Restoring SQL Server Databases

Module Overview

- Understanding the Restore Process
- Restoring Databases
- Advanced Restore Scenarios
- Point-in-Time Recovery

Understanding the Restore Process

- Phases of the Restore Process
- Types of Restores
- Preparations for Restoring Backups
- Discussion: Determining Required Backups to Restore

Phases of the Restore Process

- The restore process of a SQL Server database consists of three phases:

Phase	Description
Data copy	Creates files and copies data to the files
Redo	Applies committed transactions from restored log entries
Undo	Rolls back transactions that were uncommitted at the recovery point

- Redo and undo are also known as recovery

Types of Restores

- Complete database restores:
  - Simple recovery model
  - Full recovery model
- System database restore
- Advanced restores:
  - File or filegroup restore
  - Piecemeal restore
  - Encrypted backup restore
  - Page restore

Preparations for Restoring Backups

- Perform a tail-log backup if using full or bulk-logged recovery model
- Identify the backups to restore:
  - Last full, file, or filegroup backup
  - Last differential backup, if exists
  - Log backups if using full or bulk-logged recovery model

Discussion: Determining Required Backups to Restore

- Backup schedule:
  - Full database backups Saturday 22:00
  - Differential backups Monday, Tuesday, Thursday, Friday at 22:00
  - Log backups every hour (on the hour) from 09:00 to 18:00
- Failure occurs at Thursday at 10:30
- What restore process should you follow?

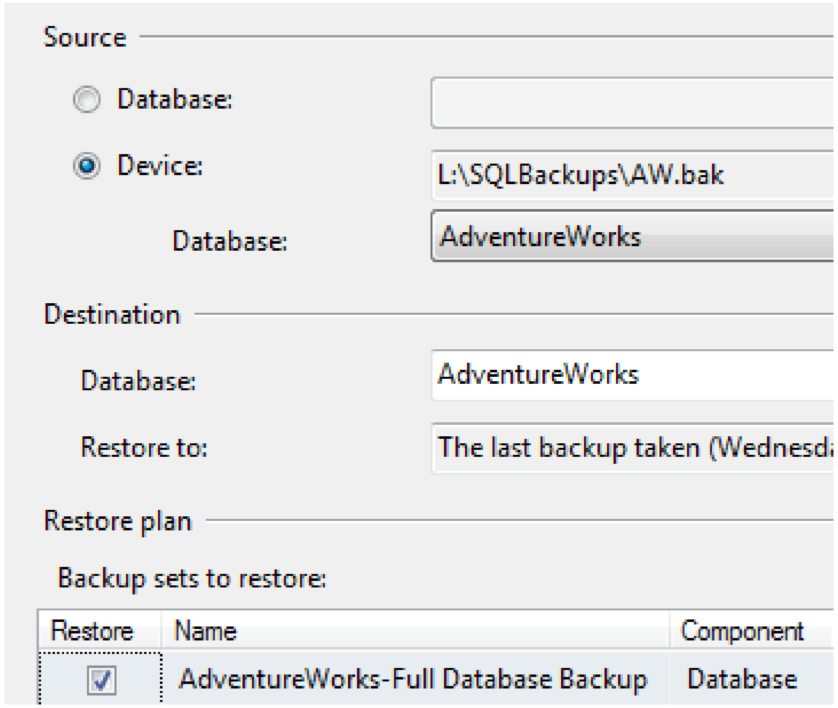
Restoring Databases

- Restoring a Full Database Backup
- Restoring a Differential Backup
- Restoring Transaction Log Backups
- Demonstration: Restoring Databases

Restoring a Full Database Backup

- Restore databases in SQL Server Management Studio, or use the RESTORE DATABASE statement
  - Use WITH REPLACE to overwrite an existing database
  - Use WITH MOVE to relocate database files

```
RESTORE DATABASE AdventureWorks
FROM DISK = 'R:\Backups\AW.bak';
```



Source

Database:

Device: L:\SQLBackups\AW.bak

Database: AdventureWorks

Destination

Database: AdventureWorks

Restore to: The last backup taken (Wednesd

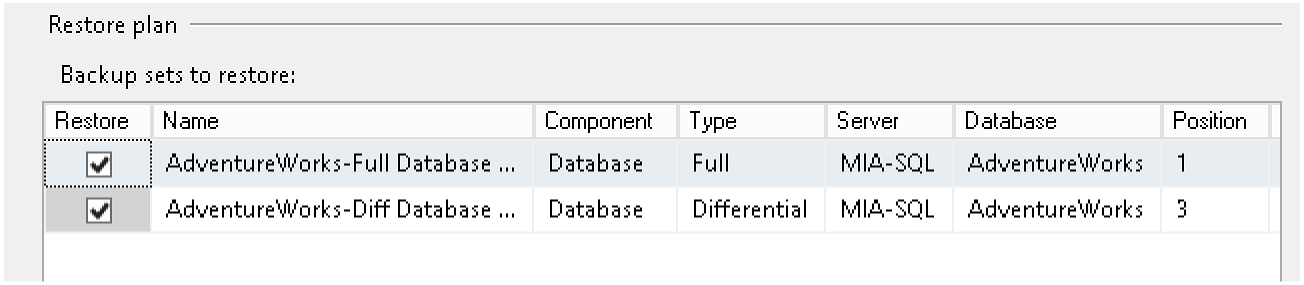
Restore plan

Backup sets to restore:

Restore	Name	Component
✓	AdventureWorks-Full Database Backup	Database

Restoring a Differential Backup

- Restore the latest full database backup WITH NORECOVERY
- Restore the latest differential backup WITH RECOVERY



Restore plan

Backup sets to restore:

Restore	Name	Component	Type	Server	Database	Position
✓	AdventureWorks-Full Database ...	Database	Full	MIA-SQL	AdventureWorks	1
✓	AdventureWorks-Diff Database ...	Database	Differential	MIA-SQL	AdventureWorks	3

```
RESTORE DATABASE AdventureWorks
FROM DISK = 'R:\Backups\AW.bak'
WITH FILE = 1, NORECOVERY;

RESTORE DATABASE AdventureWorks
FROM DISK = 'R:\Backups\AW.bak'
WITH FILE = 3,  RECOVERY;
```

Restoring Transaction Log Backups

- Restore transaction logs by using the RESTORE LOG statement
- Restore the log chain chronologically
- Use NORECOVERY for all but the last backup
- Use RECOVERY for the last backup (often the tail-log backup)

```
-- Restore last full and differential database backups...
-- Restore planned log backups
RESTORE LOG AdventureWorks FROM DISK = 'R:\Backups\AW.bak'
WITH FILE = 5,  NORECOVERY;


-- Restore tail-log backup
RESTORE LOG AdventureWorks FROM DISK = 'R:\Backups\AW-
TailLog.bak'
WITH RECOVERY;
```

Demonstration: Restoring Databases

In this demonstration, you will see how to:

- Create a tail-log backup
- Restore a database

Recovering System Databases

System database	Description
master	Backup required: Yes Recovery model: Simple Restore using single user mode
model	Backup required: Yes Recovery model: User configurable Restore using –T3608 trace flag
msdb	Backup required: Yes Recovery model: Simple (default) Restore like any user database
tempdb /resource	No backups can be performed tempdb is created during instance startup Restore resource using file restore or setup

Point-in-Time Recovery

- Overview of Point-in-Time Recovery
- STOPAT Option
- STOPATMARK Option
- Performing a Point-in-Time Recovery by Using SQL Server Management Studio
- Demonstration: Performing a Point-in-Time Recovery

Overview of Point-in-Time Recovery

- Enables recovery of a database up to any arbitrary point in time that is contained in the transaction log backups

- Point in time can be defined by:
  - A datetime value
  - A named transaction

- Database must be in FULL recovery model

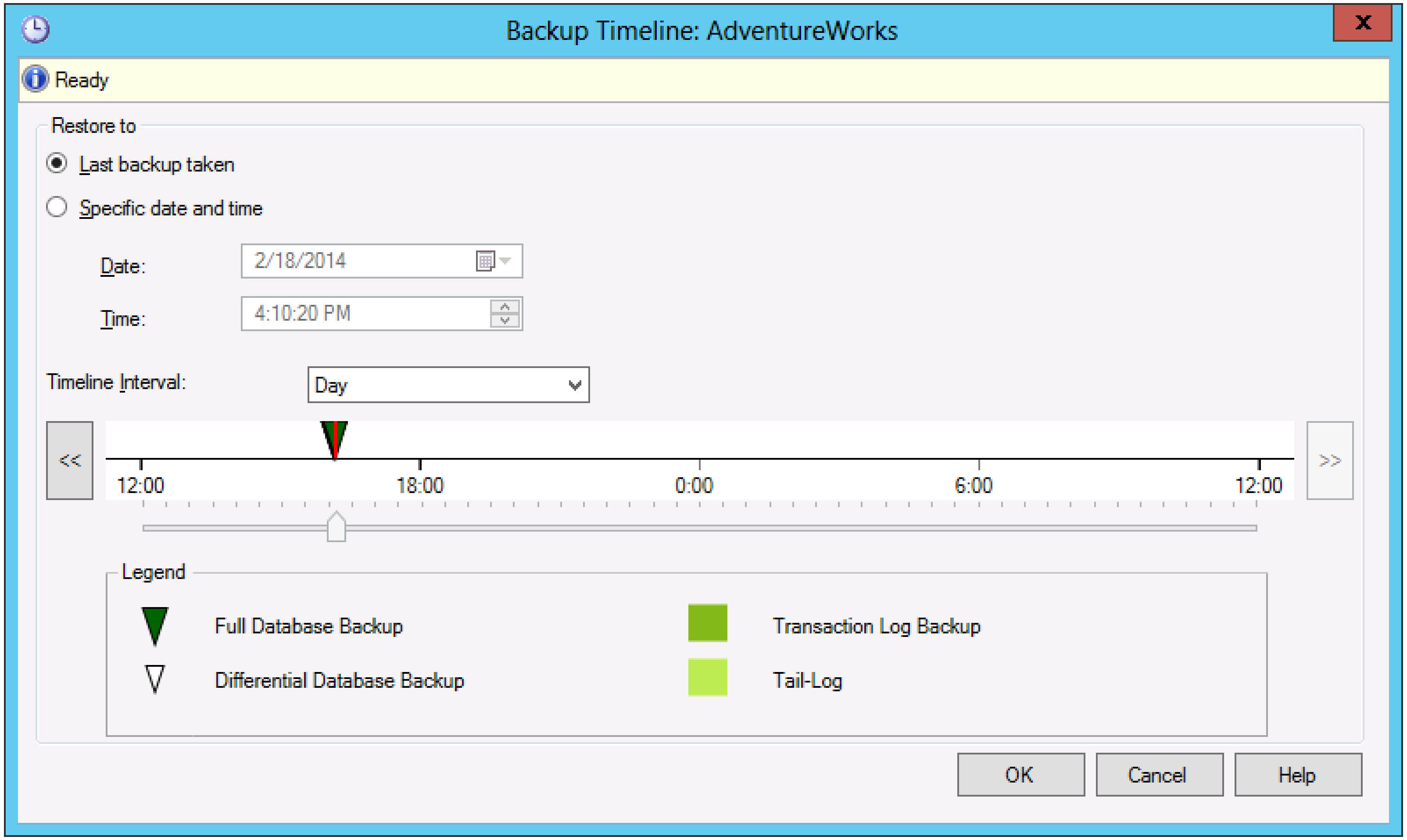
STOPAT Option

- Provide the STOPAT and WITH RECOVERY options as part of all RESTORE statements in the sequence:
  - No need to know in which transaction log backup the requested point in time resides
  - If the point in time is after the time included in the backup, a warning will be issued and the database will not be recovered after the restore completes
  - If the point in time is before the time included in the backup, the RESTORE statement fails
  - If the point in time provided is within the time frame of the backup, the database is recovered up to that point

STOPATMARK Option

- Transactions marked using:
  - BEGIN TRAN <name> WITH MARK <description>
- Restore has two related options:
  - STOPATMARK rolls forward to the mark and includes the marked transaction in the roll forward
  - STOPBEFOREMARK rolls forward to the mark and excludes the marked transaction from the roll forward
- If the mark is not present in the transaction log backup, the backup is restored, but the database is not recovered

Performing a Point-in-Time Recovery by Using SQL Server Management Studio



Backup Timeline: AdventureWorks

Ready

Restore to

(●) Last backup taken

( ) Specific date and time

Date: 2/18/2014

Time: 4:10:20 PM

Timeline Interval: Day

Legend

▼	Full Database Backup	🟩	Transaction Log Backup
▽	Differential Database Backup	🟩	Tail-Log

Demonstration: Performing a Point-in-Time Recovery

In this demonstration you will see how to:

- Perform a point-in-time recovery