Chapter-8 Backup and Restoration with MS SQL Server

Backup and restoration are essential aspects of database management in MS SQL Server to protect data from loss, ensure disaster recovery, and maintain the integrity and availability of the database. MS SQL Server provides a variety of methods to back up and restore data, allowing flexibility in how backups are performed and when they are restored.

1. Types of Backups in MS SQL Server

MS SQL Server offers different types of backups, each serving a distinct purpose. The main types of backups are:

- **Full Backup**
- **Differential Backup**
- **Transaction Log Backup**
- **File and Filegroup Backups**
- **Copy-only Backup**

1.1 Full Backup

A **Full Backup** is a complete backup of the entire database, including all objects (tables, indexes, stored procedures) and data. A full backup provides a baseline, and subsequent backups (like differential or transaction log backups) can be used to restore the database to that point in time.

Syntax:

```
BACKUP DATABASE database_name
TO DISK = 'C:\Backup\database_name.bak';
```

Example:

```
BACKUP DATABASE AdventureWorks
TO DISK = 'C:\Backup\AdventureWorksFull.bak';
```

This creates a full backup of the AdventureWorks database and saves it to the specified location.

1.2 Differential Backup

A **Differential Backup** captures only the changes made to the database since the last full backup. It is smaller in size compared to a full backup and is used to reduce the backup window and storage space needed.

Syntax:

```
BACKUP DATABASE database_name
TO DISK = 'C:\Backup\database_name_diff.bak'
WITH DIFFERENTIAL;
```

Example:

```
BACKUP DATABASE AdventureWorks
TO DISK = 'C:\Backup\AdventureWorksDiff.bak'
WITH DIFFERENTIAL;
```

This creates a differential backup of the AdventureWorks database. It only includes the data modified since the last full backup.

1.3 Transaction Log Backup

A **Transaction Log Backup** captures the changes made to the database since the last transaction log backup. This is essential for point-in-time recovery, allowing you to restore the database to a specific point in time.

Syntax:

```
BACKUP LOG database_name
TO DISK = 'C:\Backup\database_name_log.trn';
```

Example:

```
BACKUP LOG AdventureWorks
TO DISK = 'C:\Backup\AdventureWorksLog.trn';
```

This creates a transaction log backup for the AdventureWorks database, which includes all the changes since the last transaction log backup.

1.4 File and Filegroup Backups

In large databases, you may choose to back up individual files or filegroups, rather than the entire database. This is particularly useful when working with large databases spread across multiple files or filegroups.

Syntax:

```
BACKUP DATABASE database_name
FILEGROUP = 'filegroup_name'
TO DISK = 'C:\Backup\database_name_filegroup.bak';
```

Example:

```
BACKUP DATABASE AdventureWorks
FILEGROUP = 'PRIMARY'
TO DISK = 'C:\Backup\AdventureWorksPrimaryFilegroup.bak';
```

This creates a backup of the PRIMARY filegroup for the AdventureWorks database.

1.5 Copy-only Backup

A **Copy-only Backup** is a special type of backup that does not affect the sequence of transaction log backups. It is typically used when you want to take a backup for copying purposes without disrupting the regular backup chain.

Syntax:

```
BACKUP DATABASE database_name
TO DISK = 'C:\Backup\database_name_copyonly.bak'
WITH COPY_ONLY;
```

Example:

```
BACKUP DATABASE AdventureWorks
TO DISK = 'C:\Backup\AdventureWorksCopyOnly.bak'
WITH COPY_ONLY;
```

This creates a copy-only backup of the AdventureWorks database.

2. Backup with Compression

MS SQL Server supports backup compression, which reduces the size of backup files and the time it takes to create backups. Backup compression is available in SQL Server 2008 and later versions.

Syntax:

```
BACKUP DATABASE database_name
TO DISK = 'C:\Backup\database_name_compressed.bak'
WITH COMPRESSION;
```

Example:

```
BACKUP DATABASE AdventureWorks
TO DISK = 'C:\Backup\AdventureWorksCompressed.bak'
WITH COMPRESSION;
```

This creates a compressed backup of the AdventureWorks database.

---

3. Restoring Databases in MS SQL Server

Restoring a database in MS SQL Server is the process of bringing a database back to a specific state using one or more backup files. The restoration process allows you to recover from data loss, corruption, or other issues that require a rollback to a previous state.

3.1 Restoring a Full Backup

To restore a full backup, you can use the RESTORE command. This is typically the first step in the restoration process, especially when you are restoring from a full backup.

Syntax:

```
RESTORE DATABASE database_name
FROM DISK = 'C:\Backup\database_name.bak';
```

Example:

```
RESTORE DATABASE AdventureWorks
FROM DISK = 'C:\Backup\AdventureWorksFull.bak';
```

This restores the AdventureWorks database from the full backup file located at C:\Backup\AdventureWorksFull.bak.

3.2 Restoring a Differential Backup

After restoring a full backup, you can restore a differential backup to capture the changes made since the last full backup.

Syntax:

```
RESTORE DATABASE database_name
FROM DISK = 'C:\Backup\database_name_diff.bak'
WITH NORECOVERY;
```

Example:

```
RESTORE DATABASE AdventureWorks
FROM DISK = 'C:\Backup\AdventureWorksDiff.bak'
WITH NORECOVERY;
```

The WITH NORECOVERY option is used because you need to restore further transaction log backups afterward.

3.3 Restoring Transaction Log Backups

After restoring the full and differential backups, you can apply the transaction log backups to bring the database to a point in time.

Syntax:

```
RESTORE LOG database_name
FROM DISK = 'C:\Backup\database_name_log.trn'
WITH NORECOVERY;
```

Example:

```
RESTORE LOG AdventureWorks
FROM DISK = 'C:\Backup\AdventureWorksLog.trn'
WITH NORECOVERY;
```

After restoring the transaction log backup, you can restore more log backups if necessary.

3.4 Restoring to a Specific Point in Time

MS SQL Server allows you to restore a database to a specific point in time using transaction log backups. You can specify a date and time in the RESTORE statement to recover the database to a particular moment.

Syntax:

```
RESTORE DATABASE database_name
FROM DISK = 'C:\Backup\database_name.bak'
WITH STOPAT = 'YYYY-MM-DD HH:MI:SS';
```

Example:

```
RESTORE DATABASE AdventureWorks
FROM DISK = 'C:\Backup\AdventureWorksFull.bak'
WITH STOPAT = '2024-12-24 12:00:00';
```

This restores the AdventureWorks database to the state it was in at 12:00 PM on December 24, 2024.

3.5 Restoring a Database with Recovery Options

You can use the WITH RECOVERY option to bring the database online after the restoration process is complete. It is the default option if you do not specify WITH NORECOVERY.

Syntax:

```
RESTORE DATABASE database_name
FROM DISK = 'C:\Backup\database_name.bak'
WITH RECOVERY;
```

Example:

```
RESTORE DATABASE AdventureWorks
FROM DISK = 'C:\Backup\AdventureWorksFull.bak'
WITH RECOVERY;
```

This brings the AdventureWorks database online after the restoration is complete.

---

4. Backup and Restore with SQL Server Management Studio (SSMS)

SQL Server Management Studio (SSMS) provides a graphical interface for performing backup and restoration operations. Here's a quick overview of how to perform backups and restores using SSMS:

- **Backup**: Right-click the database > Tasks > Back Up. Choose the backup type (full, differential, transaction log), specify the destination, and start the backup.
- **Restore**: Right-click the database > Tasks > Restore > Database. Choose the backup file(s), select options (e.g., recovery, point-in-time recovery), and perform the restore.