≠–

Onderzoeksrapport

Document-oriented databasemanagementsystemen



Ciyan Cöcelli 548189

Roel de Man 489962

Alex Post 531671

Niels Wijers 564429

Marijn Martin 547235

Marijn Martin 547235

ICT

Webdevelopment

18 november 2016

Versie 1.0

Hogeschool Arnhem Nijmegen

DWA –Sep

Inhoudsopgave

[1. Inleiding 3](#_Toc467754639)

[2. Wat is een schemaless database? 4](#_Toc467754640)

[2.1. Inleiding 4](#_Toc467754641)

[2.2 Relationele databases 4](#_Toc467754642)

[2.3 NoSQL 4](#_Toc467754643)

[2.3.1 NoSQL Database Types 4](#_Toc467754644)

[2.5 NoSQL vs SQL 6](#_Toc467754645)

[2.6 Sharding 7](#_Toc467754646)

[2.7 conclusie 7](#_Toc467754647)

[3. Wat is een document-oriented database? 8](#_Toc467754648)

[3.1. Problemen met relationele databases 8](#_Toc467754649)

[3.2. Document-oriented databases 9](#_Toc467754650)

[3.3. Conclusie 10](#_Toc467754651)

[4. Welke criteria zijn er te stellen aan document-oriented databasemanagementsystemen en welke gevonden databases voldoen aan deze criteria? 11](#_Toc467754652)

[4.1 Inleiding 11](#_Toc467754653)

[4.2 Criteria 11](#_Toc467754654)

[4.3 DBMS-schema 12](#_Toc467754655)

[4.4 Conclusie 12](#_Toc467754656)

[5. Conclusie 13](#_Toc467754657)

[6. Literatuurlijst 14](#_Toc467754658)

# 1. Inleiding

Dit onderzoek gaat over document-oriented databasemanagementsystemen(DBMS). Hierin zal onderzocht worden welk systeem het beste te gebruiken is binnen de tijdsduur van het project.

Voor onze applicatie is het van belang dat er meerdere DBMS-en API’s (Application Programming Interface) konden worden toegevoegd, daarom was het toepasselijk om een tweede DBMS te implementeren in de applicatie.

Met de hoofdvraag: ‘Welk document-oriented databasemanagementsysteem past het beste naast MongoDB in de applicatie?’ zijn er deelvragen opgesteld.

* Wat is een schemaless database?
* Wat is een document-oriented database?
* Welke criteria zijn er te stellen aan document-oriented databasemanagementsystemen en welke gevonden databases voldoen aan deze criteria?

Door middel van het beantwoorden van deze deelvragen komen wij tot een antwoord op de hoofdvraag. In de laatste deelvraag zullen er een aantal criteria opgesteld worden waarmee de mogelijkheden met elkaar vergeleken zullen worden. Aan de hand van deze criteria wordt in de conclusie een databasemanagementsysteem gekozen die voor deze opdracht het meest van toepassing is.

# 2. Wat is een schemaless database?

## 2.1. Inleiding

Om antwoord te kunnen vinden op de hoofdvraag is het natuurlijk essentieel dat er allereerst duidelijk is wat nou precies een schemaless database is. Bij deze hoofdvraag wordt er duidelijk gemaakt wat de term inhoudt en welke databases gebruik maken van een schema en welke niet.

## 2.2 Relationele databases

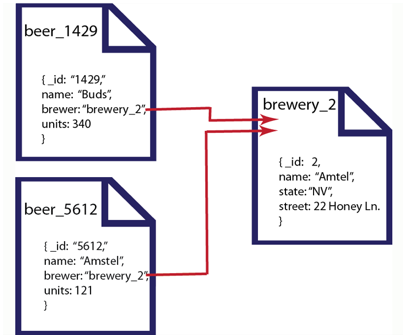
Relationele databases maken gebruik van een schema om een standaard te definiëren. De naam verklapt het al, data kan een relatie hebben met andere data. Dit is zeer voorkomend omdat je meerdere rasters hebt die met elkaar verbonden worden. Zoals je kan zien in het voorbeeld (Tabel 1), zit de data in vakjes die daarvoor bestemd zijn. Bovenaan in de rijen is gedefinieerd welke data in die kolom hoort, zoals in het voorbeeld de vakjes in het blauw. Het is in dit geval niet mogelijk om nummer in te voeren bij het beroep, omdat het vakje is gereserveerd voor een woord. (Wiegerink, 2013)

|  |  |  |  |
| --- | --- | --- | --- |
| Naam | Achternaam | Leeftijd | Beroep |
| Harry | Van Duyk | 42 | Automonteur |
| Gerrie | Janssen | 30 | Boer |
| Berend | van de Broek | 22 | Student |

Tabel 1 Relationele database tabel

## 2.3 NoSQL

NoSQL is een verzamelnaam voor een groot aantal DBMS-en die op aanmerkelijke wijze verschillen van relationele databasemanagementsystemen. NoSQL-databases maken, in tegenstelling tot relationele databases, niet gebruik van schema’s en relaties. NoSQL kan worden onderverdeeld in meerdere types databases, elke type heeft zijn eigen manier van het opslaan van data. NoSQL is ontworpen om te kunnen werken met de moderne servers en applicaties van deze tijd.

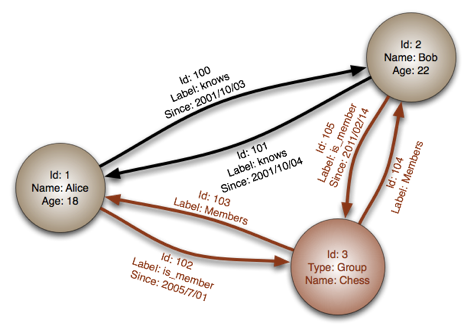


Figuur 2.1 document-oriented voorbeeld (Couchbase, Z.D.)

### 2.3.1 NoSQL Database Types

**Document-oriented:**

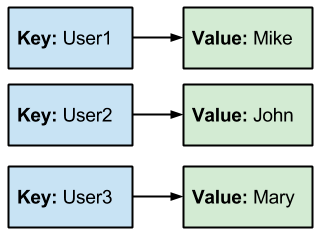
Document-oriented databases maken gebruik van een ‘key’ en een ‘value’. De key representeert een naam voor de data die daar aan vast hangt, dit is te vergelijken met een kolomnaam bij een relationele databases. De ‘value’ bestaat uit een waarde die is ingevoerd bij de ‘key’. Deze kunnen meerdere soorten datatypes zijn, zoals integer, string, array etc. Tevens maakt hij gebruik van ‘id’s’, deze bevatten een willekeurige string die een document representeert. Deze String kan ook worden aangepast naar een ander datatype. Document-oriented databases hebben de mogelijkheid om documenten met elkaar te linken door middel van deze ‘id’s’, of te embedden door een document in een ander document te nestelen (MongoDB, Z.D.). In figuur 2.1 is een voorbeeld te zien van een document-oriented database.



Figuur 2.2 Graph-store voorbeeld (Straughan, 2013)

**Graph-Stores:**

Een graph-store database maakt gebruik van grafiekstructuren om data in op te slaan. Graph databases gebruiken Nodes en Edges. Nodes kunnen worden gezien als een entiteit, zoals mensen, accounts of objecten. Edges zijn eigenschappen of relaties tussen de nodes. Een edge is gelinkt tussen twee verschillende nodes (Angles, 2008). Zie figuur 2.2 voor een voorbeeld van een graph-store database.



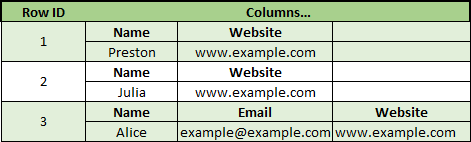
Figuur 2.3 key-value voorbeeld (Shilov, 2014)

**Key-value:**

Een key-value database is een van de simpelste manieren om data op te slaan. Zoals de naam al verklapt heeft, maakt het gebruikt van een ‘key’ en een ‘value’ (figuur 2.3). De manier van data opslaan is te vergelijken met die van de document-oriented database, namelijk zonder gebruik te maken van het embedden of linken van documenten. (Aerospike, Z.D.)

**Wide-column store:**

Wide-column stores houden zich niet aan de standpunten van NoSQL, omdat wide-column stores wél gebruik maken van schema’s en kolommen(figuur 2.4). Deze twee componenten combineren ze met de eigenschappen van een key-value database. Een schema in een wide-column store is vergelijkbaar met die van een relationele database. Het verschil is echter dat de kolomnamen niet vast staan en daarom zeer flexibel zijn. Je kan wide-column store dus zien als een flexibele versie van een relationele database. (Shinde, 2013)



Figuur 2.4 wide-column store voorbeeld (Ranjan, 2015)

## 2.5 NoSQL vs SQL

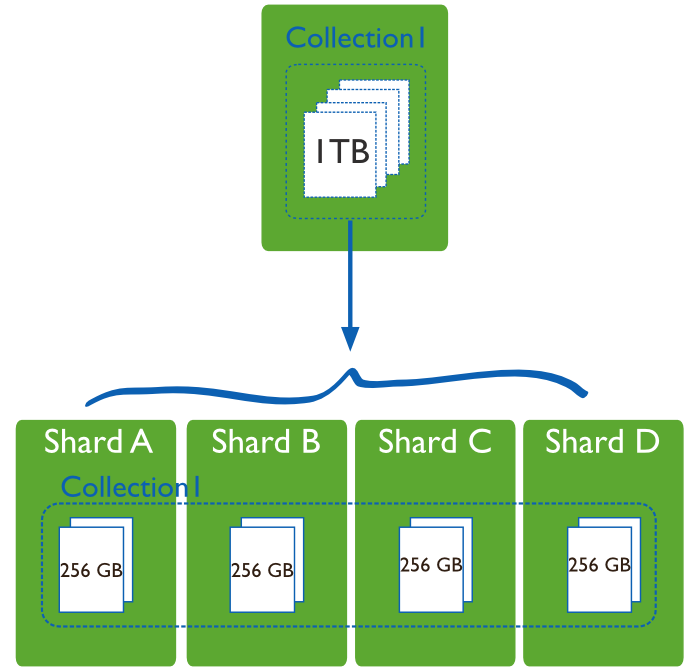
Deze paragraaf vergelijkt SQL met NoSQL door middel van figuur 2.5.

|  |  |  |
| --- | --- | --- |
|  | SQL Databases | NOSQL Databases |
| ****Types**** | Één type database met weinig variatie. | Veel verschillende type databases met veel variatie. |
| ****Ontwikkelings-geschiedenis**** | Ontwikkeld in 1970 om de eerste golf van applicaties voor gegevensopslag te verwerken. | Ontwikkeld in de late jaren 2000 om de beperkingen van SQL-databases te overkomen. |
| ****Data Storage Model**** | Afzonderlijke records worden opgeslagen als rijen in tabellen, waarbij elke kolom een specifiek deel van de gegevens over dat record opslaat, net als een spreadsheet. Gerelateerde gegevens worden opgeslagen in aparte tabellen, en vervolgens samengevoegd wanneer complexere queries worden uitgevoerd. | Varieert van database type.  Bijvoorbeeld, key-value databases slaan functies vergelijkbaar op als SQL-databases, maar hebben slechts twee kolommen ('key' en 'value'). Document databases gooien het ‘table-row’-model geheel aan de kant. Het opslaan van alle relevante gegevens wordt gedaan in een enkel 'document', waarbij gebruik wordt gemaakt van JSON, XML, of een ander formaat. |
| ****Schema’s**** | De structuur en datatypes zijn van tevoren besproken. Om een nieuw datatype toe te voegen moet de database in zijn geheel worden aangepast. Tijdens dit proces moet de database offline zijn. | Over het algemeen dynamisch, er zijn enkele afgesproken regels bij het invoegen van schema’s. Schema’s kunnen worden toegevoegd zonder de database offline te halen. Voor sommige databasetypes is het lastiger om nieuwe data dynamisch toe te voegen. |
| ****Schaling**** | Verticaal, dit betekent dat een database op één locatie moet bestaan en bij uitbreiding moet die server worden uitgebreid op dezelfde locatie. Het is wel mogelijk om een SQL-database te verspreiden over meerdere locaties, maar daarbij moet de databasestructuur volledig worden omgegooid. | Horizontaal, dit betekent dat een database uitbreidbaar is over meerdere servers op meerdere locaties, dit wordt ‘sharding’ genoemd. De databases versturen automatisch de benodigde data naar elkaar. |
| ****Ontwikkelings-model**** | Een mix van open-source en closed-source. | Een mix van open-source en closed-source. |
| ****Ondersteunt transacties**** | Ja, hij update de table helemaal of totaal niet. | In bepaalde omstandigheden en op bepaalde niveaus (bijv. document niveau versus database-niveau). |
| ****Data-aanpassingen**** | Met taal-specifieke methodes, bijvoorbeeld SQL(Select, Insert, en Update statements). | Door object-oriented API’s. |
| ****Consistentie**** | Kan geconfigureerd worden om ACID (Atomic, Consistent, Isolated, Durable) te zijn. | Hangt van het product af, sommige databases ondersteunen ACID. |

Figuur 2.5 SQL vs NoSQL (MongoDB, Z.D.)

## 2.6 Sharding

Wanneer een databaseserver uitgebreid moet worden, kan dit op de plek gedaan worden waar de database zich bevindt, hierdoor wordt de database zelf groter. Het groter worden van een database op een server wordt gezien als een probleem. Er is ook een andere manier om dit probleem op te lossen, en dat is ‘sharding’. Met sharding wordt de database uitgebreid door één of meerdere servers op andere locaties te plaatsen. Deze servers denken dat ze gezamenlijk één grote server zijn. Dit zorgt niet alleen voor een grotere server, maar ook dat de gebruiker de server sneller kan bereiken doordat een deel van de servers op een locatie dichterbij geplaatst is.



Figuur 2.6 Voorbeeld van sharding (MongoDB, Z.D.)

Wanneer een server een bepaalde taak moet uitvoeren die op het ene moment een stuk zwaarder is dan andere keren, dan kan een ‘gesharde’ server de belasting verminderen door een deel van de taken over te dragen aan andere delen van de server (Roy, 2008). Figuur 2.6 is een voorbeeld van sharding, dit voorbeeld laat zien dat data verspreid wordt over meerdere servers.

## 2.7 conclusie

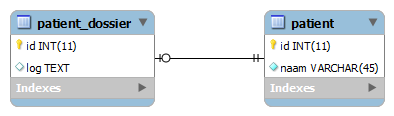
Een schemaless database wordt ook wel NoSQL genoemd en kan worden gebruikt in verschillende soorten databasemanagementsystemen. Een NoSQL-database heeft standaard geen relaties met andere documenten waardoor het zoeken binnen de database sneller gaat dan bij een relationele database. Ook hoeft een NoSQL-document geen rekening te houden met de andere kolommen, dit zorgt ervoor dat het invoegen van data een stuk sneller gaat dan bij een relationele database. Een andere reden is dat er data toegevoegd kan worden per kolom zonder dat er rekening gehouden hoeft te worden met de andere kolommen. Een NoSQL-database kan worden verspreid over meerdere locaties (sharding), dit zorgt voor sneller antwoord van de server.

Een groot nadeel van NoSQL is dat het niet gebruik maakt van een schema. Dit heeft als gevolg dat er nauwelijks structuur in de database is en dat elk veld een andere vorm kan aannemen.

# 3. Wat is een document-oriented database?

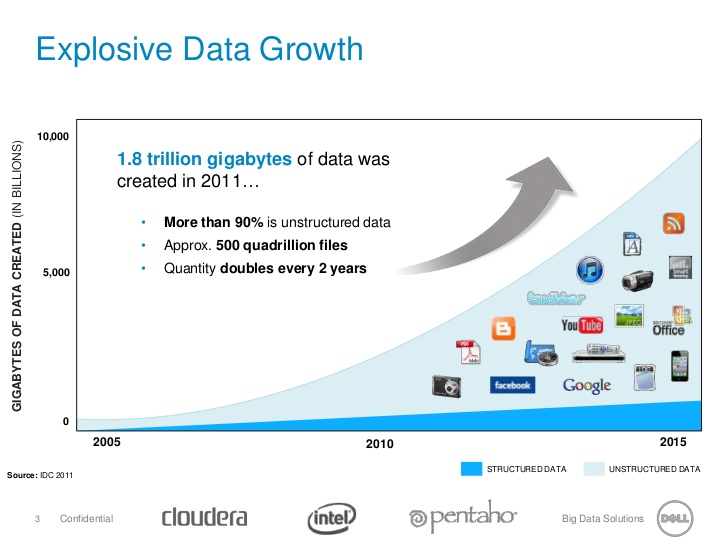
Dit hoofdstuk gaat over document-oriented databasemanagementsystemen. In dit hoofdstuk wordt uitgelegd hoe deze databases in elkaar zitten. In het vorige hoofdstuk is uitgelegd hoe NoSQL databases werken. Eerst wordt verteld waarom document-oriented databases steeds meer gebruikt worden, daarna wordt verteld hoe deze databases werken en als laatste wordt er een conclusie gegeven.

## 3.1. Problemen met relationele databases

Relationele databases bestaan al sinds de jaren zeventig. Dataschema’s toentertijd waren simpel en het was logisch om objecten in een relatie te zetten met andere objecten. Hierdoor konden zo veel mogelijk gegevens gescheiden worden gehouden. In figuur 3.1 is zo’n simpel schema weergegeven. Maar het werkveld voor databeheer is veranderd sinds de ontwikkeling van relationele databases. (MongoDB, Inc., 2016).

Figuur 3.1. De tabel patiënt\_dossier heeft een relatie met de tabel patiënt. (Databasediagrammen lezen)

De ontwikkeling van cloud computing heeft ervoor gezorgd dat het invoegen en het opslaan van data een stuk goedkoper is geworden. Maar hier kan alleen gebruik van worden gemaakt als data verspreid kan worden over meerdere servers zonder onderbreking. In vele complexe databasesystemen is dit lastig omdat queries meerdere tabellen moeten kunnen samenvoegen (joinen) om een antwoord te kunnen geven. Het uitvoeren van verschillende joins is een complex probleem in relationele databases die op verschillende servers staan. (MongoDB, Inc., 2016).



Figuur 3.2: de groei van ongestructureerde data. (Seidman, Fyfe, & Stacey, 2012)

De behoefte om ongestructeerde data op te slaan is de laatste tijd enorm gegroeid. “*Ongestructureerde data is data die niet een gespecificeerd format volgt. Het is data die niet verblijven in een traditionele rij-kolom database. Ze zijn dus het tegenovergestelde van gestructureerde data die opgeslagen worden in velden in een database.*” (Eduvision). In figuur 3.2 is te zien dat in 2011 al 90% van de data ongestructureerd was. Relationele databases zijn zeer efficiënt in het opslaan van gestructureerde data, omdat dit een schema volgt. Het opslaan van ongestructureerde data is een valkuil van relationele databases, omdat zij geen ongestructureerde gegevens kunnen bevatten. (MongoDB, Inc., 2016)

Agile development methodes zorgen ervoor dat databaseschema’s vaak veranderen tijdens een project. “*Agile (project)management is een iteratieve (zich herhalende) methode om wensen en eisen boven water te krijgen en aan die steeds veranderende wensen en eisen te blijven voldoen*.” (Vree, Z.D.) Bijvoorbeeld, als je in een later stadium erachter komt dat je ook comments moet kunnen toevoegen op je webpagina, dan kan je daardoor veel tijd sleurende alter statements uitvoeren op je relationele database omdat je hele database aangepast moet worden. Bij het gebruik van document-oriented databases bestaat dit probleem niet. In de volgende paragraaf wordt uitgelegd waarom dit probleem niet bestaat. (MongoDB, Inc., 2016)

Relationele databases hebben natuurlijk ook voordelen. Het eerste voordeel is dat relationele databases bekende technologiën zijn. Er wordt al meerdere decennia gebruik van gemaakt en elke programmeur is er ooit mee in aanraking gekomen. De software is vertrouwd omdat het een volwassen technologie is. Een tweede voordeel van relationele databases is dat het gebruikt maakt van SQL. De syntax hiervan is simpel en betrouwbaar, omdat het een lange tijd bestaat en er veel documentatie is. (Soltesz, Z.D.) Het derde voordeel is dat de normalisatie van data leidt tot nette datamanagment omdat je duplicaties van data voorkomt en hierdoor het geheugen van de harde schijf beter benut. Het laatste voordeel is dat relationele databases zich houden aan de ACID(Atomic, Concistence, Isolated, Durable) richtlijnen. Omdat er gebruik wordt gemaakt van ACID blijven alle gegevens consistent en hierdoor is de database betrouwbaar. (Basu, 2015)

## 3.2. Document-oriented databases

Een reactie op deze problemen, in de eerder beschreven paragrafen, was de ontwikkeling van NoSQL-databasemanagementsystemen. Een subklasse hiervan is de document-oriented database. In tegenstelling tot relationele databases slaan document-oriented databases data op in de vorm van een serie van autonome documenten. Er zijn geen tabellen, kolommen, rijen, relaties of strikte schema’s. De voordelen hiervan zijn:

1. Complexe objecten kunnen opgeslagen worden in hun huidige toestand als een document, dit hoeft niet vertaald te worden naar schema’s.
2. Document-oriented databases leveren betere prestaties op de servers met minder externe effecten, dit wordt bereikt door middel van sharding.
3. Data hoeft niet geherstructureerd te worden met nieuwe attributen, de aanpassing kan gelijk in het document worden gezet. (Sandeep Chanda., 2013)

Bij een relationele database moet eerst een schema gedefinieerd worden voordat de gegevens in de database geplaatst kunnen worden. Het schema is een blauwprint voor hoe gegevens in de database staan en met elkaar verbonden zijn. Dit wordt veelal geschreven in een formele taal geleverd door het databasemanagementsysteem, bijvoorbeeld SQL in MySQL. (Lerman, 2011)

In tegenstelling tot een relationele database bevat een document-oriented database documenten. Documenten bevatten de gegevens, maar hebben hiervoor geen schema nodig. De programmeur kan zelf bepalen welke gegevens hierin komen te staan. Hierdoor kunnen documenten in dezelfde collectie met elkaar verschillen, (zie figuur 3.3). In plaats van kolommen met namen en data descripties, staat de data zelf in het document beschreven. Figuur 3.4 geeft een relationele databasetabel weer waarin verschillende soorten dranken staan. In figuur 3.3 staan twee documenten die dezelfde gegevens bevatten, maar het document met ‘id’ 2 verschilt van het document met ‘id’ 1. (Couchbase, Z.D.)

{

“id”: 1,

“naam”: “Fanta”,

“bedrijf”: “Coca Cola”,

“aantal”: 809

},

{

“id”: 2,

“naam”: “Cola”,

“bedrijf”: “Coca Cola”,

“aantal”: 1244,

“score”: 10

}

Figuur 3.3

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Naam | Bedrijf | Aantal |
| 1 | Fanta | Coca Cola | 809 |
| 2 | Cola | Coca Cola | 1244 |
| 3 | Pepsi | Pepsi | 642 |
| 4 | Heineken | Heineken | 123 |

Figuur 3.4

Elke database verschilt qua implementatie over de details van documenten. Maar de verschillende databases hebben wel allemaal hetzelfde concept. De databases gaan er allemaal van uit dat documenten data coderen in een standaardformaat. Bijvoorbeeld XML, JSON en BSON. (Couchbase, Z.D.)

In document-oriented databases moeten alle documenten een unieke key bevatten. Met de key kunnen documenten gemakkelijk geïdentificeerd worden. De key is veelal een string, een URI of een path. De key gebruik je om documenten uit de database op te vragen. De database indexeert de documenten op basis van de keys, hierdoor kunnen documenten sneller opgevraagd worden. Een ander punt van een document-oriented database is dat ze niet alleen de keys nodig hebben om gegevens op te halen. Met de meegeleverde API of querytaal kunnen verschillende gegevens opgezocht worden. Bijvoorbeeld, als er in figuur 3.3 alleen op naam wordt gezocht, dan wordt er uit alle documenten de naam gepakt, zonder dat er extra gegevens gespecifieerd moeten worden. (Lerman, 2011)

## 3.3. Conclusie

Het antwoord op de vraag, ‘wat zijn document-oriented databases?’, is dat het databases zijn die gebruik maken van documenten die flexibel zijn. Een document heeft geen vaste vorm en de data die erin komt te staan hoeft zich niet aan een schema te houden. Ten eerste pakt het aan dat ongestructureerde data makkelijk verwerkt kan worden omdat het geen schema heeft. Ten tweede kan er beter gebruik gemaakt worden van cloud computing als document-oriented databases gebruikt worden. Als laatste pakken document-oriented databases het probleem aan dat eisen (met betrekking op data-storage) die tijdens een softwareontwikkelingsfase ontstaan gelijk kunnen worden verwerkt, zonder dat de structuur van de database aangepast hoeft te worden, dit bespaart een hoop tijd omdat er geen complexe alter-statements uitgevoerd hoeven te worden.

# 4. Welke criteria zijn er te stellen aan document-oriented databasemanagementsystemen en welke gevonden databases voldoen aan deze criteria?

## 4.1 Inleiding

In dit hoofdstuk wordt een schema weergegeven dat duidelijkheid geeft over welk databasemanagementsysteem naast MongoDB het beste geïmplementeerd zou kunnen worden in de applicatie. Een aantal document-oriented databasemanagementsystemen worden met elkaar vergeleken en er wordt gekeken of deze voldoen aan de aangegeven criteria. De onderstaande criteria zijn opgesteld door het projectteam in combinatie met de opdrachtgever, voor de opdrachtgever zijn ondergenoemde punten de belangrijkste criteria voor het kiezen uit een document-oriented databasemanagementsysteem.

## 4.2 Criteria

**Query resultaat in JSON/BSON:**

Het databasemanagementsysteem accepteert een universeel JSON/BSON-object. Hierdoor is het makkelijk om later meerdere DBMS-en toe te voegen aan de applicatie zonder de applicatie zelf aan te hoeven passen. De database moet dus als resultaat, na een query, een JSON- of BSON-object teruggeven aan de applicatie.

**Ondersteunt NodeJS:**

Omdat de server is opgezet met NodeJS, moet het DBMS dit natuurlijk ondersteunen. Zonder deze ondersteuning zal het DBMS niet werken.

**Maakt gebruik van http-REST protocol:**

De requests naar de server zullen RESTful geprogrammeerd zijn. Hierdoor moeten de queries ook op deze manier geschreven kunnen worden. De query zal via een RESTful http-request verzonden worden.

**Voldoende documentatie:**

Het DBMS moet voldoende documentatie bevatten zodat de API in een kort tijdsbestek begrepen en ingezet kan worden, om zo de deadline te kunnen halen.

Alleen databasemanagementsystemen die aan alle criteria voldoen zijn voor de opdracht een mogelijke optie. Als er meerdere databases zijn die aan alle criteria voldoen wordt er gekeken naar de populariteit van het DMBS en naar de beschikbare documentatie. Uiteindelijk zal na het uitzoeken van dit criteria een winnaar naar boven komen.

## 4.3 DBMS-schema

In het vorige hoofdstuk zijn er een aantal systemen gevonden die voor ons van toepassing zijn. Deze databasemanagementsystemen worden hieronder met elkaar vergeleken waarna een conclusie wordt getrokken. Hieronder zijn de gevonden databasemanagementsystemen opgesomd en worden de criteria toegepast.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DBMS | JSON/BSON | NodeJS | http REST | Documentatie | Open source |
| Elasticsearch | ✓ | ✓ | ✓ | ✓ | ✓ |
| Cloud datastore | ✓ | ✓ | ✓ | ✓ | 🗶 |
| Ms Azure documentDB | ✓ | ✓ | ✓ | ✓ | 🗶 |
| RethinkDB | ✓ | ✓ | 🗶 | ✓ | ✓ |
| CouchDB | ✓ | 🗶 | ✓ | 🗶 | ✓ |
| ToroDB | ✓ | 🗶 | 🗶 | 🗶 | ✓ |
| SequoiaDB | ✓ | 🗶 | 🗶 | 🗶 | ✓ |
| NosDB | ✓ | 🗶 | 🗶 | ✓ | ✓ |
| RavenDB | 🗶 | 🗶 | 🗶 | 🗶 | ✓ |
| Marklogic | ✓ | 🗶 | ✓ | 🗶 | ✓ |
| Clusterpoint | ✓ | ✓ | ✓ | ✓ | 🗶 |
| NeDB | ✓ | ✓ | ✓ | ✓ | ✓ |
| NodeChef | ✓ | ✓ | ✓ | 🗶 | 🗶 |
| JasDB | 🗶 | 🗶 | ✓ | ✓ | ✓ |
| DjonDB | ✓ | ✓ | ✓ | ✓ | ✓ |
| Couchbase | ✓ | ✓ | ✓ | 🗶 | ✓ |

Figuur 4.1 Onderzochte DBMS's

## 4.4 Conclusie

Door middel van figuur 4.1 is aangetoond dat Elasticsearch, NeDB en DjonDB voldoen aan alle criteria. Om een keuze te maken tussen de drie databasemanagementsystemen moet er gekeken worden naar referenties, oftewel hoe populair zijn de DBMS-en.

Op de site van Elasticsearch is aangetoond dat Elasticsearch wordt gebruikt door onder andere LinkedIn, TaskRabbit, Pandora, WordPress en Seek. (Elastic, Z.D.) Daarnaast kunnen NeDB en DjonDB geen referenties aantonen op hun website.

# 5. Conclusie

De hoofdvraag luidt: ‘welk document-oriented databasemanagementsysteem past het beste naast MongoDB in de applicatie?’. Aan de hand van een aantal deelvragen wordt hier een antwoord op deze hoofvraag gegeven.

Een schemaless database wordt ook wel NoSQL genoemd en kan worden gebruikt in verschillende soorten databasemanagementsystemen. Een NoSQL-database heeft standaard geen relaties met andere documenten waardoor het zoeken binnen de database sneller gaat dan bij een relationele database.

Document-oriented databases zijn databases die gebruik maken van documenten die flexibel zijn. Een document heeft geen vaste vorm en de data die erin komt te staan hoeft zich niet aan een schema te houden.

Samen met de opdrachtgever zijn de volgende criteria opgesteld voor het vergelijken van databasemanagementsystemen: query-resultaat in JSON/BSON, ondersteunt NodeJS, maakt gebruik van http-REST protocol, documentatie en of het open-source is. Omdat er meerdere databasemanagementsystemen uitkwamen die aan alle criteria voldeden is er een extra criterium toegevoegd, namelijk de populariteit van het systeem.

Hieruit is gebleken dat van de drie systemen die aan alle criteria voldeden (Elesticsearch, NeDB en DjonDB) Elasticsearch veruit het populairste is onder ontwikkelaars en deze ook de meeste downloads heeft. Door bovengenoemde criteria zouden wij Elasticsearch gebruiken om in de applicatie te implementeren.

# 6. Literatuurlijst

Aerospike. (Z.D.). *Aerospike*. Opgehaald van What is a key-value store?: http://www.aerospike.com/what-is-a-key-value-store/

Angles, R. (2008). Opgehaald van Survey of Graph Database Models: http://www.cse.iitk.ac.in/users/smitr/PhD%20Resources/Survey%20of%20Graph%20Databases%20Models.pdf

Basu, S. (2015, november 5). *Red Pill or Blue Pill? Choosing Between SQL & NoSQL.* Opgehaald van developer.telerik: http://developer.telerik.com/featured/decide-neo-blue-pill-red-pill/

Couchbase. (Z.D.). *Comparing document-oriented and relational data.* Opgehaald van docs Couchbase: http://docs.couchbase.com/developer/dev-guide-3.0/compare-docs-vs-relational.html

*Databasediagrammen lezen.* (sd). Opgehaald van sites google: https://sites.google.com/a/avans.nl/sql-leren/databasediagrammen-lezen/kardinaliteit-in-een-relatie

Eduvision. (sd). *Ongestructureerde data.* Opgehaald van Eduvision: https://www.eduvision.nl/big-data/kennisbank/ongestructureerde-data

Elastic. (Z.D.). Opgehaald van https://www.elastic.co/use-cases

Lerman, J. (2011, November). *Data Points - What the Heck Are Document Databases?* Opgehaald van msdn Microsoft: https://msdn.microsoft.com/en-us/magazine/hh547103.aspx

MongoDB. (Z.D.). *MongoDB*. Opgehaald van Sharding Introduction: https://docs.mongodb.com/v3.0/core/sharding-introduction/

MongoDB. (Z.D.). *Types Of NoSQL Database Management Systems*. Opgehaald van MongoDB: https://www.mongodb.com/scale/types-of-nosql-database-management-systems

MongoDB, Inc. (2016). *Document Databases.* Opgehaald van mongodb: https://www.mongodb.com/document-databases

Ranjan, N. (2015, January 03). *Devinline*. Opgehaald van NoSQL database - Different types of NoSQL database: http://www.devinline.com/2015/01/nosql-intorduction.html

Roy, R. (2008, July 28). *Roy's Technology Dairy*. Opgehaald van http://technoroy.blogspot.nl/2008/07/shard-database-design.html

Sandeep Chanda., D. F. (2013). Beginning ASP.NET 4.5 Databases. Apress.

Seidman, J., Fyfe, I., & Stacey, J. (2012, September 27). *Webinar | Using Hadoop Analytics to Gain a Big Data Advantage.* Opgehaald van Slideshare: http://www.slideshare.net/cloudera/webinar-using-hadoop-analytics-to-gain-a-big-data-advantage

Shilov, M. (2014, June 16). *Where is NoSQL practically used?* Opgehaald van http://scraping.pro/where-nosql-practically-used/

Shinde, S. (2013, January 13). *What is wide column stores*. Opgehaald van SQL server business intellegence and big data: https://bi-bigdata.com/2013/01/13/what-is-wide-column-stores/

Soltesz, L. D. (Z.D.). *The Advantages of a Relational Database Management System.* Opgehaald van techwalla: https://www.techwalla.com/articles/the-advantages-of-a-relational-database-management-system

Straughan, J. (2013, July 1). *Introduction to Graph Databases using Neo4j*. Opgehaald van http://jdstraughan.com/2013/07/01/neo4j-or-how-I-learned-to-stop-worrying-and-love-the-graph/

Vree, J. d. (Z.D.). *agile*. Opgehaald van http://www.joostdevree.nl/shtmls/agile.shtml

Wiegerink, L. (2013). *Relationele databases en SQL.* Academic Service.