

How to Move SQL Database to Another Drive: A Practical Workflow



Goal:

planned relocation of data, log, tempdb, and FILESTREAM files to another drive



Why:

capacity relief, IO tiering, data/log separation, moving off decommissioned volumes



Phases:

inventory, preparation, offline file move, verification, cleanup, documentation



Safety:

back up first; never move files while the database is online

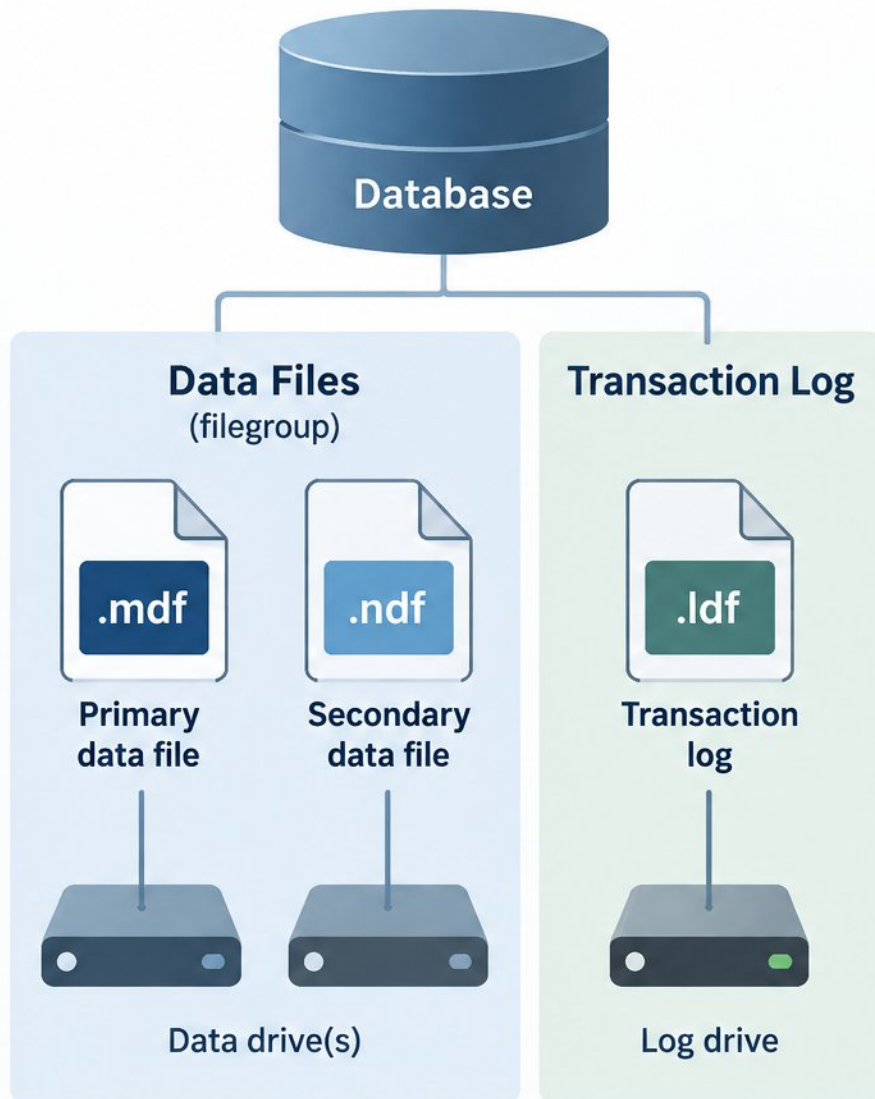


Prerequisites:

SQL Server 2016+ on Windows; basic T-SQL, SSMS, and file system skills



Database File Types and Why Placement Matters



Primary data file (.mdf):
startup info; exactly one per database



Secondary data files (.ndf):
optional; spread data across disks and filegroups



Transaction log (.ldf):
recovery information; at least one per database



Keep data and log on separate drives;
prefer NTFS over FAT



Logical name (T-SQL) vs. physical path (OS):
moves change only the physical path



Scope: planned relocation within one instance —
not backup/restore or shrink

Pre-Flight Inventory and Baseline Capture

1



Enumerate each user database: logical names, physical paths, sizes, growth, filegroups

2



List system databases separately — master, model, msdb, tempdb, Resource

3



Take a verified full backup; note recovery model and log chain status

4



Check destination: free space, drive letter, folder, no filename collisions

5



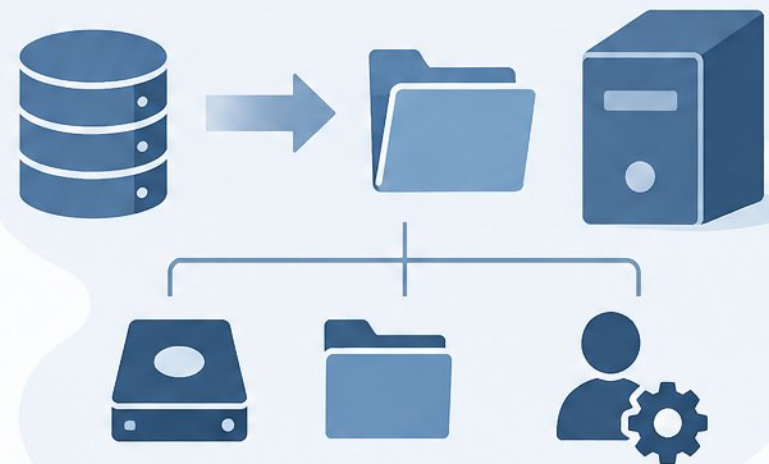
Confirm the service account has full control on the new location

6



Document dependencies and define a rollback plan plus change window

Logical Name	Physical Path	Size	Growth	Filegroups
_____	_____	_____	_____	_____
_____	_____	_____	_____	_____
_____	_____	_____	_____	_____
_____	_____	_____	_____	_____
_____	_____	_____	_____	_____

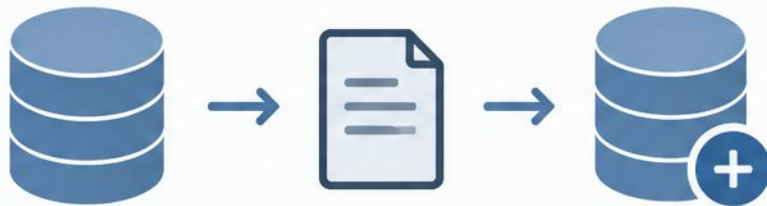


Choosing the Move Method: Detach/Attach vs. ALTER DATABASE MODIFY FILE

TWO OFFLINE APPROACHES. DIFFERENT SCENARIOS. MAKE THE RIGHT CHOICE.

Detach / Attach

Move with new paths, then reattach

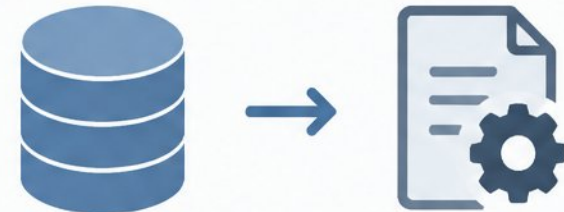


CREATE DATABASE ... FOR ATTACH

OR

ALTER DATABASE MODIFY FILE

Change file paths in metadata



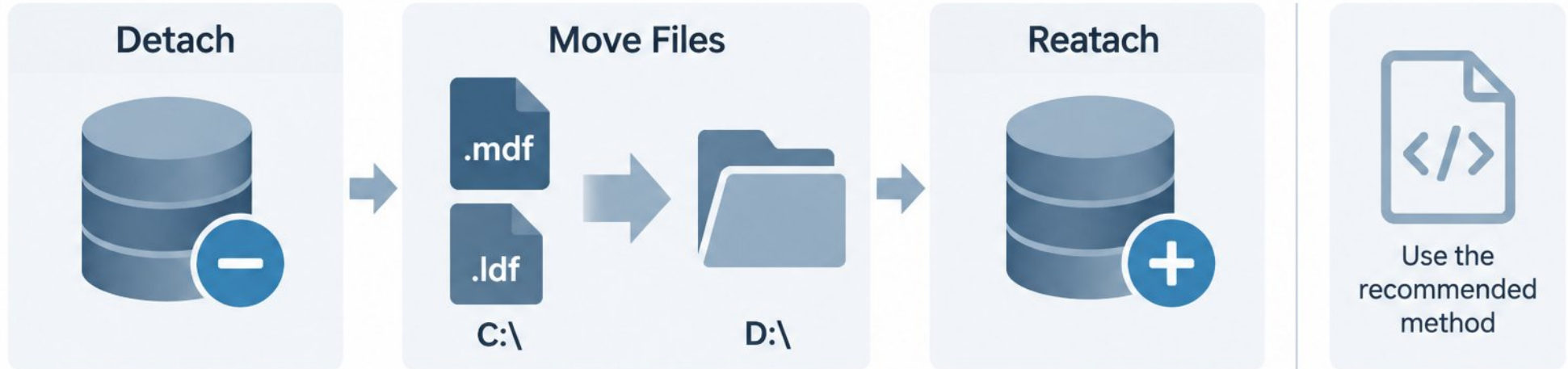
ALTER DATABASE MODIFY FILE

Apply by restart or offline/online cycle



- Two offline approaches: detach/attach with new paths, or ALTER DATABASE MODIFY FILE
- Detach/attach: best for a few user databases; reattach with CREATE DATABASE ... FOR ATTACH
- MODIFY FILE: scripted and repeatable; suits large estates where detach is undesirable
- MODIFY FILE changes metadata only; apply by restart or offline/online cycle
- tempdb and system databases: MODIFY FILE plus restart, never detach
- master uses -d and -l startup parameters; Resource database cannot be moved
- Compare on: availability impact, number of steps, error surface, database type

Method A in Practice: Detach, Move Files, Reattach a User Database



- 1 Quiesce application sessions; detach needs no active connections
- 2 List every file with `sys.database_files` before detaching
- 3 Detach with `sp_detach_db` or SSMS Tasks > Detach — files stay on disk
- 4 Move physical files to new path (robocopy), preserving permissions
- 5 Reattach: `CREATE DATABASE ... FOR ATTACH` with each new full path
- 6 Avoid deprecated `sp_attach_db` / `sp_attach_single_file_db` procedures

Method B in Practice: ALTER DATABASE MODIFY FILE for User Databases

SYNTAX PER FILE

```
ALTER DATABASE db MODIFY FILE (NAME = logical_name, FILENAME = 'new_path\os_file_name');
```

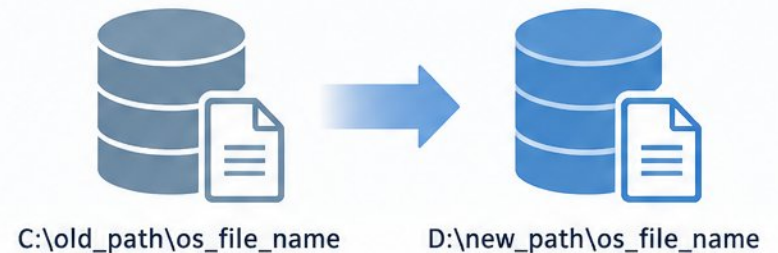
↑
NAME (logical-name
parameter)

↑
FILENAME
(new path and file name)

KEY POINTS

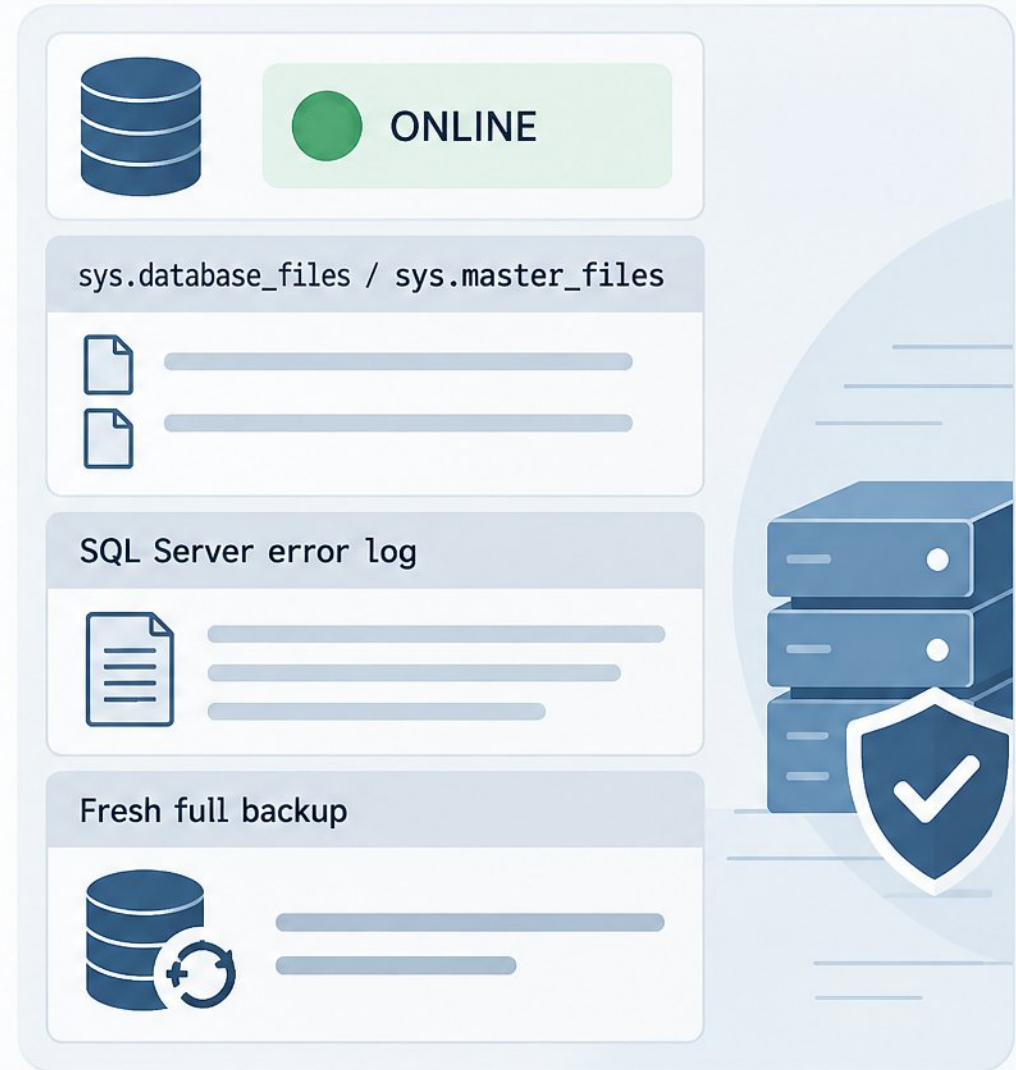
- Syntax per file: ALTER DATABASE db MODIFY FILE (NAME = logical_name, FILENAME = 'new_path\os_file_name');
- One property per statement; NAME always required
- Target path must exist; no system databases on UNC shares
- Database keeps using old files until restart or OFFLINE/ONLINE
- Move files physically, then take DB OFFLINE/ONLINE or restart
- One statement per file — including secondary filegroups
- Put log files on a separate drive for IO separation

FILE MOVE WORKFLOW (CONCEPTUAL)



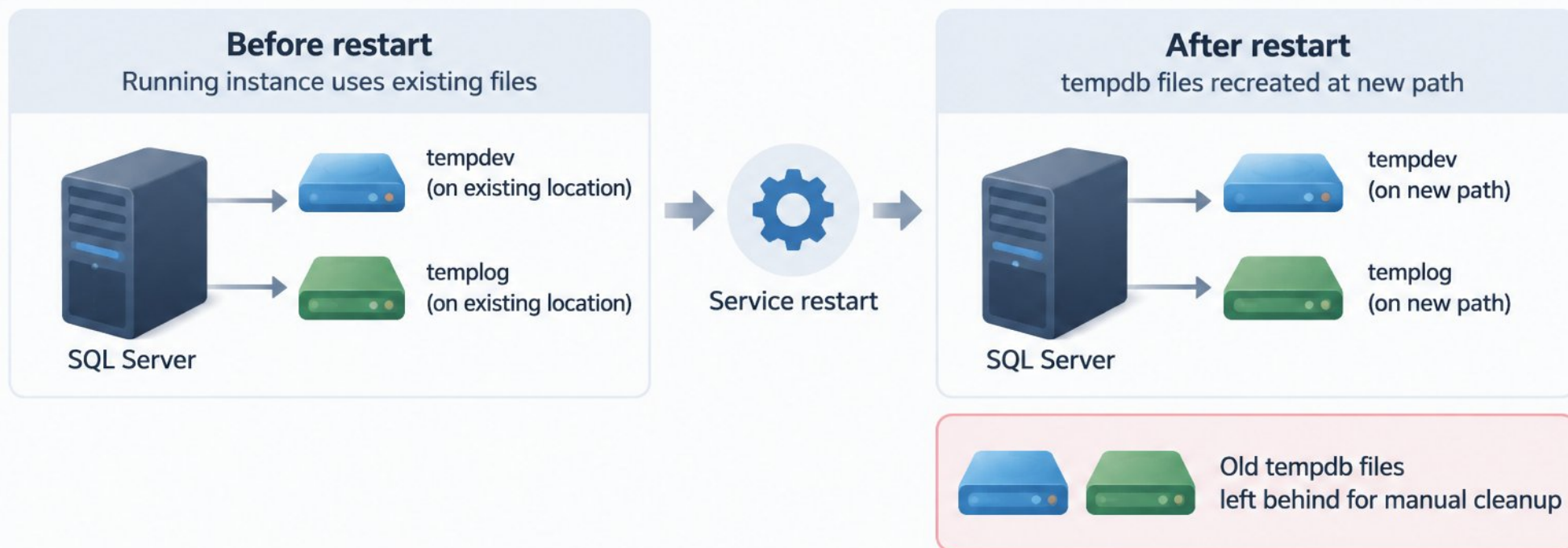
Verification After Each User Database Move

- 1 Confirm database is ONLINE and files report new paths in `sys.database_files` / `sys.master_files`
- 2 Check SQL Server error log for IO, access, or file-in-use errors
- 3 Take a fresh full backup; confirm the backup schedule still runs
- 4 Map common errors: file in use, access denied, file not found
- 5 Do not delete source files until validation of that database passes



Moving tempdb: Metadata Change Plus a Restart

- 1 Find logical names and paths: query `sys.master_files` where `database_id = DB_ID(N'tempdb')`.
- 2 One `ALTER DATABASE tempdb MODIFY FILE` statement per file — `NAME = tempdev / templog`.
- 3 No file copy needed: tempdb files are recreated at the new path on restart.
- 4 Until the service restarts, tempdb keeps using the files in the existing location.
- 5 Verify from `sys.master_files`, then `sys.database_files` inside tempdb.
- 6 Delete the old tempdb files by hand — SQL Server does not clean them up.



A Modern tempdb Gotcha: There Is Usually More Than One Data File



Since SQL Server 2016, setup creates one tempdb data file per logical processor, up to eight.



Default installs often have eight files, not one — tempdev and templog are not the whole story.



Moving only tempdev and templog leaves most of tempdb on the old drive.



Query `sys.master_files` for every tempdb row, then write one `MODIFY FILE` per row.



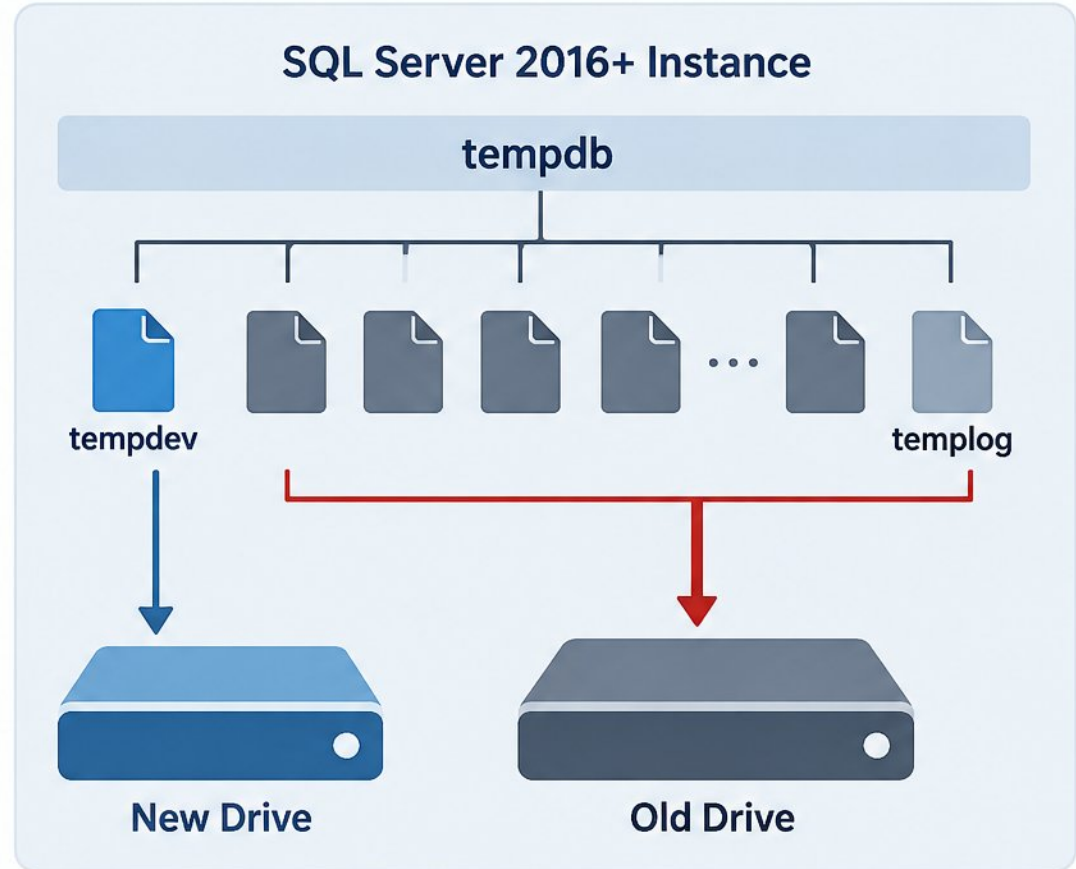
Restart is mandatory: tempdb is recreated at the new path, no file copy needed.



Risky moment: missing folder or no service-account rights means the instance will not start.



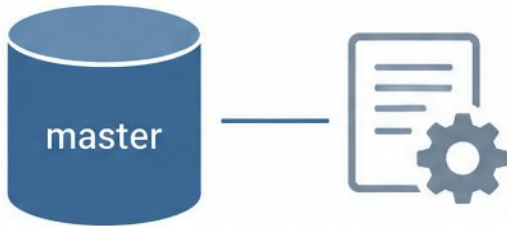
Create the folder, confirm permissions, and read paths back before restarting.



If the folder is missing or the service-account rights are not correct, the instance will not start.

Moving master, model, msdb, and the Resource Database

master



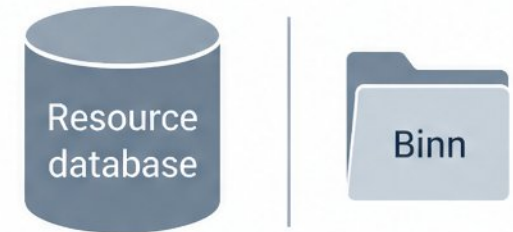
- master holds instance metadata; SQL Server will not start without it
- master path is set by -d and -l startup parameters, not MODIFY FILE alone

model and msdb



- model and msdb: MODIFY FILE, stop instance, copy files, restart

Resource database



- System databases cannot be taken offline and cannot live on UNC shares
- Resource database cannot be moved; files stay in the instance Binn folder

After the move



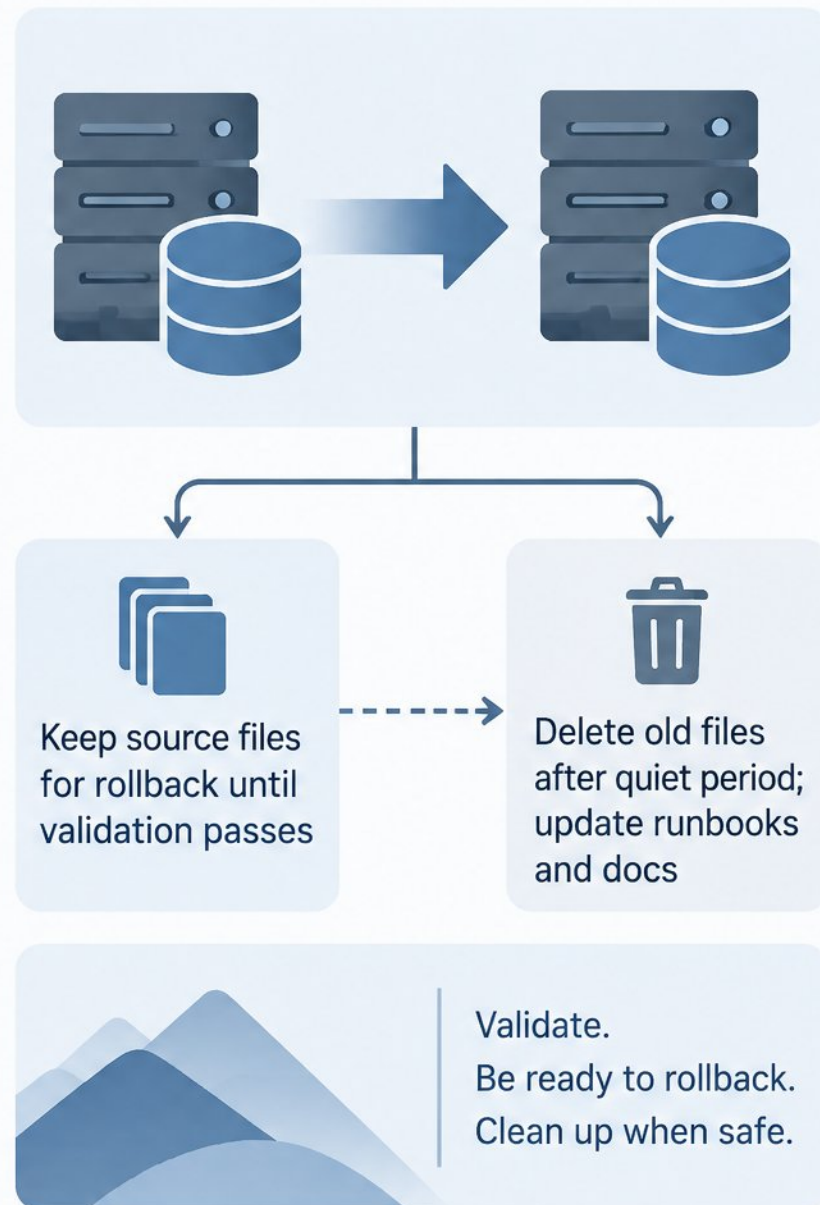
- After the move, update SQLAGENT.OUT path and Database Default Locations



Post-Move Validation, Rollback, and Cleanup

Verify, protect, and complete the move with confidence.

-  Confirm all databases ONLINE and paths correct in sys.master_files
-  Run DBCC CHECKDB clean; check error log for IO or access errors
-  Take a fresh full backup and verify restore paths
-  Test reconnection: connection strings, linked servers, Agent jobs
-  Keep source files for rollback until validation passes
-  Delete old files after quiet period; update runbooks and docs



Lab Exercise: Moving a Sample Database from Drive C to Drive D

1



Scenario: move sample database data and log files from drive C to drive D

2



Inventory: query sys.database_files and sys.master_files for names, paths, sizes, recovery model

3



Baseline: full backup, confirm service-account permissions on the new folder

4



Choose method: detach/attach or ALTER DATABASE MODIFY FILE — justify in one sentence

5



Execute: move data and log files, one database at a time

6



Validate: ONLINE status, new paths, clean DBCC CHECKDB, no error-log noise

7



Document: old and new paths, timestamps, method, evidence; keep source files



Drive C



Drive D



Extending the Lab: tempdb, System Databases, and Assessment Checkpoints



Extension 1

relocate every tempdb file, not just tempdev and templog; verify from sys.master_files and sys.database_files.



Extension 2

rehearse the master move on a throwaway instance, updating -d and -l startup parameters.



Checkpoints

When is detach/attach better than MODIFY FILE?
Why does tempdb need a restart but no copy?



Pitfalls

- moving files while online
- missing secondary files
- skipping backups
- never confirming new paths.



Next steps

rehearse tempdb and master moves, add free-space monitoring, plan a formal change window.



File



Filegroup



Database

Key Takeaways and Wrap-Up



Back up first and verify the backup before touching any file



Know your method: detach/attach for a few user databases; MODIFY FILE plus restart for system databases and larger estates



Create the destination folder and grant the service account full control first



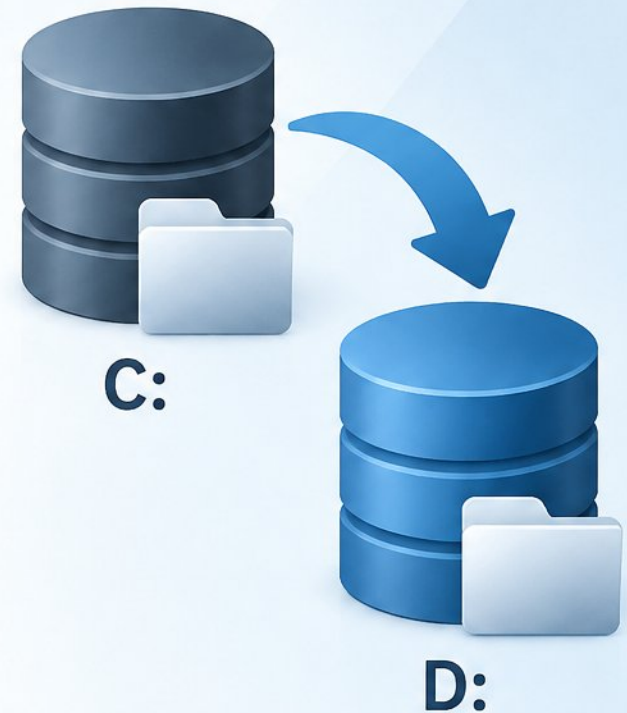
Special cases: tempdb is recreated on restart and often has many files; master uses -d / -l; Resource cannot move



Validate status, paths, DBCC CHECKDB, error log, and app connectivity; keep rollback ready



Finish: delete old files only after validation, update SQL Agent log and default database locations, document old and new paths



PersonWise.

PersonWise • AI digital-human course platform



Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/1cb3bfb5/public