

# **sqmPartitionTool**

Automatic SQL Server table partitioning, runbooks for daily DBA operations

dteSoftware · Uwe Janke · [www.powershelldb.de](http://www.powershelldb.de)  
Stand: 2026-10-02

# sqmPartitionTool — Admin-Handbuch

Automatische SQL-Server-Tabellen-Partitionierung: Konvertierung bestehender Tabellen, Sliding-Window-Wartung, Retention/Archivierung alter Partitionen und Migration ganzer Tabellen in eine separate Archiv-Datenbank — per GUI-Assistent oder vollstaendig per PowerShell/CLI.

Zielgruppe dieses Handbuchs: SQL-Server-DBAs, die das Tool operativ einsetzen (nicht die Entwicklung des Moduls selbst). Fuer die Versionshistorie siehe [CHANGELOG.md](#), fuer eine Kurzübersicht [README.md](#).

Stand: 2026-10-02, sqmPartitionTool 1.16.0.0 (mit sqmDataTransfer 0.1.23.0).

## Inhalt

- 1. Ueberblick: welcher Workflow passt zu meiner Situation?
- 2. Voraussetzungen und Installation
- 3. Ablaufplan A: Bestehende Tabelle in-place partitionieren
- 4. Ablaufplan B: Automatische Wartung einrichten (Sliding-Window + Retention)
- 5. Ablaufplan C: Tabelle in eine Archiv-Datenbank migrieren (Cutover)
- 5a. Ablaufplan D: Bereits partitionierte Tabelle mit neuer Partitionierung kopieren
- 5b. Wie die Daten kopiert werden (Kopier-Engine von sqmDataTransfer)
- 6. BoundaryType/SurrogateDateFormat — Referenz
- 7. GUI-Assistent: Schritt-fuer-Schritt
- 8. Troubleshooting und bekannte Einschränkungen
- 9. Sicherheitshinweise

## 1. Ueberblick

Das Modul deckt funf unterschiedliche, unabhaengig voneinander nutzbare Szenarien ab. Die Entscheidung, welches passt, haengt davon ab, **wo die Daten am Ende liegen sollen** und **ob die Tabelle aktiv bleibt**:

| Szenario                                  | Funktion                                                                   | Quelltabelle danach                                         | Wann sinnvoll                                                                          |
|-------------------------------------------|----------------------------------------------------------------------------|-------------------------------------------------------------|----------------------------------------------------------------------------------------|
| <b>A</b> —<br>In-Place-Partitionierung    | <code>Invoke-</code><br><code>sqmTablePartitionConversion</code>           | Bleibt in derselben DB, ist<br>jetzt partitioniert          | Tabelle soll partitioniert<br>werden, aber in derselben<br>Datenbank bleiben           |
| <b>B1</b> —<br>Sliding-Window-Erweiterung | <code>New-</code><br><code>sqmPartitionExtendJob</code><br>(SQL-Agent-Job) | Unveraendert (nur neue<br>leere Partitionen kommen<br>dazu) | Nach A: automatisch<br>dafuer sorgen, dass nie<br>"die letzte Partition"<br>volllaeuft |

|                                             |                                                                         |                                                                               |                                                                                                                                                                                                                                                                 |
|---------------------------------------------|-------------------------------------------------------------------------|-------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>B2</b> —<br>Retention/Archivierung       | <b>New-</b><br><code>sqlPartitionRetentionJob</code><br>(SQL-Agent-Job) | Alte Partitionen werden<br>geloescht oder vorher<br>archiviert                | Nach A: alte Daten nach<br>X Monaten/Jahren<br>automatisch entfernen                                                                                                                                                                                            |
| <b>C</b> — Archiv-DB-Migration<br>+ Cutover | <b>Invoke-</b><br><code>sqlTableArchiveMigration</code>                 | Umbenannt, durch eine<br>View auf die Archiv-DB<br>ersetzt                    | Ganze Tabelle soll<br>dauerhaft in eine andere<br>(typischerweise<br>kleinere/langsamer<br>angebundene) Datenbank<br>umziehen,<br>Anwendungscode aber<br>unveraendert weiterlaufen                                                                              |
| <b>D</b> — Neu-partitionierte<br>Kopie      | <b>Copy-</b><br><code>sqlPartitionedTable</code>                        | <b>Unveraendert, bleibt<br/>aktiv</b> (keine<br>Umbenennung, kein<br>Cutover) | Eine <b>bereits<br/>partitionierte</b> Tabelle soll<br>zusaeztlich als<br>eigenstaendige Kopie mit<br><b>anderer</b><br>Granularitaet/Filegroup-Strategie<br>in einer anderen<br>Datenbank existieren<br>(z.B. Reporting-Abzug mit<br>groebeerer Granularitaet) |

**Faustregel:** Wenn die Tabelle **in der Quelldatenbank bleiben** soll → A (+ optional B1/B2). Wenn die Tabelle **komplett in eine andere Datenbank** soll (z.B. Archiv-Instanz, separate Datenbank mit weniger Backup-/Storage-Anforderungen) → C. Wenn die Quelltable **bereits partitioniert ist** und **parallel** mit einer anderen Partitionierung anderswo weiterexistieren soll (kein Cutover, keine Umbenennung) → D.

## 2. Voraussetzungen und Installation

- PowerShell 5.1 oder hoeher (GUI benoetigt Desktop-CLR/WinForms, unter PowerShell 7 auf Windows weiterhin verfuegbar, nicht aber auf PowerShell 7 unter Linux/macOS).
- Module `dbatools`, `sqlSQLTool` (>= 1.9.2.0) und `sqlDataTransfer` (>= **0.1.23.0**). `sqlDataTransfer` liefert seit 1.15.0.0 die Kopier-Engine fuer Ablaufplan C, D und die Relocation (siehe Abschnitt 5b). `sqlSQLTool` und `sqlDataTransfer` liegen nicht auf der PowerShell Gallery und muessen **vorher** installiert werden.
- Ein SQL-Server-Login mit ausreichenden Rechten auf der/den Zieldatenbank(en): `ALTER` auf die betroffene(n) Tabelle(n)/Datenbank(en), `CREATE/ALTER PROCEDURE`, sowie fuer die SQL-Agent-Jobs (Szenario B) Rechte auf `msdb`.
- Fuer Szenario C und D: die **Zieldatenbank muss vom Admin vorher angelegt sein**, das Tool legt sie nicht automatisch an (bewusste Entscheidung, da Dateigroessen, Pfade und Recovery-Modell admin-spezifisch sind).

**Installation in dieser Reihenfolge** (als Administrator, aus den jeweiligen Repository-Ordern):

```
# je Repository-Ordner, als Administrator
.\sqlSQLTool\Install.ps1 -Scope AllUsers
.\sqlDataTransfer\Install.ps1 -Scope AllUsers
.\sqlPartitionTool\Install.ps1 -Scope AllUsers
# bei gesperrter Ausfuehrungsrichtlinie jeweils:
# powershell.exe -NoProfile -ExecutionPolicy Bypass -File .\<Modul>\Install.ps1 -Scope AllUsers
```

Der Installer von sqmPartitionTool prueft, ob sqmSQLTool und sqmDataTransfer im selben Scope vorhanden und aktuell genug sind, und nennt fehlende oder veraltete Module. `dbatools` wird bei Bedarf von der Gallery nachinstalliert. Welche Version tatsaechlich laeuft, zeigt der GUI-Assistent im Fenstertitel (Version und Ladepfad), per Konsole:

```
Get-Module sqmPartitionTool, sqmDataTransfer -ListAvailable |  
Select-Object Name, Version, ModuleBase
```

Verbindung erfolgt wahlweise per Windows-Authentifizierung (Standard) oder SQL Server Authentifizierung (`-SqlCredential`), auch im GUI-Assistenten in Schritt 0 waehlbar).

---

### 3. Ablaufplan A: Bestehende Tabelle in-place partitionieren

---

Ziel: eine bestehende, nicht partitionierte Tabelle wird in derselben Datenbank partitioniert.

- 1. Pruefen, ob die Tabelle geeignet ist.** `powershell Test-sqmPartitionReadiness -SqlInstance "SQL01" -Database "Sales" -Schema "dbo" -Table "OrderHistory" -PartitionColumn "OrderDate"` Meldet u.a., ob ein `-AllowKeyChange` noetig ist (PRIMARY KEY/UNIQUE-Constraint muss um die Partitionsspalte erweitert werden) und ob eingehende Fremdschlüssel/Trigger vorhanden sind (relevant fuer `-Method BatchedSwap`, siehe unten).
- 2. Granularitaet und Filegroup-Strategie festlegen** — Month/Quarter/Year, sowie `Single` (eine Filegroup fuer alle Partitionen) oder `PerPeriod` (eine eigene Filegroup je Periode — bessere I/O-Isolation, mehr Verwaltungsaufwand). Mit `-FilePath` landen die neuen Filegroup-Dateien in einem Verzeichnis **auf dem SQL Server** nach Wahl, z.B. auf einem eigenen Laufwerk (`-FilePath 'G:\SQLData\Partitions'`); ein fehlendes Verzeichnis wird angelegt, ein Laufwerk, das es auf dem Server nicht gibt, bricht vor jeder Aenderung ab. Ohne Angabe: Standard-Datenpfad der Instanz. Gilt nur fuer **neu** angelegte Filegroups, eine vorhandene wird nicht verschoben. Derselbe Parameter steht bei `Invoke-sqmTableArchiveMigration` und `Copy-sqmPartitionedTable` zur Veruegung.
- 3. Konvertierung ausfuehren:** `powershell Invoke-sqmTablePartitionConversion -SqlInstance "SQL01" -Database "Sales" -Schema "dbo" -Table "OrderHistory" -PartitionColumn "OrderDate" -Granularity Month -FilegroupStrategy Single -AllowKeyChange` Je nach Ausgangslage waehlt die Funktion automatisch die passende Umsetzung: - Clustered Index/PK ohne Partitionsspalte im Schluessel → `ALTER TABLE ... DROP/ADD CONSTRAINT`. - Clustered Index bereits passend → `CREATE CLUSTERED INDEX ... WITH (DROP_EXISTING=ON)`. - Heap → neuer Clustered Index direkt auf dem Partition Scheme, oder `-Method NewTableSwap` fuer sehr grosse Heaps. - `-Method BatchedSwap` (Heap **und** indizierte/PK-Tabellen): fuer sehr grosse Tabellen auf Datentraegern mit **wenig freiem Speicherplatz** — baut eine neue, leere partitionierte Kopie auf und verschiebt die Daten segmentweise (je Boundary-Periode, weiter unterteilt in `-BatchSize`), mit periodischem `DBCC SHRINKFILE` waehrend die alte Tabelle sich leert. **Einschraenkung:** bricht mit Fehler ab, wenn die Tabelle eingehende Fremdschlüssel oder Trigger hat — da fuer `-Method Default/NewTableSwap` verwenden.
- 4. Ergebnis pruefen:** `powershell Get-sqmPartitionStatus -SqlInstance "SQL01" -Database "Sales" -Schema "dbo" -Table "OrderHistory"`
5. Die Tabelle wird automatisch in `master.dbo.sqm_PartitionRegistry` registriert (ausser `-NoRegister`) — Voraussetzung fuer Szenario B.

---

### 4. Ablaufplan B: Automatische Wartung einrichten

---

Setzt voraus, dass die Tabelle bereits partitioniert und registriert ist (Ablaufplan A).

1. **Sliding-Window-Erweiterung** (sorgt dafuer, dass immer ein paar Perioden im Voraus als leere Partitionen existieren, -FutureBufferPeriods aus Ablaufplan A): powershell `New-sqmPartitionExtendJob -SqlInstance "SQL01"` Legt einen SQL-Agent-Job an, der `sqm_ExtendPartitionWindow` (T-SQL-Prozedur, liest die Registry) periodisch ausfuehrt — idempotent, mehrfache Ausfuehrung am selben Tag aendert nichts.

2. **Retention/Archivierung** (entfernt Partitionen, die aelter als die konfigurierte Aufbewahrungsfrist sind):

powershell `Register-sqmPartitionTable -SqlInstance "SQL01" -Database "Sales" -Schema "dbo" -Table "OrderHistory" -PartitionColumn "OrderDate" -PartitionFunctionName "..." -PartitionSchemeName "..." -Granularity Month -BoundaryType Date -RetentionValue 36 -RetentionUnit Months -ArchiveEnabled` Mit `-ArchiveDatabaseName "SalesArchive"` `New-sqmPartitionRetentionJob -SqlInstance "SQL01"` Mit `-ArchiveEnabled` werden ablaufende Partitionen **vor** dem Entfernen per `Invoke-sqmPartitionArchive` in die angegebene Archiv-Datenbank kopiert (SWITCH PARTITION, Batch-Kopie, dann MERGE RANGE) statt einfach geloescht zu werden. Ohne `-ArchiveEnabled` werden abgelaufene Partitionen unwiderruflich geloescht — vorher testen!

3. **Status/Fortschritt pruefen:** powershell `Get-sqmPartitionRegistry -SqlInstance "SQL01"`

**Wichtig:** dieser Pfad betrifft **einzelne, nach und nach ablaufende Partitionen** einer weiterhin aktiven, partitionierten Tabelle — nicht die gesamte Tabelle auf einmal. Fuer eine sofortige Komplettmigration der ganzen Tabelle siehe Ablaufplan C.

## Ad-hoc statt Job: sofortige Retention fuer eine einzelne Tabelle

`New-sqmPartitionRetentionJob` deckt den Dauerbetrieb ab (woechentlich, alle registrierten Tabellen). Fuer einen sofortigen, einmaligen Lauf — Test vor dem Einrichten des Jobs, eine Notfall-Bereinigung ("wir brauchen JETZT Platz"), oder eine Tabelle, die (noch) gar nicht registriert ist — gibt es `Invoke-sqmPartitionRetention`:

```
Invoke-sqmPartitionRetention -SqlInstance "SQL01" -Database "Sales" -Schema "dbo" -Table "OrderHistory" -RetentionValue 120 -RetentionUnit Months
```

Entfernt sofort alle Partitionen aelter als der angegebene Cutoff (hier: 120 Monate = 10 Jahre), per derselben SWITCH-PARTITION-+-MERGE-RANGE-Logik wie oben (`Invoke-sqmPartitionArchive` dahinter), optional mit `-ArchiveDatabaseName` fuer eine Kopie vor dem Entfernen. Braucht **keine** vorherige Registrierung — die Grenzwerte jeder Partition werden direkt gelesen und ausgewertet. Eine einzige Bestaetigungsabfrage fuer den ganzen Lauf (zeigt vorab die Anzahl betroffener Partitionen), nicht eine pro Partition. `-WhatIf` zeigt, wie viele Partitionen betroffen waeren, ohne etwas zu aendern.

---

## 5. Ablaufplan C: Tabelle in eine Archiv-Datenbank migrieren

Ziel: eine (noch nicht partitionierte) aktive Tabelle wird **komplett** in eine partitionierte Kopie in einer separaten Archiv-Datenbank ueberfuehrt. Am Ende wird die Quelltable umbenannt und durch eine Kompatibilitaets-View ersetzt, sodass bestehender Anwendungscode unveraendert weiter auf denselben Tabellennamen zugreifen kann (jetzt transparent gegen die Archiv-Kopie).

### Was beim Aufruf passiert:

- Beim ersten Aufruf wird eine leere Strukturkopie in der Archiv-Datenbank angelegt und monatsweise partitioniert, ueber den tatsaechlichen Wertebereich der Quelle (intern dieselbe Logik wie Ablaufplan A). Die Kopie wird automatisch fuer die Sliding-Window-Wartung registriert. `-PrimaryKeyFromUniqueIndex <Index>` macht einen eindeutigen Nonclustered Index der Quelle zum PRIMARY KEY CLUSTERED der Archiv-Kopie; ohne diese Option hat die Kopie keinen eindeutigen Schluessel.

- Danach wird **ein Monat pro Chunk** mit der Kopier-Engine von sqmDataTransfer uebertragen (Copy-sqmTableData, SqlBulkCopy, siehe Abschnitt 5b). Auf der Konsole erscheinen je Monat Zeilen, Dauer und Zeilen/s.
- **Fortsetzen nach Abbruch:** jeder Aufruf zaehlt die Zeilen je Monat in Quelle und Archiv (je ein GROUP BY). Ein Monat mit gleicher Zahl wird uebersprungen, ein Monat mit abweichender Zahl (Rest eines abgebrochenen Laufs) im Archiv geleert und neu kopiert. Ein Schluessel ist dafuer nicht noetig. Ein Monat gilt erst nach bestandener Zeilenzahl-Pruefung als Completed in dbo.sqm\_ArchiveMonthLog (Quelldatenbank).
- Standardmaessig endet die Migration beim **Vormonat**. Mit **-IncludeOpenPeriods** wird **alles** uebertragen, auch der laufende Monat und spaetere Monate mit Daten. Offene Monate werden bei jedem Aufruf komplett neu kopiert, so kommen auch Zeilen nach, die nach dem ersten Lauf in die weiterhin aktive Quelle geschrieben oder dort geaendert wurden. Sie bleiben im Log InProgress.
- **-PurgeSourceAfterArchive** loescht jeden abgeschlossenen Monat nach der Zeilenzahl-Pruefung aus der Quelle (in Batches) und gibt den Platz per DBCC SHRINKFILE zurueck (-ShrinkAfterEveryNPeriods, -AggressiveShrink). Offene Monate werden **nie** geloescht: zwischen Zaehlung und DELETE eintreffende Zeilen gingen sonst verloren.
- **-CutoverToArchiveView** benennt die Quelltable am Ende um (Suffix \_Original, -RenamedTableSuffix) und legt unter dem alten Namen eine View auf die Archiv-Kopie an. Mit **-IncludeOpenPeriods** laufen letzter Abgleich der offenen Monate, Umbenennen und View-Anlage in **einer Transaktion unter exklusiver Tabellensperre**: zwischen letztem Chunk und Umbenennen kann keine Zeile verloren gehen. Dafuer braucht dieser letzte Abgleich einen eindeutigen Schluessel (-PrimaryKeyFromUniqueIndex oder -KeyColumn, bis 5 Spalten). Die umbenannte Original-Tabelle wird **nie** automatisch geloescht.
- **Selbstheilung nach frueheren Versuchen:** existiert die Archiv-Tabelle nicht (mehr), werden veraltete Log-Eintraege dieser Tabelle zurueckgesetzt, statt Monate faelschlich als erledigt zu ueberspringen. Eine unbenutzte Partition Function/Scheme gleichen Namens aus einem frueheren Lauf wird mit den aktuellen Grenzen neu angelegt statt wiederverwendet.

**Ablauf ohne GUI, Schritt fuer Schritt.** Dieselbe Funktion wird mehrfach mit denselben Grundparametern aufgerufen, jeder Aufruf setzt dort fort, wo der vorige aufgehoert hat. Nur Schritt 5 ist nicht ohne Weiteres rueckgaengig zu machen.

```
# 0. Gemeinsame Parameter
Import-Module sqmPartitionTool
$p = @{
    SqlInstance      = 'SQL01'
    Database         = 'Sales'
    Schema           = 'dbo'
    Table            = 'Bookings'
    ArchiveDatabaseName = 'SalesArchive'
    DateColumn       = 'BOOKDATE'      # INT im Format YYYYMMDD
    PrimaryKeyFromUniqueIndex = 'UX_Bookings' # wird PK der Archiv-Kopie
    IncludeOpenPeriods = $true          # alles, inkl. laufendem Monat
}

# 1. Pruefen, ohne etwas zu aendern
Test-sqmPartitionReadiness -SqlInstance SQL01 -Database Sales -Schema dbo `
    -Table Bookings -PartitionColumn BOOKDATE
Invoke-sqmTableArchiveMigration @p -WhatIf

# 2. Optional: nur die partitionierte Archiv-Tabelle anlegen und ansehen
Invoke-sqmTableArchiveMigration @p -CreateArchiveTableOnly

# 3. Daten uebertragen, beliebig oft wiederholbar (setzt fort, gleicht den offenen Monat neu ab)
Invoke-sqmTableArchiveMigration @p
#   in Etappen:      -StartPeriod 202401 -EndPeriod 202406
#   wenig Plattenplatz: -PurgeSourceAfterArchive

# 4. Kontrolle
Get-sqmPartitionStatus -SqlInstance SQL01 -Database SalesArchive -Schema dbo -Table Bookings

# 5. Abschluss: letzter Abgleich + Umbenennen + View in einer Transaktion
```

```
Invoke-sqmTableArchiveMigration @p -CutoverToArchiveView
```

```
# 6. Kuenftige Monatspartitionen automatisch anlegen  
New-sqmPartitionExtendJob -SqlInstance SQL01
```

**Fortschritt beobachten:** neben der Konsolenausgabe je Monat laesst sich der Stand jederzeit direkt abfragen:

```
SELECT YYYYMM, Status, RowsArchived, StartedAt, CompletedAt  
FROM dbo.sqm_ArchiveMonthLog          -- in der Quelldatenbank  
WHERE TableName = N'Bookings'  
ORDER BY YYYYMM;
```

**Nach erfolgreichem Cutover:** die umbenannte Original-Tabelle (`Bookings_Original`) nach Pruefung manuell entfernen. Ohne `-IncludeOpenPeriods` enthaelt sie noch den zuletzt offenen Monat.

---

## Wann `-Method BatchedSwap` (Ablaufplan A) statt Ablaufplan C?

`-Method BatchedSwap` partitioniert **innerhalb derselben Datenbank** (kein Datenbankwechsel, schnellere Batches da keine Cross-DB-Kommunikation noetig) — geeignet, wenn die Tabelle in der Quelldatenbank bleiben soll, aber wenig Plattenplatz fuer eine klassische Ein-Schritt-Konvertierung vorhanden ist. Ablaufplan C ist die richtige Wahl, wenn die Daten **dauerhaft in eine andere Datenbank** sollen (typischerweise eine separate, guenstiger/anders gesicherte Archiv-Instanz).

---

## 5a. Ablaufplan D: Bereits partitionierte Tabelle mit neuer Partitionierung kopieren

Ziel: eine **bereits partitionierte**, weiterhin aktive Tabelle soll zusaetzlich (nicht statt dessen) als eigenstaendige, **neu partitionierte** Kopie in einer anderen Datenbank derselben Instanz existieren, z.B. mit groeerer Granularitaet fuer Reporting, oder als Testabzug vor einer geplanten Umstellung der Produktionstabelle. Im Unterschied zu Ablaufplan C gibt es **keinen Cutover**: die Quelltable wird nie umbenannt, nie durch eine View ersetzt und bleibt vollstaendig unveraendert.

1. **Zielfdatenbank anlegen** (Admin-Aufgabe, nicht automatisiert, gleiche Begrueendung wie bei C).
2. **Kopie starten** (ohne `-TargetTableName` bekommt die Kopie den Namen der Quelle): powershell  
`Copy-sqmPartitionedTable -SqlInstance "SQL01" -Database "Sales" -Schema "dbo" -Table "OrderHistory" -TargetDatabaseName "SalesReporting" -Granularity Year -Confirm:$false`  
Ablauf im Detail: - Prueft, dass die Quelltable tatsaechlich bereits partitioniert ist (sonst Fehler mit Verweis auf Ablaufplan A/C) und leitet die Partitionsspalte aus dem bestehenden Partition Scheme ab, sofern `-PartitionColumn` nicht ausdruuecklich eine andere Spalte vorgibt. - Legt beim ersten Aufruf die neue Partitionierung (Filegroups, Partition Function/Scheme) in der Zielfdatenbank an und erstellt dort eine strukturell identische Tabelle (Spalten, Indizes, PK/UNIQUE-Constraints; Fremdschluessel und Trigger werden **nicht** mitgenommen). - Kopiert **eine Partition der neuen Zieltabelle pro Chunk** mit der Kopier-Engine von `sqmDataTransfer` (Abschnitt 5b). Gezaehlt wird je Partition ueber `$PARTITION` der neuen Partition Function: vollstaendige Partitionen werden uebersprungen, eine unvollstaendige wird im Ziel geleert und neu kopiert. **Kein Schluessel noetig**; ein angegebenes `-KeyColumn` wird seit 1.15.0.0 ignoriert (Warnung). - `-MaxDurationMinutes` beendet den Lauf sauber zwischen zwei Partitionen, ein erneuter Aufruf setzt fort. - `-CreateTableOnly` legt nur die neu partitionierte, leere Zieltabelle an (keine Daten, keine Registrierung), z.B. um sie vor der Kopie zu pruefen oder die Kopie in ein spaeteres Wartungsfenster zu legen. Ein Aufruf ohne den Schalter kopiert dann in die vorhandene Tabelle. Existiert die Zieltabelle schon, passiert nichts (Status `TargetTableExists`). - `-FilePath` legt die neuen Filegroups der Zielfdatenbank auf ein Laufwerk der Wahl: powershell `Copy-sqmPartitionedTable -SqlInstance "SQL01" -Database "Sales" -Schema`

```
"dbo" -Table "OrderHistory" ` -TargetDatabaseName "SalesReporting" -Granularity Year -FilePath 'G:\SQLData\Reporting' -CreateTableOnly - Registriert die neue Tabelle in sqm_PartitionRegistry (ausser -NoRegister), Ablaufplan B1 kann danach wie fuer jede andere partitionierte Tabelle eingerichtet werden.
```

3. **Verifikation:** Zeilenzahlen von Quelle und Kopie werden am Ende abgeglichen. Weichen sie ab (z.B. weil waehrend der Kopie neue Zeilen in die weiterhin aktive Quelle kamen), meldet die Funktion einen Fehler; ein erneuter Aufruf kopiert die Differenz nach.

## 5b. Wie die Daten kopiert werden

Seit 1.15.0.0 laufen alle Kopien **zwischen Datenbanken** ueber die Kopier-Engine von **sqmDataTransfer**, dieselbe Strecke wie dessen Chunk-Transfer, die produktiv mit Tabellen von mehreren hundert Millionen Zeilen eingesetzt wird. Vorher nutzte sqmPartitionTool eigene T-SQL-Batches (MERGE bzw. `Keyset-INSERT ... SELECT TOP`); ohne passenden Index las jeder Batch die ganze Tabelle, und auf der Konsole kam stundenlang keine Rueckmeldung.

Was die Engine mitbringt:

- **SqlBulkCopy** mit expliziter Spaltenzuordnung ueber den Namen (eine berechnete Spalte mitten in der Tabelle kann keine Spalten verschieben).
- Bricht ein Chunk ab, wird die Quellabfrage auf dem Server **abgebrochen**, statt den Rest des Chunks noch uebers Netz zu lesen.
- Batchgroesse bei Columnstore-Zielen automatisch gedeckelt.
- Fortschritt je Chunk (`Write-Progress`) plus eine Konsolenzeile mit Zeilen/s.
- **Lesen per Index-Seek statt Scan je Chunk** (seit 1.15.1.0): auf einem Heap bzw. einem Clustered Index, der nicht mit der Datums-/Partitionsspalte beginnt, liest SQL Server fuer `WHERE <Monat>` sonst bei grossen Tabellen die ganze Tabelle je Chunk. Gibt es einen Index mit der Spalte vorne und mindestens 4 Chunks, wird jeder Chunk `WITH (FORCESEEK)` gelesen. Ohne einen solchen Index vor einem grossen Lauf einen anlegen.

| Funktion                                     | Ein Chunk ist                                                                                                        | Fortsetzen nach Abbruch                                    |
|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| <code>Invoke-sqmTableArchiveMigration</code> | ein Kalendermonat                                                                                                    | Zeilenzahl je Monat, Quelle gegen Archiv                   |
| <code>Copy-sqmPartitionedTable</code>        | eine Partition der neuen Zieltabelle                                                                                 | Zeilenzahl je Zielpartition ueber <code>\$PARTITION</code> |
| <code>Invoke-sqmTableRelocation</code>       | automatisch erkannte Chunk-Spalte (Datum, Periode oder <code>YYYYMMDD</code> , monatsweise), sonst die ganze Tabelle | Zeilenzahl je Chunk                                        |

In allen drei Faellen wird ein Chunk mit gleicher Zeilenzahl auf beiden Seiten uebersprungen und ein Chunk mit abweichender Zahl im Ziel geleert und neu kopiert. Ohne `-BatchSize` gilt die Standard-Batchgroesse von sqmDataTransfer (`Get-sqmTransferConfig DefaultBatchSize`, 500000).

**Bewusst unveraendert serverseitig:** die Retention-Archivierung einzelner abgelaufener Partitionen (`Invoke-sqmPartitionArchive`, Staging-Tabelle in die Archiv-DB derselben Instanz per `DELETE ... OUTPUT INTO`,

transaktional je Batch, laeuft unbeaufsichtigt im Retention-Job) und die In-Place-Umwandlung `-Method NewTableSwap` (`INSERT ... WITH (TABLOCK)` in derselben Datenbank). Ueber den Client waeren beide nur ein zusaetzlicher Netzweg.

### Praxis bei sehr grossen Tabellen:

- Den Lauf **auf dem SQL-Server-Host selbst** starten. SqlBulkCopy liest ueber den Client; von einer Workstation aus geht jede Zeile zweimal uebers Netz.
- `ASYNC_NETWORK_IO` an der **schreibenden** Session (`INSERT BULK` im Ziel) heisst: das Ziel wartet auf den Client, der Engpass liegt beim Lesen der Quelle oder im Client. Pruefen mit `Docs/Diagnose-ChunkTransfer-Quelle.sql` aus `sqmDataTransfer`.
- `ASYNC_NETWORK_IO` an der **lesenden** Session ist normal, solange das Schreiben langsamer ist als das Lesen. Der eigentliche Engpass zeigt sich an der **schreibenden** Session (`INSERT BULK`): `WRITELOG` bzw. Log-Wachstum (bei FULL Recovery wird jede Zeile protokolliert, Log vorher passend vergroessern), `PAGEIOLATCH` (Storage), `LCK_M_*` (Blockierung).
- Waehrend einer Kopie **keine Index-Wartung, kein TRUNCATE und kein Partitions-SPLIT** auf der Quelltable: die lesende Session haelt eine Schema-Sperre, die wartende DDL blockiert alles, was sich dahinter einreihet.

---

## 6. BoundaryType/SurrogateDateFormat — Referenz

Alle Partitionierungs-/Migrationsfunktionen unterstuetzen drei Spaltentypen fuer die Partitions-/Periodenspalte, ueber `-BoundaryType` (ohne Angabe automatisch aus dem SQL-Spaltentyp abgeleitet):

| BoundaryType | Passender SQL-Spaltentyp                    | Beispielwert                        | Hinweis                                          |
|--------------|---------------------------------------------|-------------------------------------|--------------------------------------------------|
| Date         | <code>date/datetime/datetime2/smallc</code> | <code>2024-01-15tetimeoffset</code> | Standardfall, keine weitere Konfiguration noetig |
| Int          | <code>int/bigint/smallint/tinyint</code>    | <code>20240115</code>               | Numerischer Datums-Surrogatschluessel            |
| Text         | <code>char/varchar/nchar/nvarchar</code>    | <code>'20240115'</code>             | String-Datums-Surrogatschluessel                 |

Bei `Int/Text` steuert `-SurrogateDateFormat` die Genauigkeit:

- `yyyyMMdd` (Standard) — Tagesgenauigkeit, z.B. `20240115`.
- `yyyyMMdd` — Monatsgenauigkeit ohne Tag, z.B. `202401` (typisch, wenn die Quellspalte selbst nur auf Monatsebene gefuehrt wird).

Beispiel: eine Tabelle mit einer `INT`-Spalte im Format `YYYYMMDD` (z.B. `BOOKDATE`) braucht `-BoundaryType Int`, ohne Angabe wird das aus dem Spaltentyp abgeleitet.

## 7. GUI-Assistent: Schritt-fuer-Schritt

```
Show-sqmPartitionToolGui -SqlInstance "SQL01"
```

| Schritt                      | Inhalt                                                                                                                                                                                                                                                               |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 — Connection               | Instanz + Authentifizierung (Windows oder SQL Server Login), Datenbank waehlen                                                                                                                                                                                       |
| 1 — Select Table             | Kandidatentabellen (mit Zeilenzahl/Groesse/Heap-oder-Clustered)                                                                                                                                                                                                      |
| 2 — Select Column            | Partitions-/Datumsspalte auswaehlen                                                                                                                                                                                                                                  |
| 3 — Min/Max Preview          | Tatsaechlicher Wertebereich der Quelldaten (oder manuelle Werte bei leerer Tabelle)                                                                                                                                                                                  |
| 4 — Granularity & Filegroups | Month/Quarter/Year, Single/PerPeriod, bei Nicht-Datumsspalten zusaetzlich Surrogate Date Format; <i>Data file folder (server)</i> : Verzeichnis fuer neue Filegroups, Auswahlliste mit den Laufwerken des Servers und deren freiem Platz (leer = Standard-Datenpfad) |
| 5 — Boundary Preview         | Berechnete Partitions Grenzen zur Kontrolle vor der Ausfuehrung                                                                                                                                                                                                      |
| 6 — Archive & Retention      | Zwei sich gegenseitig ausschliessende Modi (siehe unten)                                                                                                                                                                                                             |
| 7 — Summary & Execute        | Zusammenfassung, Ausfuehren-Button, Live-Log                                                                                                                                                                                                                         |

### Schritt 6 haengt von der gewaehlten Tabelle ab:

- **Noch nicht partitioniert:**
- **"Migrate to archive database now"** fuehrt Ablaufplan C aus. Optionen: *Create the partitioned archive table only* oder *Create the archive table AND transfer the data; Include the current month* (vorausgewaehlt, uebertraegt alles, siehe `-IncludeOpenPeriods`); *Delete each archived month from the source; rename the source table and replace it by a view*. Bei einem Heap mit passendem eindeutigem Index bietet der Assistent an, ihn zum Clustered PRIMARY KEY der Archiv-Tabelle zu machen. Ein "Key Column(s)"-Feld erscheint nur, wenn ein Schluessel gebraucht wird und nicht abgeleitet werden kann.
- **"Set up automatic maintenance"** fuehrt Ablaufplan A + B aus (in-place partitionieren, Sliding-Window, Retention). Bei aktivem "Migrate now" ist dieser Bereich ausgeblendet.
- **Bereits partitioniert:** Copy-Modus (Ablaufplan D) mit Zieldatenbank und optionalem Zielnamen, ohne Schluesselauswahl. Wahlweise *Create the partitioned target table only* (`-CreateTableOnly`) oder *Create the target table AND copy the data*.

"Finish" fragt nach, falls noch nichts ausgefuehrt wurde. Der Fenstertitel zeigt Modulversion und Ladepfad.

---

## 8. Troubleshooting und bekannte Einschränkungen

---

- **Modul lädt nicht ("required module sqmDataTransfer"):** sqmDataTransfer >= 0.1.23.0 fehlt im selben Scope. Zuerst sqmDataTransfer installieren, dann sqmPartitionTool (Abschnitt 2).
- **Die GUI verhält sich wie eine alte Version** (z.B. nur eine Schlüsselspalte erlaubt): es läuft eine ältere installierte Kopie. Der Fenstertitel zeigt Version und Ladepfad; `Get-Module sqmPartitionTool -ListAvailable` listet alle installierten Versionen.
- **'-KeyColumn' ist Pflicht** bei Ablaufplan C: nur noch beim Cutover mit `-IncludeOpenPeriods` auf einem Heap ohne eindeutigen Schlüssel. `-PrimaryKeyFromUniqueIndex` oder `-KeyColumn 'Spalte1','Spalte2',...` (max. 5) angeben.
- **"date ist inkompatibel mit int":** die Datumsspalte ist kein echter DATE/DATETIME-Typ, sondern ein numerischer/String-Surrogatschlüssel, `-BoundaryType Int/Text` (+ ggf. `-SurrogateDateFormat`) angeben, siehe Abschnitt 6.
- **Alle Zeilen landen in der letzten Partition der Archiv-Tabelle:** seit 1.13.0.0 behoben (eine unbenutzte alte Partition Function wird neu angelegt). Bei älteren Versionen die leere Archiv-Tabelle und die alte Function/Scheme vor dem Lauf manuell entfernen.
- **Monate fehlen im Archiv, obwohl das Log "Completed" zeigt:** seit 1.13.0.0 behoben (veraltete Log-Einträge werden zurückgesetzt, wenn die Archiv-Tabelle neu angelegt wird).
- **-Method BatchedSwap bricht mit Fehler ab** ("eingehende Fremdschlüssel/Trigger"): diese Methode unterstützt keine Tabellen, auf die andere Tabellen per Fremdschlüssel verweisen, oder die Trigger haben. Fremdschlüssel/Trigger vorher entfernen, oder `-Method Default/NewTableSwap`.
- **Performance bei sehr grossen Tabellen (Ablaufplan C):** ohne Index mit der Datumsspalte als führender Spalte liest jeder Monats-Chunk die ganze Tabelle. Das Tool warnt, legt aber keinen Index an (Admin-Entscheidung). Weitere Hinweise zu Wartezeiten und Log in Abschnitt 5b.
- **Umbenannte Original-Tabelle nach Cutover** (Ablaufplan C) wächst nicht weiter, enthält ohne `-IncludeOpenPeriods` aber noch den zuletzt offenen Monat, vor dem endgültigen Löschen prüfen.
- **-Method BatchedSwap und Ablaufplan C laufen nicht online:** beide sperren während der jeweiligen Batches kurzzeitig die betroffenen Zeilen/Partitionen, der atomare Cutover die ganze Quelltable für die Dauer des letzten Abgleichs. Für produktive Systeme Wartungsfenster oder Zeiten mit geringer Last einplanen.

---

## 9. Sicherheitshinweise

---

- Alle destruktiven Operationen (Retention-Löschung ohne `-ArchiveEnabled`, `-PurgeSourceAfterArchive`) löschen Daten unwiderruflich aus der Quelle — vor dem produktiven Einsatz an einer Testtabelle mit repräsentativen Daten ausprobieren.
- Die umbenannte Original-Tabelle (Ablaufplan C, `_Original-Suffix`) wird **nicht** automatisch gelöscht — das ist Absicht, kein Bug. Das Tool löscht generell keine Tabellen automatisch, sondern verlässt sich auf den Admin für den letzten, endgültigen Schritt.
- Für produktive Migrationen empfiehlt sich vorab ein vollständiges Backup der Quelldatenbank sowie ein Testlauf mit `-WhatIf` (alle Kernfunktionen unterstützen `SupportsShouldProcess`).
- SQL-Server-Authentifizierung (`-SqlCredential`) sollte nur über Mixed-Mode-Logins mit minimal notwendigen Rechten erfolgen, nicht über `sa`.