SQL Administration

Setup

Umgebung

- Virtualisierung
  - NUMA identisch in VM mit Host
  - Arbeitsspeicher nicht dynamaisch
  - Festplatte mit genügend großer Leistung
    Perfmon

    Disk reads/sec + disk writes/sec = **IOPS**

    Disk read bytes/sec + disk write bytes/sec = **DURCHSATZ**

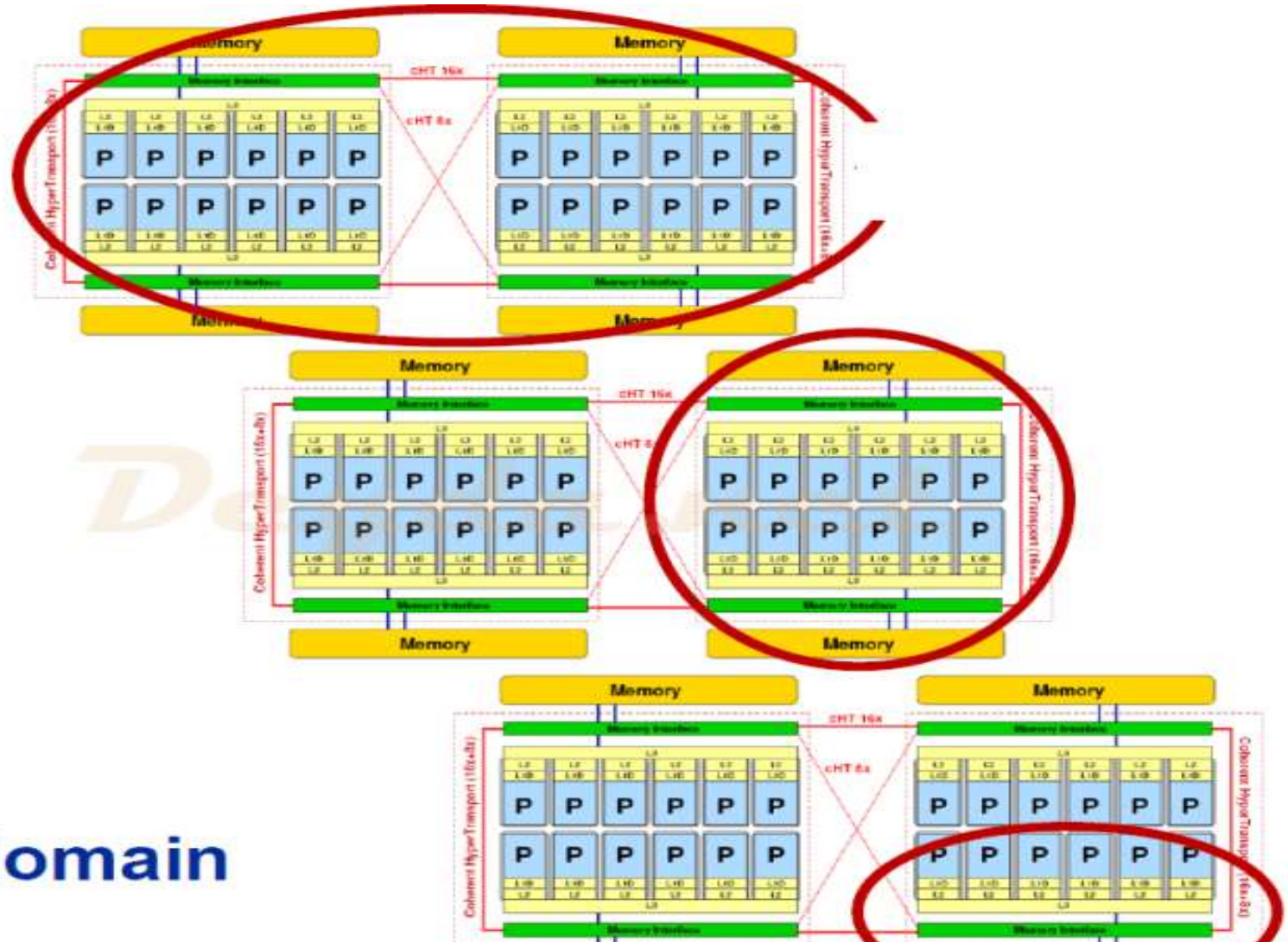
    Latenzzeiten erfassen

NUMA

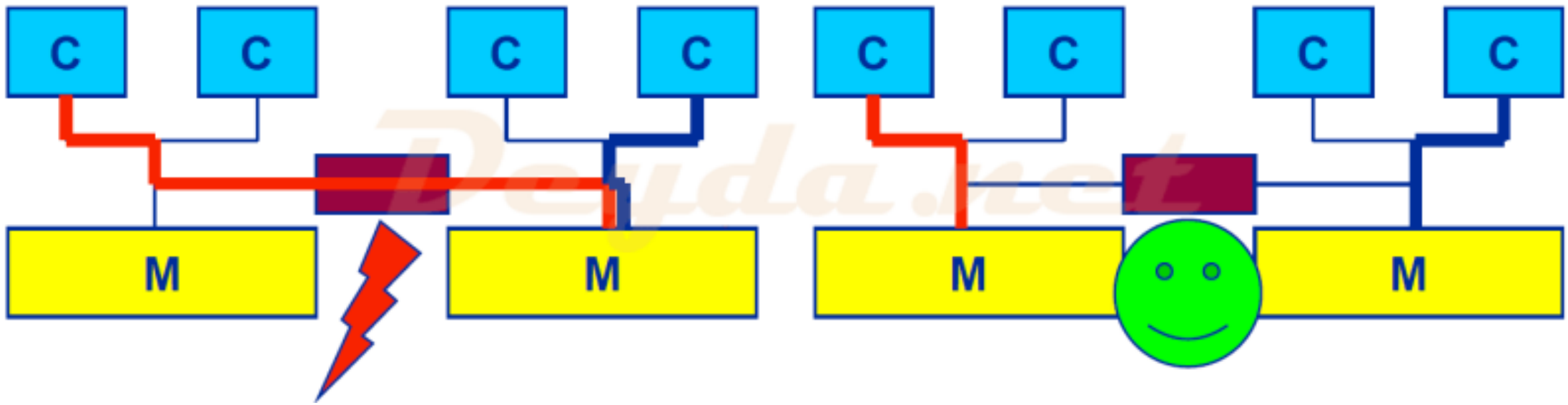
node

socket

NUMA domain



NUMA



Setup

- Lizenz
  - Versionen:
    - Developer entspricht Enterprise, nicht produktiv einsetzbar
    - Standard .. ab SQL 2016 SP1 fast Enterprise
    - Express 0 Euro – ab SQL 2016 SP1 fast Enterprise
    - WebEdition
    - Enterprise (alles drin)
- Mein Tipp: SQL 2016 mit SP1 oder höher
- Clientlizenzen (Prozessor oder CAL)

ServerSettings

- RAM
  - MIN RAM
    - Garantie für SQL Server
  - MAX RAM
    - Garantierter Speicher für andere (zB. OS)
- Datenbankeinstellungen
  - Trenne Daten von Logs
  - Kompression der Backups
- CPU
  - SQL verwaltet NUMA selbständig
- Aktivierung von Features

Arbeitsspeicher

- Für das OS die richtige Größe reservieren
  - 1 GB für OS + (4 bis 16GB)/4GB*1GB + (>16GB)/8GB*1GB

- 256 GB= 1GB + 12/4*1GB + 240/16*1GB
  = 1 + 3 + 15
  = **20 GB**

- 16 GB = ?

Datenträger Kontrolle

- Wichtige Werte
  - IOPS
  - Latenzzeiten
  - IOPS x Latenzzeit = Durchsatz

- DiskSpd
- Standardmessungen aktivieren
  - Perfmon

- TempDB Kontrolle

MAXDOP

- Anzahl der CPUs bei Abfragen
  - Abhängig von Kostenschwellwert
    - Default: 5
  - Anzahl der CPUs
    - Default: 0 (bis SQL 2017 inklusive)

- Besser:
  - Maximale Anzahl der CPUs auf Anzahl der Kerne max 8 beschränken
  - Kostenschwellwert: mit 25 (OLAP) bzw 50 (OLTP) beginnen
  - Tipp: ein wenig experimentieren. Der eingest. Wert zählt ab nächster Abfrage
  - Kein Neustart

Pfade

- Binärkram
  - C:\Program Files\.. OK!
- Datenbankdateien
  - DB besteht aus
    - .mdf (master datafile, Systemtabellen..)
    - .ldf (log datafile) .. Transaktionsprotokoll
    - TRENNE LOG VON DATEN PHYSIKALISCH (HDDs)! (Faustregel)
  - SystemDatenbanken
    - Master, model, msdb
  - Tempdb
    - Trenne Log und Daten und mehr!!! Datendateien

TempDB

- „Mülleimer“
  - #tabellen
  - Zeilenversionierung
  - Indizierung
- Sollte eigene HDDs haben
  - Log und Daten trennen
- bis zu 8 Datendateien
  - Soviele Dateien wie Kerne max 8
  - -T1117 alle Dateien sind immer gleich groß
  - -T1118 alle Blöcke gehören exklusiv Tabellen (uniform extents)

Security

- Authentifizierung
  - NT
  - NT + SQL Logins ➔ gemischte Authentifzierung
  - sa
    - Komplexes Kennwort (14 Zeichen)
    - Am besten deaktivieren und Ersatzkonto
- Windows Administratoren sind keine SQL Administratoren
- Datenträgervolumewartungstask
  - Ausnullen der Dateien während des Vergrößerns ist nicht mehr notwendig
    - SQL Dienst darf selbst vergößern ohne auszunullen

Instanzen

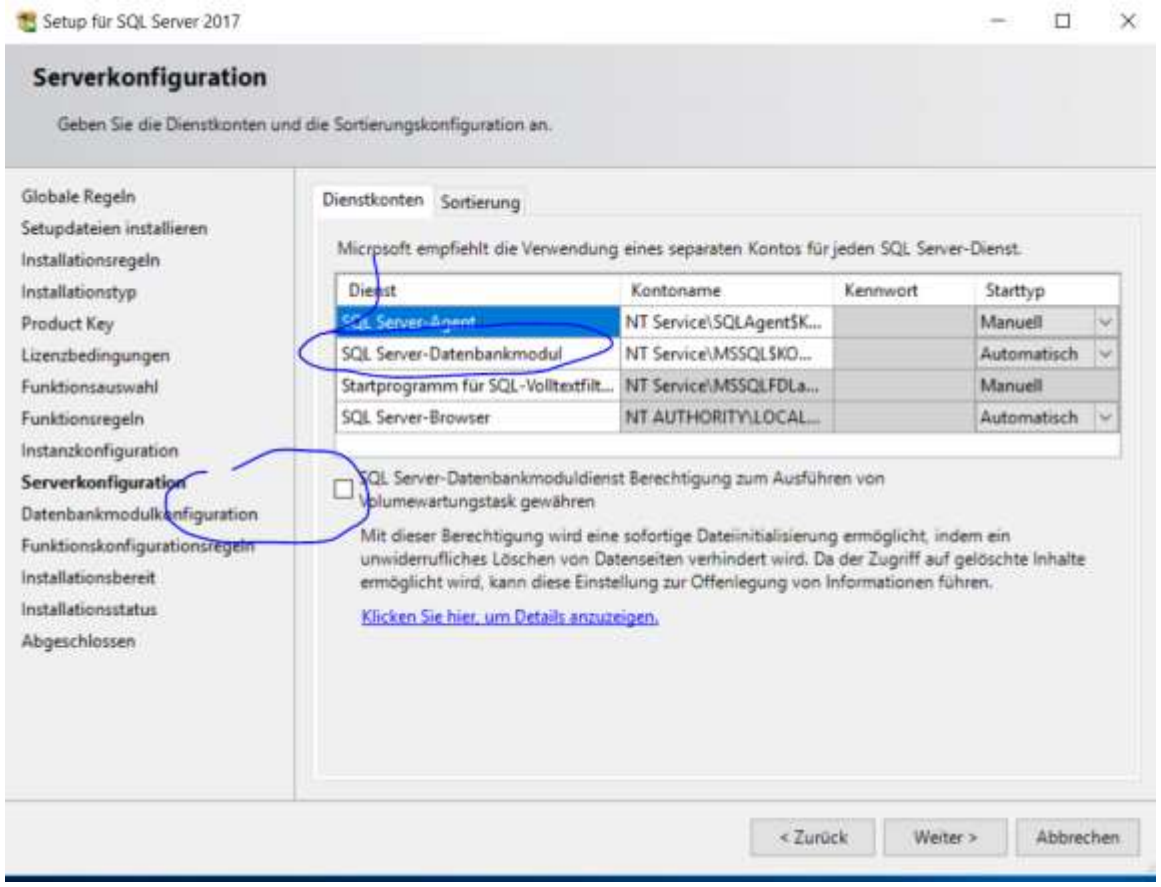
- SQL Server kann auf einem Rechner mehrfach (50) installiert werden
- Instanzen sind streng isoliert
- Haben eigene Prozesse, Arbeitsspeicher, SystemDBs
  - Sortierung
  - Man braucht evtl mehr Instanzen wg Versionen , Editionen

- Standardinstanz
  - Port: 1433 TCP Aufruf: „PC“
  - Benannt. Instanz: ??? (random) Aufruf „PC\Instanzname“
  - Browser regelt Zugriffe auf random Port (Browserdienst: 1434 UDP)

    https://docs.microsoft.com/de-de/sql/sql-server/install/work-with-multiple-versions-and-instances-of-sql-server?view=sql-server-ver15

Lokale Sicherheitsrichtlinie

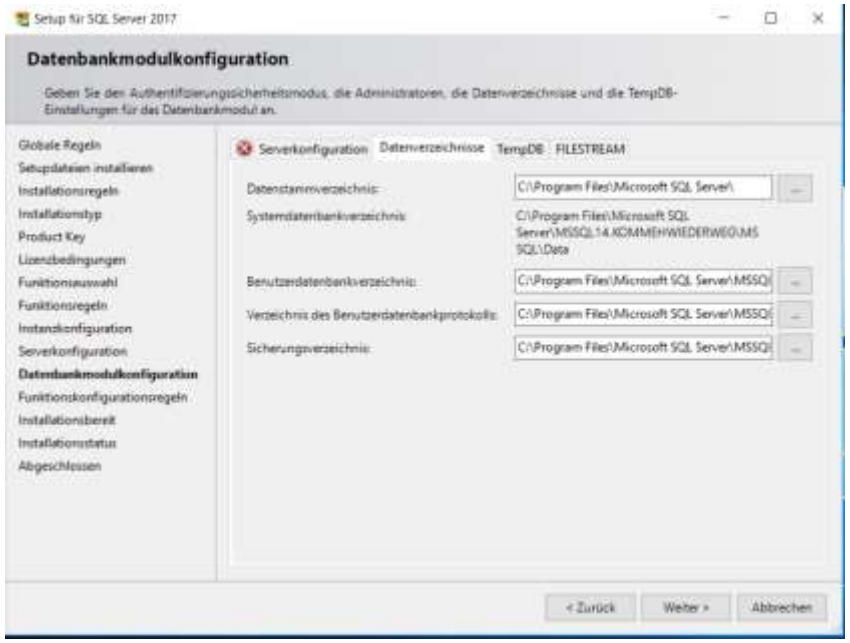
- Ausnullen



Setup

- Sortierung
  - ….CS… Groß und Kleinschreibung!
  - …CI….
- Filestream
  - \\PC\XY\Akten --> Filetable
  - Backup und Security in SQL Server

Datenbankverzeichnisse

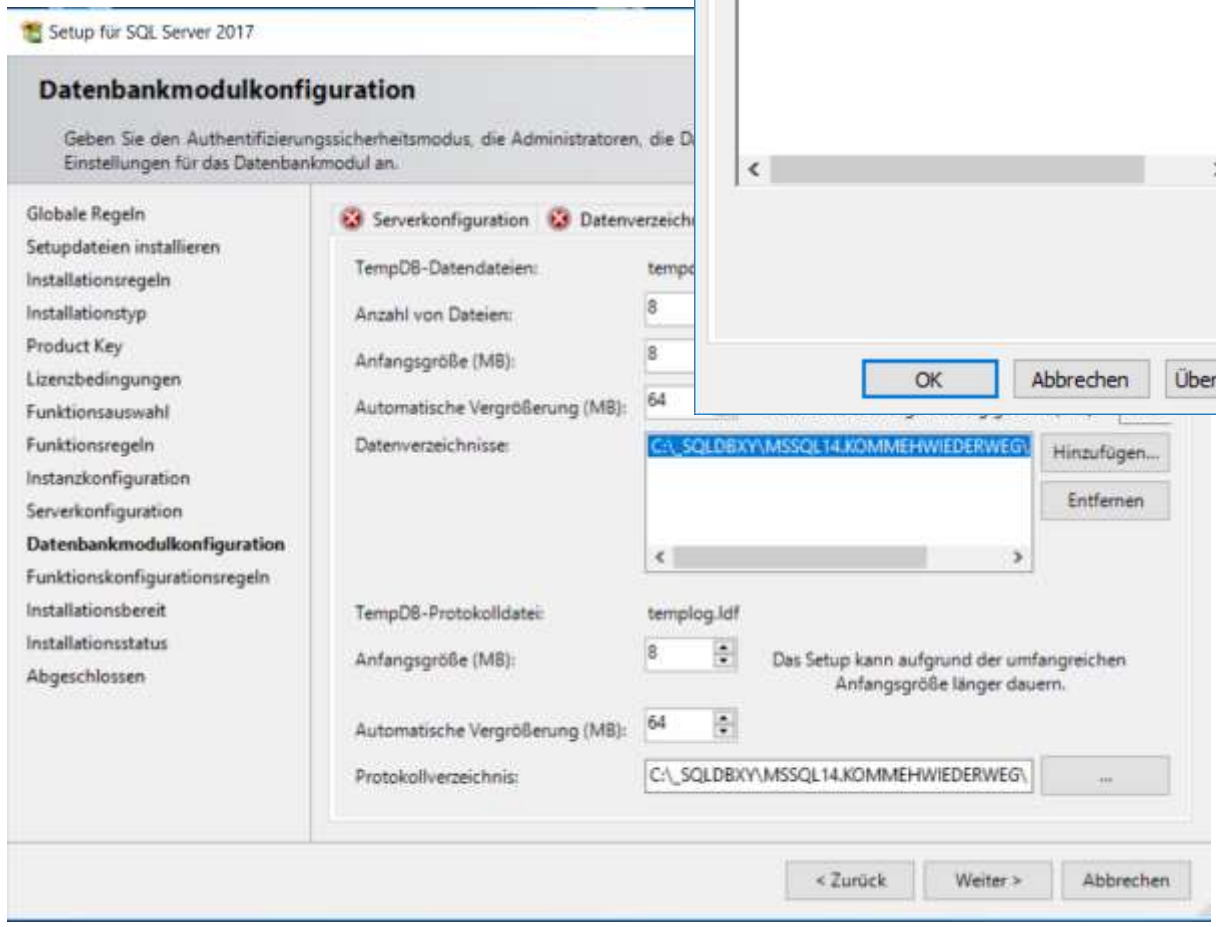


- Trenne Log von Daten
  - Lgfile sollte ungehindert Schreiben können
  - Logfile schreibt sequentiell
  - Datenfile random Zugriff
- Pauschalregel!

  50 LDF auf einer HDD sind auch kein Spaß

TempDB

- Gleiche Regel wie für nornale DBs
  - Trenne Log von Daten
- Anzahl von Dateien:
  - Anzahl der Kerne; max 8
  - Soviele Dateien wie Kerne
- Traceflags
  - T1118; T1117
  - Gleich große Datendateien
  - Universal Extents
- Seit SQL 2016 default



Server Settings

- RAM
  - MIN Arbeitsspeicher (Garantie für SQL Server)
  - MAX Arbeitsspeicher (Garantie für andere)
  - Mit Arbeitsspeicher ist nur der BufferPool gemeint
    - Datenpuffer und Plancache, Tables, Indexes, Proc Cache, Lock Hash Tables
  - !! Der Min Speicher wird erst fix reserviert, wenn er auch von SQL verwendet wird
    - → GPO
- HDD
  - Trenne Log von Daten
  - Keine Regel für ewig → 100DBs → 100 Logfiles auf einem Datenträger
- CPU
  - Affinitäten sind gut eingestellt

MAXDOP

- Maximaler Grad der Parallelität
  - Wieviele CPUs verwendet SQL Server für eine einzelne Abfrage?
  - Default: 5 – 0
    - Ab 5 SQL Dollar alle CPUs
  - Laufende Abfragen sind von Änderungen nicht betroffen
  - kein Neustart
- Faustregel:

  OLAP 50 und max 8 Core bzw 50%

  OLTP 25

- CXPACKET

Editionvergleich und Limits

- https://docs.microsoft.com/en-us/sql/sql-server/maximum-capacity-specifications-for-sql-server?redirectedfrom=MSDN&view=sql-server-ver15

- https://docs.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-version-15?view=sql-server-ver15

SQL Administration

After Installation

- SQL Server Konfiguration Manager
  - Kontrolle
    - Nur hier das Dienstkonto ändern!
    - Ports für benannte Instanzen kontrollieren
    - Filestreaming aktivieren /deaktivieren
    - Startparameter
    - Dienste gestartet?
  - SQL Alias

ServerSettings

- RAM
  - MIN RAM
    - Garantie für SQL Server
  - MAX RAM
    - Garantierter Speicher für andere (zB. OS)
- Datenbankeinstellungen
  - Trenne Daten von Logs
  - Kompression der Backups
- CPU
  - SQL verwaltet NUMA selbständig
- Aktivierung von Features

Arbeitsspeicher

- Für das OS die richtige Größe reservieren
  - 1 GB für OS + (4 bis 16GB)/4GB*1GB + (>16GB)/8GB*1GB

- 256 GB= 1GB + 12/4*1GB + 240/16*1GB
  = 1 + 3 + 15
  = **20 GB**

- 16 GB = ?

Datenträger Kontrolle

- Wichtige Werte
  - IOPS
  - Latenzzeiten
  - IOPS x Latenzzeit = Durchsatz

- DiskSpd
- Standardmessungen aktivieren
  - Perfmon

- TempDB Kontrolle

MAXDOP

- Anzahl der CPUs bei Abfragen
  - Abhängig von Kostenschwellwert
    - Default: 5
  - Anzahl der CPUs
    - Default: 0 (bis SQL 2017 inklusive)

- Besser:
  - Maximale Anzahl der CPUs auf Anzahl der Kerne max 8 beschränken
  - Kostenschwellwert: mit 25 (OLAP) bzw 50 (OLTP) beginnen
  - Tipp: ein wenig experimentieren. Der eingest. Wert zählt ab nächster Abfrage
  - Kein Neustart

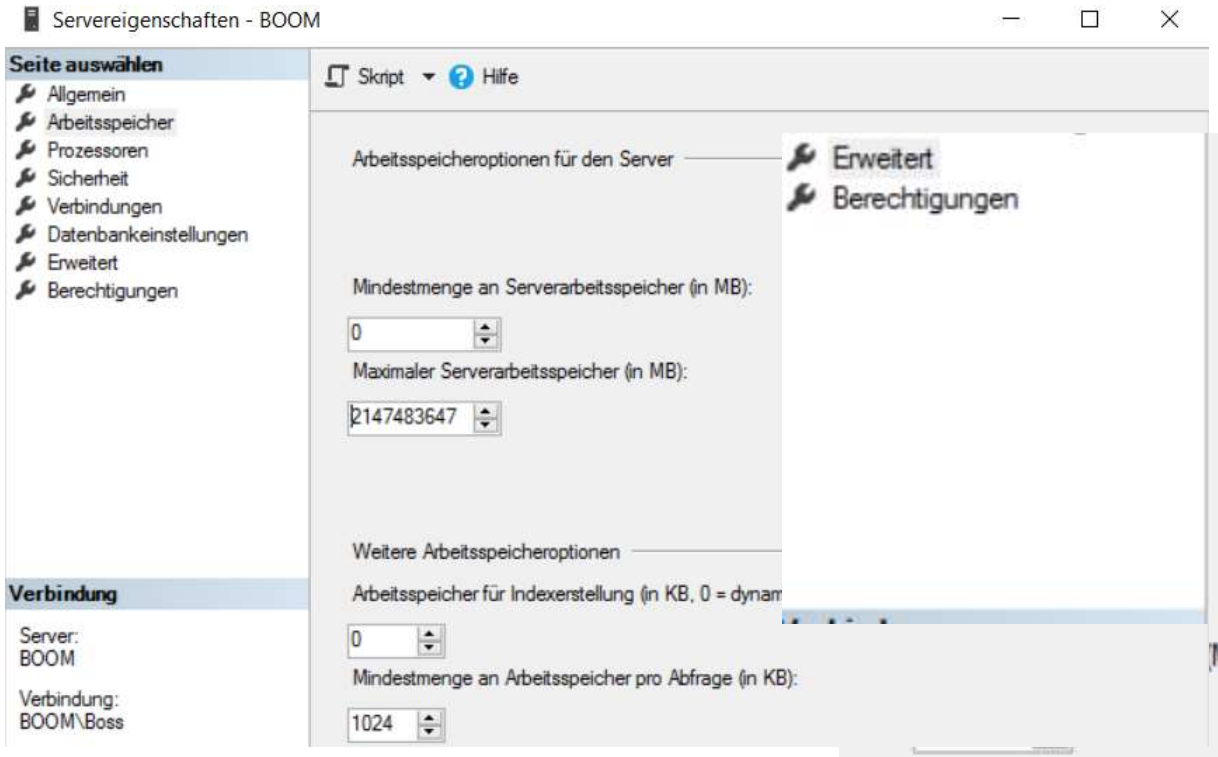
SQL Admin Tuning

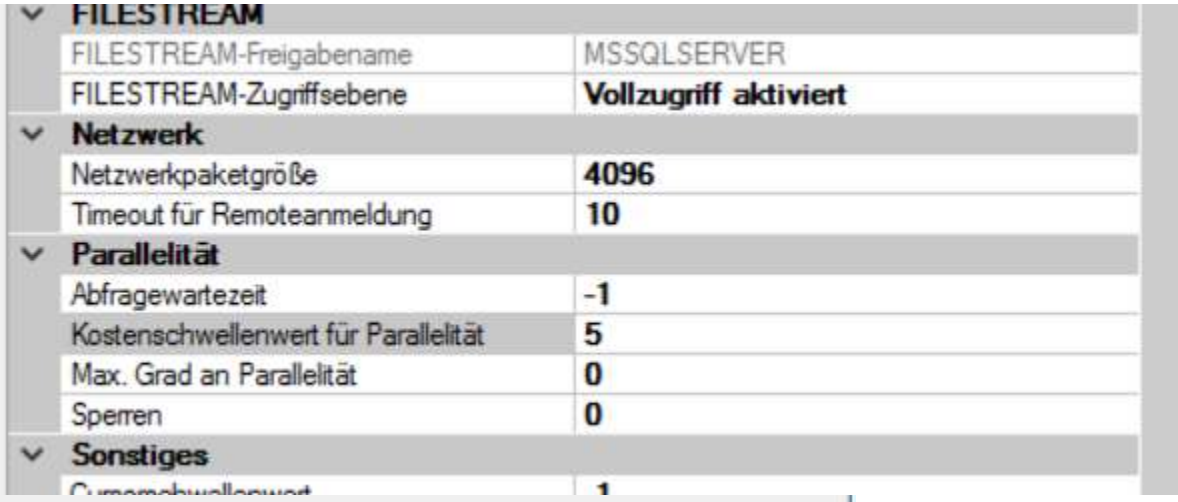
Modul 2 Serversettings

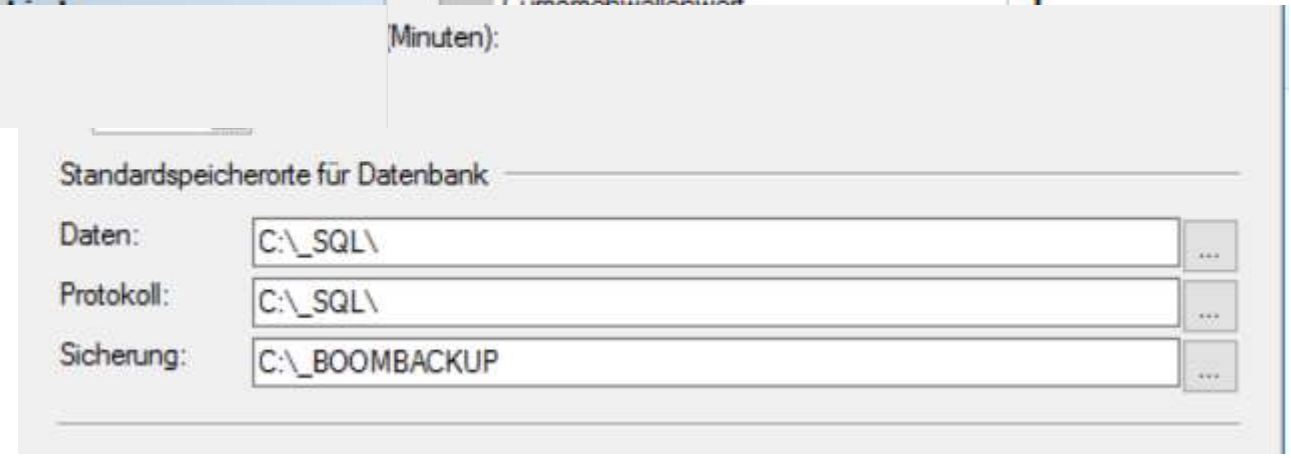
Setup

- Seit SQL 2016 bereits best practice Gedanken umgesetzt
  - TempDB
  - MAXDOP (SQL 2019)
  - MAX Memory (SQL 2019)
  - Benutzerdatenbank Verzeichnisse

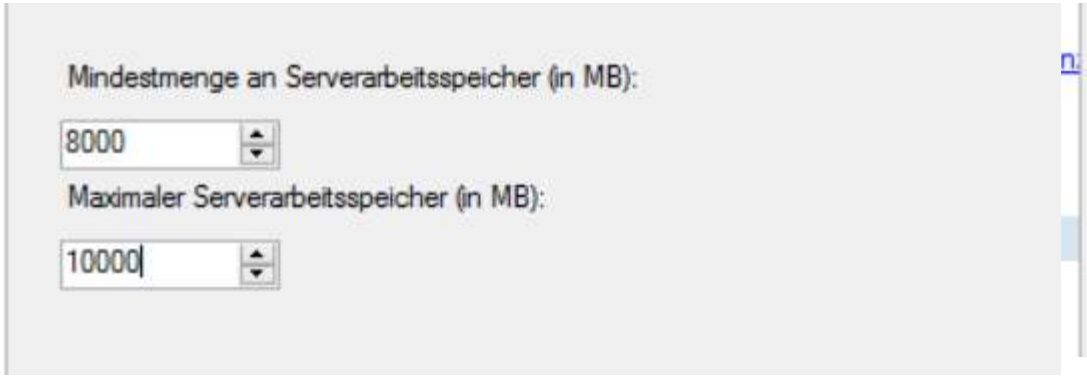
Serversettings - vorher

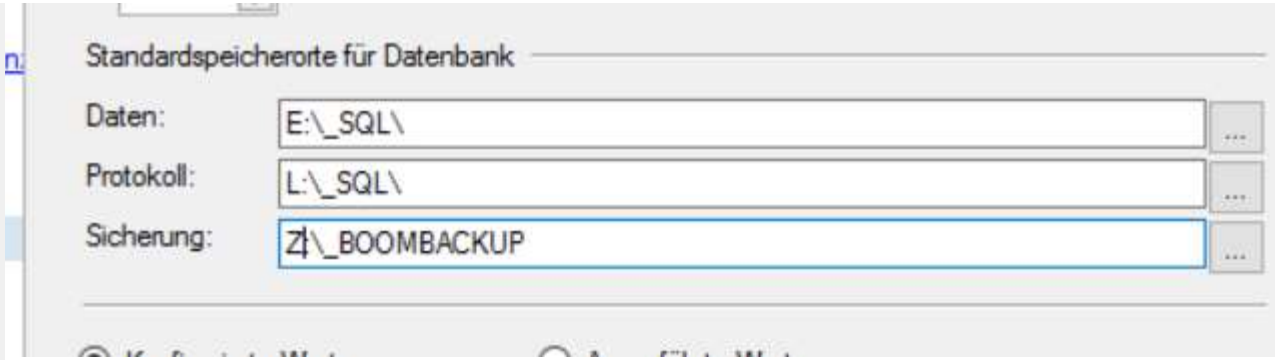


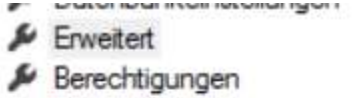


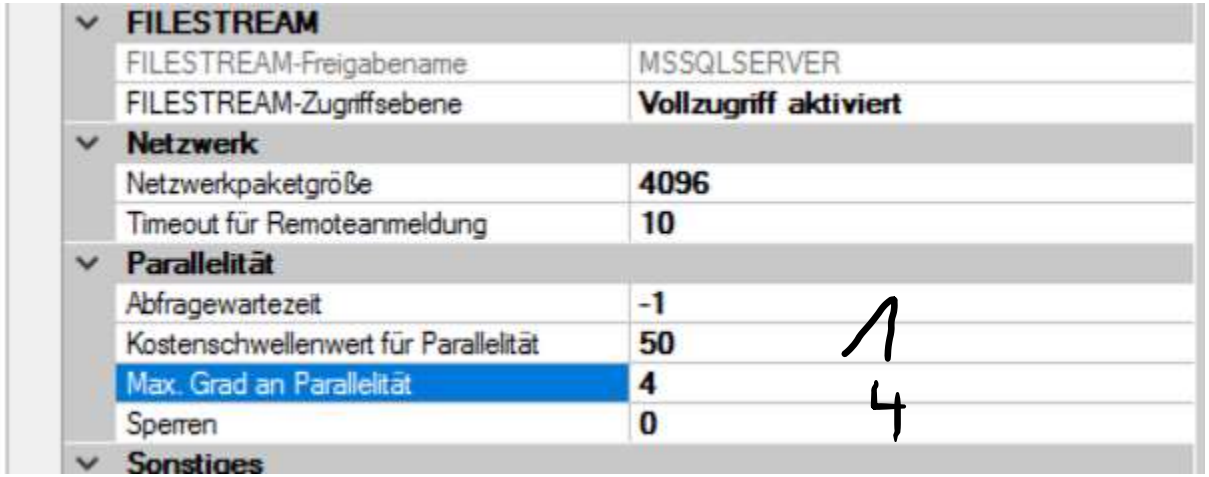


Serversettings - nachher









MAXDOP

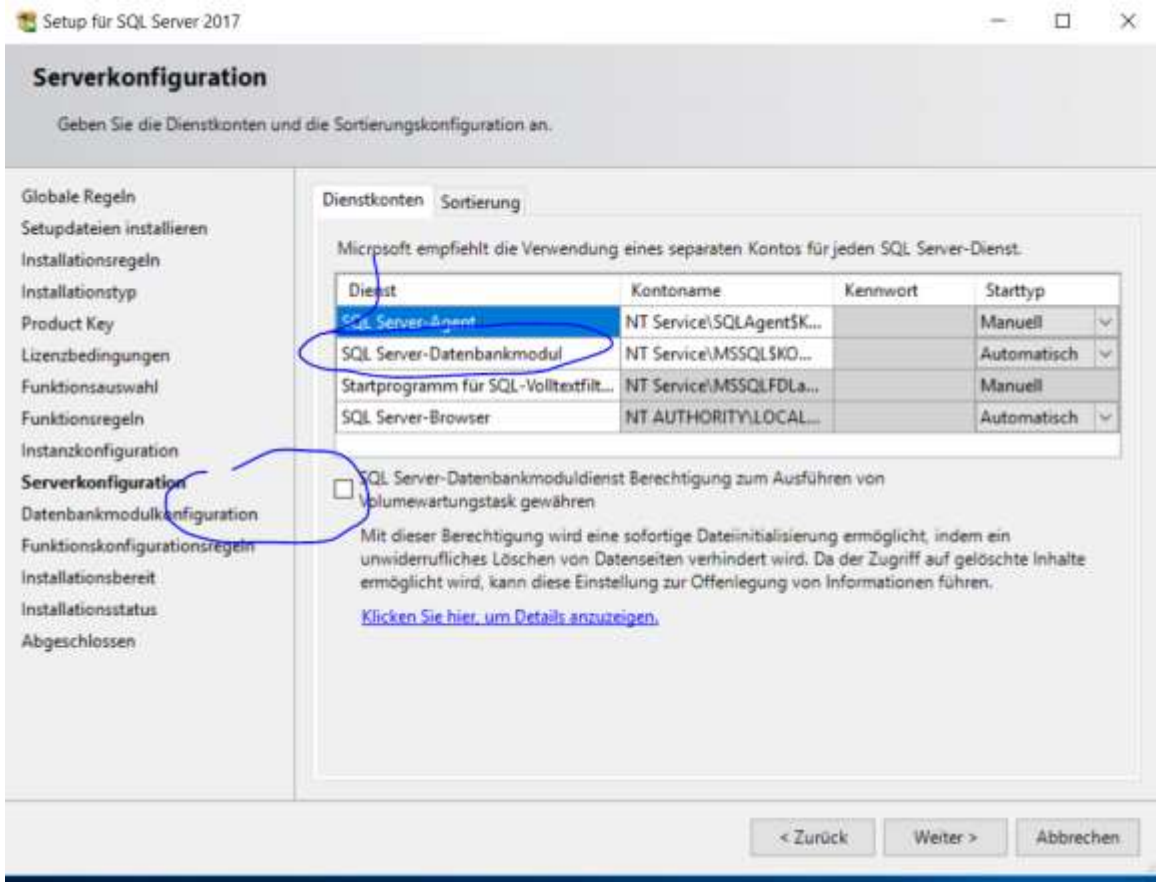
- Maximaler Grad der Parallelität
  - Wieviele CPUs verwendet SQL Server für eine einzelne Abfrage?
  - Default: 5 – 0
    - Ab 5 SQL Dollar alle CPUs
  - Laufende Abfragen sind von Änderungen nicht betroffen
  - kein Neustart
- Faustregel:
  OLAP 25 und max 8 Core bzw 50%
  OLTP 50

..seit SQL 2016 auch pro DB setzbar (MAXDOP)

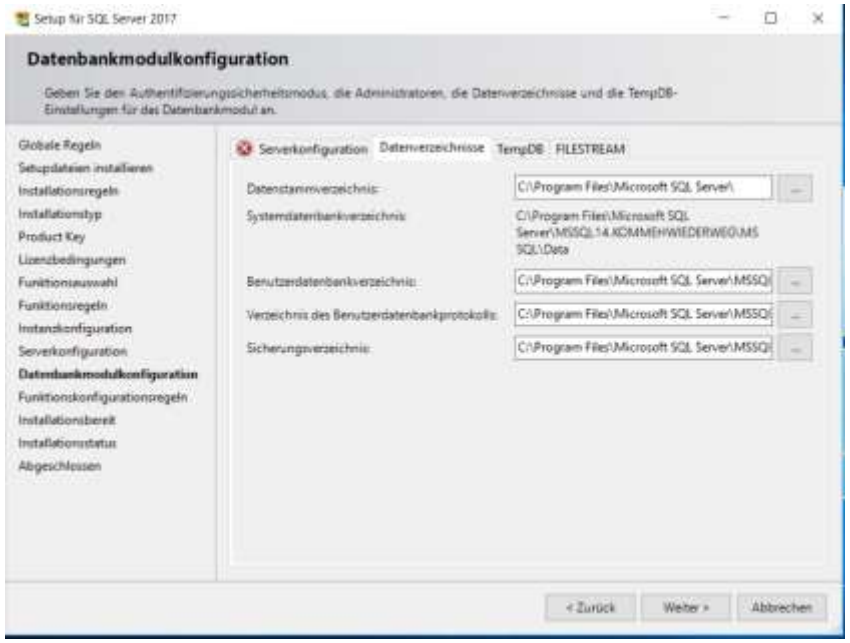
- Ressource: CXPACKET

Lokale Sicherheitsrichtlinie

- Ausnullen



Datenbankverzeichnisse

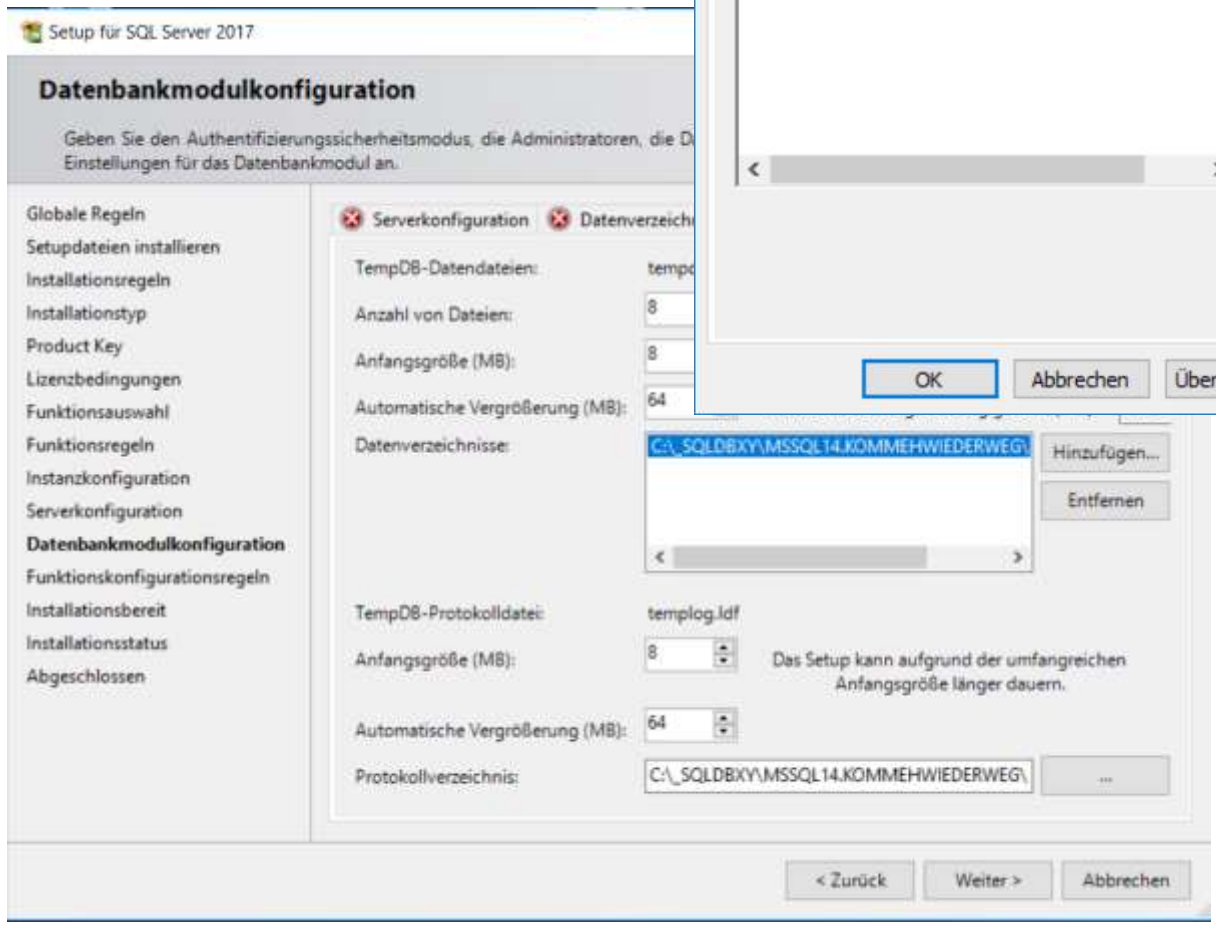


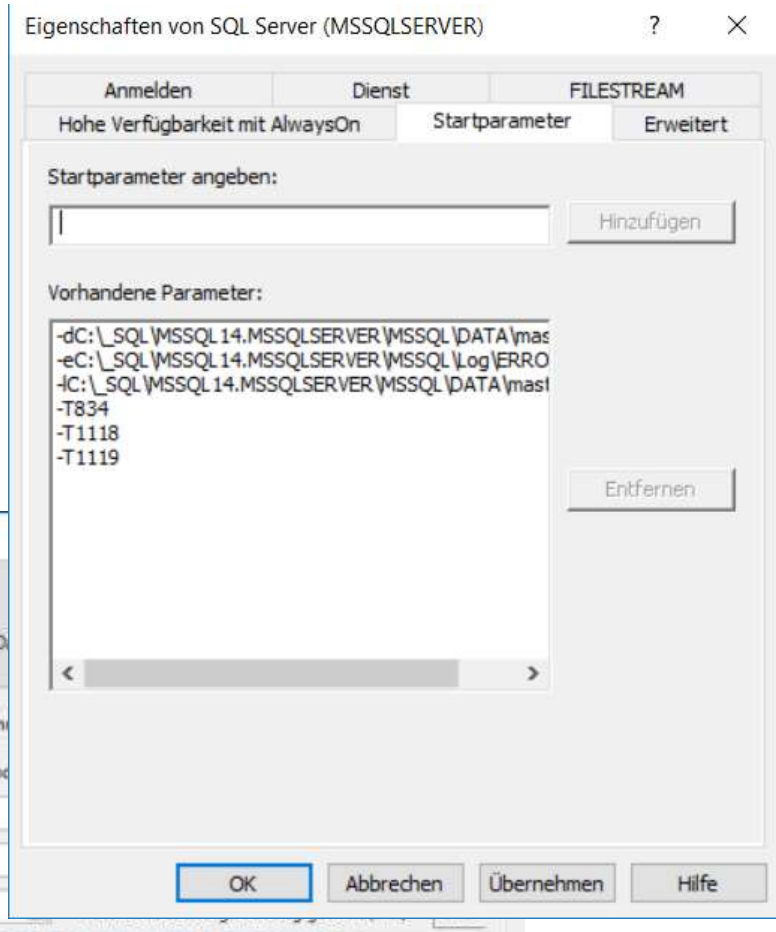
- Trenne Log von Daten
  - Logfile sollte ungehindert Schreiben können
  - Logfile schreibt sequentiell
  - Datenfile random Zugriff
- Pauschalregel!

  50 LDF auf einer HDD sind auch kein Spaß

TempDB

- Gleiche Regel wie für normale DBs
  - Trenne Log von Daten
- Anzahl von Dateien:
  - Anzahl der Kerne; max 8
  - Soviele Dateien wie Kerne
- Traceflags
  - T1118; T1117
  - Gleich große Datendateien
  - Uniform Extents
- Seit SQL 2016 default





Server Settings

- RAM
  - MIN Arbeitsspeicher (Garantie für SQL Server)
  - MAX Arbeitsspeicher (Garantie für andere)
  - Mit Arbeitsspeicher ist nur der BufferPool gemeint
    - Datenpuffer und Plancache, Tables, Indexes, Proc Cache, Lock Hash Tables
  - !! Der Min Speicher wird erst fix reserviert, wenn er auch von SQL verwendet wird
    - → GPO
- HDD
  - Trenne Log von Daten
  - Keine Regel für ewig → 100DBs → 100 Logfiles auf einem Datenträger
- CPU
  - Affinitäten sind gut eingestellt

SQL Administration

Datenbanksettings

- Mindestgröße (8MB Daten + 8MB Log ?)
- Wachstumsraten (64MB ?)
- Pfade
  - Trenne Daten von Logfile
    - Eigtl für jede DB extra überdenken
- Wiederherstellungsmodell
- Auto Statistiken
- Indirekte Checkpoints
- Verzögerte Dauerhaftigkeit

DB Objekte

- Tabellen
- Sichten
  - Gemerkte Abfragen
  - Verhalten sie wie Tabellen (INS; UP; DEL)
  - → Security
  - Besitzen keine Daten
- Prozeduren
  - Wie Batchdatei (mit Parameter) … exec procname wert
- Funktionen
  - Vielfältig einsetzbar … select f(spalte), f(wert) from f(wert) where f(sp) = f(wert)
  - Eingabewert→ Rückgabewert

Tipp

- Diagramm erstellen
  - Datentypen
  - DB Design
    - Normalisierung

SQL Admin Tuning

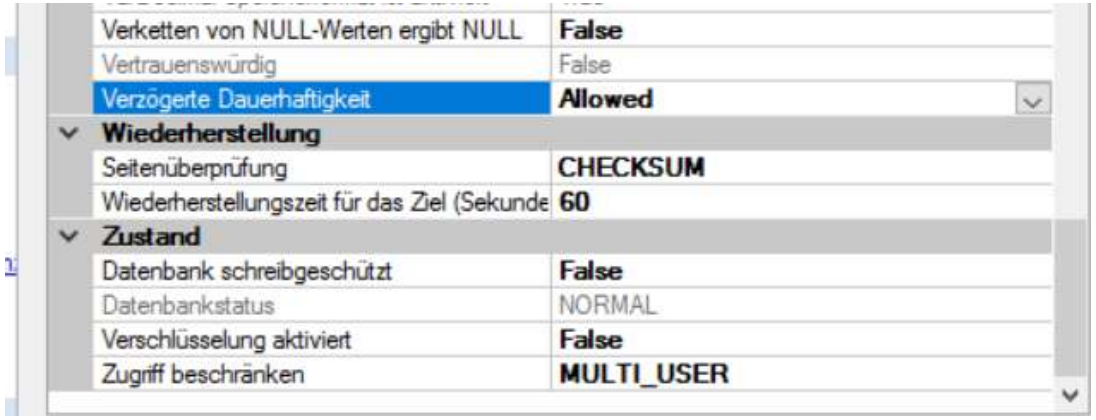
Modul 3: Datenbanksettings

Datenbank Settings

- Grundeinstellungen sind
  - Merkwürdig
  - SQL 2016: 8 MB Daten und 8 MB Logfile ; 64 MB Wachstumsrate
  - SQL 2014: 5MB Daten und 2 MB Logfile: 1 MB bzw 10% Wachstumsrate
- Wiederherstellungsmodel
  - Vollständig
- Ansatz : wie groß in 3 Jahren

Datenbank - Initialgrößen

- Wie groß ist die Dateien 3 Jahren
  - Hardwarewechsel
- Wachstumsraten
  - 10%??
- Wiederherstellungsmodel
  - Einfach – Voll – Massenprotokolliert
- Checkpoint
  - Indirekt ..alle 60 Sekunden

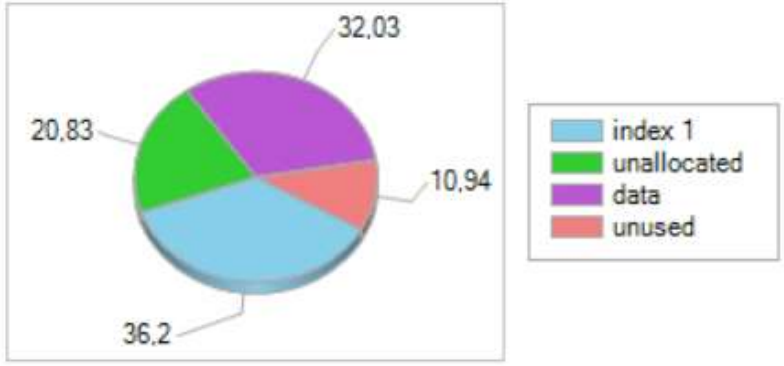


Datenbank – default -

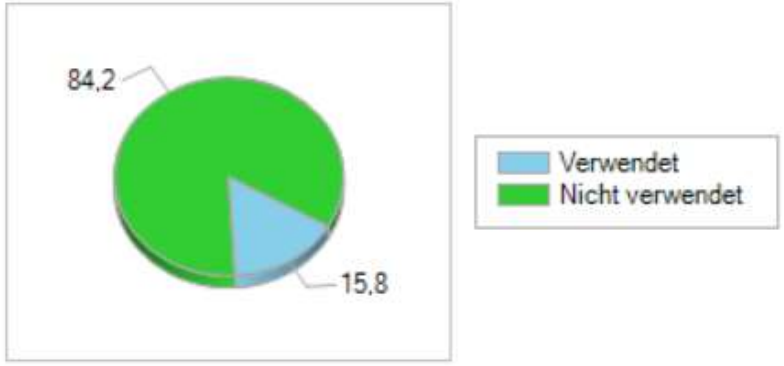
Dieser Bericht bietet einen Überblick über die Nutzung des Speicherplatzes in der Datenbank.

Gesamter reservierter Speicherplatz	6,88 MB
Reservierter Speicherplatz für Datendateien	3,00 MB
Reservierter Speicherplatz für Transaktionsprotokoll	3,88 MB

Speicherplatz für Datendateien (%)



Speicherplatz für Transaktionsprotokoll (%)



Es wurde kein Ereigniseintrag für automatische Vergrößerung/Verkleinerung für die Datenbank "mucdb" im Ablaufverfolgungsprotokoll gefunden.

⊞ **Von Datendateien verwendeter Speicherplatz**

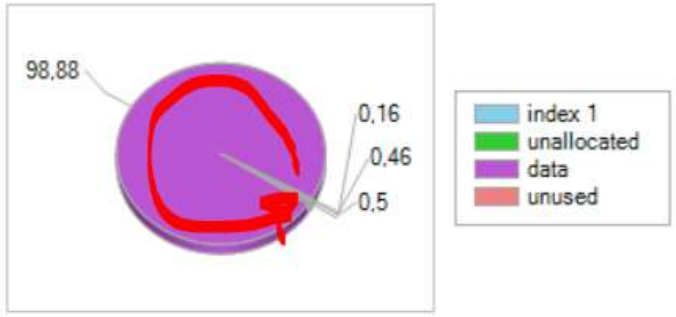
Unmanaged Database

Datenträgerverwendung
[mucdb]
am BOOM um 22.11.2018 14:27:57

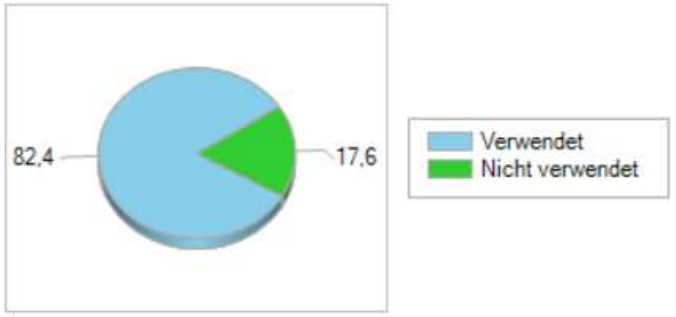
Dieser Bericht bietet einen Überblick über die Nutzung des Speicherplatzes in der Datenbank.

Gesamter reservierter Speicherplatz	265,88 MB
Reservierter Speicherplatz für Datendateien	238,00 MB
Reservierter Speicherplatz für Transaktionsprotokoll	27,88 MB

Speicherplatz für Datendateien (%)



Speicherplatz für Transaktionsprotokoll (%)



⊞ **Ereignisse zur automatischen Vergrößerung/Verkleinerung für Daten-/Protokolldateien**

⊞ **Von Datendateien verwendeter Speicherplatz**

Datenbank Design

- Theorie und Realität
  - Normalisierung vs Redundanz
  - Seiten und Blöcke
- Diagramm
  - Primärschlüssel ohne Fremdschlüssel
  - Datentypen
  - „Breite“ Tabellen

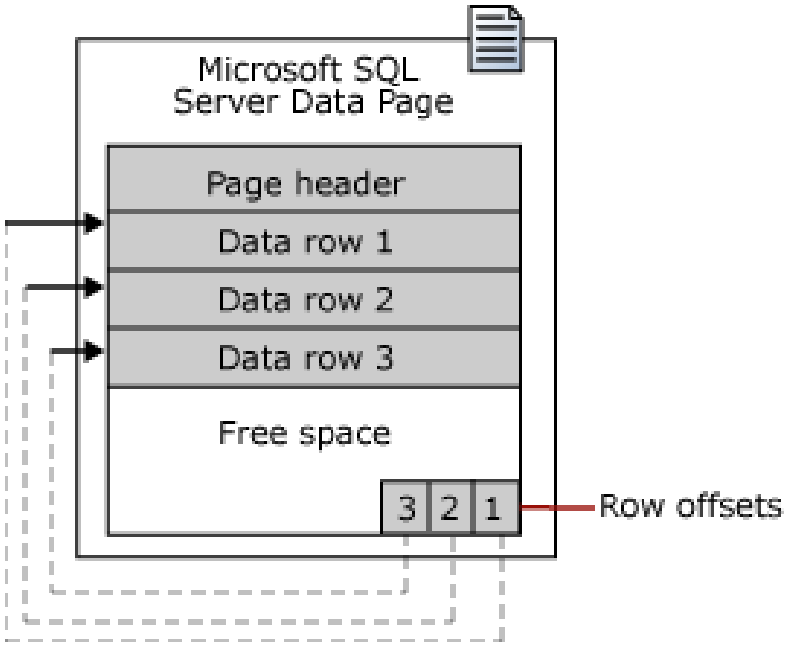
Datenbank Design

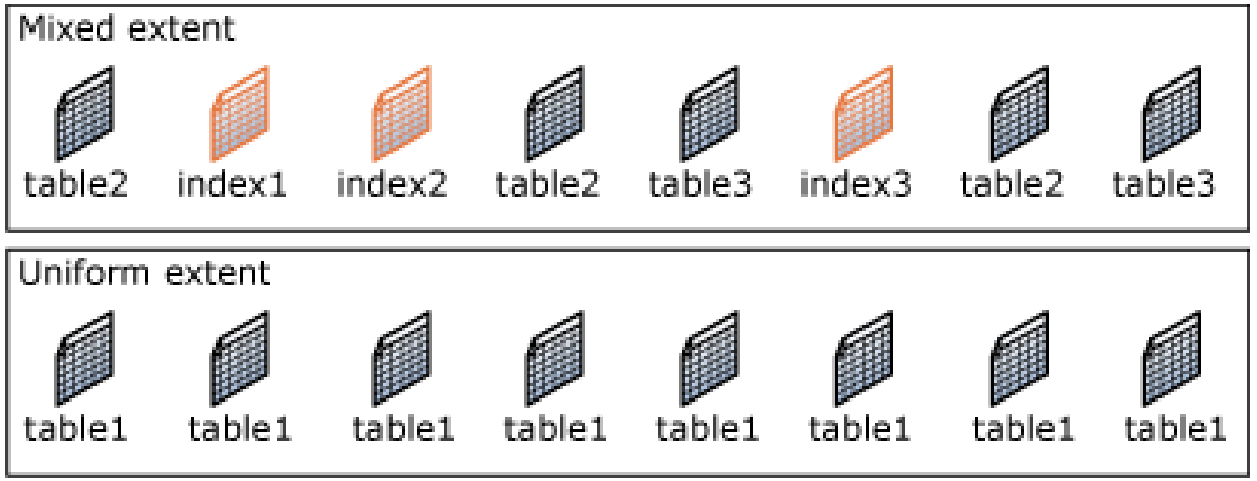
- Normalisierung schafft
  - Besseres Sperrverhalten durh Granularität
    - -> Indizes
  - Vermeidung von Redundanz
  - Abfragen benötigen mehr Joins
    - vs Perfomance
- Physikalisches Layout muss aber beachtet werden
  - Dbcc showcontig()
  - sys.dm_db_index_pyhsical_stats
  - Füllgrad der Seiten (8060 bytes)

Seiten und Blöcke

- Datensätze werden in Seiten gespeichert
  - 8192 bytes pro Seite
  - 8060 bytes für Datensätze verwendbar
  - Max 700 Datensätze pro Seite
  - Fixe Längen müssen in Seiten passen
  - Seiten werden 1:1 vom Datenträger gelesen und in Arbeitsspeicher übertragen
- 8 Seiten werden zu einem Block zusammengefasst

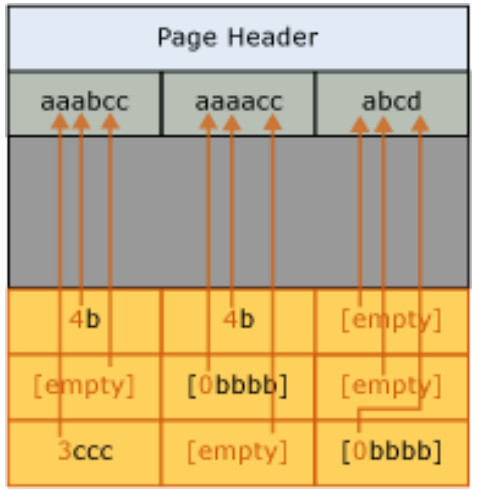
Seite und Blöcke





Kompression

- Variable Datentypen vs fixe Datetypen
  - Kompression schafft beste Leistung mit fixen Längen und Leeräumen
  - Char(50) du keine volle Belegung
  - Zu erwartende Kompressionsrate 40-60%
- Zeilenkompression
  - Eliminieren der Leeräume
  - Zusammenziehen der Datensätze auf weniger Seiten
- Seitenkompression
  - Zuerst Zeilenkompression
  - Dann Präfixkompression



Kompression

- Gut bei Archivtabellen
  - Abfragen auf große Tabellen benötigen nicht mehr so viel RAM
  - Abfragen benötigen allerdings mehr CPU Leistung
  - Abfragen werden durch Kompression selten schneller
  - Client bekommt immer Originalgrößen übermittelt
    - Transparent für Client!!

SQL Admin Tuning

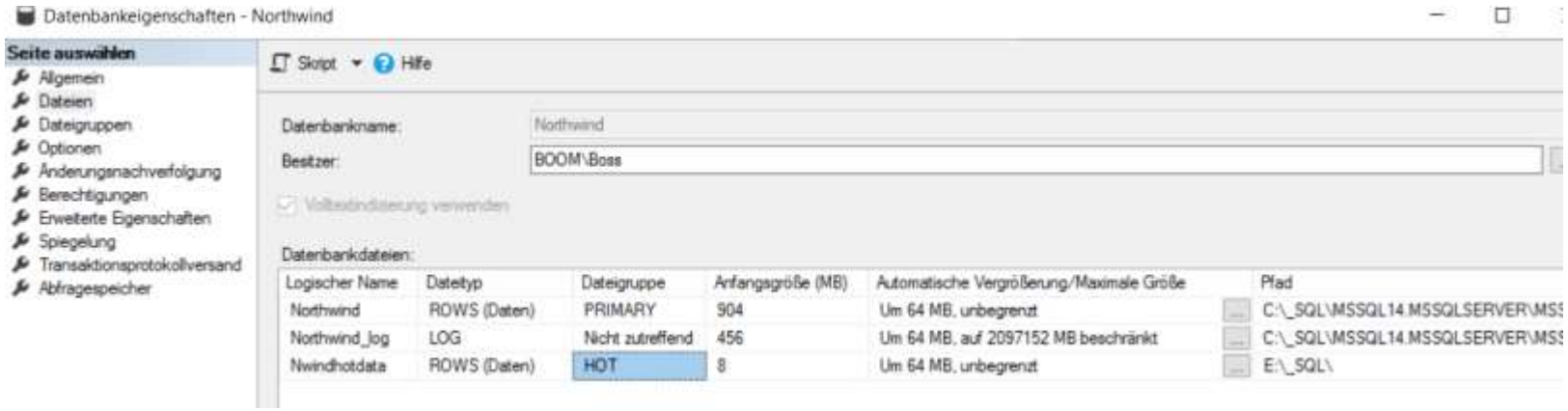
Modul 5 : Advanced Database Design

Optimierung per HDD - Salamitaktik

- Lastverteilung
- Dateigruppen
  - Tabellen auf andere Datenträger legen
- Partitionierte Sicht
  - Große Tabellen in viele kleine Tabellen splitten
- Partitionierung – für User/Software unsichtbar
  - Physikalische Partitionierung

Dateigruppen und CO

- Einfache Methode mehr HDDs ins Spiel zu bringen
- Tabellen können auf Dateigruppen gelegt werden
  - Create table () on Dateigruppe



Partitionierte Sicht

- Große Tabellen auf viele kleine Tabellen verteilen
  - Problem: Wo ist die große Tabelle?
- Sicht
  - Performance allerdings erst dann besser, wenn anstatt den vielen Tabellen nur noch die „richtige“ verwendet wird
- Konzept steht und fällt mit der im where verwendeten Spalte

Partitionierte Sicht

- Sicht
  - Enthält UNION ALL
    - Keine Suche nach doppelten Werten
    - Tabelle müssen Check Constraints besitzen
- Einschränkungen
  - PK auf allen Tabellen
    muss Eindeutigkeit über die Sicht besitzen
  - Kein Identity

```
Create View V_Sec_Fact_01_Umsatz_01 as

Select *
From T_Fact_01_Umsatz_01

Union all

Select *
From T_Fact_01_Umsatz_02

Union all

Select *
From T_Fact_01_Umsatz_03
```

Partitionierung

- Physisches Pendant zur partitionierten Sicht
  - Es bleibt bei einer Tabelle
- Mit Hilfe einer Funktion wird der Bereich eines Datensatzes bestimmt
- Ein Schema entscheidet, wo dieser Bereich tatsächlich zu finden ist

Partitionierung

- Verteilung der Daten mit Hilfe einer
  - Partitionierungsfunktion
    - Bestimmt in welchen Bereich ein Datensatz fällt
    - ----------100----------200----------
    - 1 2 3
  - Partitionierungsschema
    - Bestimmt mit Hilfe der Funktion den Ort der Daten
    - DG1 ---- DG2 ---- DG3
  - Tabellen werden auf das Schema gelegt unter Angabe der zu part. Spalte

Partionierung

```
Create partition function fname(datentyp)
As
Range left|right for values (Grenze1, Grenze2, …)


Create partition scheme schName
as
partition fname to (Dateigruppe1, Dateigruppe2,..)
```

Partitionierung

- Flexibel anpassbar
- Neue Grenzen
  - Scheme: next used
  - F() :SPLIT RANGE
- Grenzen entfernen
  - MERGE RANGE
- Partition zu Tabelle
  - Switch partition Nr to Tabelle
  - Tabelle muss allerdings auf der derselben Dateigruppe liegen wie Tabelle

Partitionierung

```
Alter partition function fname()
split range (neuer Grenzwert)

Alter partition function fname()
merge range („Grenze entfernen“)

Alter partition scheme schName
next used Dateigruppe

Alter Table PartTabelle switch partition (NR)
     (Zahl der Partition)
     to Archivtabelle
```

SQL Admin Tuning

Modul 8 Monitoring

Monitoring

- DMV
  - Datamanagementviews in jeder DB abfragbar
  - Abfrage und Funktionen über den Zustand des SQL Servers
  - Detailierte Informationen
  - Nach Neustart werden die gespeicherten Werte gelöscht!!
- Wichtige Systemsichten
  - Sys.dm_os_wait_stats
  - Sys.sysprocesses
  - Sys.dm_tran_locks
  - ..von über 500

sys.dm_os_wait_Stats

- Werte werden seit dem Start des SQL Server kumuliert dargstellt
  - Script für regelmäßige Erfassung
- Wait_time_ms: gesamte Wartezeit
- Signal_time_ms: Anteil der Wartezeit auf CPU
  - Sollte unter 25% liegen
- Wait_time_ms - Signal_time_ms = Wartezeit auf Ressource

Monitoring

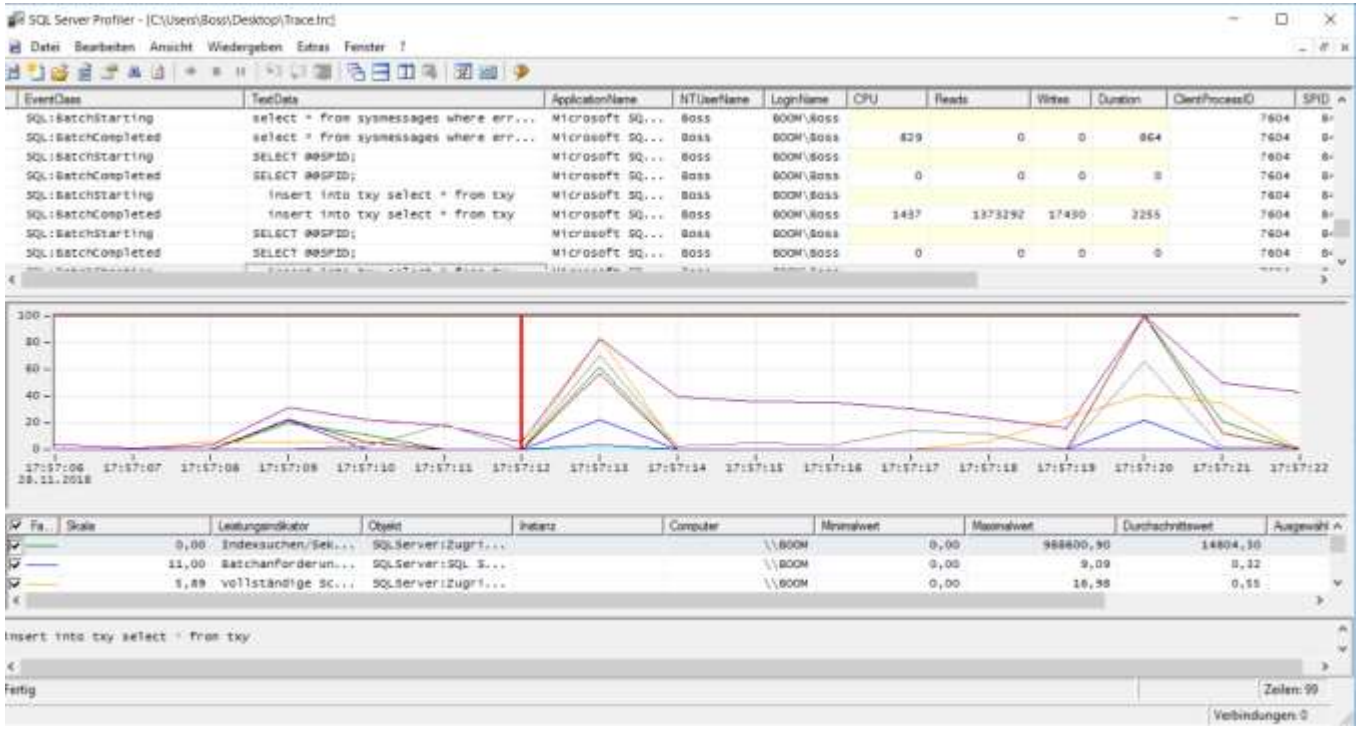
- Aktivitätsmonitor
  - Basiert auf DMVs
  - Aktueller Zustand des SQL Servers
    - Aktuelle Sperren,
    - Traffic auf den Datenbankdateien
    - Kategorisierte Wartezeiten (Summe und letzten 1000ms)
  - Teuerste Abfragen

Perfmon

- Leistungsindikatoren
  - Messwerte zu OS
    - Arbeitsspeicher: Seiten/sek
    - Prozessor: Prozessorzeit in %
    - Physikalischer Datenträger: unter 2
  - Messwerte zu SQL
    - Puffercache: Puffercache Trefferquote < 90%
    - Puffercache: Lebenserwartung der Seiten > 300 .. Eigtl deutlich mehr
    - General Statistics: Benutzerverbindungen
    - SQL Locks: durchschn. Wartezeit
    - SQL Plancache: > 60%
    - SQL Statsistics: Kompilierungen/sek; Batchanforderungen

Profiler

- Leistungsdaten können importiert werden
- Perfmondaten und Profilerdaten korrespondieren per Zeit



Profiler

- Was will man messen?
  - Für IX Tuning reichen die Statements
    - Tuning Vorlage
  - Für Troubleshooting ist die Startzeit und Endzeit hilfreich
    - Vor allem in Kombi mit Perfmon
    - Vorlage Standard
      - Login eigtl nicht notwendig
      - Aber Filter setzen!
        - Filter funktionieren nur , wenn Spalten auch zu sehen sind
        - Ideen: Dauer, CPU Zeit, Applikation, Datnbank, Login.
  - Sperren.. Hier wird sehr viel protokolliert.. Also Vorsicht
    - Vorlage : Locks (Deadlock Graph ;-))

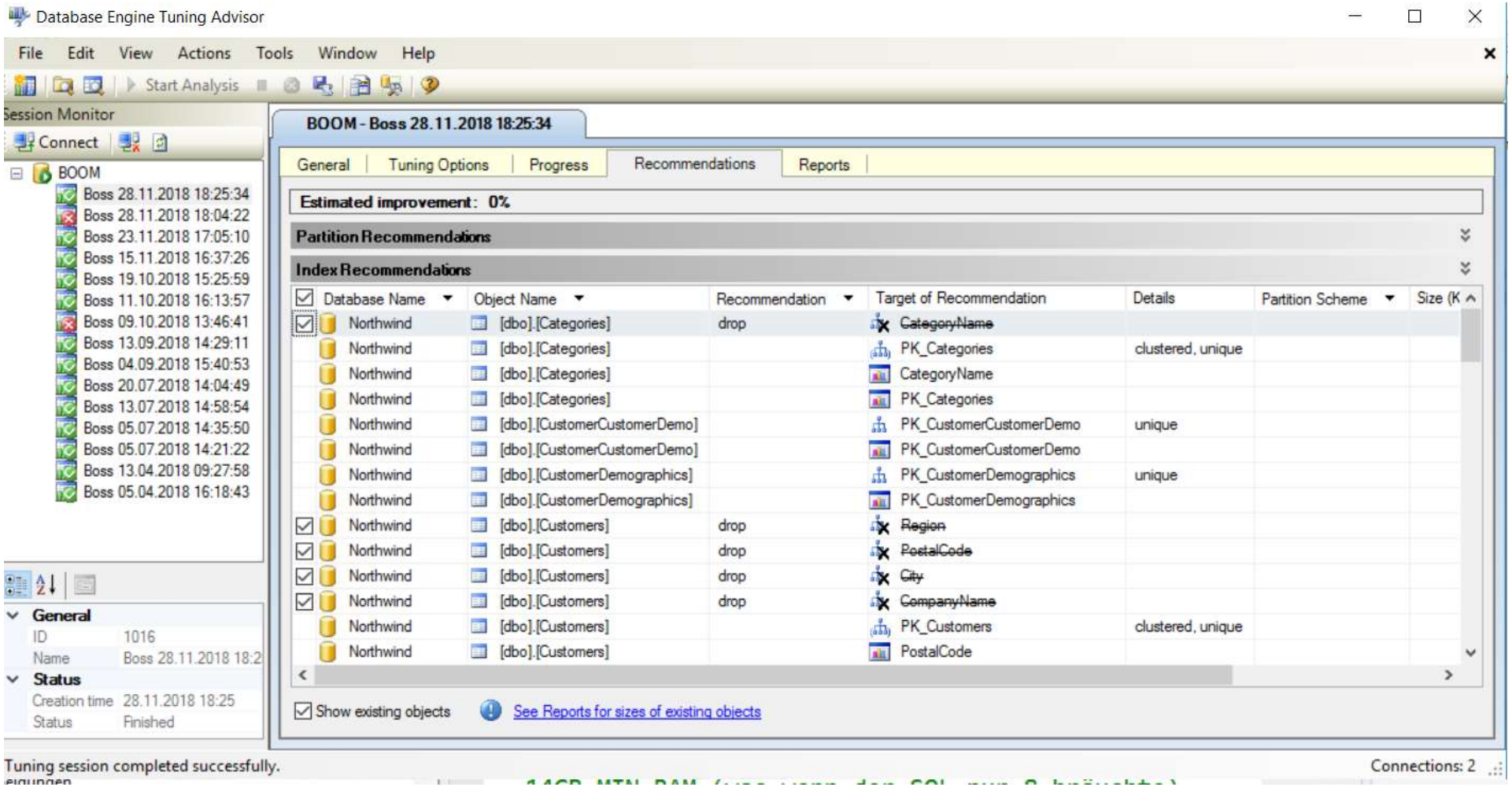
Perfmonintegration in Profiler

- Eine Aufzeichnung muss geöffnet sein, um die Perfmondaten laden zu können
- Datei → Leistungsdatenimportieren

Datenoptimierungsassitent

- Macht gutes Werk!
- Workloadanalyse
  - Quellen: Profilerdatei-, Tabelle, Query, QueryStore, Plancache
  - Workload sollte repräsentativ sein
- Finden von Indizes, Statistiken und Partitionierung
- Auch Löschen kann ein guter Vorschlag sein
- Berichte
  - Welche Spalte welcher Tabelle wurde wie oft verwendet

Datenbankoptimierungsassistent

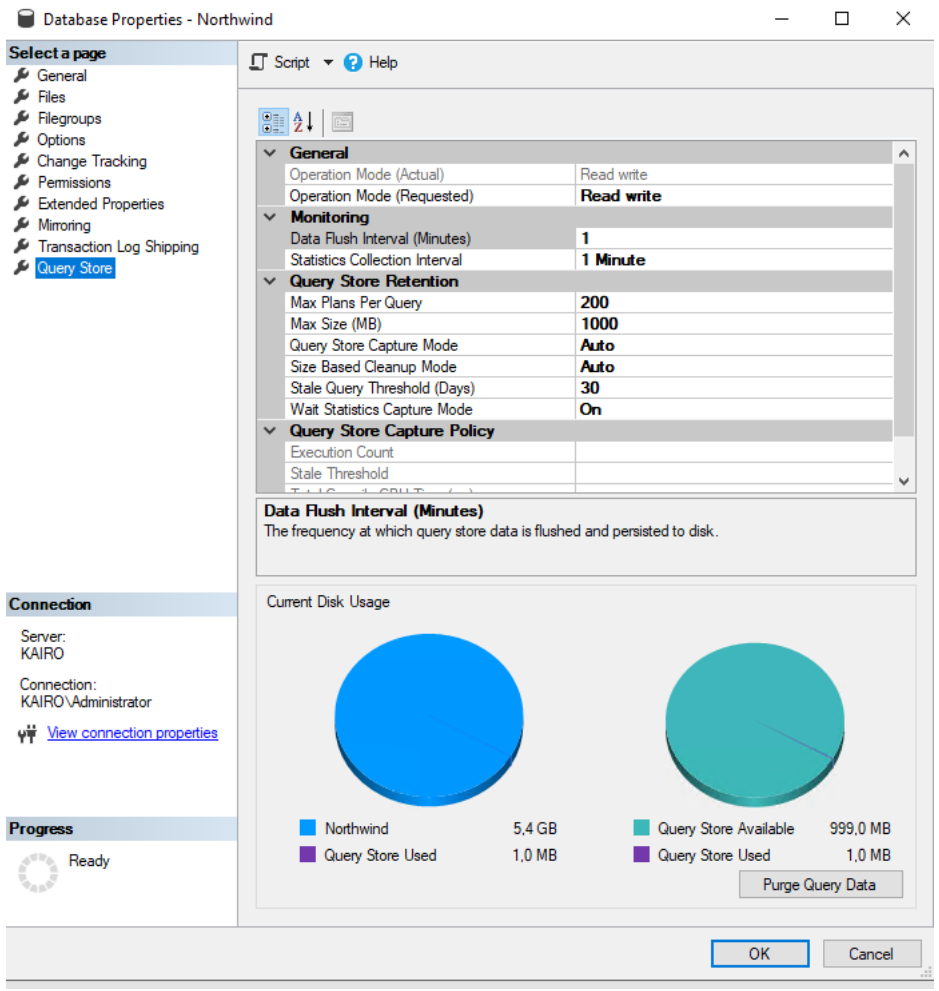


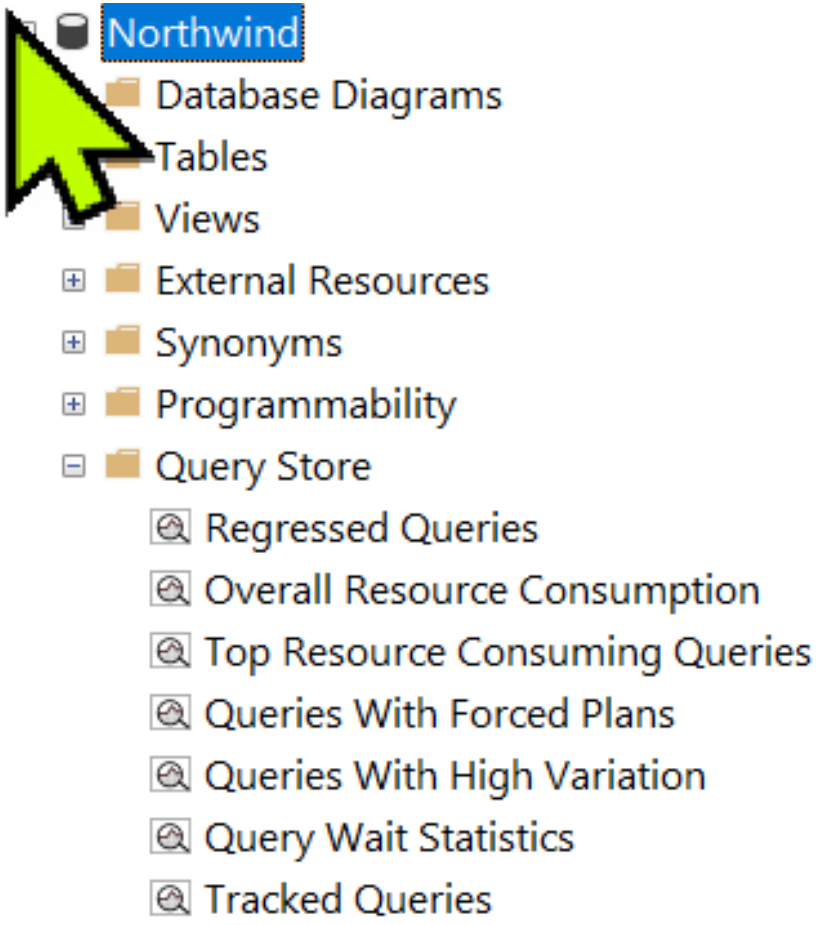
QueryStore

- Wichtige Informationen wie aus
  statistics io, time , Ausfürhungspläne, waits

  werden in DB sehr effizient gesammelt
- Ab SQL 2022 standardmäßig aktiv und dient dort als Optimierungsplattform
- Viele nützliche Bereicht
- Einschränkbar auf Datums und Zeitbereich
- Läßt sich auch per TSQL abfragen ;-)

Aktivierung des QueryStore





SQL Admin Tuning

Modul 6. Indizes

Indizes und Statistiken

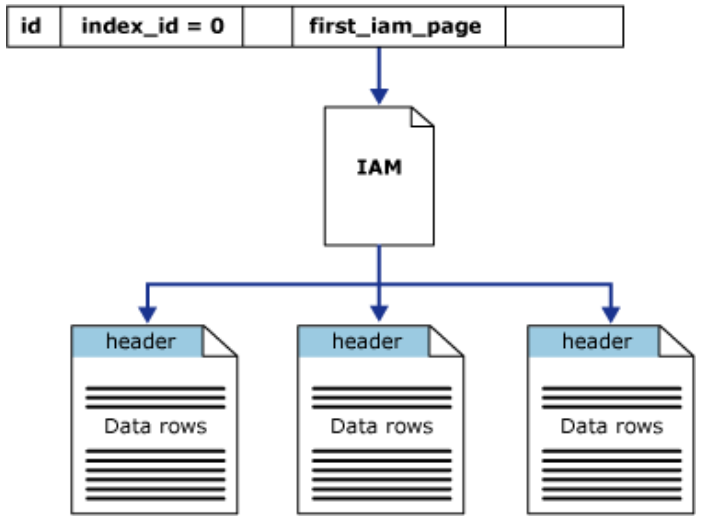
- Clustered Index (gruppierte)
- Non Clustered Index
- Eindeutiger IX
- Zusammengesetzten IX
- IX mit eingeschlossen Spalten
- Gefilterter IX
- Partitionierter IX
- Ind Sicht
- Abdeckender IX
- Realen hypothetischer IX
- Columnstore IX

Arbeitsweise der Indizes

- Indizes werden wie Datenbanken in Seiten verwaltet
- Seiten enthalten 8192 bytes
- Tabellen ohne Clustered Index = Heap
- B-Tree (balancierter Baum)
- Suche ab Wurzelknoten
  - Wie Telefonbuch

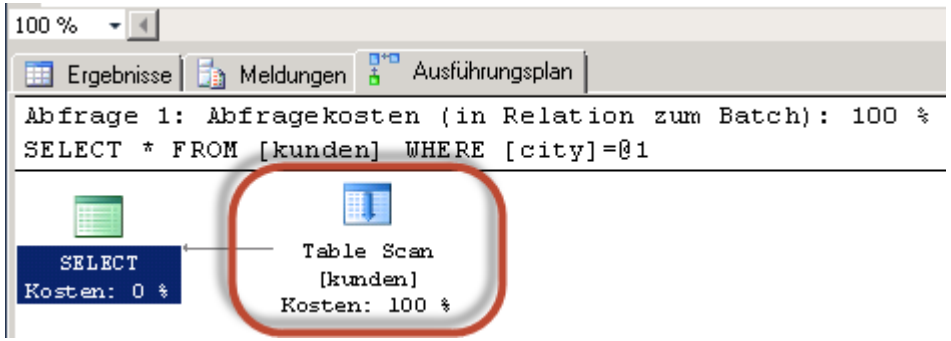
Heap

- Ein „Sau“-Haufen an Daten
- Eigtl keine Reihenfolge der Datensätze vorhersagbar
- Heap besteht aus vielen Seiten



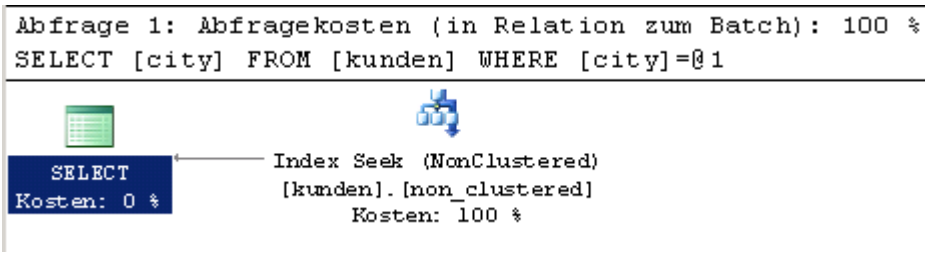
Heap

- Suche nach bestimmten Datensätzen muss immer den kompletten Heap durchlaufen
- Suche = Durchsuchen aller Seiten
  - SET STATISTICS IO ON
- Suche = TABLE SCAN

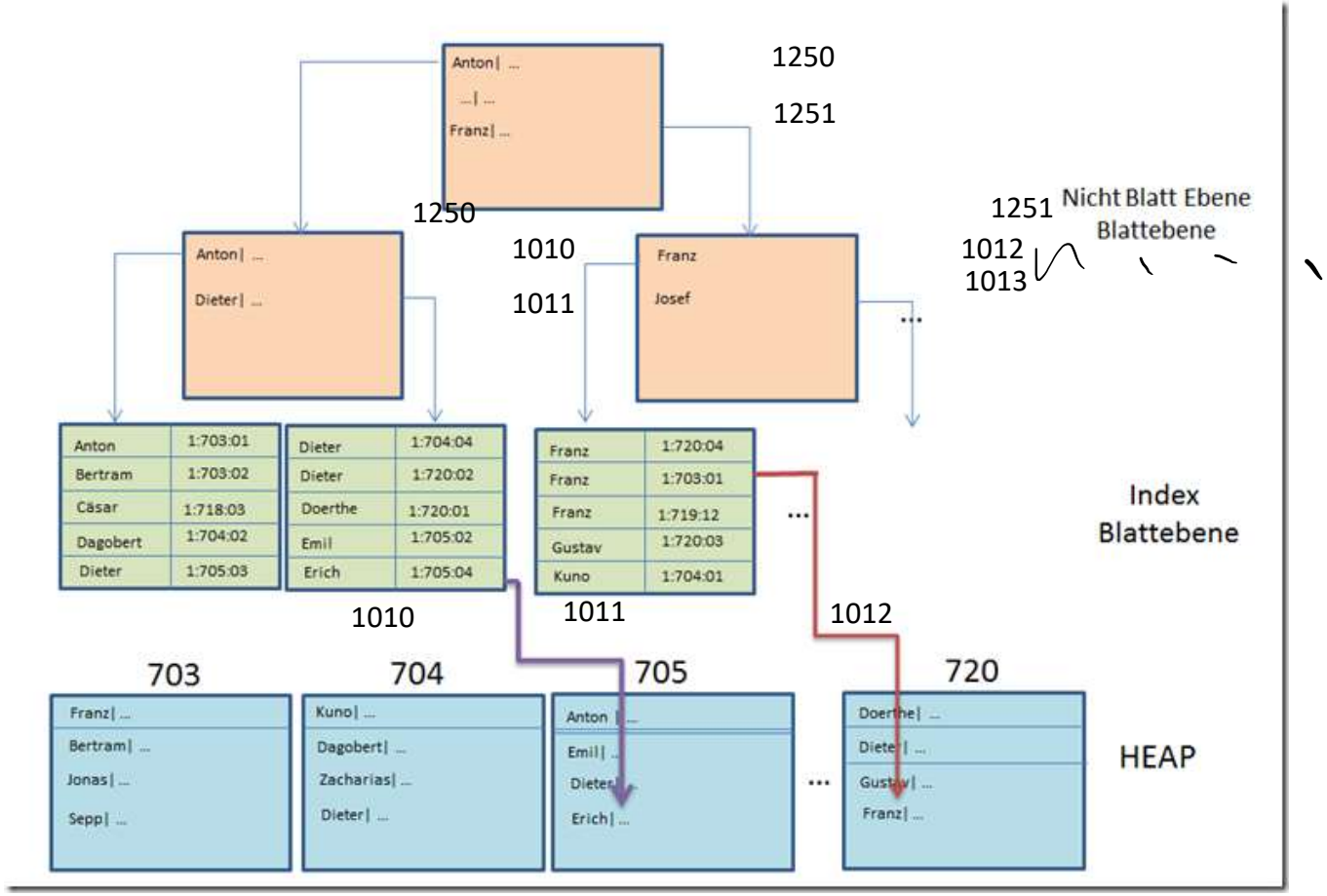


Wie funktioniert denn der Index?

- Wer das weiss, weiss auch welcher Index verwendet werden sollte
- Indizes werden ähnlich wie Telefonbücher verwaltet
  - Suche nach Tel von „Maier Hans“
    Gezieltes Suchen im Telefonbuch..
    …Treffer.. TelNr gefunden.
- Gezieltes Suchen im Index ist ein „Seek“



Wie funktioniert der Index?



Wie funktionieren Indizes

- Man kann auch nachschauen ;-)
  - sys.dm_db_index_physical_stats
  - DBCC IND (DB, Tabelle, 1)
  - DBCC PAGE (DB, Datei, Seite, [1,2,3])
  - DBCC TRACEON (3604)

Welche Indizes gibt es denn?

- Spaß bei Seite!
  - Nur 2!
  - Bzw. 3

  Nicht gruppierter Index

  Gruppierter Index

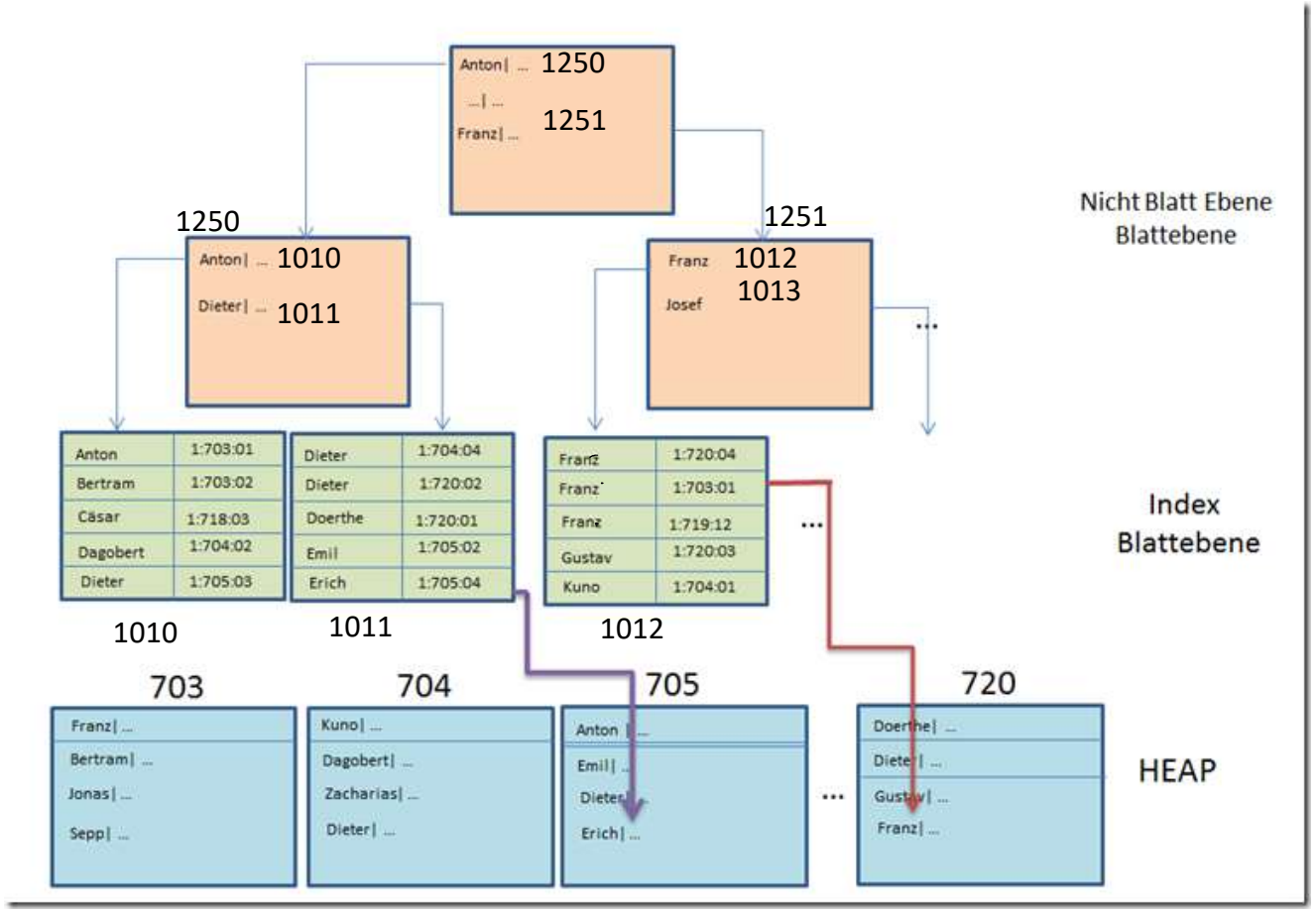
  Columnstored Index

  Spezialindizes: XML, Geo-Indizes

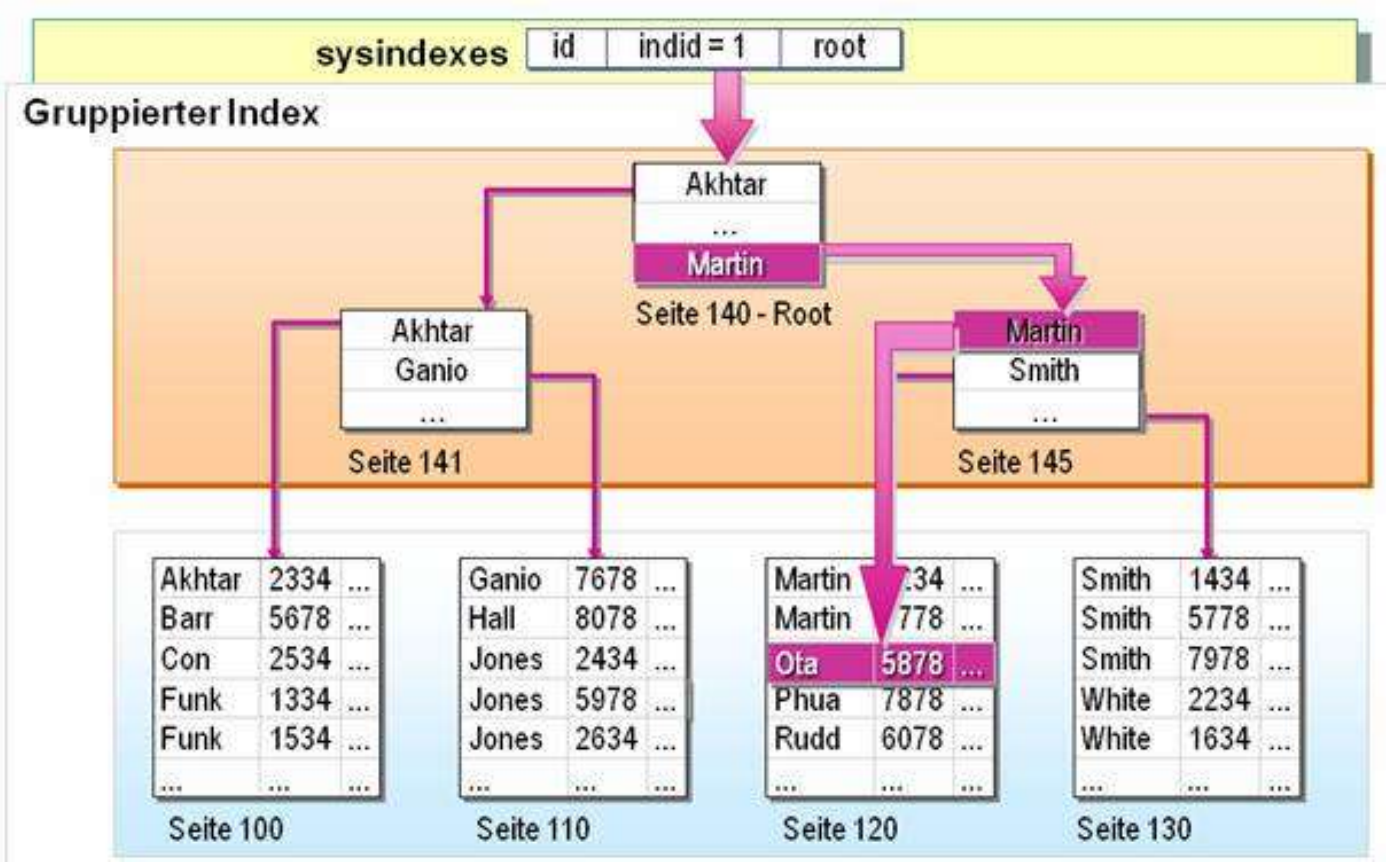
Wie funktioniert der Index?

- Nicht gruppierter Index lediglich sortierte Kopie der Indexspalten mit Zeiger auf den Originaldatensatz (1:204:02)
- Gruppierter Index ist Tabelle in physikalischer sortierter Form

Nicht gruppierter Index



Gruppierter Index



Indizes

- Nicht gruppierte Indizes besitzen Kopien der Daten und verwenden Zeiger auf den Originaldatensatz
- Gruppierte Indizes sind die Tabellen!
  …in physikalisch sortierter Form

Einsatzgebiete

- Gruppierter Index
  - Sehr gut bei Abfragen nach Bereichen und rel. Großen Ergebnismengen: < , > , between, like
    Kandidaten: Bestelldatum, PLZ,..
    Gibt’s nur 1-mal, daher zuerst vergeben!
- Nicht gruppierter Index
  - Sehr gut bei Abfragen auf rel. eindeutige Werte bzw. geringen Ergebnismengen: =
    Kandidaten: ID; Firmenname, …
    kann mehrfach verwendet werden (999-mal)
  - ➔ PK oft Gruppierter Index!! = Verschwendung

..und die anderen Indizes?

- Gefilterter Index:
  - Es müssen nicht mehr alle Datensätze in den Index mit aufgenommen werden.
- Mit Eingeschlossenen Spalten
  - Der Index kann zusätzliche Werte enthalten (→ SELECT), der Indexbaum wird dadurch nicht belastet.
- Partitionierter Index
  - Physikalische Verteilung der Indexdaten per Partitionierung

..und die anderen Indizes?

- Eindeutiger Index
  - Erzwingt eindeutige Werte.
    Kandidat: Primary Key
- Zusammengesetzter Index
  - Index besteht aus mehreren Spalten. Auch im Indexbaum enthalten.
  - Kandidat: where umfaßt mehrere Spalten
  - Land , Stadt
- Abdeckender Index
  - ;-) leider nicht per „CREATE“, sondern ergibt aus der Abfrage. Bester Index! Alle Eregbnisse werden aus dem Index geliefert.
    Keine Lookup Vorgänge!

..und die anderen Indizes?

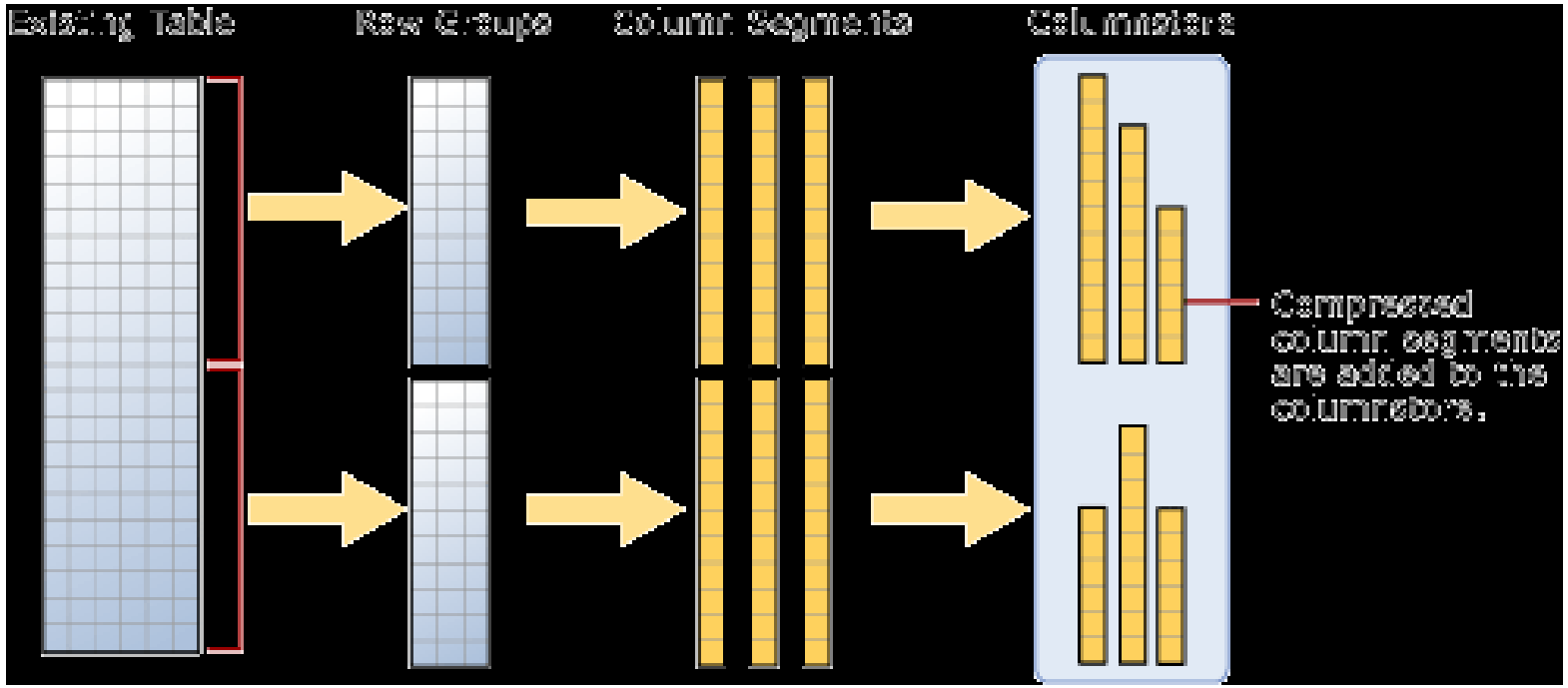
- Indizierte Sicht
  - Perfekt für Aggregate!
  - = Clustered Index (materialized View)
  - Viele Bedingungen
    - Schemabinding, big_count()
  - In Enterprise Version können Statements „überschrieben“ werden
    Statt Abfrage auf Tabelle, verwendet SQL Server die Sicht
  - Aber auch Probleme: Locks

..und die anderen Indizes?

- Columnstored Index (ab SQL 2012)
  - Statt Datenätze werden Spalten in Seiten verwalten
  - Sehr gut bei Datawarehouse Szenarien
    - Mehrfach vorkommende Werte lassen sich gut komprimieren
  - Abfragen verwenden nur noch die Seiten, in denen die entsprechenden Daten vorhanden sind

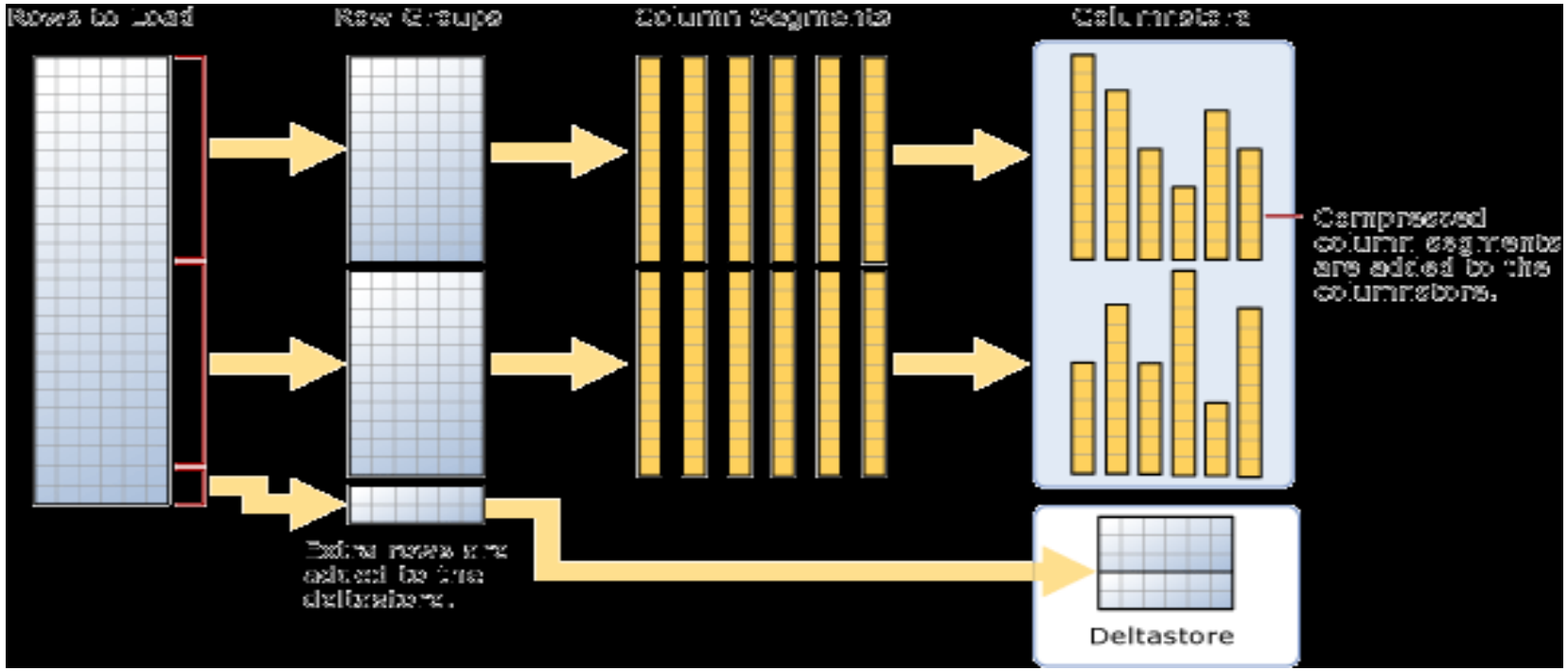
Columstore Index

- Datensätze werden nicht zeilenorientiert abgelegt
  - Sondern spaltenorientiert
  - Deutlich höhere Kompressionsraten möglich



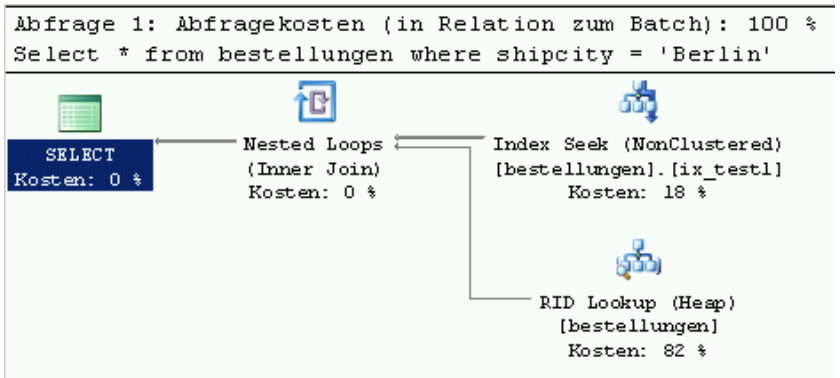
Columnstore Index

- Seit SQL 2014 sind Columnstore Indizes aktualisierbar
- Dazu wird eine Deltastore eingerichtet, der die neuen und alten Daten enthält



Welchen Indizes sollte man denn erstellen?

- Nur die, die man benötigt!
  - Jeder weitere Index stell bei INS, UP …eine Last dar
  - Keine überflüssigen Indizes (ABC, AB, A)
    - Wieviele Telefonbücher benötigen man pro Stadt?
- Die, die fehlen!
  - SQL Server merkt sich fehlende Indizes
- Nicht nur das WHERE ist entscheidend
  - Sondern auch der SELECT



Wie wirken sich Indizes auf die Leistung aus?

- Hervorragend,
  - Sofern keine Messdatenerfassung erfolgt
- Entscheidend ist die Anzahl der Indexebenen
  - Statt 100000 Seiten im Heap für 1 DS durchlaufen zu müssen, benötigt man über den Index soviele Seiten wie Ebenen vorhanden sind. (3 bis 4 Ebenen)
  - Ob 1 Mio oder 100 Mio DS, oft kaum mehr als 3 Ebenen

Worauf sollte man Indizes achten?

- Indizes müssen gewartet werden?
  - Reorg oder Neuerstellung
- Suche nach korrekten Indizes
- Suche nach doppelten, überflüssigen, fehlenden Indizes
- Gute Übersicht durch Systemsichten
  - Sys.dm_db_index_physical_Stats

..was ist besser?

- Table Scan
- Index Scan
- Index Seek
- Clustered Index Seek