Netflix DATABASE

*Storm Visser  
15-06-2021*

Snelheidstesten op de database van Netflix

# Achtergronden

Netflix wil weten of hedendaagse modernere technieken een positieve bijdragen kunnen leveren aan de snelheid van de database. De afgelopen weken zijn deze onderzoeken uitgevoerd en in dit rapport worden de resultaten besproken.

# Testmethode

In dit hoofdstuk wordt kort besproken hoe de snelheden zijn getest.

## Test Software

Om de testen uit te voeren is er een eigen programma gemaakt. Dit programma test alle CRUD-operaties voor 1, 100, 1000 en 10000 rijen. De testen zullen allemaal drie keer worden uitgevoerd, waarna er een gemiddelde kan worden uitgerekend. Dit wordt gedaan zodat de resultaten betrouwbaarder zijn omdat eventuele abnormale resultaten worden voorkomen.

De software en het bijbehorende versiebeheer is te vinden op <https://github.com/Storm-Visser/Databases2> en de software zal ook worden meegestuurd in de RAR-file.

## Hardware

De test software en de database server worden uiteraard gedraaid op een computer. De hardware van de computer zal invloed hebben op de testresultaten. Daarom zal de computer voor alle tests gebruikt worden. Ook zullen de tests op dezelfde dag plaatsvinden zodat eventuele invloeden van temperatuur worden geminimaliseerd.

De computer die wordt gebruikt heeft de volgende specificaties:

* CPU: AMD Ryzen 5 3600
* RAM-geheugen: G-Skill Trident-Z RGB 16GB DDR4 3600MHz
* GPU: MSI GeForce GTX 980
* SSD: Samsung 970 EVO 500 GB M.2

# Resultaten

In dit hoofdstuk worden de gemiddelde resultaten van elke CRUD-operatie getoond. Daardoor zijn de resultaten het beste te vergelijken. Voor een volledig overzicht van alle testresultaten, zie het Exel bestand genaamd TestResultaat.xlsx.

## Create

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Gemiddeld resultaat in MS | 1 rij | 100 rijen | 1000 rijen | 10000 rijen |
| ADO.NET | 228,333 | 21 | 191 | 1790 |
| Entity Framework | 300,667 | 215 | 1683,333 | 16155,333 |
| NoSQL | 1 | 27,667 | 264,667 | 2347 |

Zoals hier te zien is, is ADO.NET de snelste optie en daaropvolgend is NoSQL met als langzaamste Entity Framework.

## Read

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Gemiddeld resultaat in MS | 1 rij | 100 rijen | 1000 rijen | 10000 rijen |
| ADO.NET | 7 | 9 | 84,333 | 849 |
| Entity Framework | 25,333 | 45,333 | 397,333 | 3980,667 |
| NoSQL | 52 | 2923,333 | 29387,667 | 319414,333 |

Zoals hier te zien is, is ADO.NET de snelste optie en daaropvolgend is Entity Framework met als langzaamste NoSQL.

## Update

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Gemiddeld resultaat in MS | 1 rij | 100 rijen | 1000 rijen | 10000 rijen |
| ADO.NET | 1,333 | 26,667 | 198,667 | 1893,667 |
| Entity Framework | 50,667 | 205,333 | 1802,333 | 16411 |
| NoSQL | 74 | 2997,667 | 30012,333 | 328617,333 |

Zoals hier te zien is, is ADO.NET de snelste optie en daaropvolgend is Entity Framework met als langzaamste NoSQL.

## Delete

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Gemiddeld resultaat in MS | 1 rij | 100 rijen | 1000 rijen | 10000 rijen |
| ADO.NET | 11,333 | 23,667 | 342,667 | 9635,667 |
| Entity Framework | 3 | 38 | 365,667 | 3641,333 |
| NoSQL | 32,667 | 2921,667 | 29425,667 | 296272,333 |

Zoals hier te zien is, is ADO.NET weer de snelste optie met daaropvolgend is Entity Framework met als langzaamste NoSQL.

# Conclusie

In dit hoofdstuk wordt een conclusie getrokken aan de hand van de eerder besproken resultaten. En worden er waar mogelijk formules gemaakt, deze formules zijn schattingen aan de hand van de verkregen resultaten, en zijn dus niet volledig betrouwbaar. Ook zijn er grafieken die de tijd (Y as) laten zien aan de hand van het aantal rijen (X as)

### Create

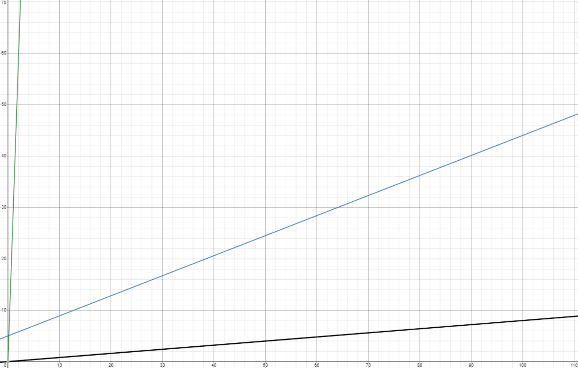
Bij de test van ADO.NET en Entity Framework is er te zien dat 1 rij langer duurt dan 100 rijen, wat natuurijk een vreemd fenomeen is. Ik denk dat dit komt omdat de tabel eerst volledig leeg is, maar deze theorie is niet getest.

Vanwege dit vreemde verschijnsel kan er geen duidelijke formule getrokken worden uit de genoemde twee opties. De formule kan wel gemaakt worden bij de overige optie, NoSQL. De formule hiervoor is;

* **Tijd in MS = aantal rijen \* 0,25 ;**

### Read

Bij de tests voor het lexen van de data zijn geen vreemde fenomen voorgekomen, het enige wat opvalt is dat NoSQL veel langer duurt dan de andere opties. De formules die dit benaderen zijn als volgt;

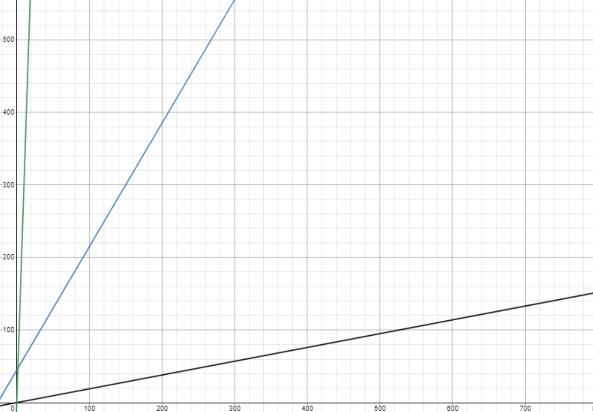
* **ADO.NET: (zwart)**

**Tijd in MS = aantal rijen \* 0,08;**

* **Entity Framework: (blauw)  
  Tijd in MS = aantal rijen \* 0.39 + 5**
* **NoSQL: (groen)  
  Tijd in MS = aantal rijen \* 30 – aantal rijen;**

Dit levert de volgende grafiek;  
Tijd is de Y as en aantal rijen de X as

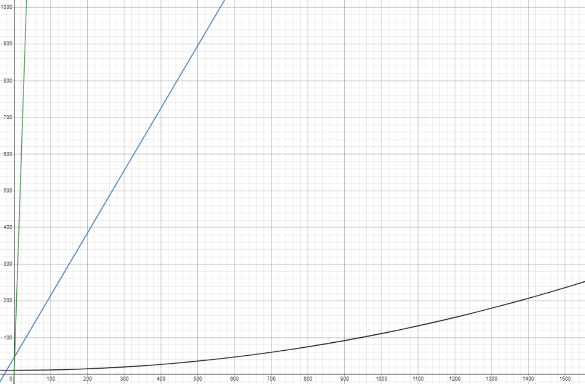
## Update

Bij de tests voor het lexen van de data zijn weer geen vreemde fenomen voorgekomen, het enige wat opvalt is dat NoSQL weer veel langer duurt dan de andere opties. De formules die dit benaderen zijn als volgt;

* **ADO.NET: (zwart)  
  Tijd in MS = aantal rijen \* 0,19 ;**
* **Entity Framework: (blauw)  
  Tijd in MS = aantal rijen \* 1.7 + 45;**
* **NoSQL: (groen)  
  Tijd in MS = aantal rijen \* 30;**

Dat levert de volgende grafiek:   
Tijd is de Y as en aantal rijen de X as

### Delete

****Bij de tests voor het lexen van de data zijn weer geen vreemde fenomen voorgekomen, het enige wat opvalt is dat NoSQL weer veel langer duurt dan de andere opties. De formules die dit benaderen zijn als volgt;

* **ADO.NET: (zwart)  
  Tijd in MS = 0.0001 \* aantal rijen^2 + 11;**
* **Entity Framework: (blauw)  
  Tijd in MS = aantal rijen \* 0,36 + 2;**
* **NoSQL: (groen)  
  Tijd in MS = aantal rijen \* 29.5 + 2;**

Dat levert de volgende grafiek:   
Tijd is de Y as en aantal rijen de X as

Hier is te zien dat, ondanks dat het hiervoor niet het geval was, er voor ADO.NET een kwadratische vergelijking is. Dit betekent dat ADO.NET voor meer delete statements steeds langer gaat duren.

# Aanbeveling

In dit hoofdstuk wordt er aan de hand van de conclusie een aanbeveling gedaan voor Netflix, over welke optie het beste is om te kiezen. Ook worden een aantal discussiepunten in kaart gebracht.

### Discussie

Tijdens het testen is er zo veel mogelijk rekening gehouden met vergelijkbare omstandigheden, maar ondanks dat is er bij de resultaten van NoSQL te zien dat de tweede en derde poging extreem veel langer duurden. Er is hier geen logische verklaring voor gevonden en dit kan dus de resultaten hebben beïnvloed. Omdat er is gekozen om drie keer te testen en het gemiddelde te gebruiken is die invloed relatief klein gemaakt en zijn de resultaten zo betrouwbaar mogelijk gebleven. Er is overwogen om de eerste poging door te strepen of om hem opnieuw te proberen, maar daar is niet voor gekozen.

Daarnaast wordt er voor het deleten van ADO.NET gebruik gemaakt voor een iets ingewikkeldere query, wat ervoor kan zorgen dat de resultaten langer duren dan met een simpelere query.

Het was eerst het idee om met 1, 1000, 100000 en 1000000 rijen te testen, maar dit is niet gedaan omdat de testen dan simpelweg te lang duren; voor 3 keer testen ongeveer 74 uur. Daarnaast zijn er aan de hand van het huidige resultaat formules gemaakt, daarmee kan een schatting gemaakt worden van de tijd die het kost om zoveel rijen uit te voeren.

### Aanbeveling

Aan de hand van de testresultaten en de conclusie zal er worden aangeraden dat Netflix gebruik blijft maken van de ADO.NET database. Dit is de snelste manier in bijna alle tests. Het enige nadeel is dat als er data wordt gedeletet, de tijd kwadratisch omhooggaat. Maar omdat de desbetreffende formule zo langzaam gaat voor een kwadratische formule, is hier toch voor gekozen.

Dit heeft als bijkomend voordeel dat aangezien Netflix momenteel ook ADO.NET gebruikt, er geen grote migratie hoeft te komen naar een nieuwe database en systeem. En dit bespaart veel tijd en kosten.