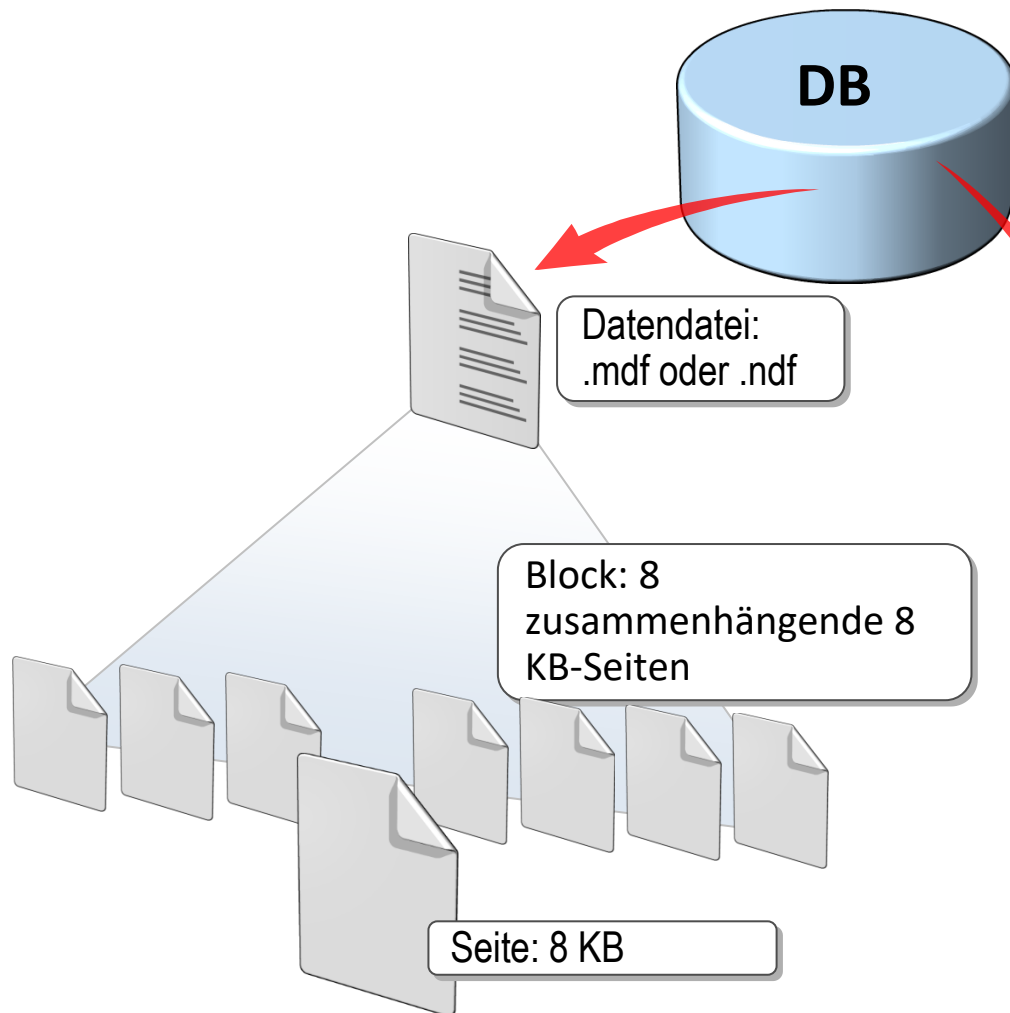

Datenbanken und Dateien

Wie werden Datenbanken gespeichert?

Dateiart	Erweiterung	Dateityp
Primäre Dateien	.mdf (Master Data File)	Enthält Startinformationen für die Datenbank und Zeiger auf die anderen Dateien
Sekundäre Dateien	.ndf (NibleGen Design File; SQL secondary Data File)	Mit sekundären Dateien können Benutzerdaten auf mehrere Datenträger verteilt werden, indem jede Datei auf einem anderen Datenträger gespeichert wird
Transaktions-Protokoll	.ldf (Log Data File)	Protokolldateien enthalten sämtliche Informationen, welche zum Wiederherstellen der Datenbank benötigt werden

Speicherung

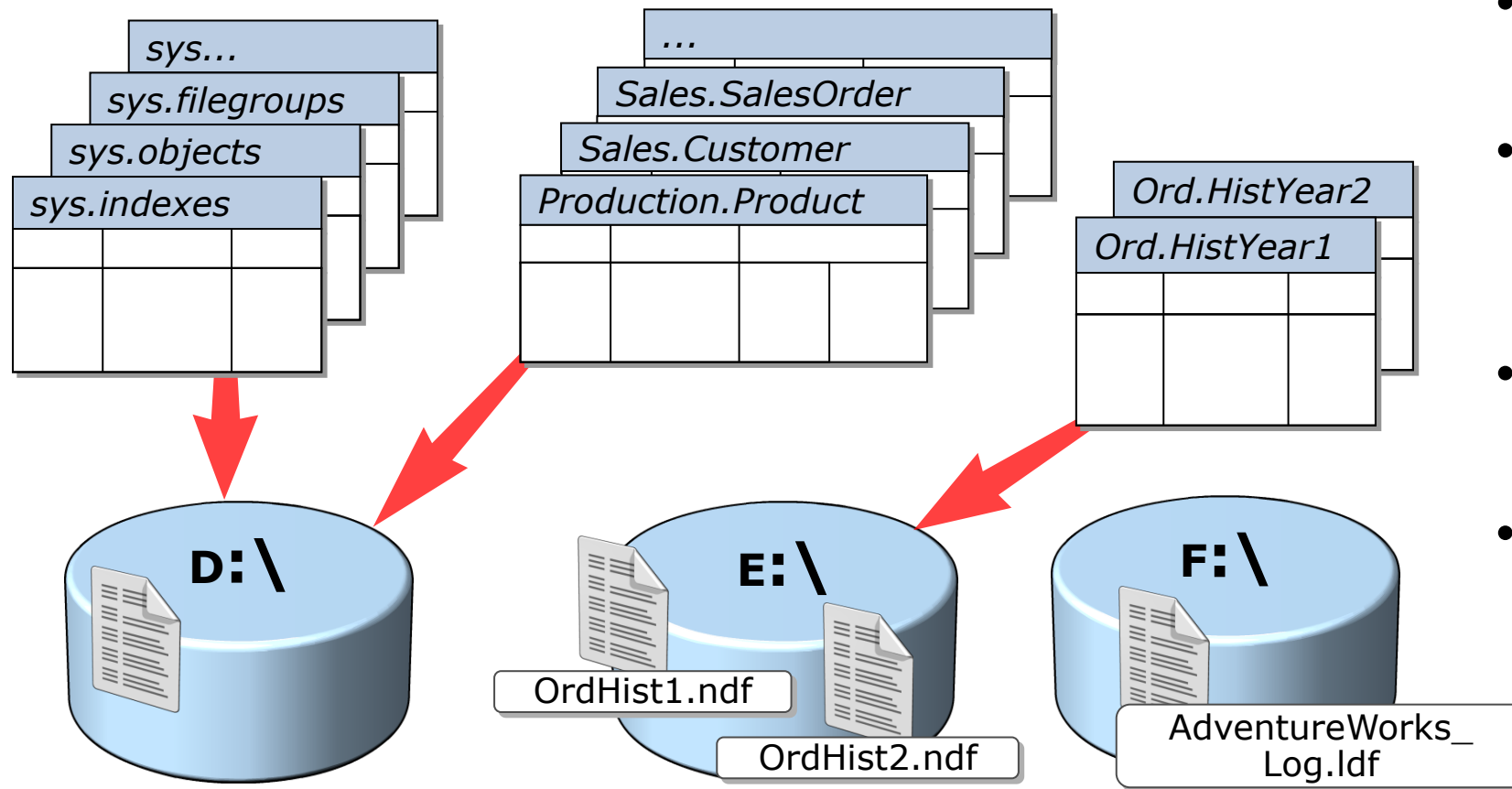


.mdf und .ndf: bei Verwendung vieler Tabellen und bei viel Nutzung in separaten Dateigruppen und auf separaten Laufwerken ablegen.

Viele Platten parallel um IO-Leistung zu erhöhen

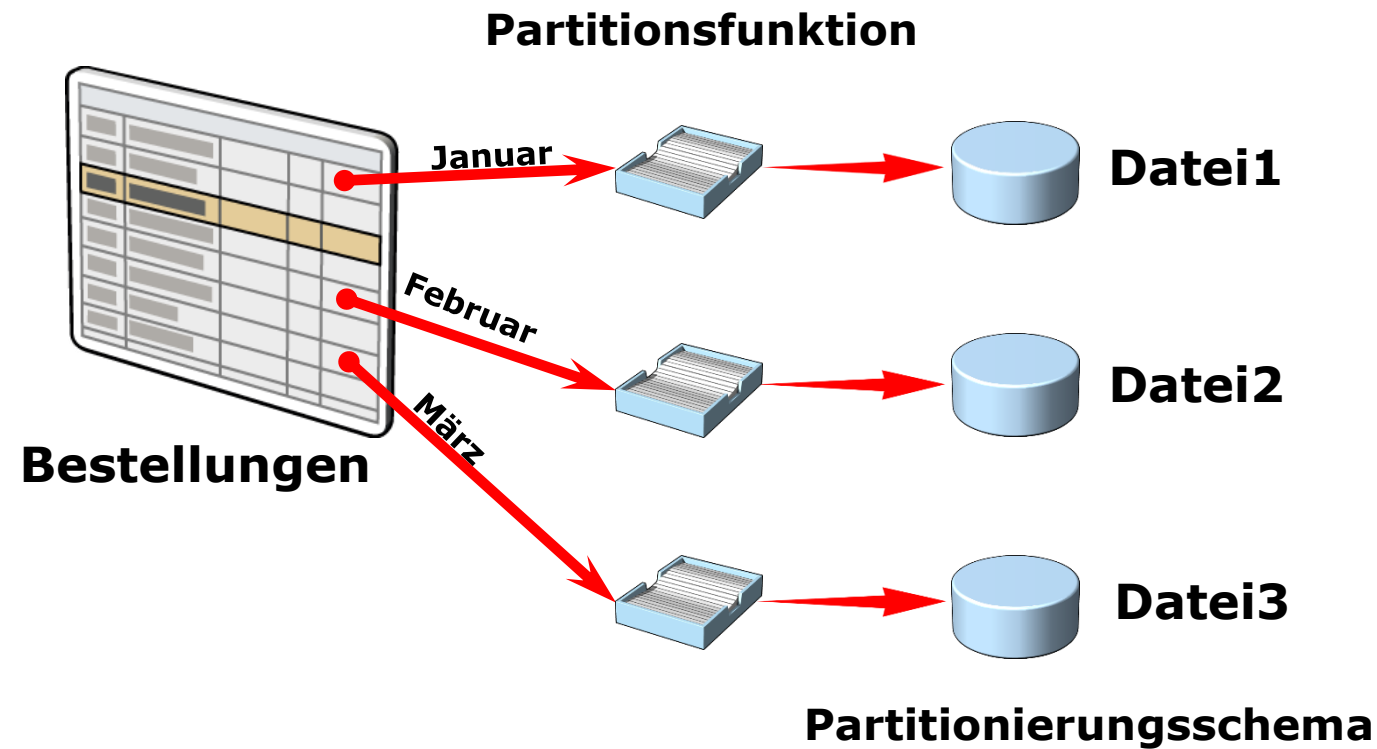
.ldf auf physisch separaten Datenträger oder RAID-Array

Dateigruppen



- Können nachträglich erstellt werden
- Häufig verwendete Tabellen in verschiedene Dateigruppen legen
- Ebenso Tabellen, welche in Joins genutzt werden
- Metadaten in die primäre Dateigruppe

Partitionen



- Partitionieren splittet Tabellen in Teiltabellen auf; diese Teile können in separaten Dateigruppen / auf verschiedenen Datenträgern gespeichert werden.

→ **Partitio-
nierung**

Kapazitätsplanung

- Abschätzen der maximalen Größe der Datenbank
 - Dateigrößen direkt anpassen, Fragmentierung verhindern, automatisches Vergrößern nicht zu klein festlegen
 - <https://technet.microsoft.com/de-de/library/cc298801.aspx>

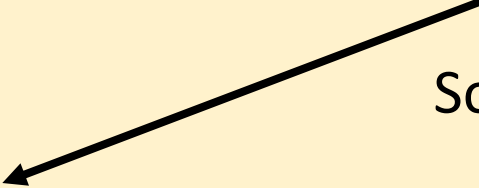
Erstellen der Datenbanken

- Bereits bei Erstellung Möglichkeit, die Dateigruppen anzugeben
- Primäre Datenbankdatei PRIMARY
- Sekundäre Dateien FILEGROUP, .ldf ist LOG ON

```
CREATE DATABASE database_name  
[ ON [ PRIMARY ] [ filespec [, n] ]  
[ FILEGROUP filegroup_name [DEFAULT]  
filespec [, n] ] [, n] ]  
[ LOG ON [filespec [, n] ]  
[ COLLATE collation_name ] ]
```

```
SELECT name, description  
FROM fn_helpcollations();
```

Sortierung nach Kulturstandard



Beispiel

```
CREATE DATABASE NeueDatenbank
ON PRIMARY
( NAME = N'Primaerfile',
  FILENAME = N'E:\DB\nochnblubb.mdf' )
, ←
  FILEGROUP [Sekundaerfile]
( NAME = N'FG2',
  FILENAME = N'G:\DB\FG2.ndf'
)
LOG ON
( NAME = N'nochnblubb_log',
  FILENAME = N'H:\Log\nochnblubb_log.ldf' )
```

Mehrere Dateigruppen
werde durch das Komma
„ , “ separiert

Mehrere Dateien in einer
Dateigruppe werden
auch durch das Komma
separiert

Datenbankoptionen

Option	Beschreibung
AUTO_CLOSE	Die Datenbank wird geschlossen und ordnungsgemäß heruntergefahren, wenn der letzte Benutzer die Datenbank beendet
AUTO_SHRINK	Wenn für diese Option der Wert ON festgelegt wurde, werden die Datenbankdateien möglicherweise periodisch verkleinert
READ_ONLY	Wenn READ_ONLY festgelegt ist, können Benutzer Daten abrufen, jedoch nicht ändern
RECOVERY FULL	Mithilfe von Sicherungen von Datenbank- und Transaktionsprotokollen wird die Möglichkeit gewährleistet, Daten bei einem Medienfehler vollständig wiederherzustellen

Weiteres I

- Startgröße einer DB-Datei mit SIZE-Parameter festlegen
- Maximale Größe mit MAXSIZE festlegen
- Automatische Erweiterung mit FILEGROWTH-Parameter

```
(NAME = FG2Dat2,  
FILENAME = N'G:\NDB1NDF\sekundaer.ndf',  
SIZE = 10 MB,  
MAXSIZE = 100 MB,  
FILEGROWTH = 10 MB), ...
```

- Fehler 1105, wenn kein Speicherplatz mehr zur Verfügung steht

Weiteres II

- DB-Dateien können auch verkleinert werden (auch .ldf)
- Werden vom Ende her verkleinert
 - Tabellen können fragmentieren, ev. Reorganisation nötig
- Row-Kompression: Daten fester Größe werden als Daten variabler Größe gespeichert
- Page-Kompression: Ähnlich dem ZIP-Verfahren

Datenbanken ändern

- Datenbankoptionen

```
ALTER DATABASE <db_name> SET OPTION <Wert>;
```

- Dateigruppen

```
ALTER DATABASE <db_name>  
{ <add_or_modify_files> | < add_or_modify_filegroups> }
```

- Name ändern

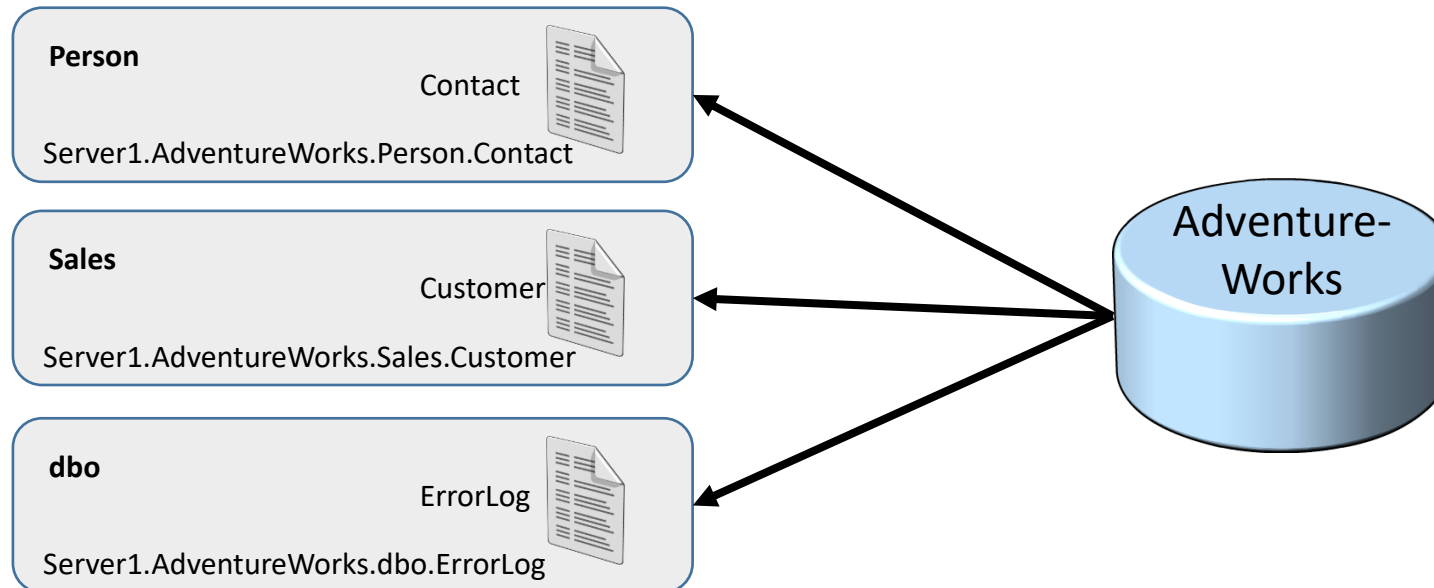
```
ALTER DATABASE <db_altname> MODIFY NAME = <db_neuname>
```

- Datenbank löschen

```
DROP DATABASE db_name
```

Schemas

- Schemas könnte man als „Container für Datenbankobjekte“ bezeichnen (oder namespaces)
- Standardschema dbo (= **D**atabase **o**wner)



Erstellen von Schemas

- Syntax:

```
CREATE SCHEMA  
Name | AUTHORIZATION Besitzer | Name AUTHORIZATION Besitzer  
[ Tabellendefinition | Sichtdefinition | GRANT-Anweisung |  
REVOKE-Anweisung | DENY-Anweisung ]
```

- Beispiel

```
CREATE SCHEMA Schema1 AUTHORIZATION SQLUSER1;
```

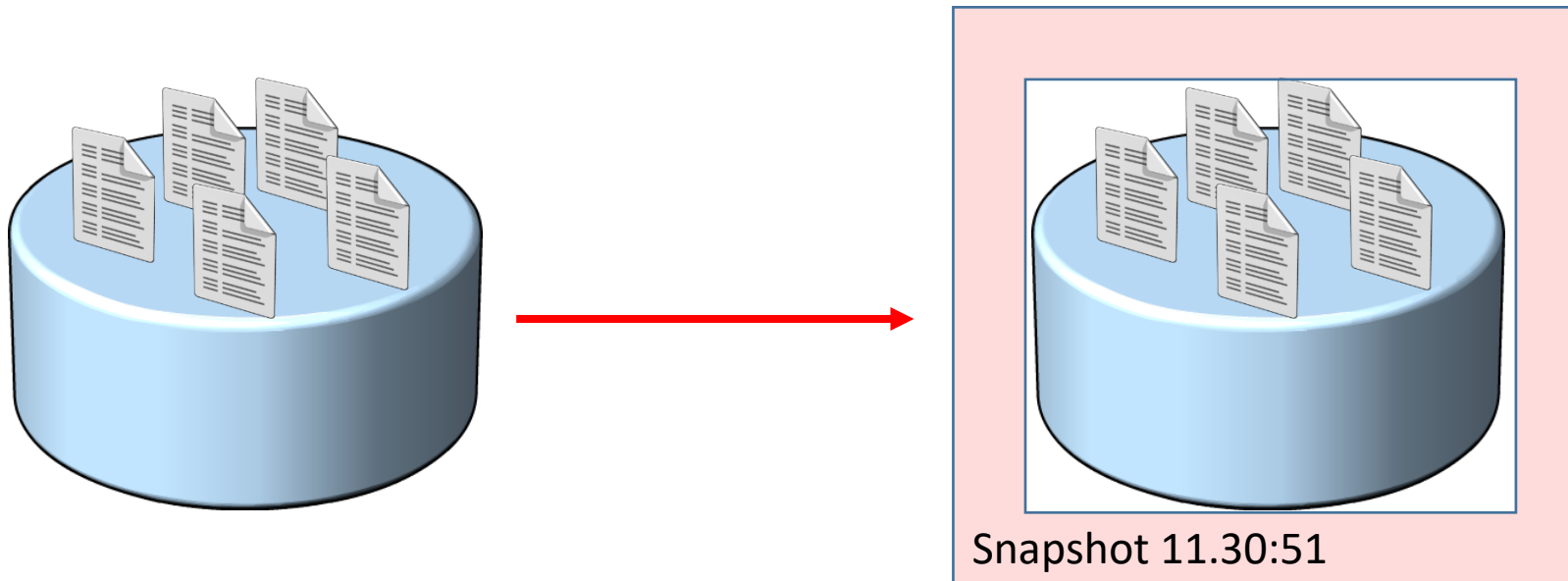
Collations

- Kollationen bestimmen die Sortierreihenfolge des SQL-Servers
- <https://msdn.microsoft.com/en-us/library/ms143726.aspx>

Option	Bedeutung
_CS	„case-sensitive“: es wird zwischen Groß- und Kleinschreibung unterschieden
_AS	„accent-sensitive“: Unterscheidung von Buchstaben mit Akzentzeichen von gleichen Buchstaben ohne Akzentzeichen
_KS	„kana-sensitive“: Unterscheidung zwischen den beiden Arten der japanischen Kana-Zeichen: Hiragana und Katakana
_WS	„width-sensitive“: Unterscheidet zwischen gleichen Zeichen, wenn sie einmal als Single-Byte und einmal als Doppel-Byte dargestellt sind. Ohne diese Option interpretiert SQL-Server die Zeichen als identisch

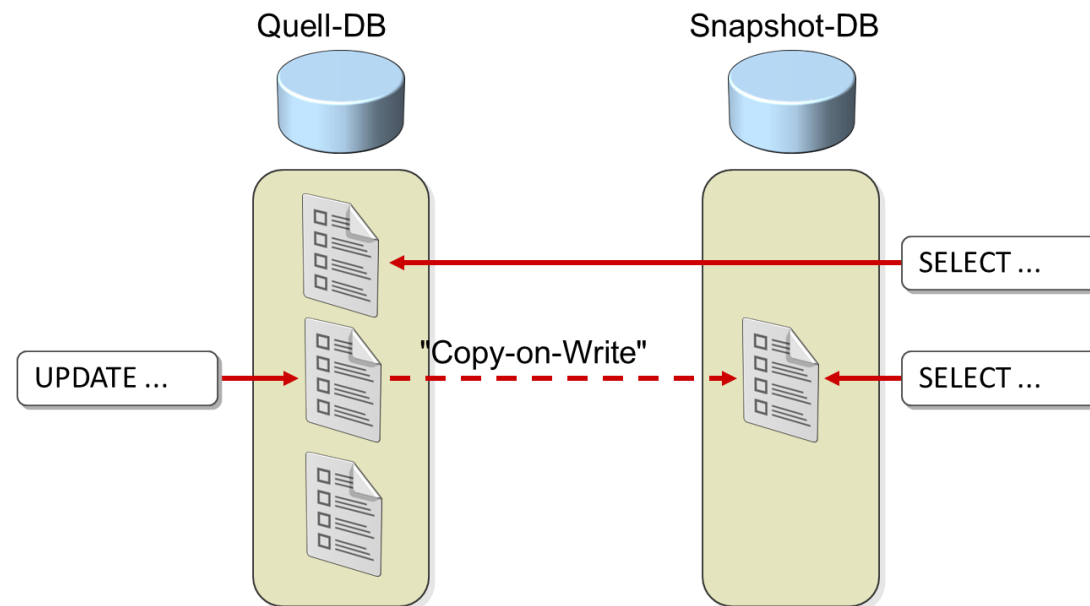
Snapshots

- Schreibgeschützte, konsistente Sicht einer Datenbank zu einem bestimmten Zeitpunkt
- Hilfreich als Test- oder Entwicklungs-DB und für Berichterstattung
- Muss auf demselben Server wie die Quelldatenbank liegen



Funktionsweise von Snapshots

- Snapshot ist statische Ansicht der Quelldatenbank zum Zeitpunkt des Snapshots
- Vor einer Veränderung einer Seite wird die Seite auf den Snapshot kopiert, d.h. Snapshot wächst mit jeder Änderung



Syntax

- Datenbanksnapshots können nicht über das SSMS direkt erstellt werden, sondern nur über T-SQL

```
CREATE DATABASE <snapshot-name>
(
    Dateiangabe
    [ , Dateiangabe [,...] ]
)
AS SNAPSHOT OF <Originalname>
;
```

Eigenschaften Snapshots

- Können nicht gesichert werden
- Müssen in derselben Instanz wie das Original liegen
- Unterstützen keine Volltextindizes und FILESTREAM-Daten
- Können nicht für System-Datenbanken angelegt werden
- Quelldatenbank kann nicht gelöscht, wiederhergestellt oder getrennt werden

Beispiel

- Snapshot erstellen

```
CREATE DATABASE Snapshot_Name
ON (
    NAME = Logischer_Dateiname,
    FILENAME = 'Betriebssystem_Dateiname'
)
AS SNAPSHOT OF Quelldatenbank_Name;
```

Logischer Dateiname der .mdf-Datei. Besteht die DB aus mehreren Dateien, müssen alle angegeben werden, ohne Log.

Dateien kann man mit

```
SELECT *
FROM sys.database_files
WHERE type_desc != 'LOG'
```

erfassen; Spalte *name* ist relevant

- Datenbank aus Snapshot wiederherstellen

```
RESTORE DATABASE Quelldatenbank_Name
FROM DATABASE_SNAPSHOT = 'Snapshot_Name';
```