

2022-2023

DerbOven Maxim & Jonas leijzen

IAO3

Optimalisatie dossier

data persistency

Inhoud

[2 Logische Indexed View [S1] 2](#_Toc117541061)

[2.1 Inleiding 2](#_Toc117541062)

[2.2 Query keuze 2](#_Toc117541063)

[2.3 Werkwijze 2](#_Toc117541064)

[2.3.1 De logische reads zonder indexes 2](#_Toc117541065)

[3 Partitionering [S2] 3](#_Toc117541066)

[3.1 Inleiding 3](#_Toc117541067)

[3.2 Horizontal Partitioning 3](#_Toc117541068)

[3.3 Vertical Partitioning 3](#_Toc117541069)

[3.4 Partities aanmaken 3](#_Toc117541070)

[3.5 Werkelijke optimalisatie in werking 4](#_Toc117541071)

[3.5.1 Plan voor en na 4](#_Toc117541072)

[3.5.2 Hoe hebben we de optimalisatie gedaan 4](#_Toc117541073)

[4 Column storage [S2] 6](#_Toc117541074)

[4.1 Inleiding 6](#_Toc117541075)

[5 Table Compression [S1] 7](#_Toc117541076)

[5.1 Inleiding 7](#_Toc117541077)

[5.2 Query keuze 7](#_Toc117541078)

# Logische Indexed View [S1]

## Inleiding

Een logische indexed view is een virtuele tabel die fysiek (logisch) wordt opgeslagen is op de harde schijf, dus niet virtueel. Hierdoor moeten we bij het ophalen van de view niet telkens weer alle data uit de basis tabel ophalen. Dit wil ook zeggen dat het aanpassen van de data in de indexed view niet performant is. Pas deze optimalisatie dus alleen toe in systemen die infrequent of zelden geupdate worden met nieuwe rijen, zoals een datawarehouse!

Wanneer uw basis tabel veel veranderd is het niet aangeraden een indexed view te gebruiken.

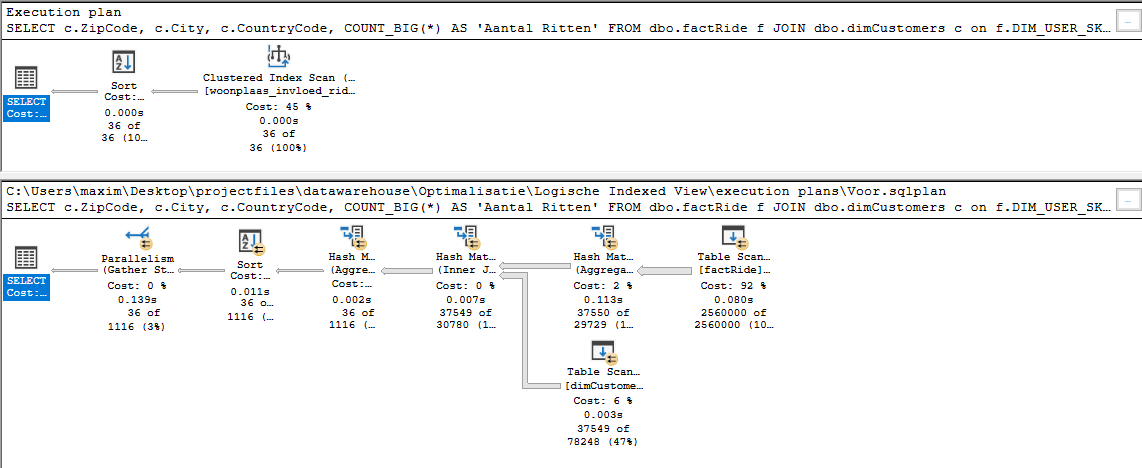
## Query keuze

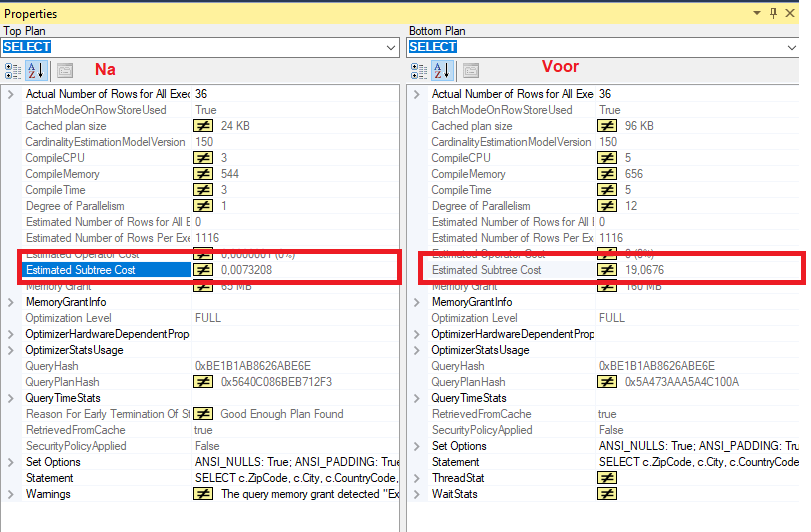
Queries met veel joins en veel rijen hebben die niet vaak geupdate moeten worden / up to date data moeten hebben hebben een grotere kans op optimalisatie voordelen.

Voor dit voorbeeld hebben we gebruik gemaakt van een querie die joins deed op 3 verschillende tabellen.

## Werkwijze

### De logische reads zonder indexes





# Partitionering [S2]

https://www.sqlshack.com/database-table-partitioning-sql-server/

## Inleiding

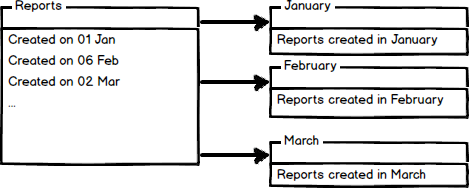
Partitioneren is het splitsten van tabellen in subtabellen op basis van een bepaalde voorwaarde.

Als er in de where clausule de voorwaarde verwerkt zit dan wordt enkel de relevante subtabel aangesproken. Het doel van partitioneren is het verbeteren van de efficiëntie en prestaties van het laden van gegevens.

Datums opsplitsen in bepaalde ranges.

## Horizontal Partitioning

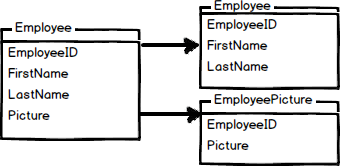
Horizontale partitionering verdeelt een tabel in meerdere tabellen die hetzelfde aantal kolommen bevatten, maar minder rijen. Op deze manier verwijzen zoekopdrachten die gegevens voor een specifiek jaar vereisen, alleen naar de juiste tabel. Tabellen moeten zo worden gepartitioneerd dat query's naar zo weinig mogelijk tabellen verwijzen.



## Vertical Partitioning

Verticale tabelpartitionering wordt meestal gebruikt om de prestaties van SQL Server te verbeteren.

* kolommen ophalen die zeer brede tekst- of BLOB-kolommen bevat
* beperken van de toegang tot gevoelige gegevens



## Partities aanmaken

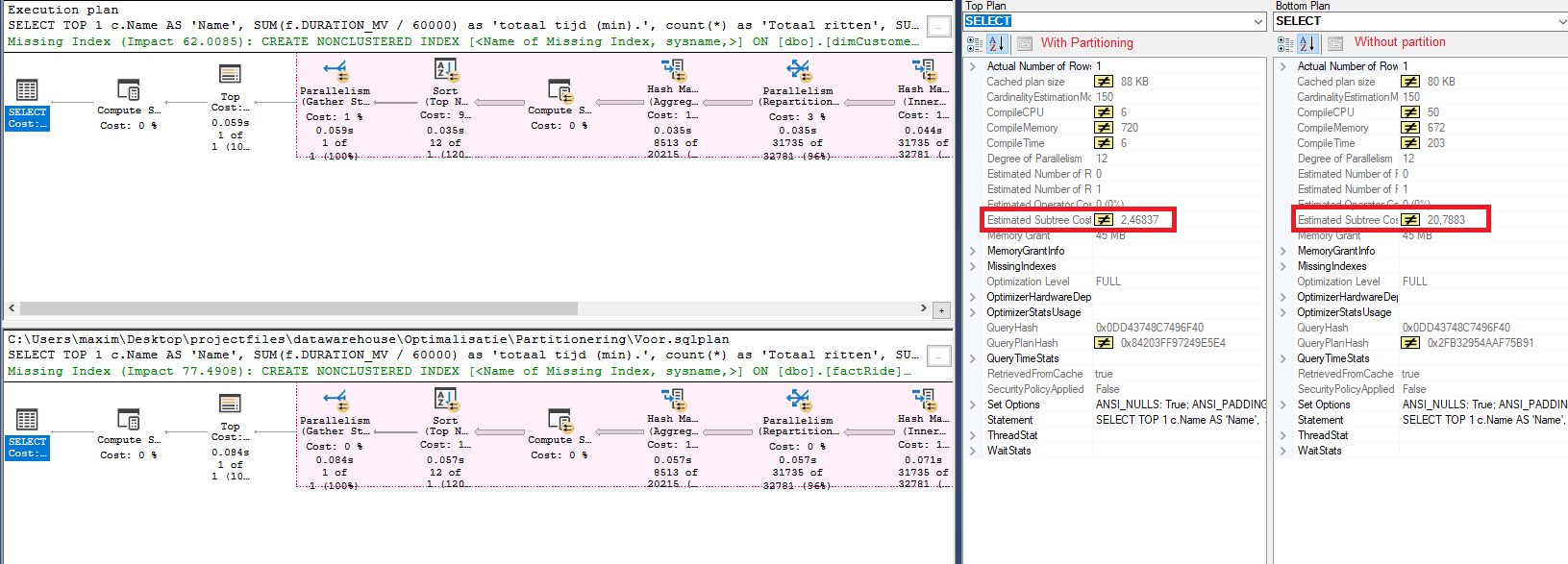
Om partities aan te maken gebruiken we in dit hoofdstuk SQL Server Management Studio.

We zullen opteren voor een SQL-Syntax optie i.p.v. een grafische omgeving. Zo kunnen we meer in detail gaan op de werking van partities zoals vorig jaar gezien bij Oracle.

## Werkelijke optimalisatie in werking

### Plan voor en na

We zien een duidelijke verbetering met een cost van 18, als we dit op grotere schaal gaan uitvoeren halen we hier veel voordelen uit.



### Hoe hebben we de optimalisatie gedaan

Filegroepen aanmaken voor het opslaan van de partitie op de harde schijf.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Voor elke filegroep een bestand

Afbeelding met tekst

Automatisch gegenereerde beschrijvingAfbeelding met tekst

Automatisch gegenereerde beschrijving

Afbeelding met tekst

Automatisch gegenereerde beschrijvingAfbeelding met tekst

Automatisch gegenereerde beschrijving

Ranges opstellen per maand / jaar (in dit geval simpel gehouden omdat we even puur een test willen doen voor die oktober)



Partities toewijzen aan de filegroepen

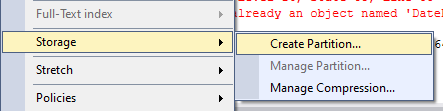
Afbeelding met tekst, oranje, sluiten, schermafbeelding

Automatisch gegenereerde beschrijving

De tabel indelen in deze partities kunnen we doen met de MSMS wizard:

Afbeelding met tafel

Automatisch gegenereerde beschrijvingAfbeelding met tafel

Automatisch gegenereerde beschrijving

Afbeelding met tekst

Automatisch gegenereerde beschrijving

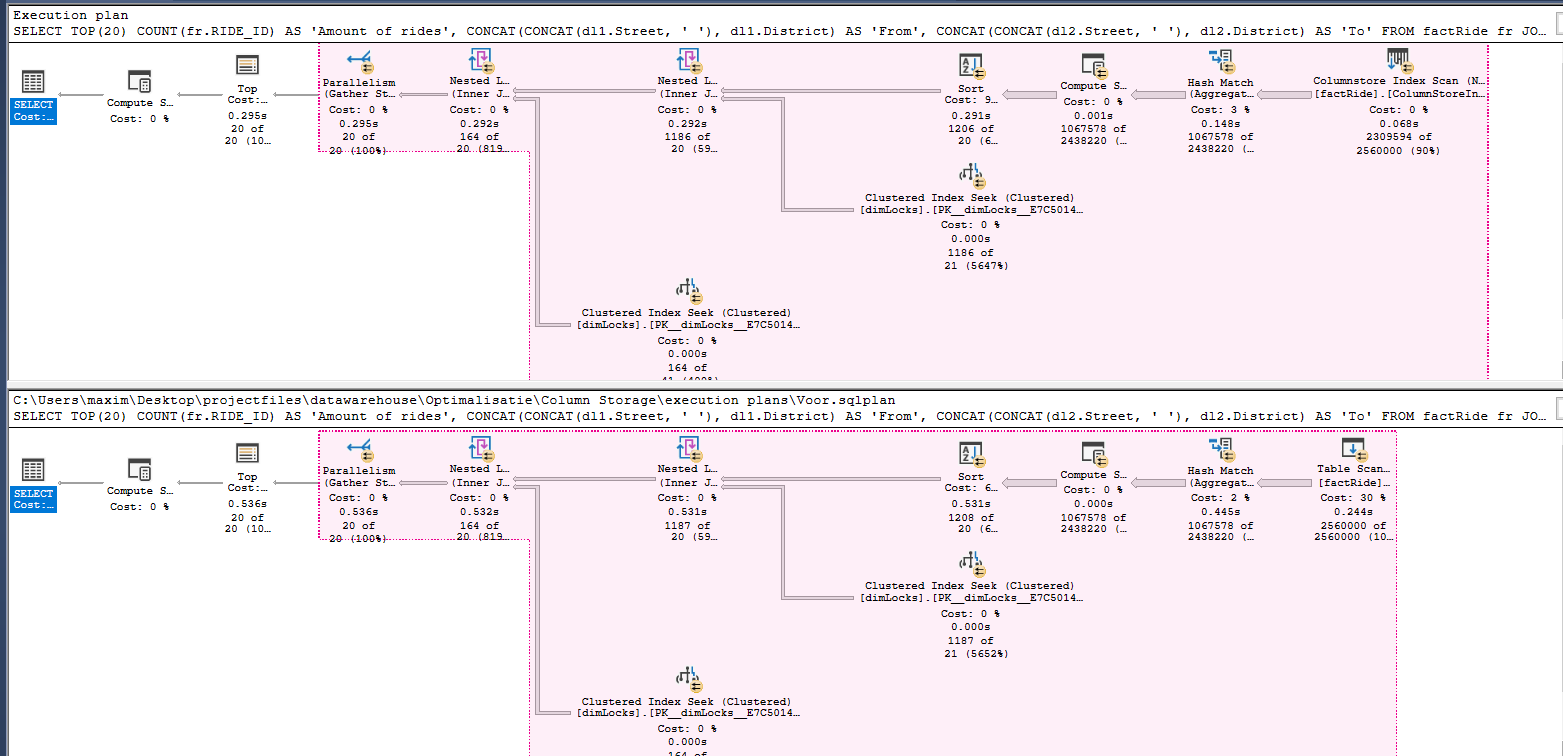
Afbeelding met tekst

Automatisch gegenereerde beschrijving

# Column storage [S2]

## Inleiding

Columnstore indexen zijn de standaard voor het opslaan en bevragen van grote gegevenstabellen voor datawarehousing. Deze index maakt gebruik van kolomgebaseerde gegevensopslag en queryverwerking om in de datawarehouse tot 10 keer betere queryprestaties te bereiken dan met traditionele rijgeoriënteerde opslag.



Afbeelding met tekst

Automatisch gegenereerde beschrijving

# Table Compression [S1]

## Inleiding

Het idee is simpel: Repitieve data in een kolom zoeken en en unieke informatie als rij of page of column store index opslagen.

Moderne compressie-algoritmen kunnen de voetafdruk op de schijf met 40-60% of zelfs meer verminderen, afhankelijk van het type gegevens.

Het comprimeren van uw SQL Server-gegevens is een eenvoudige manier om meer in uw beperkte schijfruimte te proppen

## Query keuze

Bij het kiezen van een tabel om compressie op toe te passen is het handig te kijken naar het aantal rijen in de tabel. Grote tabellen met veel rijen hebben het meeste voordeel bij compressie. Compressie heeft natuurlijk wel een relatief grote impact op CPU time, aangezien de data ingepakt/uitgepakt moet worden bij vele database operaties.