log only succeeds if the transaction log file is not damaged, the database is in a state that supports tail-log backups, and the database does not contain any bulk-logged data changes. But remember, you need to take Tail-log backup (last backup of interest in the recovery plan) before you start restoring your database.

### File and Filegroup Backup

Sometimes when the size of your database grows significantly, it becomes difficult to take a full database backup; when that happens you can consider taking a file or filegroup backup, which change frequently (some files or file groups will have static data or will be read-only and hence it would not be required to frequently take a backup of these files or file groups). This is the copy of the files or file groups of the database.

For a larger database, there might be several files or file groups; some might change frequently and some rarely (i.e. a file or file group containing archived data). In this case you can setup a process to backup the frequently changing file or filegroup. You can choose to backup and restore each individual file or can choose the whole filegroup instead of specifying each constituent file individually. This greatly increases the speed of recovery by letting you restore only damaged files, without worrying about restoring the rest of the database.

File or File group backup and restore comes with additional administrative complexity i.e. maintaining and keeping track of a complete set of these backups can be a time-consuming task that might outweigh the space needed and time required for a full database backup. Also, a table and all of the indexes must be backed up in the same backup and hence if you intend to use file or file group backup you need to plan placing a table and all its indexes on the same file/filegroup.

--Back up filegroups FileGroup1 and FileGroup2

BACKUP DATABASE AdventureWorks2012

   FILEGROUP = 'FileG roup1

',

   FILEGROUP = 'FileGroup2'

   TO DISK = 'D:\backup\AdventureWorks2012FG.Bak';

GO

### Partial Backup

If you have a very large database with a couple of file groups and only a few of them are changing, you can reduce the time to take the backup by backing up only those parts of the database which are changing. A partial backup is similar to a full database backup but takes a backup of only the primary filegroup and every read-write file group (optionally you can include one or more read-only file groups as well). If you take a partial backup of a read-only database it only contains the primary filegroup. Partial backup might be helpful if you want to exclude read-only file groups to reduce overall backup time.

Like differential backup works with full backup, you can take a differential partial backup based on a partial backup as base. As in the case of differential backup, differential partial backup captures all the extent modified since the last partial backup.

### Copy-Only Backup

Copy-Only backup, as its name implies, takes the copy of the database without changing the normal sequence of conventional backup operation. Copy-Only backup can be done at the database level or transaction log level.

As said before, when you take a full database backup, SQL Server resets bits to 0 for all the extents in the DCM page, likewise with transaction log backup, it resets how the next transaction log backup should happen but when you take Copy-Only full backup, it does not interfere with normal backup\restore operations and does not reset bits to 0 for all the extents in DCM page at the same time; when you take Copy-only transaction log backup, it preserves the existing log archive point and the transaction log is never truncated after a Copy-Only log backup.

Copy-Only database backup and restore might be helpful if you want to setup development\testing\QA environment without impacting the regular backup and restore rhythm setup in production.