

Backing Up and Restoring a Database

NET3000

Database and SQL

Week 12

Kambiz Ghazinour

Recovery Models

- Common recovery models are:
 - Full Recovery Model
 - All transactions are logged.
 - Simple Recovery Model
 - No transaction log backups.
 - Bulk-Logged Recovery Model
 - Bulk-logged operations are not logged.

Backup Destination Media Types

- Disk
 - Local drive
 - Network drive (UNC)
- Tape
- Named pipe

System Databases

- master
 - Simple recovery model
 - Backup after any significant changes are made to SQL Server
- model
 - Full recovery model
- Msdb
 - Simple recovery model
 - Best practice: alter to full recovery model
- tempdb
 - Simple recovery model
 - Reset after each restart of SQL Server; therefore, don't need to backup

Backing Up a Database

- Maintaining a duplicate copy of data that you can recover in the event of data loss is critical.
- SQL Server provides a variety of features that you can use to accomplish this goal.

Performing Full Backups

- The purpose of a *full database backup* is to capture all the data that is stored in the database.
- The *backup engine* accomplishes this task by extracting every extent in the database that is allocated to an object.
- You can then use a full backup by itself to re-create the entire database.
- Notice that this backup method is always available, regardless of the *recovery model* you configure for a database.

Full Backup

- The backup engine is configured to perform a backup as quickly as possible while using a minimum of resources.
- When you initiate a backup, the backup engine writes pages to the backup device without regard to the order of pages.
- Because the backup is not concerned with the precise ordering of pages, SQL Server can open multiple threads to write data as fast as it can be accepted by the media.
- The only limiting factor in the backup speed is how fast data can be written to a device.

Inconsistency?

- Because a backup is not instantaneous and can occur while users are connected to the database and issuing queries, logical inconsistency in the database is a possibility.
- If a page of data were written to the backup media and then modified by another request, for example, restoring this backup would place the database in an inconsistent state.
- SQL Server, however, does NOT allow this to happen.

Backup Steps

1. Lock the database, blocking all transactions.
2. Place a mark in the transaction log.
3. Release the database lock.
4. Back up all pages in the database.
5. Lock the database, blocking all transactions.
6. Place a mark in the transaction log.
7. Release the database lock.
8. Extract transactions between the two log marks and append to the backup.
 - This process ensures that the database is completely consistent as of the time that the backup completes.

Syntax

BACKUP DATABASE <database name> TO DISK =
'<directory>\<filename>' WITH INIT

- The TO clause specifies the backup device to send the backup to, which can be the name of a logical backup device that is created, or you can specify an explicit path to either DISK or TAPE.
- The INIT parameter tells SQL Server to overwrite anything in the backup device that might already exist before starting the backup operation.

Performing Differential Backups

- A differential backup captures all the extents that have changed since the last full backup.
- The main purpose is to reduce the number of transaction log backups that need to be restored.
- You use a differential backup along with a full backup.
- If a full backup does not exist, you cannot create a differential backup.
- You can perform a differential backup of a database no matter what recovery model is specified for the database.
 - Just like a full backup.

Differential Backup

- A differential backup is NOT an incremental backup (!).
- An incremental backup captures any changes since the previous incremental backup.
- Therefore, restoring an incremental backup requires all other incremental backups.
- A differential backup always captures every extent that has changed since the last full backup.
- So each differential backup contains everything that any previous differential backup taken after a full backup contains.

Example of a Differential Backup

- Suppose that a full backup occurs at midnight, with differential backups taken every four hours during the day.
- The differential backup at 04:00 contains all extents that have changed since midnight.
- The differential backup at 08:00 contains all extents that have changed since midnight.
- And the noon differential backup contains all extents that have changed since midnight.

Syntax

BACKUP DATABASE <database name> TO DISK =
'<directory>\<filename>' WITH DIFFERENTIAL

- Note the use of the DIFFERENTIAL parameter

Transaction Log Backups

- You can perform transaction log (t-log) backups only for databases you have configured to use the Full or Bulk-Logged recovery model and that have not yet had a minimally logged transaction executed.
- T-log backups are also allowed only after a full backup has been performed.
- A t-log backup contains only a subset of data and requires that you have at least a full backup to recover the database.

Syntax

```
BACKUP LOG <database name> TO DISK =  
    '<directory>\<filename>' WITH INIT
```


Performing Filegroup Backups

- Filegroup backups provide an alternative backup strategy to full backups.
- Instead of backing up the entire database, you can perform a filegroup backup to back up individual filegroups within the database.
- The starting point for a filegroup backup strategy must include a backup of all filegroups within the database so that you can reassemble all the filegroups within that database.
- Configure the database in either Full or Bulk-Logged model so that you can perform a filegroup backup that is read/write.
- To restore, you can then use filegroup, differential, and transaction log backups.

Syntax

BACKUP DATABASE <database name> FILEGROUP =
'<filegroup name>' TO DISK = '<directory>\<filename>'

- To take a differential backup:

BACKUP DATABASE <database name> FILEGROUP =
'<filegroup name>' TO DISK = '<directory>\<filename>'
WITH DIFFERENTIAL

Summary of Restore Steps

1. Find the backup device.
 - ▮ Find the right backup set
2. Verify the backup device.
 - ▮ Establish that your backup is usable
3. Restore the database form the backup device.
 - ▮ RESTORE DATABASE command

Restoring a Database

- The ability to restore a backup determines how quickly your databases can resume responding to business requests after damage occurs.

Restoring a Full Backup

- Most restore operations begin by re-creating the database at a specific point in time and then applying subsequent backups to bring the database up to a particular point in time
- This process begins with a restore of a full backup.
- The restore operation must place the pages back into the database in sequential order.
- This process ensures a completely coherent database when finished.
- It also requires additional time.
 - Generally requires about 30% more time to complete than the backup being restored took to generate.

Syntax

RESTORE DATABASE <database name> FROM DISK =
'<directory>\<filename>' WITH REPLACE

- The REPLACE option overwrite the existing database
- Other important clauses:
 - WITH RECOVERY
 - WITH NORECOVERY

WITH RECOVERY

- The database is brought online and the database is allowed to accept transactions.
- No further restore operations are allowed after you recover a database.

WITH NORECOVERY

- The database or filegroup state remains set to RESTORING.
- In this state, you can restore additional backups, such as differential and transaction log backups, to apply any changes that have occurred since the full backup was taken.

Restoring a Differential Backup

- To restore a differential backup, you must first restore a full backup while ensuring that the database is NOT recovered.
- The most recent differential backup is then applied to the database.
- Filegroup differential restore
 - The process for restoring a filegroup differential backup is very similar to restoring a differential backup.
 - It requires that you execute a full filegroup restore first and that you do not recover the filegroup.

Example

```
RESTORE DATABASE PUBS FROM DISK =  
    'C:\DEMO\BACKUP\PUBSFULL.BAK' WITH NORECOVERY
```

```
RESTORE DATABASE PUBS FROM DISK =  
    'C:\DEMO\BACKUP\PUBSDIFF.BAK' WITH RECOVERY
```

- The first command restores the full backup, leaving the database unrecovered.
- The second command applies to a differential backup and then recovers the database.

Restoring a Transaction Log Backup

- You use transaction log backups to roll a database forward to a specific point in time.
- This point in time is generally the last operation that was executed against the database, but you can select a different point.
- Transaction logs can be applied to a full backup or after a differential backup has been restored.
- A transaction log backup contains a sequence of transactions.
- Transactions can also be explicitly named by placing a mark in the transaction log.
- The exact time a transaction was executed is logged along with the change that was made.

Example

--Restore sequence using a full, differential, and transaction log backup.

--Full

```
RESTORE DATABASE AdventureWorks FILEGROUP = 'FG1' FROM DISK = 'C:\TEST\AWFG1.BAK'  
    WITH NORECOVERY
```

--Differential

```
RESTORE DATABASE AdventureWorks FROM DISK = 'C:\TEST\FG1DIFF1.BAK' WITH  
    NORECOVERY
```

--Transaction log

```
RESTORE LOG AdventureWorks FROM DISK = 'C:\TEST\AW2.TRN' WITH RECOVERY
```

Another Example

--Restore sequence using a full backup and multiple transaction log backups.

--Full

```
RESTORE DATABASE AdventureWorks FILEGROUP = 'FG1' FROM DISK = 'C:\TEST\AWFG1.BAK'  
    WITH NORECOVERY
```

--Transaction log

```
RESTORE LOG AdventureWorks FROM DISK = 'C:\TEST\AW1.TRN' WITH NORECOVERY
```

```
RESTORE LOG AdventureWorks FROM DISK = 'C:\TEST\AW2.TRN' WITH RECOVERY
```

Validating a Backup

- You have performed several backups, but how do you know the backups are stable?
- The only way to guarantee that a backup is usable is to restore it and verify all the data.
- This process can be very time-consuming and is rarely practical.
- However, SQL Server provides a way to verify the integrity of a backup.
- Although not the same as actually restoring a database, it provides a very thorough check of the backup integrity.

Syntax

RESTORE VERIFYONLY FROM <backup device> [, ...n]

- When you execute this command, SQL Server checks the media header to ensure that it is intact.
- It then verifies the backup checksum, reads the internal page chains, and recalculates the backup checksum for comparison.
- A variety of checks are preformed to ensure that the backup is intact.
- However, SQL Server does not check the actual data structures in the backup.

Questions?