BACK UP AND RECOVERY

Types of Backups:

1) **Full Database Backup:**
A full database backup captures the entire database content.

A full database backup copies all the pages from a database onto a backup device/media set.

Full backup contains:
a) The path/location and structures of all the MDF, NDF and LDF and their sizes

b) Copy all the pages to the media set. Entire Active Log and Last LSN number

Contents of the Active Portion of the Log file (i.e. Log Records) are backed up by Full Backup.

Command:
```
BACKUP DATABASE DBName TO DISK='F:\DBName.bak'
```

2) **Differential Backup:**
Differential backup tracks and backups up all the changes that have happenned from Last Full Backup till the backup completion time.

All differential backups are cumulative. Even if one backup in the mid of the week is lost, the recent backup can be used.

During differential backup, SQL Server refers DCM(Differential Changed Map) Page to track all the extents modified and copying the pages inside them into the BAK file. Also the active log is backed up as part of Differential backup.

Command:
```
backup database DBName to disk=N'F:\DBName_Diff.bak' WITH DIFFERENTIAL
```

3) Transaction Log Backups:
Backups all the transaction log records from the last FULL backup (or) last Transaction Log backup. Log backups are Incremental.

Command:
```
backup log DBName to disk=N'F:\DBName1.trn'
```

4) File and Filegroup Backup:
File and Filegroup backups are individual backups of Files/Filegroups in a database.

F&FG backups are helpful to devise backup strategies for Very Large DBs.

Individual Files and Filegroups are backed up using this method, also tables can be placed in FG and can be backed up individually.

Command:
```
backup database FGTest
```

```
FILEGROUP='Primary' to disk=N'F:\FGTest.bak'

backup database FGTest
FILE='FGTest' to disk=N'F:\FGTest1.bak'
```

5) Striped Backup:
During space constraints backups can be split into different locations.

Command:

```
BACKUP DATABASE DBname TO
DISK=N'F:\DBNamePart1.bak',
DISK=N'E:\DBNamePart2.bak'
```

Maximum 64 striped backupsets are possible.

6) Mirror Backup:
For additional safety to backup files it is possible to create mirrored copies of database backups. If one file gets corrupted we have another as safety measure.

```
BACKUP DATABASE DBname
TO DISK=N'F:\DBName_Mirr1.bak' MIRROR
TO DISK=N'E:\DBName_Mirr2.bak' WITH FORMAT
```

Mirrored backupsets cannot reside with other NonMirrored backupsets, hence WITH FORMAT option is mandatory to mention.

Maximum 4 mirrored sets are possible.

7) COPY_ONLY Backups:
Backups that are taken without affecting the Backup Cycle. COPY_ONLY. This concept was implemented from SQL Server 2005 onwards.

Command:

```
BACKUP DATABASE DBName TO
DISK=N'f:\DBName.bak'
WITH COPY_ONLY
```

Note:
1) COPY_ONLY backups are possible for FULL and TLogs Only.

2) If COPY_ONLY, DIFFERENTIAL is mentioned in the backup command then COPY_ONLY is ignored.

```
Backup database DBName to disk=N'f:\DBName.bak' WITH DIFFERENTIAL,COPY_ONLY
```

3) COPY_ONLY backup cannot be taken through GUI and is possible through commands only (2005 Only). In 2008 it is possible to take COPY_ONLY through GUI.

Summary:-
COPY_ONLY full backup will not disturb the differential backup chain.

COPY_ONLY log backup will not disturb the log chain.

Command:

```
backup log Friendship to
disk=N'D:\Backup\Friendship_CopyOnly_Log1.trn'
WITH COPY_ONLY
```

8) Tail Log Backup:
Tail log backup is a Tlog backup, that can be attempted during a crash situation. Tail log also takes backup of Active Log.

Command:

```
BACKUP LOG DBName TO
DISK=N'f:\DBName_Tail.trn'
WITH NO_TRUNCATE
```

Tail log can fail if Log file gets corrupted.

9) Partial Backup:
Partial backups are useful whenever we want to exclude read-only filegroups. A partial does not contain all the filegroups.

Command:

```
BACKUP DATABASE DBName READ_WRITE_FILEGROUPS,
FG1,FG2 TO DISK=N'F:\DBName_PBKP.bak'
```

10) Partial Differential Backup:
Is a differential backup for Partial backup.

```
backup database DBName READ_WRITE_FILEGROUPS,READONLY1,READONLY2 to
disk=N'F:\DBName_PBKP.bak' WITH DIFFERENTIAL
```

11) Cold Backups in SQL Server:
If backup is being taken for one specific database then.

```
ALTER DATABASE DBName SET OFFLINE
WITH ROLLBACK IMMEDIATE
```

Copy MDF & LDF files of this database in Data Directory. This files backup taken can be attached to another database if required to restore.

Parameters for CUI based backups:

1) NOINIT/INIT:
NOINIT appends backupsets into the media set.

INIT overwrites existing backupsets into the media set.

2) NOFORMAT/FORMAT:
NOFORMAT will keep the existing media set.

FORMAT will create a new media set and overwrites all the contents in the file.

3) NOSKIP/SKIP:
NOSKIP instructs the BACKUP statement to check the expiration date of all backup sets on the media before allowing them to be overwritten. This is the default behavior.

SKIP ignores the backup set expiration.

4) BLOCKSIZE
Specifies the physical block size, in bytes. The supported sizes are 512, 1024, 2048, 4096, 8192, 16384, 32768, and 65536 (64 KB) bytes. The default is 65536 for tape devices and 512 otherwise.

BLOCKSIZE is selected automatically based on the device chosen.

5) BUFFERCOUNT
Specifies the total number of I/O buffers to be used for the backup operation. You can specify any positive integer.

The total space used by the buffers is determined by: BUFFERCOUNT * MAXTRANSFERSIZE

6) MAXTRANSFERSIZE
Specifies the largest unit of transfer in bytes to be used between SQL Server and the backup media. The possible values are multiples of 65536 bytes (64 KB) ranging up to 4194304 bytes (4 MB).

7) STATS [=percentage]

Displays a message each time another percentage completes, and is used to gauge progress. If percentage is omitted, SQL Server displays a message after each 10 percent is completed.

Types of Recovery

1) Restart Recovery:
Everytime an instance is restarted/started the consistency of all the databases including master,model,msdb and tempdb is checked. This process is an internal operation and initiated just to keep the entire instance clean and with integrity.

Taking database offline/online involves restart recovery for that database.

2) Restore Recovery
As per backup strategy whenever a restore is started and recovery is done per backup sequence. This entire process of recovery initiated manually is called as Restore recovery.

Recovery Models:

Recovery models are designed to control transaction log maintenance.

Recovery models control the behavior of the log file.

The recovery models in SQL Server are Full, Bulk Logged and Simple.

Full Recovery Model:
Every Transaction in Full Recovery Model is logged into the Transaction Log file.

Growth of the Tlog is very fast and drastic in this recovery model, so it is a recommended Practice to take regular Tlog Backups.

When tlog backup is taken all the committed log records will be backed up and truncated from Transaction Log file.

Pros:
1)Full Recovery Model is recommended in Production Environment.
2)Point-in-time recovery is possible only in this recovery model.
3)Almost no data loss

Cons:
1) Growth of log file is faster and requires additional storage
2) Costly due to disk space requirements

Bulk Logged Recovery Model:
Every transaction is logged into the transaction log file except Bulk Insert statements which are minimally logged.

Minimally logged operation means only few details are available in the log file like LSN number, Page Details and Transaction ID when compared to Full Recovery Model.

1) Point-in-time recovery not possible. Possible if no BULK INSERT commands are run.

2) Log file growth is not too large, but some amount of growth would be there.

3) Used especially during BULK INSERT statements ONLY.

4) It is always recommended practice to take a FULL backup pre and post changing recovery model to BULK LOGGED.

Simple Recovery Model:
Every transaction is logged into the Transaction Log file. But at regular intervals Transaction Log file gets Truncated.

Pros:
1) This recovery model is used in Non-Critical environments (Testing and Development)

2) Automatic Maintenance of Log file, log file growth can be controlled.

3) In simple recovery model, Log File Truncation occurs when ever Checkpoint occurs.

Cons:
1)It is not possible to take a Tlog backup in Simple Recovery Model.

2)Point-in-time recovery is not possible

Restore Commands are:

Full Backup Restore:
```
restore database Test from disk=N'F:\Test.bak' WITH NORECOVERY
```

Differential Backup Restore:
```
restore database Test from disk=N'F:\Test_Diff.bak' WITH NORECOVERY
```

Tlog Backup Restore:
```
restore log Test from disk=N'F:\Test_Tlog1.trn' WITH NORECOVERY
```

Point-in-time Recovery:
```
restore log Test from disk=N'F:\Test.trn' WITH NORECOVERY
STOPAT '06-21-2012 06:50:00AM'
```

Tlog restore with Recovery:
```
restore log Test from disk=N'F:\Test_Tlog2.trn' WITH NORECOVERY
restore database Test with recovery
```

F&FG Restore:
```
restore database NewDB123
FILEGROUP='Primary',
Filegroup='Secondary'
from disk='c:\dummy\NewDB_FG.bak'
with recovery
```

Types of Restores:
1) Restore an entire database from a full database backup (a complete restore).

2) Restore part of a database (a partial restore/Piece Meal Restore).

3) Restore specific files or filegroups to a database (a file restore).

4) Restore specific pages to a database (a page restore).

5) Restore a transaction log onto a database (a transaction log restore).

6) Revert a database to the point in time captured by a database snapshot.

RESTORE SCENARIOS

1) Complete Database Restore:
A complete database restore involves restoring a full database backup followed by restoring and recovering a differential backup.

Backup Steps:

```
backup database Friendship
to disk=N'D:\Backup\Friendship_Full_041420150659.bak'

backup database Friendship
to disk=N'D:\Backup\Friendship_Diff1_041420150701.bak'
with differential

backup log Friendship
to disk=N'D:\Backup\Friendship_Log1_041420150702.trn'
```

---:Restore Steps:---

```
RESTORE DATABASE [Friendship]
FROM  DISK = N'D:\Backup\Friendship_Full_041420150659.bak'  WITH MOVE N'Friendship' TO
N'D:\Backup\Friendship.mdf',
MOVE N'Friendship2' TO N'D:\Backup\Friendship2.ndf',
MOVE N'Friendship3' TO N'D:\Backup\Friendship3.ndf',
MOVE N'Friendship_log' TO N'D:\Backup\Friendship.ldf',  NORECOVERY,  REPLACE,  STATS = 10
GO

RESTORE DATABASE [Friendship]
FROM DISK = N'D:\Backup\Friendship_Diff1_041420150701.bak'
WITH NORECOVERY,  STATS = 10
GO

RESTORE LOG [Friendship]
FROM DISK=N'D:\Backup\Friendship_Log1_041420150702.trn'
WITH NORECOVERY, STATS = 10
GO

RESTORE DATABASE [Friendship] WITH RECOVERY
```

2) File/Filegroup Restore:
Restore one or more damaged read-only files, without restoring the entire database. File restore is available only if the database has at least one read-only filegroup

Create a database with two filegroups and create two tables in each filegroup.

```
create table Table1
(Sno int, sname varchar(50))
on FG1

create table Table2
(Sno int, sname varchar(50))
on FG2
```

Take Full Backup and Differential and Tlog with some inserts between them.

```
BACKUP DATABASE FFGRESTORE
TO DISK=N'D:\BACKUP\FFGRESTORE_FULL_041520150653.BAK'
```

```
insert into FFGRestore.dbo.Table1 select * from FFGRestore.dbo.Table1
insert into FFGRestore.dbo.Table2 select * from FFGRestore.dbo.Table2

BACKUP DATABASE FFGRESTORE
TO DISK=N'D:\BACKUP\FFGRESTORE_DIFF1_041520150656.BAK'
WITH DIFFERENTIAL

insert into FFGRestore.dbo.Table1 select * from FFGRestore.dbo.Table1
insert into FFGRestore.dbo.Table2 select * from FFGRestore.dbo.Table2

BACKUP LOG FFGRESTORE
TO DISK=N'D:\BACKUP\FFGRESTORE_TLOG1_041520150657.TRN'
```

----:RESTORE COMMANDS:----

```
RESTORE DATABASE FFGRESTORE
FILEGROUP='PRIMARY'
FROM DISK=N'D:\BACKUP\FFGRESTORE_FULL_041520150653.BAK'
WITH REPLACE,NORECOVERY

RESTORE DATABASE FFGRESTORE
FILEGROUP='FG1'
FROM DISK=N'D:\BACKUP\FFGRESTORE_FULL_041520150653.BAK'
WITH NORECOVERY

RESTORE DATABASE FFGRESTORE
FILEGROUP='FG2'
FROM DISK=N'D:\BACKUP\FFGRESTORE_FULL_041520150653.BAK'
WITH NORECOVERY

RESTORE DATABASE FFGRESTORE
FILEGROUP='PRIMARY',
FILEGROUP='FG1',
FILEGROUP='FG2'
FROM DISK=N'D:\BACKUP\FFGRESTORE_DIFF1_041520150656.BAK'
WITH NORECOVERY

RESTORE LOG FFGRESTORE
FILEGROUP='PRIMARY'
FROM DISK=N'D:\BACKUP\FFGRESTORE_TLOG1_041520150657.TRN'
WITH NORECOVERY

RESTORE DATABASE FFGRESTORE WITH RECOVERY
```

3) Page Restore:
Restores one or more damaged pages. An unbroken chain of log backups must be available, up to the current log file, and they must all be applied to bring the page up to date with the current log file

4) Point-in-time restore

```
create table Table1 (sno int, sname varchar(50))
insert into Table1 values (1,'Pit1')
insert into Table1 values (2,'Pit2')
insert into Table1 values (3,'Pit3')
insert into Table1 values (4,'Pit4')

--08:03AM (Full Backup)
backup database PIT to disk=N'D:\Backup\PIT_Full.bak'

-- 4 Inserts after Full
insert into Table1 values (5,'Pit5')
insert into Table1 values (6,'Pit6')
insert into Table1 values (7,'Pit7')
insert into Table1 values (8,'Pit8')

--08:04AM (Diff Backup)
backup database PIT to disk=N'D:\Backup\PIT_Diff.bak'
with differential

-- 4 Inserts after Diff
insert into Table1 values (9,'Pit9')
insert into Table1 values (10,'Pit10')
insert into Table1 values (11,'Pit11')
insert into Table1 values (12,'Pit12')

--08:05AM (TLog1 Backup)
backup log PIT to disk=N'D:\Backup\PIT_Tlog1.trn'

-- 4 Inserts after Tlog1
insert into Table1 values (13,'Pit13')
insert into Table1 values (14,'Pit14')
insert into Table1 values (15,'Pit15')
insert into Table1 values (16,'Pit16')

--08:06AM (TLog2 Backup)
backup log PIT to disk=N'D:\Backup\PIT_Tlog2.trn'

-- 4 Inserts after Tlog1
insert into Table1 values (17,'Pit17')
insert into Table1 values (18,'Pit18')
insert into Table1 values (19,'Pit19')
insert into Table1 values (20,'Pit20')

begin tran
insert into Table1 values (21,'Pit21')
insert into Table1 values (22,'Pit22')
insert into Table1 values (23,'Pit23')
insert into Table1 values (24,'Pit24')

--08:07AM DB Corrupt so Tail Log Backup
```

```
backup log PIT to disk=N'D:\Backup\PIT_Tail.trn'\
WITH NO_TRUNCATE
```

Restore Commands (Point Of Failure)

```
restore database PIT
from disk=N'D:\Backup\PIT_Full.bak'
with replace,norecovery

restore database PIT
from disk=N'D:\Backup\PIT_Diff.bak'
with norecovery

restore log PIT
from disk=N'D:\Backup\PIT_Tlog1.trn'
with norecovery

restore log PIT
from disk=N'D:\Backup\PIT_Tlog2.trn'
with norecovery

restore log PIT
from disk=N'D:\Backup\PIT_Tail.trn'

--For Point in Time recovery mention STOPAT parameter with appropriate timeline.

RESTORE LOG [PIT] FROM
DISK = N'D:\Backup\PIT_Tlog2.trn'
WITH STOPAT = N'2015-04-15T08:06:00'
GO
```

5) Piecemeal Restore

```
RESTORE DATABASE FFGRESTORE
FILEGROUP='PRIMARY'
FROM DISK=N'D:\BACKUP\FFGRESTORE_FULL_041520150653.BAK'
WITH REPLACE,NORECOVERY,PARTIAL

RESTORE DATABASE FFGRESTORE
FILEGROUP='FG1'
FROM DISK=N'D:\BACKUP\FFGRESTORE_FULL_041520150653.BAK'
WITH NORECOVERY

RESTORE DATABASE FFGRESTORE
FILEGROUP='PRIMARY',
FILEGROUP='FG1'
FROM DISK=N'D:\BACKUP\FFGRESTORE_DIFF1_041520150656.BAK'
WITH NORECOVERY

RESTORE LOG FFGRESTORE
FROM DISK=N'D:\BACKUP\FFGRESTORE_TLOG1_041520150657.TRN'
```

```
WITH RECOVERY
```

---Restoring FG2 seperately.

```
RESTORE DATABASE FFGRESTORE
FILEGROUP='FG2'
FROM DISK=N'D:\BACKUP\FFGRESTORE_FULL_041520150653.BAK'
WITH NORECOVERY

RESTORE DATABASE FFGRESTORE
FILEGROUP='FG2'
FROM DISK=N'D:\BACKUP\FFGRESTORE_DIFF1_041520150656.BAK'
WITH RECOVERY

RESTORE LOG FFGRESTORE
FROM DISK=N'D:\BACKUP\FFGRESTORE_TLOG1_041520150657.TRN'
WITH RECOVERY
```

Steps for Page_Restore

```
create database TestPageLevelRestore

use TestPageLevelRestore
go
create table Shift (sno int,sname varchar(50))

insert into Shift values (1,'Hello')
insert into Shift values (2,'Hi')
insert into Shift values (3,'How Are You')

BACKUP DATABASE TestPageLevelRestore
TO DISK=N'C:\OurBackups\TestPageLevelRestore_Full.bak'

insert into Shift values (4,'Hell')
insert into Shift values (5,'H')
insert into Shift values (6,'How')

BACKUP DATABASE TestPageLevelRestore
TO DISK=N'C:\OurBackups\TestPageLevelRestore_Diff.bak'
WITH DIFFERENTIAL

insert into Shift values (7,'H1')
insert into Shift values (8,'H2')
insert into Shift values (9,'H3')

BACKUP LOG TestPageLevelRestore
TO DISK=N'C:\OurBackups\TestPageLevelRestore_Tlog.trn'

Use TestPageLevelRestore
Select * from sys.indexes where OBJECT_NAME(object_id)='Shift'
```

```
DBCC TRACEON(3604,-1)
DBCC IND('TestPageLevelRestore','Shift',0)

DBCC PAGE(8,1,276,3)

ALTER DATABASE TestPageLevelRestore SET OFFLINE
WITH ROLLBACK IMMEDIATE

--Corrupt the pages using HexEditor or DBCC WRITEPAGE commands
--When corrupting page, multiply Page Number*8192. Press Ctrl+G to go page number in decimal format.

ALTER DATABASE TestPageLevelRestore SET ONLINE

use TestPageLevelRestore
go
select * from Shift
```

```
Msg 8928, Level 16, State 1, Line 1
Object ID 2105058535, index ID 0, partition ID 72057594038779904, alloc unit ID 72057594039697408 (type In-row data): Page (1:153) could not be processed.  See other errors for details.
Msg 8939, Level 16, State 98, Line 1
Table error: Object ID 2105058535, index ID 0, partition ID 72057594038779904, alloc unit ID 72057594039697408 (type In-row data), page (1:153). Test (IS_OFF (BUF_IOERR, pBUF->bstat)) failed. Values are 12716041 and -4.
```

Run DBCC CHECKDB, to identify if just a single page is corrupt or if multiple pages are corrupt.

```
--Take Backup with NORECOVERY, this will put database in RESTORING State.
BACKUP LOG TestPageLevelRestore
TO DISK=N'C:\OurBackups\TestPageLevelRestore_NoRecovery.trn'
```

--Start the restore sequence.

```
restore database TestPageLevelRestore
PAGE='1:277'
from disk=N'C:\OurBackups\TestPageLevelRestore_Full.bak'
WITH NORECOVERY

restore database TestPageLevelRestore
from disk=N'C:\OurBackups\TestPageLevelRestore_Diff.bak'
WITH NORECOVERY

restore log TestPageLevelRestore
from disk=N'C:\OurBackups\TestPageLevelRestore_Tlog.trn'
WITH NORECOVERY
restore log TestPageLevelRestore
from disk=N'C:\OurBackups\TestPageLevelRestore_NoRecovery.trn'
WITH RECOVERY
```