Mutatiedocument Casus
Relational Database Integration

Datum: 5-7-2019
Versie: 1.0

Klas: ADB DT

Docenten:
Mark Giesen
Bram Laumans

Marc Groenhout 527698
Nick Hartjes 423064
Joey Stoffels 609589

Aangepaste hoofdstukken

- Constraints
- Concurrency
- Indexeren

Inhoudsopgave

Constraints

Motivatie wijzigingen

De eerder door ons gemaakte constraints zijn gewijzigd. Deze wijzigingen variëren van kleine tot grote wijzigingen. Derhalve zijn alle constraints in dit mutatiedocument opgenomen. Hieronder zijn de verwijzingen naar de .sql bestanden opgenomen, deze zijn als bijlage in het .zip bestand terug te vinden.

Opdracht 1

Deze constraint zorgt ervoor dat een film of spel altijd bij minimaal één genre hoort. Dit geldt alleen voor de nieuwe films en spellen welke worden toegevoegd. Hierbij hebben we de reeds aanwezige films en spellen zonder genre buiten beschouwing gelaten.

Triggers

Zie Bijlage sql\herkansing\constraint-1-triggers.sql

Stored Procedures

Zie Bijlage sql\herkansing\constraint-1-stored-procedures.sql

Opdracht 2

Deze constraint zorgt ervoor dat een film met een previous part, altijd uitgebracht na de previous part. Dit geldt alleen voor de nieuwe films welke worden toegevoegd. Hierbij hebben we de reeds aanwezige films buiten beschouwing gelaten.

Triggers

Zie Bijlage sql\herkansing\constraint-2-triggers.sql

Stored Procedures

Zie Bijlage sql\herkansing\constraint-2-stored-procedures.sql

Opdracht 3

Deze constraint zorgt ervoor dat de abonnements periodes van een klant niet met elkaar kunnen overlappen. Het kan namelijk zo zijn dat een klant meerdere abonnements periodes heeft, maar deze dienen dus wel gescheiden van elkaar te zijn.

Triggers

Zie Bijlage sql\herkansing\constraint-3-triggers.sql

Stored Procedures

Zie Bijlage sql\herkansing\constraint-3-stored-procedures.sql

Opdracht 4

Deze constraint zorgt ervoor dat een film alleen kan worden bekeken als er een geldige abonnementsperiode actief is. Wanneer een klant geen geldige abonnementsperiode heeft, is deze niet in staat om een film te kijken.

Triggers

Zie Bijlage sql\herkansing\constraint-4-triggers.sql

Stored Procedures

Zie Bijlage sql\herkansing\constraint-4-stored-procedures.sql

Opdracht 5

Deze constraint zorgt ervoor dat er voor een filmrecensie altijd een cijfer voor Plot en Acting aanwezig is, daarnaast dient eveneens de rubriek Cinematography of Music and Sound beoordeeld te zijn. Zodra hieraan is voldaan, is de recensie geldig en wordt deze geaccepteerd.

Triggers

Zie Bijlage sql\herkansing\constraint-5-triggers.sql

Stored Procedures

Zie Bijlage sql\herkansing\constraint-5-stored-procedures.sql

Opdracht 6

Deze constraint bestaat uit twee delen. Deel één zorgt ervoor dat genres verschillende zijn voor films en spellen. Deze mogen niet bij het verkeerde media item worden gebruikt. Hetzelfde geldt echter voor het tweede gedeelte van deze constraint welke van toepassing is op reviews. De review categorieën zijn ook hier verschillende voor films en spellen.

Triggers

Zie Bijlage sql\herkansing\constraint-6-stored-procedures-categories.sql
Zie Bijlage sql\herkansing\constraint-6-stored-procedures-genres.sql

Stored Procedures

Zie Bijlage sql\herkansing\constraint-6-triggers-categories.sql
Zie Bijlage sql\herkansing\constraint-6-triggers-genres.sql

Concurrency

Motivatie wijzigingen

De grootste wijzigingen zijn in constraint 1 en 6 zijn doorgevoerd. Deze constraints zijn compleet herschreven, hierdoor dienden ook de eerder bepaalde isolation levels te worden herzien. Hieronder zijn de nieuwe isolation levels terug te vinden. Per trigger en constraint is vervolgens wederom beargumenteerd waarom er voor een bepaald transaction level is gekozen.

Concurrency constraint 1

Deze constraint zorgt ervoor dat een film of spel altijd bij minimaal één genre hoort. Dit geldt alleen voor de nieuwe films en spellen welke worden toegevoegd. Hierbij hebben we de reeds aanwezige films en spellen zonder genre buiten beschouwing gelaten.

Constraint	**Isolation level**	**Waarom?**
ProductGenre_AI	Read Committed	Trigger start impliciet een transactie met het standaard isolation level Read Committed. Bij het uitvoeren van deze trigger vindt er slechts één enkele DELETE plaats wanneer er een andere genre, niet zijnde het standaard genre, wordt toegevoegd. In dit geval gaan we het isolation level niet wijzigen, standaard waarde volstaat voor deze opdracht waardoor we niet te maken krijgen met een Lost Update, we willen immers de genre verwijderen. Daarnaast waarborgen we eveneens het feit dat er geen Dirty Read plaats kan vinden, onze transactie is in die tussentijd namelijk nog niet definitief. Deze informatie dient ondertussen niet in te zien te zijn.
ProductGenre_AD	Read Committed	Trigger start impliciet een transactie met het standaard isolation level Read Committed. Bij het uitvoeren van deze trigger wordt gecontroleerd of er minimaal nog 1 genre voor de film of het spel opgegeven is. Als dit niet het geval is, wordt er een foutmelding getoond. Door het standaard isolation level Read Committed te gebruiken voorkomen we dat de genre kan worden gewijzigd of gelezen, terwijl deze eigenlijk op het punt staat om verwijderd te worden. Pas na het afhandelen van de transactie zal het uiteindelijke resultaat zichtbaar zijn.

Product_AI	Read Committed	Trigger start impliciet een transactie met het standaard isolation level Read Committed. Bij het uitvoeren van deze trigger wordt er, nadat er reeds een film of spel is toegevoegd aan de tabel Product, ook een standaard genre toegevoegd. Ook hier willen we niet dat er sprake is van een Lost Update, evenals een Dirty Read. Het toevoegen van de gegevens is voor het beëindigen van de transactie namelijk nog niet definitief.
SP_ProductGenreInsert	Read Committed	In deze stored procedure wordt een transactie gestart, zonder daarbij een andere dan het standaard isolation level te kiezen. We willen zonder dat de transactie afgerond geen Lost Update of Dirty Read. Het standaard isolation level is hier derhalve afdoende. Bij het uitvoeren van deze stored procedure vindt er een delete en insert in dezelfde tabel plaats. Hierbij wordt de standaard genre, indien aanwezig, vervangen voor de nieuwe genre.
SP_ProductGenreDelete	Read Committed	In deze stored procedure wordt een transactie gestart, zonder daarbij een andere dan het standaard isolation level te kiezen. We willen zonder dat de transactie afgerond geen Lost Update of Dirty Read. Het standaard isolation level is hier derhalve afdoende. Bij het uitvoeren van deze stored procedure wordt er een genre verwijder uit de tabel. Indien dit de laatste genre is, wordt er een foutmelding getoond.
SP_ProductInsert	Read Committed	In deze stored procedure wordt een transactie gestart, zonder daarbij een andere dan het standaard isolation level te kiezen. We willen zonder dat de transactie afgerond geen Lost Update of Dirty Read. Het standaard isolation level is hier derhalve afdoende. Bij het uitvoeren van deze stored procedure wordt er voor ieder toegevoegd product een standaard genre toegevoegd. Vanwege de transactie slaagt of faalt dit al geheel.
SP_ProductDelete	Read Committed	In deze stored procedure wordt een transactie gestart, zonder daarbij een andere dan het standaard isolation level te kiezen. We willen

		zonder dat de transactie afgerond geen Lost Update of Dirty Read. Het standaard isolation level is hier derhalve afdoende. Bij het uitvoeren van deze stored procedure wordt niet alleen het product, maar ook de genres verwijderd. Vanwege de transactie slaagt of faalt dit al geheel.

Concurrency constraint 6

Deze constraint bestaat uit twee delen. Deel één zorgt ervoor dat genres verschillende zijn voor films en spellen. Deze mogen niet bij het verkeerde media item worden gebruikt. Hetzelfde geldt echter voor het tweede gedeelte van deze constraint welke van toepassing is op reviews. De review categorieën zijn ook hier verschillende voor films en spellen.

Constraint	**Isolation level**	**Waarom?**
ReviewCategory_AI_AU	Read Committed	Trigger start impliciet een transactie met het standaard isolation level Read Committed. Bij het uitvoeren van deze trigger wordt gecontroleerd of de gebruikte review categorie voor het product type mag worden gebruikt. Hierbij is het standaard isolation level Read Committed voldoende en zal deze ook niet worden gewijzigd. Hiermee voorkomen we dat er een Lost Update of een Dirty Read kan plaatsvinden wanneer de transactie nog niet afgerond is.
ProductGenre_AI_AU	Read Committed	Trigger start impliciet een transactie met het standaard isolation level Read Committed. Bij het uitvoeren van deze trigger wordt gecontroleerd of de gebruikte genre voor het product type mag worden gebruikt. Hierbij is het standaard isolation level Read Committed voldoende en zal deze ook niet worden gewijzigd. Hiermee voorkomen we dat er een Lost Update of een Dirty Read kan plaatsvinden wanneer de transactie nog niet afgerond is.
SP_ReviewCategoryInsert	Read Committed	In deze stored procedure wordt een transactie gestart, zonder daarbij een andere dan het standaard isolation level te kiezen. We willen zonder dat de transactie afgerond geen Lost Update of Dirty Read. Het standaard isolation level

		is hier derhalve afdoende. Er wordt enkel een review voor een categorie toegevoegd, ondertussen kan o.a. het product type niet worden gewijzigd, maar kunnen de gegevens van het product wel gewoon worden gelezen.
SP_ReviewCategoryUpdate	Read Committed	In deze stored procedure wordt een transactie gestart, zonder daarbij een andere dan het standaard isolation level te kiezen. We willen zonder dat de transactie afgerond geen Lost Update of Dirty Read. Het standaard isolation level is hier derhalve afdoende. Er wordt enkel een review categorie gewijzigd, ondertussen kan o.a. het product type niet worden gewijzigd, maar kunnen de gegevens van het product wel gewoon worden gelezen.
SP_ProductGenreInsert	Read Committed	In deze stored procedure wordt een transactie gestart, zonder daarbij een andere dan het standaard isolation level te kiezen. We willen zonder dat de transactie afgerond geen Lost Update of Dirty Read. Het standaard isolation level is hier derhalve afdoende. Er wordt enkel een genre voor een product toegevoegd, ondertussen kan o.a. het product type niet worden gewijzigd, maar kunnen de gegevens van het product wel gewoon worden gelezen.
SP_ProductGenreUpdate	Read Committed	In deze stored procedure wordt een transactie gestart, zonder daarbij een andere dan het standaard isolation level te kiezen. We willen zonder dat de transactie afgerond geen Lost Update of Dirty Read. Het standaard isolation level is hier derhalve afdoende. Er wordt enkel een genre van een product gewijzigd, ondertussen kan o.a. het product type niet worden gewijzigd, maar kunnen de gegevens van het product wel gewoon worden gelezen.

Indexeren

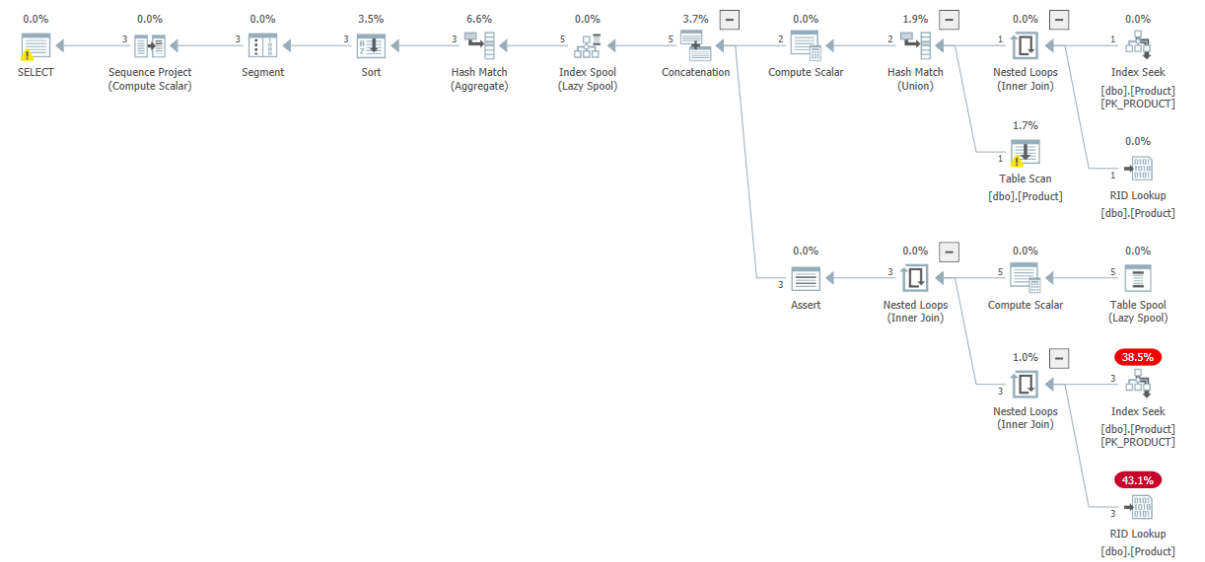
Motivatie wijzigingen

Naar aanleiding van de eerdere beoordeling is gebleken dat de indexering beter kon. Vanuit PowerDesigner werden er standaard nonclustered indexes op kolommen geplaatst. Dit had tot gevolg dat er bij het uitvoeren van verschillende queries RID-lookups werden gedaan. Het doel was om alle RID-lookups te voorkomen en de performance te verbeteren, waardoor de benodigde waarden inmiddels als primary key clustered indexes terug te vinden zijn. De resultaten zijn hieronder beschreven.

Query 1

Zie Bijlage sql\herkansing\index-query-01.sql

Initieel query plan:



Afbeelding 01 - Query plan query 1 zonder index

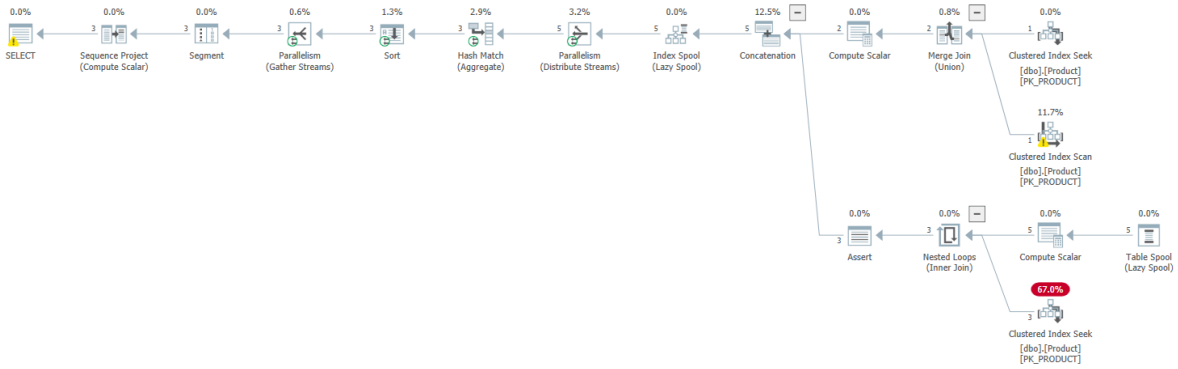
```
--  SQL Server Execution Times:
--    CPU time = 0 ms,  elapsed time = 0 ms.
-- SQL Server parse and compile time:
--    CPU time = 0 ms, elapsed time = 0 ms.
--
--  SQL Server Execution Times:
--    CPU time = 0 ms,  elapsed time = 0 ms.
--
-- (3 rows affected)
-- Table 'Worktable'. Scan count 2, logical reads 29, physical reads 0, read-ahead reads 0, lob
logical reads 0, lob physical reads 0, lob read-ahead reads 0.
-- Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob
logical reads 0, lob physical reads 0, lob read-ahead reads 0.
-- Table 'Product'. Scan count 1, logical reads 3366, physical reads 0, read-ahead reads 0, lob
logical reads 0, lob physical reads 0, lob read-ahead reads 0.
--
```

```
-- (1 row affected)
--
--  SQL Server Execution Times:
--    CPU time = 65 ms,  elapsed time = 64 ms.
```

Een merendeel van de query tijd gaat verloren in de RID Lookup. Na een index seek, vult de Database engine aan met specifieke extra data door middel van een RID Lookup. Alleen gebeurd deze RID Lookup alleen wanneer er geen clustered index op de tabel staat, anders zou het een key lookup moeten zijn.

Nadat we de non-clustered index van de Product table hebben verwijderd, en vervangen hebben voor een clustered index zien we het volgende resultaat.

Initieel query plan:



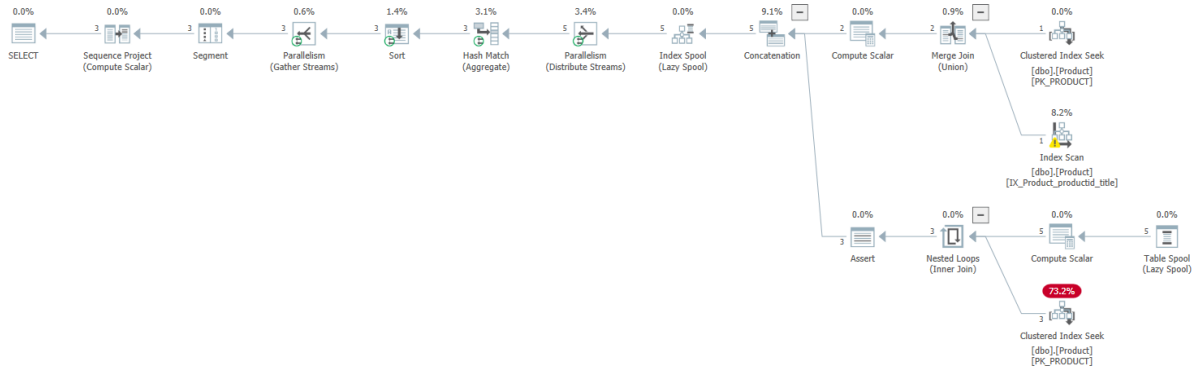
Afbeelding 02 - Query plan query 1 met nieuwe clustered Primary Key

```
SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(3 rows affected)
Table 'Product'. Scan count 1, logical reads 3374, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Worktable'. Scan count 2, logical reads 29, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.

(1 row affected)

 SQL Server Execution Times:
   CPU time = 50 ms,  elapsed time = 52 ms.
```

Dit levert al goede optimalisatie op. Alleen de WHERE previous_product_id = @MovieInReeks verloopt nog via een Clustered scan. Deze kan overigens toch nog enigszins geoptimaliseerd worden door de volgende non clustered index te zetten

```
--  ------------------------------------------------------
--  Create index
--  ------------------------------------------------------
CREATE NONCLUSTERED INDEX IX_Product_productid_title ON Product
(product_id
) INCLUDE(title, previous_product_id, publication_year);
```

Query plan na wijzigingen:



Afbeelding 03 - Query plan query 1 met nieuwe clustered Primary Key en index

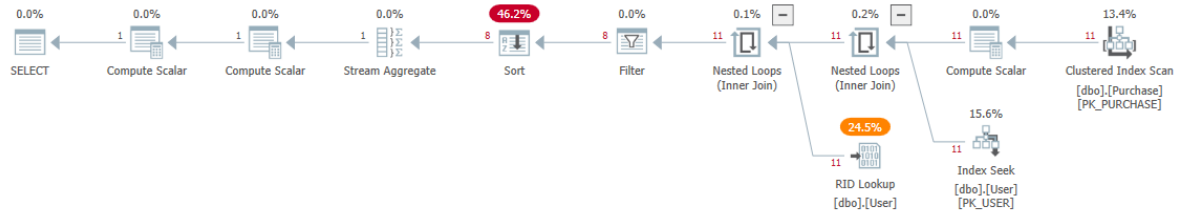
```
 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

(3 rows affected)
Table 'Product'. Scan count 1, logical reads 1957, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Worktable'. Scan count 2, logical reads 29, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.

(1 row affected)

 SQL Server Execution Times:
   CPU time = 56 ms,  elapsed time = 58 ms.
```

De snelheidswinst met deze optimalisatie is minimaal, derhalve zou ik er niet voor kiezen om deze index in het systeem te houden, tenzij deze query veelvuldig per minuut wordt uitgevoerd en daardoor waarschijnlijk belangrijk is.

Query 2a

Zie Bijlage sql\herkansing\index-query-02a.sql

Initieel query plan:

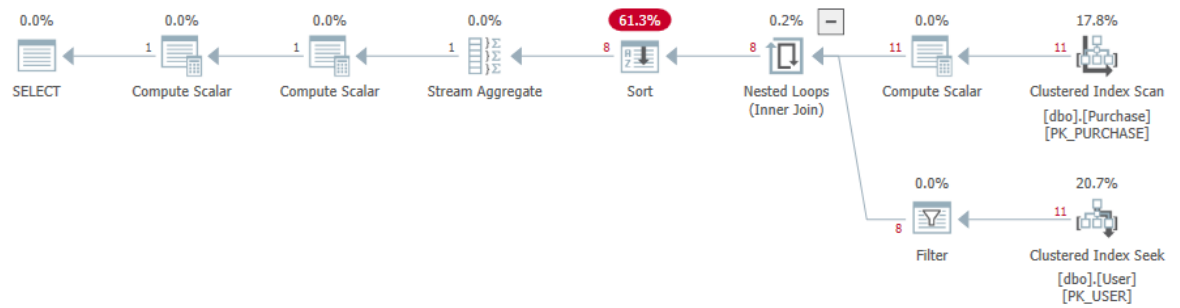


Afbeelding 04 - Query plan query 2a zonder index

```
--  SQL Server Execution Times:
--    CPU time = 0 ms,  elapsed time = 2 ms.
-- SQL Server parse and compile time:
--    CPU time = 15 ms, elapsed time = 15 ms.
--
--  SQL Server Execution Times:
--    CPU time = 0 ms,  elapsed time = 0 ms.
--
--  SQL Server Execution Times:
--    CPU time = 0 ms,  elapsed time = 0 ms.
--
--  SQL Server Execution Times:
--    CPU time = 0 ms,  elapsed time = 0 ms.
--
-- (1 row affected)
-- Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob
logical reads 0, lob physical reads 0, lob read-ahead reads 0.
-- Table 'User'. Scan count 0, logical reads 33, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.
-- Table 'Purchase'. Scan count 1, logical reads 2, physical reads 0, read-ahead reads 0, lob
logical reads 0, lob physical reads 0, lob read-ahead reads 0.
--
-- (1 row affected)
--
--  SQL Server Execution Times:
--    CPU time = 0 ms,  elapsed time = 79 ms.
```

Wederom hebben we hier RID Lookups en zullen we eerst de Primary Key van User moeten optimaliseren. Na het zetten van clustered Primary Key, krijgen we het volgende query plan.

Query plan na wijzigingen:



Afbeelding 05 - Query plan query 2a met nieuwe Primary Key

In dit geval zie je dat de 2 data streams moet worden samengevoegd, dit gebeurd in de Stream Aggregate. Deze werkt overigens alleen maar wanneer de data al gesorteerd wordt aangeleverd. En in ons geval kost dit heeft veel tijd.

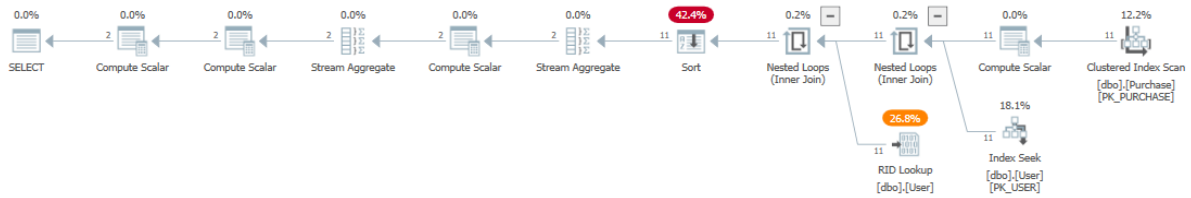
```
--  ------------------------------------------------------
--  Create index
--  ------------------------------------------------------
CREATE NONCLUSTERED INDEX IX_User_emailaddress_countryname ON [User](email_address)
INCLUDE(country_name);
CREATE NONCLUSTERED INDEX IX_Purchase_emailaddress_purchase_date ON [Purchase](email_address)
INCLUDE(purchase_date);

--  ------------------------------------------------------
--  Remove index
--  ------------------------------------------------------
DROP INDEX IX_User_emailaddress_countryname ON [User];
DROP INDEX IX_Purchase_emailaddress_purchase_date ON [Purchase];
```

We hebben geprobeerd om verschillende indexen te zetten, alleen heeft dit niet geleid tot performance winst. De queryoptimzer blijft gebruik maken van de sort. Dit heeft er toe geleid dat we alle indexen weer verwijderd hebben.

Query 2b

Zie Bijlage sql\herkansing\index-query-02b.sql

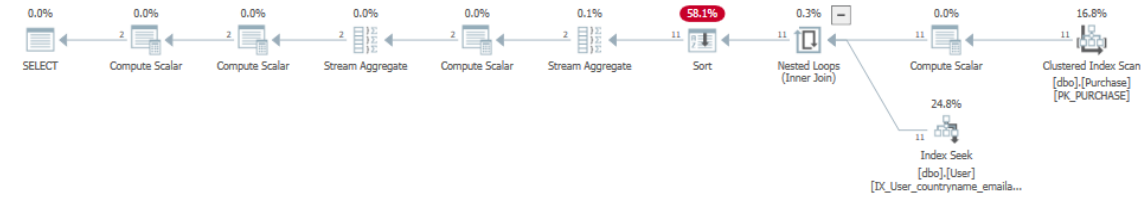
Initieel query plan:



Afbeelding 06 - Query plan query 2b zonder index

Voor opdracht 2b geldt dezelfde optimalisatie als voor 2a. Door de een clustered Primary Key op de User tabel te zetten wordt de query al geoptimaliseerd.
Meerdere pogingen met het zetten van indexen, hebben niet geleid tot een verbetering van de perfomance.

Query plan na wijzigingen:

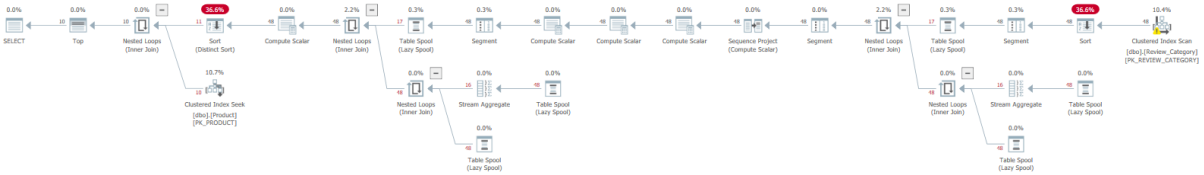


Afbeelding 07 - Query plan query 2b met index

Query 3

Zie Bijlage sql\herkansing\index-query-03.sql

Query plan:

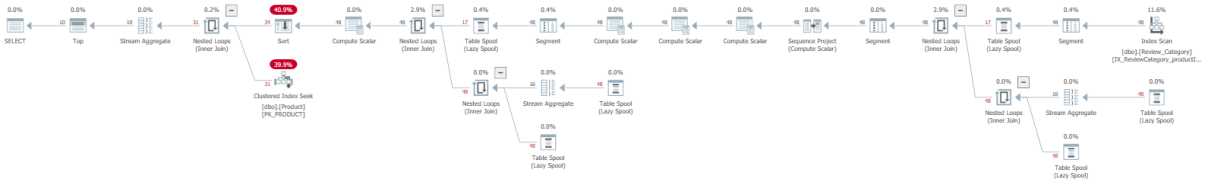


Afbeelding 08 - Query plan query 3 zonder index

Na de clustered index scan wordt de data gesorteerd, dit kunnen we oplossen door de data al direct op de juiste manier uit een index te halen. Dit levert een kleine performance winst op.

```
--  ------------------------------------------------------------
--  Create index
--  ------------------------------------------------------------
CREATE NONCLUSTERED INDEX IX_ReviewCategory_productId_score ON Review_Category(product_id,score);
```

Query plan:

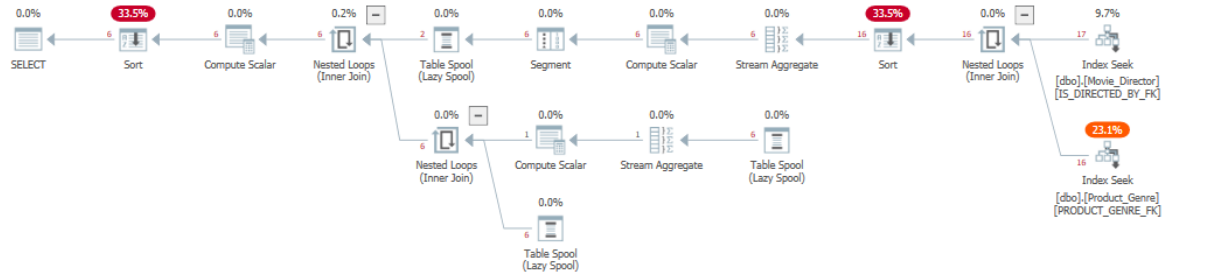


Afbeelding 09 - Query plan query 3 met index

Query 4

Zie Bijlage sql\herkansing\index-query-04.sql

Query plan:



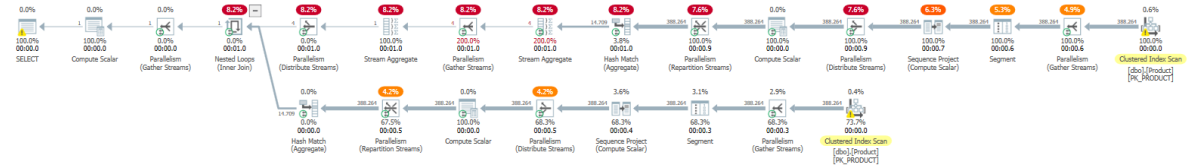
Afbeelding 10 - Query plan query 4 zonder index

Conclusie:
De meeste tijd gaat zitten in het sorteren na een join. Dit gebeurd 2 keer. Helaas kun je door het zetten van een specifieke index hier geen invloed op uitoefenen. De Movie_director en Product_genre haal hij op via een index seek. Veel sneller dan dat kan het niet. Zou deze query een limitatie in het systeem worden, dan zou het mogelijk zijn om de data die nodig is bij elkaar te zetten in een tijdelijke tabel of een view. Zodat alles beter geoptimaliseerd zou kunnen worden.

Query 5a

Zie Bijlage sql\herkansing\index-query-05a.sql

Query plan:



Afbeelding 11 - Query plan query 5a zonder index

```
-- SQL Server parse and compile time:
--    CPU time = 0 ms, elapsed time = 0 ms.
--
--  SQL Server Execution Times:
--    CPU time = 0 ms,  elapsed time = 0 ms.
-- SQL Server parse and compile time:
--    CPU time = 0 ms, elapsed time = 0 ms.
--
--  SQL Server Execution Times:
--    CPU time = 0 ms,  elapsed time = 0 ms.
--
--  SQL Server Execution Times:
--    CPU time = 0 ms,  elapsed time = 0 ms.
-- SQL Server parse and compile time:
--    CPU time = 18 ms, elapsed time = 18 ms.
--
-- (1 row affected)
-- Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob
logical reads 0, lob physical reads 0, lob read-ahead reads 0.
-- Table 'Product'. Scan count 10, logical reads 6700, physical reads 0, read-ahead reads 0, lob
logical reads 0, lob physical reads 0, lob read-ahead reads 0.
-- Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob
```

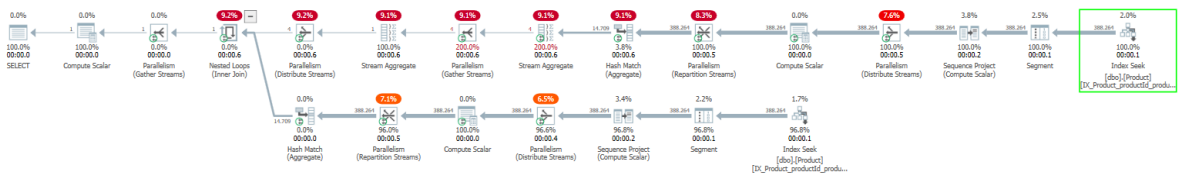
```
logical reads 0, lob physical reads 0, lob read-ahead reads 0.
--
--  SQL Server Execution Times:
--    CPU time = 2219 ms,  elapsed time = 1277 ms.
```

Omdat er 2 keer de Product tabel wordt geraadpleegd, om te achterhalen welke product_type er is. Dit wordt in beide keren met een clustered index scan gedaan. Nu gaat hij elk record af om te kijken of er aan de waarde wordt voldaan. We kunnen dit redelijk goed versnellen door een nonclustered index te zetten op de kolom product_id en product_type erbij includen

Na de index maakt de query geen gebruik meer van de clustered index maar van een index seek. Het aantal logical reads is daarmee ook bijna twee derde afgenomen.

```
--  ------------------------------------------------------------
--  Create index
--  ------------------------------------------------------------
CREATE NONCLUSTERED INDEX IX_Product_productId_productType ON Product(product_id, product_type);

SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 18 ms, elapsed time = 18 ms.

(1 row affected)
Table 'Product'. Scan count 2, logical reads 1836, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.

(1 row affected)

 SQL Server Execution Times:
   CPU time = 2079 ms,  elapsed time = 1229 ms.

--  ------------------------------------------------------------
--  Remove index
--  ------------------------------------------------------------
CREATE NONCLUSTERED INDEX IX_Product_productId_productType ON Product(product_type)
INCLUDE(product_id);
```

Query plan:

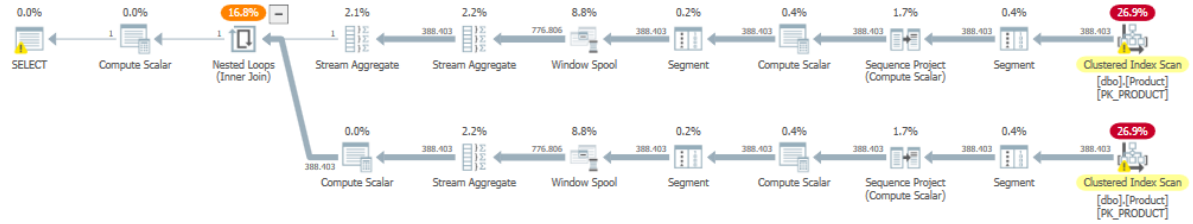


Afbeelding 12 - Query plan query 5a met index

Query 5b

Zie Bijlage sql\herkansing\index-query-05b.sql

Query plan:



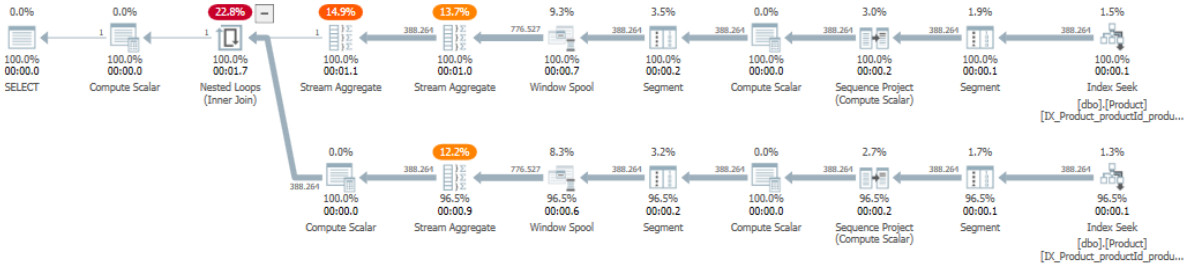
Afbeelding 13 - Query plan query 5b zonder index

```
 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Product'. Scan count 2, logical reads 6724, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.

(1 row affected)

 SQL Server Execution Times:
   CPU time = 2246 ms,  elapsed time = 2247 ms.
```

De queries 5a en 5b zijn bijna identiek, de analyse en de index geldt voor beide query's.
De query 5b kan dus dezelfde index gebruiken als 5a.

```
 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 9 ms, elapsed time = 9 ms.
Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Product'. Scan count 2, logical reads 1836, physical reads 0, read-ahead reads 0, lob logical
reads 0, lob physical reads 0, lob read-ahead reads 0.

(1 row affected)

 SQL Server Execution Times:
```

```
   CPU time = 1922 ms,  elapsed time = 1923 ms.
```



Afbeelding 14 - Query plan query 5b met index