Microsoft SQL Server

Einer der bekanntesten Datenbankmanagementsysteme. In der Express Version ist dieses kostenlos einsetzbar, jedoch gibt es viele gute Funktionen erst in den kostenpflichtigen Standard und Enterprise Versionen.

- Installation mit Docker
- Beispieldaten
- Monitoring
- Einzelbenutzer Modus
- Snapshots
- Backup wiederherstellen
- TempDB
- Änderungen mit Change Tracking verfolgen

Installation mit Docker

Seit der SQL Server Version 2017 ist der Microsoft SQL Server auch unter Linux als Docker Container verfügbar. Mittlerweile sind auch die meisten Funktionen unter Linux verfügbar, sodass der Docker Container eine gute Möglichkeit bietet, um SQL Datenbanken mit Microsofts SQL Server zu nutzen. Besonders für Testumgebungen ist dies ein besonders großer Mehrwert, damit schon früh im Entwicklungsstadium mit "echten" MS SQL Servern getestet werden kann ohne extra einen SQL Server unter Windows installieren zu müssen, was doch einen erheblichen Aufwand darstellt.

Für die Bereitstellung eines SQL Servers in der aktuellen Version 2022 mit persistentem Speicher genügt die folgende Docker Compose Konfiguration.

```
services:
  sql:
    image: mcr.microsoft.com/mssql/server:2022-latest
    container_name: sql
    environment:
      - ACCEPT_EULA=Y
      - MSSQL_SA_PASSWORD=EinNichtSoTolles08.15Passort
      - MSSQL_BACKUP_DIR=/var/opt/mssql/backup
      - MSSQL_LOG_DIR=/var/opt/mssql/log
    volumes:
      - /pfad/zu/sql/data:/var/opt/mssql/data
      - /pfad/zu/sql/log:/var/opt/mssql/log
      - /pfad/zu/sql/backup:/var/opt/mssql/backup
```

In dieser Konfiguration wurden bereits die Daten des SQL Servers in eigene Ordner geteilt, sodass die Logs und Backups separat von den aktuellen Daten liegen. Dies ist für optionale zusätzliche Konfigurationen wie Log Dateien abschneiden und Backups wichtig. Auch Migrationen werden hierdurch vereinfacht.

Um die Docker Compose Konfiguration auszuführen, kann am besten in das Verzeichnis der YAML Datei gewechselt werden. Danach wird je nach nach gewählter Installation `sudo docker-compose up -d` oder `sudo docker compose up -d` (keine Bindestrich zwischen docker und compose) eingegeben, um die Standard Konfiguration `docker-compose.yml` zu starten. Compose erstellt dann die gewünschten Container mit den angegeben Optionen. Sollten die Container bereits mit dieser Compose Konfiguration erstellt worden sein, so werden die Container in dieser neu erstellt, dessen Konfiguration geändert wurde.

Beispieldaten

Wer gerade keine ordentliche Datenbank samt Testdaten zur Hand hat, kann z. B. die AdventureWorks Beispiele von Microsoft herunterladen und in den eigenen Microsoft SQL Server wiederherstellen.

Die .bak Dateien können hier heruntergeladen werden: https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms#download-backup-files

Zuerst wird eine Docker Compose Datei erstellt, welche die Konfiguration des SQL-Servers vornimmt.

```
services:
  sql:
    image: mcr.microsoft.com/mssql/server:2022-latest
    container_name: sql
    environment:
      - ACCEPT_EULA=Y
      - MSSQL_SA_PASSWORD=EinNichtSoTolles08.15Passort
      - MSSQL_BACKUP_DIR=/var/opt/mssql/backup
      - MSSQL_LOG_DIR=/var/opt/mssql/log
    volumes:
      - /pfad/zu/sql/data:/var/opt/mssql/data
      - /pfad/zu/sql/log:/var/opt/mssql/log
      - /pfad/zu/sql/backup:/var/opt/mssql/backup
```

Nun werden die 3 Ordner für die Daten, Logs und Backups erstellt. Mit dem folgenden Skript werden auch direkt die Rechte angepasst (bitte den Pfad in der Variable anpassen).

```
cd /pfad/zu/sql/
mkdir data log backup
sudo chown -R 10001 data log backup
```

Um die Docker Compose Konfiguration auszuführen, kann am besten in das Verzeichnis der YAML Datei gewechselt werden. Danach wird je nach nach gewählter Installation `sudo docker-compose up -d` oder `sudo docker compose up -d` (keine Bindestrich zwischen docker und compose) eingegeben, um die Standard Konfiguration `docker-compose.yml` zu starten. Compose erstellt dann die gewünschten Container mit den angegeben Optionen. Sollten die Container bereits mit dieser

Compose Konfiguration erstellt worden sein, so werden die Container in dieser neu erstellt, dessen Konfiguration geändert wurde.

Während der SQL-Server startet, kann die bak-Datei der AdventureWorks in den Ordner `/pfad/zu/sql/backup` heruntergeladen werden. Eine Übersicht mit allen verfügbaren Sicherungsdateien inklusive Downloads gibt es hier: AdventureWorks-Beispieldatenbanken

Wer es automatisieren möchte, kann es auch mittels folgender Bash-Zeile herunterladen.

```
cd /pfad/zu/sql/backup
wget https://github.com/Microsoft/sql-server-
samples/releases/download/adventureworks/AdventureWorks2022.bak
```

Sobald der SQL-Server gestartet und bereit ist, kann das Backup mit folgendem Befehl eingespielt werden.

```
sudo docker exec -it sql /opt/mssql-tools/bin/sqlcmd \
   -S localhost -U SA -P 'EinNichtSoTolles08.15Passort' \
   -Q 'RESTORE DATABASE AdventureWorks2022 FROM DISK =
"/var/opt/mssql/backup/AdventureWorks2022.bak" WITH MOVE "AdventureWorks2022" TO
"/var/opt/mssql/data/AdventureWorks2022.mdf", MOVE "AdventureWorks2022_Log" TO
"/var/opt/mssql/data/AdventureWorks2022.ldf"'
```

Nun steht eine SQL-Server Instanz samt AdventureWorks2022 Datenbank zum Spielen bereit.

Monitoring

Wenn es mal im SQL-Server hakt und die aktiven Aktivitäten und Abfragen geprüft werden müssen, können folgende Befehle helfen.

Zum einen das Skript sp_whoisactive und zum anderen sp_who2. Während sp_who2 bereits vorhanden ist, muss sp_whoisactive nachinstalliert werden, da dieses Skript nicht von Microsoft stammt. sp_whoisactive zeigt alle gerade vom SQL Server ausgeführten Abfragen an, mit Details wie z. B. durch welche andere Abfrage sie blockiert werden oder wie lange sie schon laufen. sp_who2 zeigt diverse Informationen über alle aktiven Verbindungen zum SQL Server an. Beide Skripte werden im folgenden vorgestellt.

sp_whoisactive

Um die aktiven Prozesse auf einem Microsoft SQL-Server anzeigen zu können, bietet sich das Skript `sp_whoisactive` an. Jedoch handelt es sich nicht um einen Standard-Befehl, sondern dieser Befehl muss erst mittels eines Skriptes nachgerüstet werden.

Dazu einfach den Source Code von der Github Seite herunterladen und entpacken. Dann z. B. das *Microsoft SQL Server Management Studio* öffnen und die Master Datenbank als aktive Datenbank markieren. Danach das Skript *sp_WhoIsActive.sql* in der Master Datenbank ausführen.

> Der Befehl muss in der Datenbank Master ausgeführt werden, ansonsten steht dieser nur in der Datenbank zur Verfügung, in der das Skript ausgeführt wurde, da nur dort sp_whoisactive als Stored Procedure gespeichert wird.

Nun steht der Befehl in allen Datenbanken zur verfügung und kann einfach in einer beliebigen SQL-Query ausgeführt werden mit `sp_whoisactive`. Eine vollständige Dokumentation des Befehls ist hier zu finden: sp_whoisactive Doku

sp_who2

Um die aktiven Verbindungen eines SQL-Servers einsehen zu können, kann der Befehl `sp_who2` verwendet werden. Es können mehrere Verbindungen pro Rechner und User sein. Deswegen empfiehlt es sich folgende Abfrage zu verwenden, um die Informationen etwas aufzubereiten.

```
CREATE TABLE #sp_who2 (
SPID INT,Status VARCHAR(255)
,Login VARCHAR(255)
```

```
,HostName  VARCHAR(255)
,BlkBy VARCHAR(255)
,DBName VARCHAR(255)
,Command VARCHAR(255)
,CPUTime INT
,DiskIO INT
,LastBatch VARCHAR(255)
,ProgramName VARCHAR(255)
,SPID2 INT
,REQUESTID INT
)
INSERT INTO #sp_who2 EXEC sp_who2
SELECT
Login
,HostName
,DBName
,Command
,ProgramName
,COUNT(*) AS Verbindungen
,SUM(CPUTime) AS CPUTime
,SUM(DiskIO) AS DiskIO
,MAX(LastBatch) AS LastBatch
FROM #sp_who2
WHERE SPID > 50 AND DBName = 'Testdatenbank'
GROUP BY
Login
,HostName
,DBName
,Command
,ProgramName
ORDER BY Login ASC
DROP TABLE #sp_who2
```

Der Befehl sp_who2 wurde um eine temporäre Tabelle erweitert, in welche die ermittelten Daten geschrieben werden. Dadurch steigt zwar der Aufwand für die Ausführung, jedoch können die Daten so sinnvoller gefiltert werden, da sonst sehr viele Daten dargestellt werden. In diesem Beispiel werden einige Systemprozesse ausgeschlossen und nur Verbindungen zur Datenbank *Testdatenbank* angezeigt. Zusätzlich wird gruppiert, um mehrfache Verbindungen zusammenzufassen.

Einzelbenutzer Modus

Für einige administrative Tätigkeiten kann es sinnvoll sein, die Datenbank in den sog. *Single-User Mode* zu versetzen. In diesem Modus kann nur ein User mit der Datenbank interagieren.

Eine bestimmte Datenbank in den *Single-User Mode* versetzen:

```
ALTER DATABASE N'<Database Name, sysname,>'
SET SINGLE_USER
WITH ROLLBACK IMMEDIATE;
```

Eine Datenbank zurück in den *Multi-User Mode* versetzen:

```
ALTER DATABASE N'<Database Name, sysname,>'
SET MULTI_USER;
```

Snapshots

Mittels eines Snapshots kann der aktuelle Stand einer Datenbank schnell und einfach fixiert werden. Dabei werden die Änderungen in einen separaten Container geschrieben, sodass keine Kopie der Daten notwendig ist. Der benötigte Speicherplatz wächst mit den Änderungen. Wichtig dabei ist, dass die Snapshots nicht mit einer Kopie vergleichbar sind, denn liegt ein Defekt im Datenbanksystem vor, sodass die originale Datenbank aufgrund eines Fehlers nicht mehr gelesen werden kann, dann bringen auch die Snapshots nichts mehr. Snapshots können aufeinander aufbauen (z. B. jeden Monat inkrementell ein Snapshot), aber dann müssen alle lückenlos vorhanden sein.

Snapshot erstellen

Diese können nur mittels T-SQL erstellt werden, nicht über das SQL Server Management Studio. Der Snapshot ist vergleichbar mit einer eigenen Datenbank. Er kann wie diese angesteuert werden. Der Unterschied ist nur, dass keine Änderungen möglich sind. Es können nur Daten gelesen werden.

```
CREATE DATABASE db_snapshot_name
ON (
□NAME = db_original_name,
□FILENAME = 'D:\Data\db_snapshot_name.ss'
)
AS SNAPSHOT OF db_original_name;
GO
```

Weitere Informationen: Create a Database Snapshot

Snapshot wiederherstellen

```
-- Zum Wiederherstellen in die Master-DB wechseln
USE master;
RESTORE DATABASE db_original_name
FROM DATABASE_SNAPSHOT = 'db_snapshot_name';
GO
```

Weitere Informationen: Revert a Database to a Database Snapshot

> Sollte das Wiederherstellen fehlschlagen, weil noch aktive Verbindungen bestehen, dann kann die Datenbank zeitweise in den Einzelbenutzer Modus geschaltet werden.

Einzelbenutzer Modus

Für einige administrative Tätigkeiten kann es sinnvoll sein, die Datenbank in den sog. *Single-User Mode* zu versetzen. In diesem Modus kann nur ein User mit der Datenbank interagieren.

Eine bestimmte Datenbank in den *Single-User Mode* versetzen:

```
ALTER DATABASE N'<Database Name, sysname,>'
SET SINGLE_USER
WITH ROLLBACK IMMEDIATE;
```

Eine Datenbank zurück in den *Multi-User Mode* versetzen:

```
ALTER DATABASE N'<Database Name, sysname,>'
SET MULTI_USER;
```

Snapshot löschen

Das Löschen eines Snapshots ist vergleichbar mit dem Löschen einer Datenbank. Es werden die selben Rechte benötigt und alle Verbindungen zur DB werden gekappt.

```
DROP DATABASE db_snapshot_name;
```

Weitere Informationen: Drop a Database Snapshot

Reportings auf Snapshots

Lesevorgänge auf Snapshots haben keine Locks zur Folge. Somit wäre es denkbar, ein Reporting auf Basis von Snapshots zu bauen. Beispielsweise könnte 1x täglich ein Snapshot erstellt werden, sodass das Reporting auf diesen Snapshot basierend komplexe Abfragen ausführen kann. Eine Veränderung der Daten in den Snapshots ist nicht möglich, sie sind sozusagen zum Zeitpunkt XY eingefroren.

Backup wiederherstellen

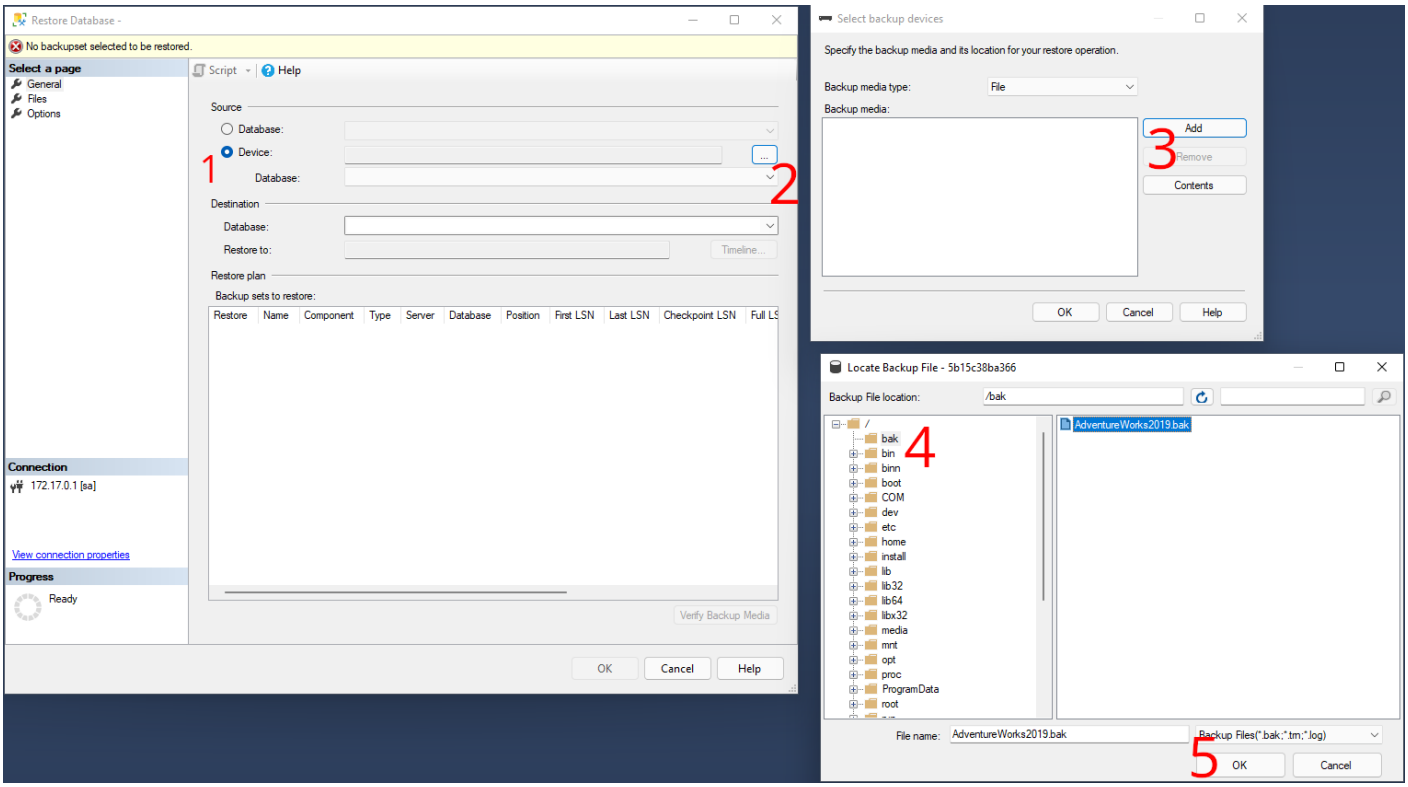
Beim Wiederherstellen (Restore) von Microsoft SQL Datenbanken gibt es verschiedene Möglichkeiten.

> Soll das Backup auf einem anderen Server wiederhergestellt werden, so sind das Vollbackup und alle darauf folgenden gesicherten Transaktionslogs in das Backupverzeichnis des Zielservers zu übertragen. Anschließend kann wie unten gezeigt das Backup wiederhergestellt werden.

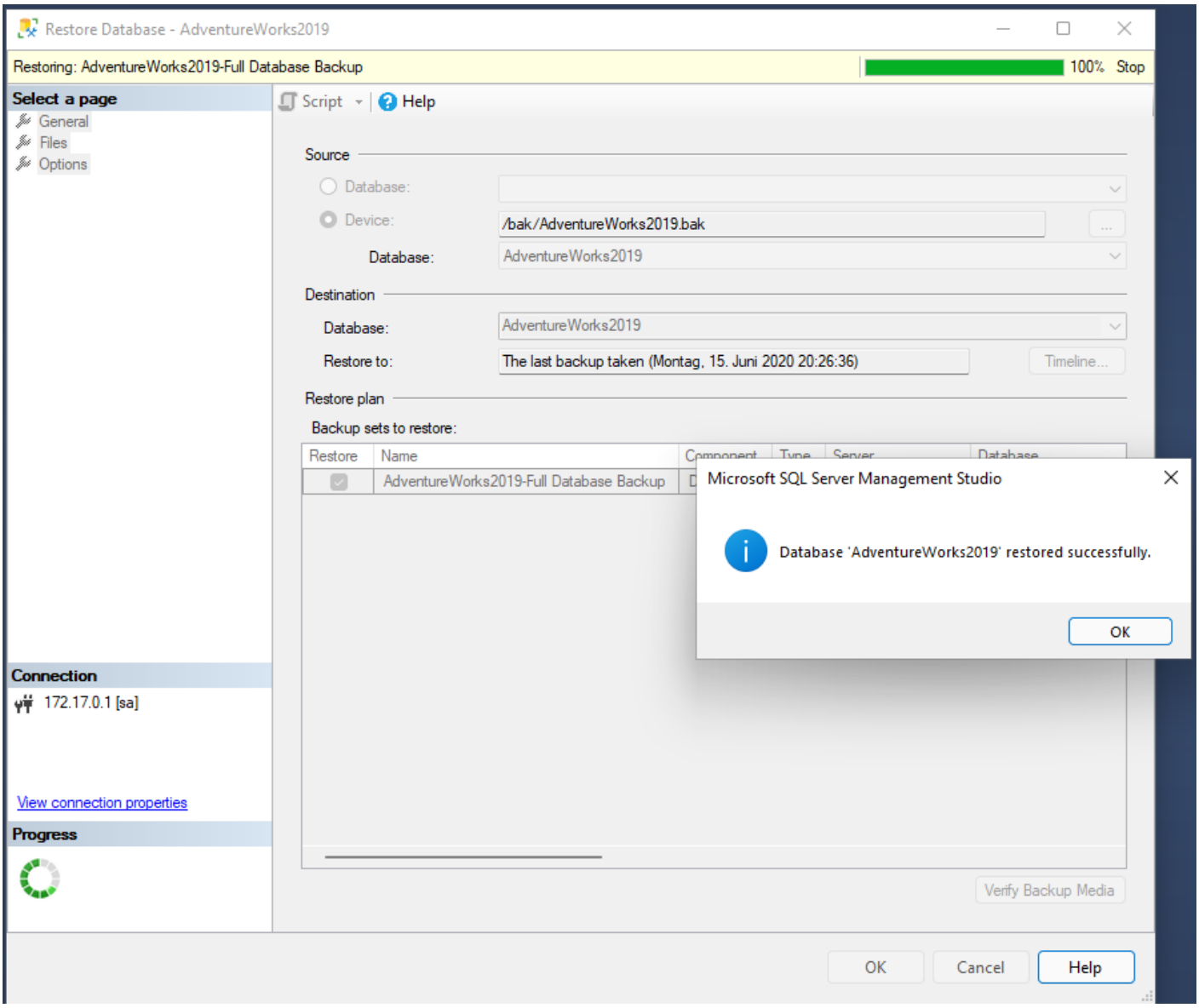
Restore eines Vollbackups

Dies ist der einfachste Fall, eine Datenbank wird einfach aus einer .bak Datei wiederhergestellt. Dies geht einfach und schnell.

1. Microsoft SQL Server Management Studio öffnen und in Server einloggen
2. Rechtsklick auf *Databases -> Restore Database...*
3. Danach folgende Schritte umsetzen (1 - 5)
    1. im Reiter *General* unter *Source* von *Database:* auf *Device:* wechseln
    2. die 3 Punkte auswählen
    3. mit *Add* den Dialog zum Auswählen der Backup Datei öffnen
    4. den Order der Backups auswählen
    5. die Backup Datei auswählen und mit *OK* bestätigen



4. Es wird nun ein *Backup media:* angezeigt, dieses nochmal mit *OK* bestätigen.
5. Nun wird ein Restore Punkt angezeigt, diesen nur noch mit *OK* bestätigen und danach steht die Datenbank in der Version zur Verfügung.



Restore eines Vollbackups mit Logs

Dieser Fall ist aufwendiger als der einfache Restore eines Vollbackups, aber es kann fast auf die Minute sogar Sekunde genau wiederhergestellt werden. Bei dieser Art des Restores wird zuerst ein Vollbackup eingespielt, danach werden dann die Transaktionslogs eingespielt. Wird z. B. 1x täglich um 20 Uhr ein Vollbackup erstellt und alle 30 Minuten werden die Log Dateien gesichert und abgeschnitten, so kann beispielsweise die Datenbank von 15:30 Uhr wiederhergestellt werden. Eventuell können sogar die Änderungen auf die Minute z. B. 15:35 Uhr eingespielt werden, sofern der Zeitpunkt exakt bekannt ist und das Transaktionslog nicht beschädigt ist, wie im Fall von gelöschten Datensätzen (denn diese sind dann ordnungsgemäß ins Transaktionslog geschrieben).

> Egal ob das Backup über das Management Studio oder T-SQL eingespielt wird, in beiden Fällen muss das _Recovery Model_ der Datenbank auf _Simple_ gesetzt werden.

Mit T-SQL

1. Damit das Vollbackup + Logs zurückgespielt werden kann, ist das *Recovery Model* der Datenbank auf *Simple* zu setzen.

```
USE [master];
ALTER DATABASE [AdventureWorks2019]
SET RECOVERY SIMPLE;
```

2. Nun kann das Vollbackup eingespielt werden, dabei ist insbesondere der Parameter `WITH NORECOVERY` wichtig, da dieser dafür sorgt, dass die Datenbank im Wiederherstellungsmodus bleibt:

```
RESTORE DATABASE [AdventureWorks2019]
FROM DISK = '/backup/FULL/AdventureWorks2019_20230419_171225.bak'
WITH NORECOVERY
```

3. Nun werden der Reihe nach alle gesicherten Transaktionslogs eingespielt:

```
RESTORE LOG [AdventureWorks2019]
FROM DISK = '/backup/LOG/AdventureWorks201920230419_171315.trn'
```

WITH NORECOVERY `Dabei ist wichtig, dass solange noch weitere Log Dateien folgen, der Parameter `WITH NORECOVERY` mit übergeben wird. 4. Die letzte Log Datei wird ohne den `WITH NORECOVERY` Parameter eingespielt, damit der Wiederherstellungsprozess ordnungsgemäß beendet wird:`
sql RESTORE LOG [AdventureWorks2019] FROM DISK = '/backup/LOG/AdventureWorks201920230419_171441.trn' ```

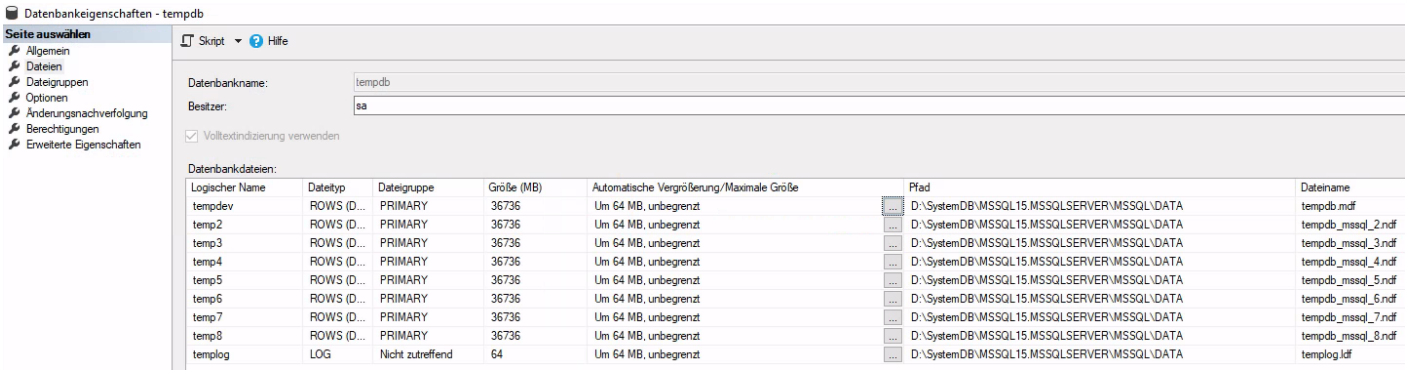
Mit Management Studio

Tatsächlich erfolgt das Wiederherstellen mittels Management Studio genauso wie bei Restore eines Vollbackups. Es ist nur vorher das Recovery Model zu ändern.

TempDB

Größe der TempDB

Es können mehrere Dateien zusammen die TempDB bilden z. B. 8x 8MB bei 8 CPU-Kernen. Außerdem ist in der Regel ein automatisches Wachsen eingestellt, sodass die Dateien größer werden, je länger die Laufzeit des Servers. Über das SQL Management Studio können die Größen der Temp-Dateien eingesehen werden, jedoch nicht die initiale Größe:



Logischer Name	Dateityp	Dateigruppe	Größe (MB)	Automatische Vergrößerung/Maximale Größe	Pfad	Dateiname
tempdev	ROWS (D...	PRIMARY	36736	Um 64 MB, unbegrenzt	D:\SystemDB\MSSQL15.MSSQLSERVER\MSSQL\DATA	tempdb.mdf
temp2	ROWS (D...	PRIMARY	36736	Um 64 MB, unbegrenzt	D:\SystemDB\MSSQL15.MSSQLSERVER\MSSQL\DATA	tempdb_mssql_2.ndf
temp3	ROWS (D...	PRIMARY	36736	Um 64 MB, unbegrenzt	D:\SystemDB\MSSQL15.MSSQLSERVER\MSSQL\DATA	tempdb_mssql_3.ndf
temp4	ROWS (D...	PRIMARY	36736	Um 64 MB, unbegrenzt	D:\SystemDB\MSSQL15.MSSQLSERVER\MSSQL\DATA	tempdb_mssql_4.ndf
temp5	ROWS (D...	PRIMARY	36736	Um 64 MB, unbegrenzt	D:\SystemDB\MSSQL15.MSSQLSERVER\MSSQL\DATA	tempdb_mssql_5.ndf
temp6	ROWS (D...	PRIMARY	36736	Um 64 MB, unbegrenzt	D:\SystemDB\MSSQL15.MSSQLSERVER\MSSQL\DATA	tempdb_mssql_6.ndf
temp7	ROWS (D...	PRIMARY	36736	Um 64 MB, unbegrenzt	D:\SystemDB\MSSQL15.MSSQLSERVER\MSSQL\DATA	tempdb_mssql_7.ndf
temp8	ROWS (D...	PRIMARY	36736	Um 64 MB, unbegrenzt	D:\SystemDB\MSSQL15.MSSQLSERVER\MSSQL\DATA	tempdb_mssql_8.ndf
templog	LOG	Nicht zutreffend	64	Um 64 MB, unbegrenzt	D:\SystemDB\MSSQL15.MSSQLSERVER\MSSQL\DATA	templog.ldf

Initialgröße der TempDB-Dateien ändern

Die aktuelle Größe der TempDB-Dateien lässt sich nicht ändern, da in diesen Temp-Tabellen usw. gespeichert sind, die der SQL Server benötigt. Jedoch lässt sich die Größe angeben, wie viel Speicherplatz die Dateien beim Start des SQL Server Dienstes aufweisen dürfen. Standardmäßig sind dies 8 MB. Zwar wachsen die Dateien dynamisch mit, jedoch sollte die Anfangsgröße an die benötigte Größe angepasst werden, da jede Vergrößerung Performance kostet.

```
ALTER DATABASE tempdb MODIFY FILE (Name=tempdev, SIZE = 16GB);
ALTER DATABASE tempdb MODIFY FILE (Name=temp2, SIZE = 16GB);
ALTER DATABASE tempdb MODIFY FILE (Name=temp3, SIZE = 16GB);
ALTER DATABASE tempdb MODIFY FILE (Name=temp4, SIZE = 16GB);
ALTER DATABASE tempdb MODIFY FILE (Name=temp5, SIZE = 16GB);
ALTER DATABASE tempdb MODIFY FILE (Name=temp6, SIZE = 16GB);
ALTER DATABASE tempdb MODIFY FILE (Name=temp7, SIZE = 16GB);
ALTER DATABASE tempdb MODIFY FILE (Name=temp8, SIZE = 16GB);
```

Damit die Änderungen wirksam werden, muss der SQL Server Dienst neu gestartet werden.

Initialgrößen der TempDB-Dateien

```
SELECT
 name
 ,size*8.0/1024  'Initial Size in MB'
FROM master.sys.sysaltfiles
WHERE dbid = 2
```

Änderungen mit Change Tracking verfolgen

Der MS SQL Server bietet ein Feature mit dem Namen Change Tracking an, welches die Änderungen von Zeilen in einer Tabelle mittels Versionierung verfolgbar machen kann. Im Gegensatz zum Change Data Capture (CDC) werden jedoch nicht die Änderungen an sich erfasst und gespeichert, sondern es wird eine Versionsnummer über die Änderungen vergeben, sodass z. B. nur alle geänderten Abfragen seit einer bestimmten Version zurückgegeben werden können.

Dieses Feature bietet sich an, um inkrementelle SQL-Abfragen zu gestalten, da so z. B. jede Stunde ein SELECT ausgeführt werden kann, welches alle geänderten Zeilen zurückgibt, um diese in eine andere Datenbank zu kopieren oder mittels einer BI-Software zu verarbeiten.

Damit die Änderungen erfasst werden können, muss zuerst Change Tracking in der Datenbank und dann für die zu überwachenden Tabellen aktiviert werden.

Um Change Tracking in einer Datenbank verfügbar zu machen, kann folgendes SQL verwendet werden.

```
ALTER DATABASE ÄndereDenDatenbanknamen
SET CHANGE_TRACKING = ON
(CHANGE_RETENTION = 7 DAYS, AUTO_CLEANUP = ON)
```

In diesem SQL wurde eine Verfallszeit von 7 Tagen angegeben und dass die Logs ab dann abgeschnitten werden. Es können auch kürzere oder längere Zeiträume definiert oder sogar das Aufräumen komplett deaktiviert werden. Jedoch können bei vielen Änderungen eine Menge Daten anfallen, sodass ein Aufräumen zu empfehlen ist.

Mit dem folgenden SQL kann nun das Tracking für eine Tabelle aktiviert werden.

```
ALTER TABLE ÄndereDenNamenDerTabelle
ENABLE CHANGE_TRACKING
WITH (TRACK_COLUMNS_UPDATED = ON)
```

Nun werden alle Änderungen in der Tabelle überwacht und führen zu einer Aktualisierung der Version. Außerdem werden auch die Spalten überwacht, sodass nicht nur festgestellt werden kann, welche Zeilen geändert wurden, sondern auch welche Spalten. Diese Option sollte auf OFF gesetzt werden, wenn nur die geänderten Zeilen benötigt werden und sowieso immer bestimmte Spalten oder sogar alle Spalten ausgegeben werden.

Die Versionierung startet bei 0 und wird mit jeder Änderung um 1 hochgezählt, die aktuelle Version lässt sich mit folgendem SQL ermitteln.

```
SELECT CHANGE_TRACKING_CURRENT_VERSION();
```

Wenn nun Änderungen an den Daten vorgenommen werden, wird die Version automatisch erhöht. Mit Hilfe des folgenden SQL können nun nur die Änderungen ab einer bestimmten Version zurückgegeben werden.

```
SELECT aenderungen.SYS_CHANGE_VERSION, emp.*
FROM CHANGETABLE (CHANGES HumanResources.Employee, 0) AS aenderungen
INNER JOIN HumanResources.Employee emp on emp.[BusinessEntityID] =
aenderungen.[BusinessEntityID];
```

Als Beispiel wurde die Tabelle *HumanResources.Employee* aus der AdventureWorks Beispieltabelle verwendet.

Hier wurde die Version auf 0 gesetzt, wodurch alle Zeilen zurückgegeben werden, welche jemals verändert wurden (innerhalb der letzten 7 Tage, da diese die Löschzeit ist).

Nachdem das Change Tracking aktiviert wurde, sind 2 Zeilen in der Tabelle verändert worden, wodurch die Abfragen nun folgende Ergebnisse liefern.

```
11  SELECT CHANGE_TRACKING_CURRENT_VERSION() AS 'Version';
12
13  SELECT aenderungen.SYS_CHANGE_VERSION, emp.*
14  FROM CHANGETABLE (CHANGES HumanResources.Employee, 0) AS aenderungen
15  INNER JOIN HumanResources.Employee emp on emp.[BusinessEntityID] = aenderungen.[BusinessEntityID];
```

Results Messages

	Version
1	2

	SYS_CHANGE_VERSION	BusinessEntityID	NationalIDNumber	LoginID	OrganizationNode	OrganizationLevel	JobTitle	BirthDate	MaritalStatus	Gender	HireDate	SalariedFlag	Vacat
1	1	5	695256908	adventure-works\gail0	0x5ADA	3	Senior Design Engineer	1952-09-27	M	F	2008-01-06	1	5
2	2	6	998320692	adventure-works\josef0	0x5ADE	3	Senior Design Engineer	1959-03-11	M	M	2008-01-24	1	6