Database Security

CT069-3-3

Backup and Recovery

Risks of Losing Data

- Data and database can be corrupted or lost due
  - to accidental
    - Unreliable hardware, software programs
    - human error
  - intentional activities
    - Data is always the target and remain vulnerable for various threats
- Data loss can incur cost to repair or redo work, may lead to temporary shutdown of operation or even permanent closure business

Backup

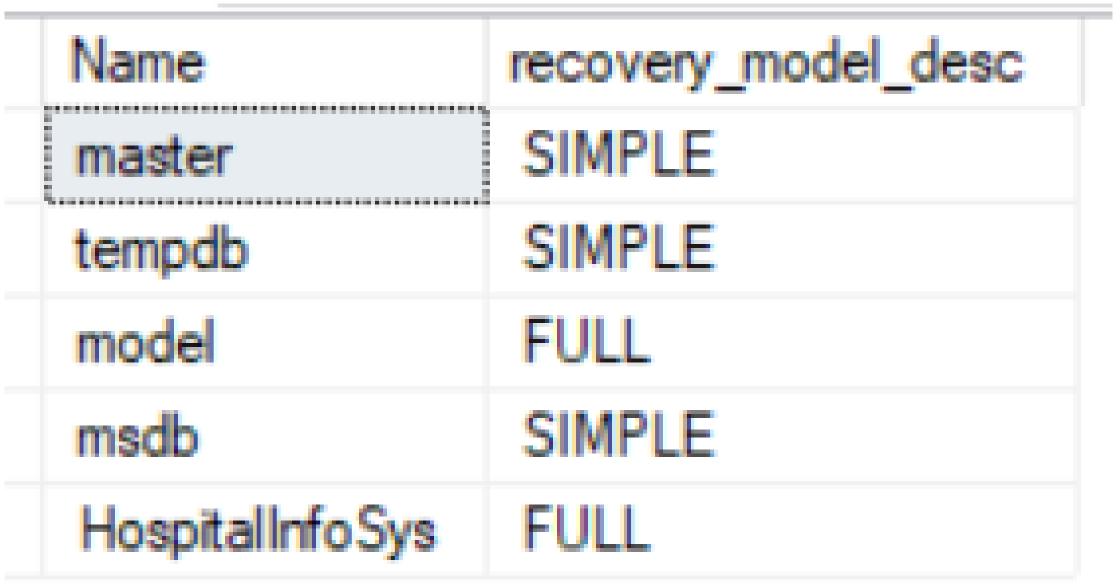
- Database Backups are an essential part of every good disaster recovery and security strategy to protect against the losses/corruption
- In order to recover a corrupted database, we need to restore a database to a point in time before the corruption occurred.
- That recovery process is done using one or more database backups

MS-SQL Server Backup

- SQL Server backup and restore operations occur within the context of the recovery model of the database.
- Recovery models are designed to control transaction log maintenance.
- A *recovery model* is a database property that controls how transactions are logged, whether the transaction log requires (and allows) backing up, and what kinds of restore operations are available.
- Three recovery models exist: **simple, full, and bulk-logged.** Typically, a database uses the full recovery model or simple recovery model.
- A database can be switched to another recovery model at any time.

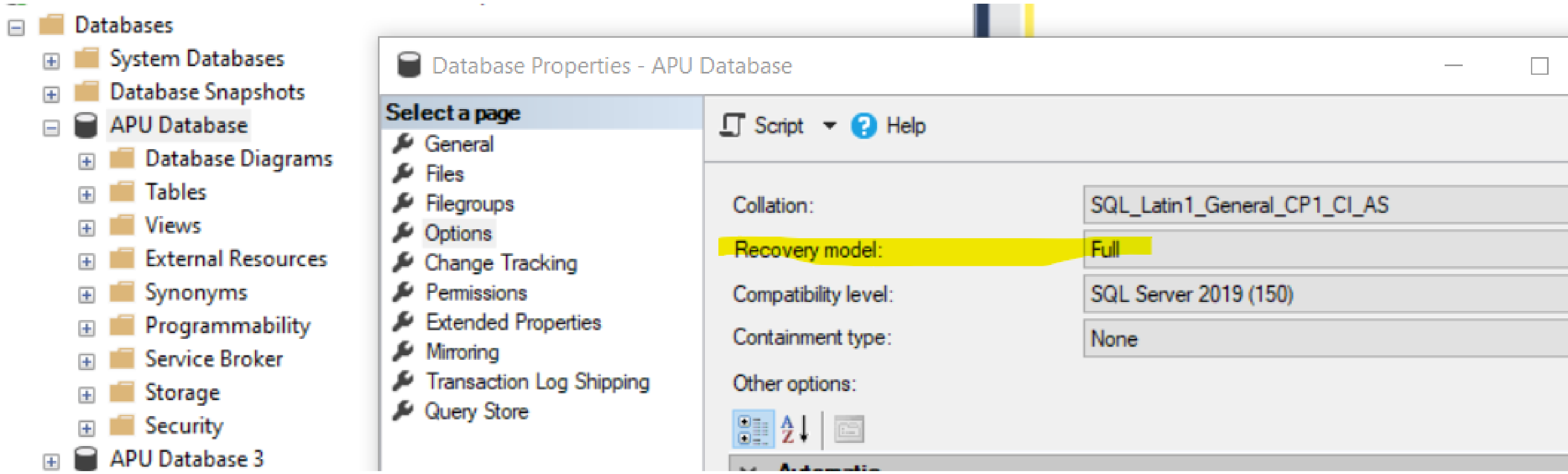
Recovery Model

```sql
Select Name, recovery_model_desc
From sys.databases
```



Name	recovery_model_desc
master	SIMPLE
tempdb	SIMPLE
model	FULL
msdb	SIMPLE
HospitalInfoSys	FULL

Recovery Model



Databases
- System Databases
- Database Snapshots
- APU Database
  - Database Diagrams
  - Tables
  - Views
  - External Resources
  - Synonyms
  - Programmability
  - Service Broker
  - Storage
  - Security
- APU Database 3

Database Properties - APU Database

Select a page
- General
- Files
- Filegroups
- Options
- Change Tracking
- Permissions
- Extended Properties
- Mirroring
- Transaction Log Shipping
- Query Store

Script Help

Collation:	SQL_Latin1_General_CP1_CI_AS
Recovery model:	Full
Compatibility level:	SQL Server 2019 (150)
Containment type:	None

Other options:

Simple Recovery Model

- The most basic of recovery models
- Uses less disk space for transaction logs because the transaction logs records will be removed automatically for completed transactions (when data is saved to the database data files)
- However, simple recovery model doesn’t support using transaction log backups
- With the simple recovery model, you can only perform full and differential backups.
- Doesn’t support point in time restores

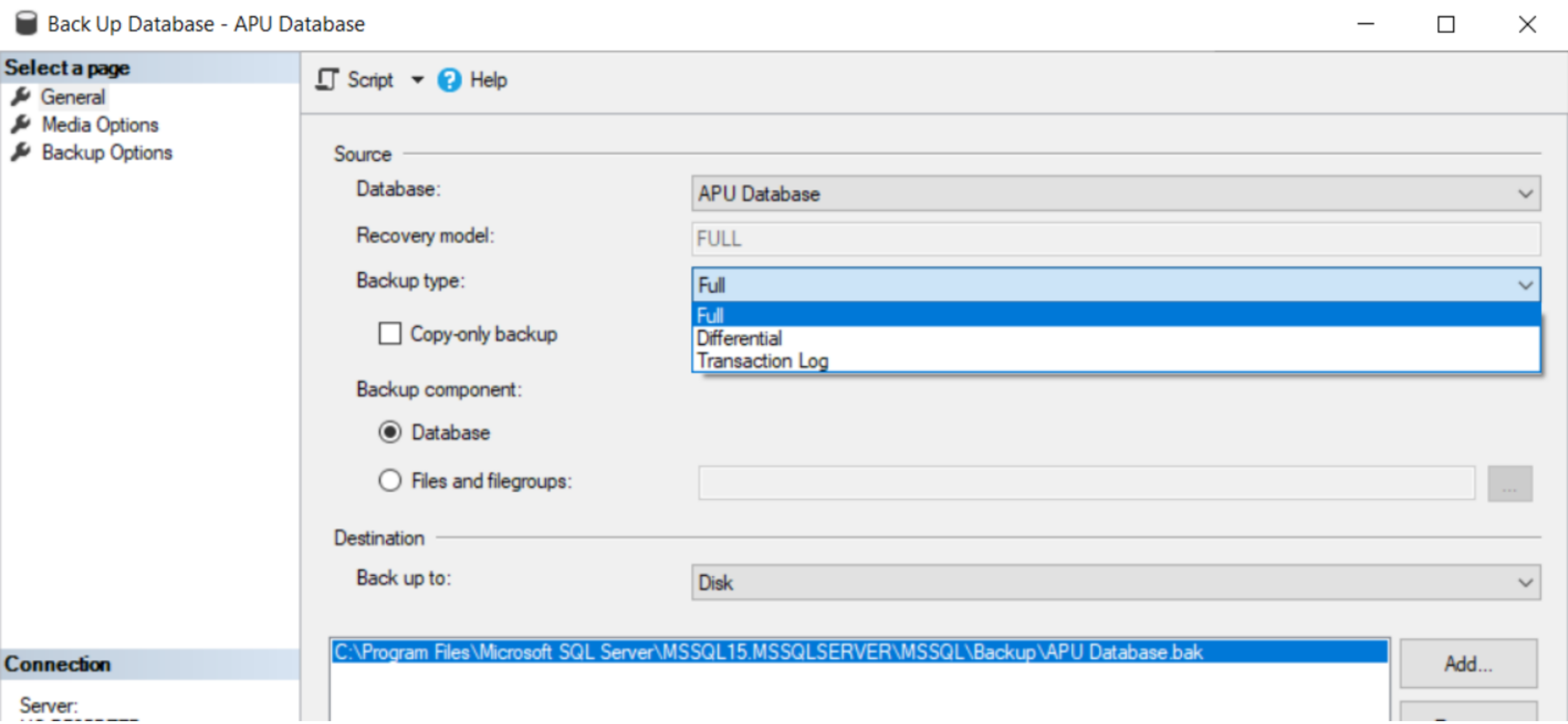
Full Recovery Model

- Transaction log file size increases as the log records are NOT removed for completed transactions (when data is saved to the database data files)
- The transaction log records stay in the transaction log until a log backup is performed. When a transaction log backup is performed against a database that is in full recovery mode, the log records are written to the transaction log backup, and the completed transaction log records are removed from the transaction log.
- Support point in time restores

Types of Backups

- Full
- Differential
- Transactional Log(T-Log)
- Copy-Only
- Mirror
- File and FileGroup

SQL Backup



Back Up Database - APU Database

Source

Database: APU Database

Recovery model: FULL

Backup type: Full

- Full
- Differential
- Transaction Log

☐ Copy-only backup

Backup component:

(●) Database

( ) Files and filegroups:

Destination

Back up to: Disk

C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\Backup\APU Database.bak

Full Backup

- It is the foundation for other types of backup
- Backs up everything (complete copy)
- Stores all the objects of the database: Tables, procedures, functions, views, indexes etc
- With a full backup, we can easily restore a database in the same form as it was at the time of the backup
- A full backup creates a complete backup of the database as well as part of the **transaction log**, so the database can be recovered

Full Backup

- Full backup allows for the simplest form of database restoration, since all of the contents are contained in one single backup.
- A full backup must be done at least once before any of the other types of backups can be run—this is the foundation for every other kind of backup.

SQL Statement

```sql
BACKUP DATABASE BackupRestoreTest
TO DISK = 'C:\Temp\BRT.bak'
   WITH INIT,
   NAME = 'BackupRestoreTest';
GO
```

Differential Backup

- A differential database backup contains all changes that have been made *since the last full backup*.
- It is typically faster as only the newly added data are added after full backup is taken
- Differential backups are cumulative

Differential Backup

```
BACKUP DATABASE BackupRestoreTest
TO DISK = 'C:\Temp\BRT_Diff.bak’
   WITH DIFFERENTIAL
   NAME = 'BackupRestoreTest Diff';
GO
```

Transaction Log Backup

- Every SQL Server database has a transaction log that records all transactions and the database modifications made by each transaction
- The transaction log is a critical component of the database. If there is a system failure, you will need that log to bring your database back to a consistent state
- Transaction log backup backs up the transaction log
- A transaction log backup contains all log records that have not been included in the last transaction log backup.
- It allows the database to be recovered to a specific point in time.
- This means that the transaction log backups are incremental

Transaction Log Backup

```
BACKUP LOG BackupRestoreTest
TO DISK = 'C:\Temp\BRT.log’
   WITH CHECKSUM
   NAME = 'BackupRestoreTest Log';
GO
```

Copy-Only Backup

- A *copy-only backup* is a SQL Server backup that is independent of the sequence of conventional SQL Server backups.
- For example, let’s say we need to provide a copy of our database to the app development team for their development work - Copy-only backups serve this purpose as it will not impact the server performance

```
BACKUP DATABASE Test1
TO DISK = 'D:\SQLBackup\Test1.BAK’
WITH COPY_ONLY
```

Mirror Backup

- Creates multiple copies of the backup files into different locations
- Typically used to protect ourselves from being dependent only one backup
- So, if one backup file is corrupted, we can access the backups from another location

```
BACKUP DATABASE Test1
TO DISK = 'D:\SQLBackup\Test1.BAK'
MIRROR TO DISK =  'E:\SQLBackup\Test1.BAK'
MIRROR TO DISK =  'F:\SQLBackup\Test1.BAK'
WITH FORMAT
```

File and File Group

- If we have more than one file for our database, we have the option of backing up only certain files in the database

Strategy

- A typical backup strategy is a combination of
  - A **full backup** usually taken once a day during off peak hours as it is a resource intensive operation which may slow down the server
  - A **differential backup** usually taken a few times day and uses increasing resource as it gets further way from full backup
  - A **transaction log backup** usually taken every hour or so

Recovery / Restore

- Restore steps depends on the backup strategy
- If you want to restore the database to a specific point in time, you need restore :-
  - a full backup
  - and recent differential
  - and all the corresponding transaction log records

  which are necessary to build the database up to that specific point, or to a point very close to the desired point in time, just before the occurrence of the accident that resulted in the data loss.

Sample

Assumption/Stategy

- Full backup - once a day at 12 midnite
- Incremental backup - 6 hourly - 6am, 12 noon, 6pm
- Transaction log - hourly -
- Assume current time is 2.20pm - db is corrupted

Steps

1. restore the latest full backup taken on 12 midnite
2. restore the latest incr backup taken on 12 noon
3. restore all the trans log backups taken after the latest incr backup (on 1pm, 2pm)

Note: that means 20 mins data is lost

Restore Steps

```
RESTORE DATABASE [BackupRestoreTest]
FROM DISK = 'C:\Temp\BRT.bak'
WITH NORECOVERY

RESTORE DATABASE [BackupRestoreTest]
FROM DISK = 'C:\Temp\BRT_Diff.bak'
WITH NORECOVERY

RESTORE LOG [BackupRestoreTest]
FROM DISK = 'C:\Temp\BRT.log'
```

Note: Use the key word WITH NORECOVERY for all except the last restore command

Restoring An Encrypted Database To Another Instance

- Before we can restore an encrypted database to another instance, we need to first backup the certificate with its private key

```
Use master
Go
BACKUP CERTIFICATE CertforEncrDBBRTest
TO FILE = N'C:\Temp\CertforEncrDBBRTest.cert'
WITH PRIVATE KEY (
    FILE = N'C:\Temp\CertforEncrDBBRTest.key',
ENCRYPTION BY PASSWORD = 'QwErTy12345!@#$%'
);
Go
```

Restoring An Encrypted Database in Another Instance

- Then we need to create the master key and recreate the certificate in the target instance
- And finally run the restore command

```
USE MASTER
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD =
'MyBackUpPassword$$12345’

USE MASTER
GO
Create CERTIFICATE CertforEncrDBBRTest
From FILE = N'C:\Temp\CertforEncrDBBRTest.cert'
WITH PRIVATE KEY (
    FILE = N'C:\Temp\CertforEncrDBBRTest.key',
DECRYPTION BY PASSWORD = 'QwErTy12345!@#$%'
);
```

Encrypting Backup

- Backups of databases can also be encrypted
- This ensures that if any backup (of databases that were not encrypted) are stolen, it CANNOT be used by unauthorized persons

Encrypting Backup

```
USE MASTER
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'MyBackUpPassword$$12345'

CREATE CERTIFICATE MyCert WITH SUBJECT = 'MyBackupCertificate'

BACKUP DATABASE [Test1]
TO DISK = 'C:\Temp\Test1.bak'
WITH
   ENCRYPTION
   (
   ALGORITHM = AES_256,
   SERVER CERTIFICATE = MyCert
   )
```

System Database Backups

- System databases are an essential component of SQL Server engine for the functioning of a server instance.
- The system databases are critical as it stores meta-data of the user-defined databases.
- It must be backed up after every significant update as we do it for user-defined databases.
- The system databases that you must always back up include
  - msdb - It is used by SQL Server Agent for job management, and it also stores a history of every backup and restore operation
  - master - The database is used to record all of the system level information
  - Model - It’s a template for all databases.

Q & A