



**HoGent**

Faculteit Bedrijf en Organisatie

NoSQL: Cassandra

Lorenz Verschingel

Scriptie voorgedragen tot het bekomen van de graad van  
Bachelor in de toegepaste informatica

Promotor:  
Sabine De Vreese  
Co-promotor:  
Jean-Jacques De Clercq

Instelling: HoGent

Academiejaar: 2015-2016

Tweede examenperiode



Faculteit Bedrijf en Organisatie

NoSQL: Cassandra

Lorenz Verschingel

Scriptie voorgedragen tot het bekomen van de graad van  
Bachelor in de toegepaste informatica

Promotor:  
Sabine De Vreese  
Co-promotor:  
Jean-Jacques De Clercq

Instelling: HoGent

Academiejaar: 2015-2016

Tweede examenperiode

## **Samenvatting**

# Voorwoord

# Inhoudsopgave

<b>1</b>	<b>Inleiding</b>	<b>4</b>
1.1	NoSQL . . . . .	4
1.2	Cassandra . . . . .	5
1.3	Probleemstelling en Onderzoeksvragen . . . . .	5
<b>2</b>	<b>Methodologie</b>	<b>6</b>
<b>3</b>	<b>Architectuur</b>	<b>8</b>
3.1	Partitionering . . . . .	8
3.2	Replicatie . . . . .	8
3.3	Snitches . . . . .	8
3.4	Seed node . . . . .	8
3.5	Gossip en foutdetectie . . . . .	8
3.6	Herstelmechanisme . . . . .	8
<b>4</b>	<b>Data opslag in Cassandra</b>	<b>9</b>
4.1	Cluster . . . . .	9
4.2	Keyspace . . . . .	9
4.3	Table / Column Family . . . . .	9
4.4	Row . . . . .	9
4.5	Column . . . . .	9
<b>5</b>	<b>Opzetten van de Cassandra cluster</b>	<b>10</b>
5.1	Apache Cassandra . . . . .	10
5.2	DataStax OpsCenter . . . . .	10
5.3	Toevoegen van een node . . . . .	14
<b>6</b>	<b>Datamodellering in Cassandra</b>	<b>16</b>
6.1	Primaire sleutel . . . . .	16
6.2	Partitie kolom . . . . .	16
6.3	Clustering kolom . . . . .	17

6.4	De WHERE clause . . . . .	17
6.4.1	Restricties opgelegd door de partitie kolommen . . . . .	17
6.4.2	Restricties opgelegd door de clustering kolommen . . . . .	18
6.5	Doelen bij datamodellering in Cassandra . . . . .	19
6.6	Datamodel opstellen voor Cassandra . . . . .	19
<b>7</b>	<b>Data importeren in Cassandra</b>	<b>22</b>
7.1	Importeren via cqlsh . . . . .	22
7.2	Importeren via sstableloader . . . . .	23
7.3	Importeren via cassandra-loader . . . . .	23
7.4	Een vergelijking van de drie methodes . . . . .	23
<b>8</b>	<b>Gedrag bij uitvallen van een node</b>	<b>24</b>
<b>9</b>	<b>Back-ups in Cassandra</b>	<b>25</b>
<b>10</b>	<b>Conclusie</b>	<b>26</b>

# Hoofdstuk 1

## Inleiding

### 1.1 NoSQL

Sinds de term voor het eerst in 1998 gebruikt werd door Carlo Strozzi, is er veel te doen rond NoSQL databanken. Toen hij deze term de wereld instuurde bedoelde Strozzi dat de databank die op dat moment besproken werd geen SQL interface aanbood. In 2009 werd de term NoSQL opnieuw gebruikt oor Johan Oskarsson als hashtag voor een meetup waar de problemen met relationele databanken het de huidige manier van programmeren besproken gingen worden. Nu wordt NoSQL begrepen als not only SQL, wat erop wijst dat er meerdere manieren zijn om data op te slaan. (Fowler, 2013)

NoSQL databanken vinden hun oorsprong in de verschillen tussen een relationeel model en het object georiënteerd programmeren, het zogenoemde "impedance mismatch", het feit dat relationele databanken vaak niet goed werken op een cluster of met grote hoeveelheden realtime data. . . Hierdoor maken NoSQL databanken vaak geen gebruik van een relationeel model, zijn gemaakt om op clusters te werken, zijn ze schema-less. . . Kortom databanken onder de noemer NoSQL zijn aangepast om de problemen die zich nu voordoen binnen de Informatica aan te pakken. (Fowler, 2012)

Sadalage (2014) zegt dat binnen de NoSQL databanken vier grote types naar voor zijn gekomen, namelijk key-value stores (Riak, Redis. . .), document stores (MongoDB, CouchDB. . .), Column Family Stores (Cassandra, HBase. . .) en Graph databases (Neo4J, Infinite Graph. . .). Elk van deze types heeft zijn eigen specifieke use cases. Zelf binnen de verschillende types zijn er nog verschillende use cases.



## **1.2 Cassandra**

In het vervolg van deze bachelorproef wordt de focus gelegd op de NoSQL databank Cassandra. Cassandra is een NoSQL database die focust op schaalbaarheid en beschikbaarheid, zonder aan performantie in te boeten. Cassandra is op dit moment een productiewaardige database. Enkele bekende gebruikers van Cassandra zijn Facebook, Apple, Netflix, GitHub, Instagram, GoDaddy. . .

Cassandra is een project dat zijn oorsprong vond bij Facebook. In 2008 was Cassandra de oplossing, bedacht door Lakshman en Malik, voor het Inbox Search probleem van Facebook. De moeilijkheid bij hier was dat er een systeem nodig was die een hoge throughput nodig heeft, miljarden write operaties per dag moet aankunnen en kan mee schalen met het aantal gebruiker (Lakshman and Malik, 2010). Om tot deze oplossing te komen baseerden Lakshman en Malik zich op twee andere projecten.

Een eerste project waarop gebaseerd is, is Google BigTable. Google BigTable had namelijk al een oplossing voor een eerste probleem dat Lakman en Malik moesten oplossen: een schaalbare database zonder realtime antwoorden op te offeren (Chang et al., 2008). Lakhmeshs vorige project, Amazon Dynamo, was de tweede inspiratiebron voor Cassandra. Dynamo loste eerder al voor een hoge betrouwbaarheid bij een schaalbare database (DeCandia et al., 2007).

Hoewel er nu nog steeds voldaan wordt aan de voorwaarden om een oplossing te zijn voor het oorspronkelijk probleem van Facebook, is Cassandra verder blijven groeien sinds 2008. Zo kan Cassandra nu bijvoorbeeld ook overweg met gestructureerde, semi-gestructureerde en ongestructureerde data (Kan, 2014).

## **1.3 Probleemstelling en Onderzoeksvragen**

Cassandra belooft een groot aantal zaken en binnen deze bachelorproef is het de bedoeling om na te gaan. Eerst en vooral wil de schaalbaarheid van Cassandra gecontroleerd worden. Is het werkelijk eenvoudig om de database op verschillende eenvoudige servers te installeren? Een tweede punt dat nagegaan wordt is de betrouwbaarheid van Cassandra. Is er werkelijk geen single point of failure en in hoever zijn backups nodig binnen deze omgeving.

# Hoofdstuk 2

## Methodologie

Om een antwoord te bieden op alle onderzoeksvragen werd deze bachelorproef opgesplitst in twee luiken. Het eerste luik omvat het eerder theoretisch gedeelte, waar een literatuurstudie aan te pas kwam. Het tweede luik omvat het praktisch gedeelte die nodig was om op een aantal vragen een antwoord te krijgen.

In het theoretisch gedeelte komt zoals eerder vermeld de literatuurstudie aan bod. Hierin wordt dieper in gegaan op de architectuur, hoe data opgeslagen wordt binnen Cassandra, wat er juist bedoeld wordt met het meervoudig opslaan van data, hoe belangrijk back-ups zijn binnen dit systeem en voor welke problemen Cassandra een oplossing biedt. . .

Om dit alles te kunnen nagaan werd het praktisch gedeelte opgezet. Eerst moest er een Cassandra cluster opgezet worden. Dit gebeurde aan de hand van Vagrant virtuele machines. Er werd geopteerd voor Vagrant omdat dit een snelle manier is om verschillende identieke virtuele machines op te zetten. Ook kon aan de hand van één enkel script de volledige omgeving gecontroleerd worden. Voor de installatie van Cassandra werd eerst gekozen om met de apache versie te werken. Het idee hiervan was om met de meest recente versie te werken. Om praktische reden werd later verkozen om via het OpsCenter Community Edition van Datastax te werken. Deze tool maakte het mogelijk om via een webinterface de databank Cassandra te beheren en te monitoren. Door gebruik te maken van deze opzet konden snel nodes toegevoegd of verwijderd worden binnen de cluster, via deze opzet kon de schaalbaarheid makkelijk getest worden.

Toen deze cluster opgezet was, werd de data die voorzien werd door de Universiteit van Gent ingeladen in deze virtuele cluster. Voor deze data ingeladen kon worden moest eerst stilgestaan worden bij het datamodel van deze data. Deze data werd dus ook gebruikt om uit te leggen hoe je het best een datamodel opstelt binnen Cassandra.

Hier werd eveneens kort stil gestaan bij het verschil tussen datamodellering binnen een relationele databank en Cassandra.

In een laatste deel moest ook nog de betrouwbaarheid van Cassandra getest worden. Hier werd doelbewust een van de virtuele machines, een van de nodes van de databank, uitgeschakeld om te zien hoe Cassandra hierop reageert. Doordat dit in een virtuele omgeving gebeurde is er geen risico op verlies van kritieke data.

# **Hoofdstuk 3**

## **Architectuur**

### **3.1 Partitionering**

### **3.2 Replicatie**

### **3.3 Snitches**

### **3.4 Seed node**

### **3.5 Gossip en foutdetectie**

### **3.6 Herstelmechanisme**

# **Hoofdstuk 4**

## **Data opslag in Cassandra**

### **4.1 Cluster**

### **4.2 Keyspace**

### **4.3 Table / Column Family**

### **4.4 Row**

### **4.5 Column**

# Hoofdstuk 5

## Opzetten van de Cassandra cluster

### 5.1 Apache Cassandra

Om de cluster op te zetten werd geopteerd om gebruik te maken van virtuele machines, die geconfigureerd werden met Vagrant.

In een eerste poging om een werkende Cassandra cluster te bekomen werd op elke Vagrant machine Cassandra 3.3, op het moment van dit onderzoek de meest recente versie, geïnstalleerd. Nadat dit gebeurd was, dienden nog enkele stappen voltooid te worden om een werkende cluster te bekomen (DataStax, 2016). Deze configuratie gaf echter veel problemen. Cassandra werd na enkele bewerkingen telkens onbruikbaar en gaf de volgende foutmelding weer: "could not access pidfile for Cassandra". Een eerste oplossing voor dit probleem was om ervoor te zorgen dat de user 'cassandra' toegang had tot de pidfile, wat namelijk niet het geval was doordat de installatie van Cassandra werd uitgevoerd door de Vagrant setup. Maar ook dit leverde weinig resultaat op. Wel moet opgemerkt worden dat de installatie van Cassandra zelf geen problemen met zich meebracht. Voor de aanpassing van de configuratie file van Cassandra werkte deze perfect op iedere node.

### 5.2 DataStax OpsCenter

Uiteindelijk werd er geopteerd om gebruik te maken van OpsCenter omdat dit een gemakkelijke manier is om snel een Cassandra cluster te bekomen en omdat er ook goede mogelijkheden tot observeren van de database aanwezig zijn. Er werd voor

OpsCenter community edition 5.2.4 gekozen. Bij deze versie diende Cassandra 2.1.11 gebruikt te worden (Cantoni, 2016).

De setup bestaat uit één master node waarop het OpsCenter runt en verder uit drie slave nodes waarop de uiteindelijke Cassandra database komt te runnen. Na de NAT router van Oracle Virtual Box werd een privaat netwerk opgezet opdat deze machines met elkaar zouden kunnen communiceren. Hiervoor moest iedere machine een uniek ip-adres krijgen binnen het netwerk en moest de file /etc/hosts op iedere node aangepast worden. De scripts hiervoor kunnen in bijlage A teruggevonden worden.

Eenmaal de virtuele machines correct geconfigureerd waren, kon overgegaan worden tot de eigenlijke installatie van Cassandra. Zoals eerder vermeld werd, werd hiervoor gebruik gemaakt van het OpsCenter. Daarom werd op de master node naar 'localhost:8888' gesurft om de installatie te kunnen starten. Op de pagina werden verschillende opties aangeboden en hier werd voor de optie 'brand new cluster' gekozen.

In het volgende venster wordt er om verschillende zaken gevraagd. Tabel 5.1 en figuur 5.1 geven weer hoe dit venster werd ingevuld.

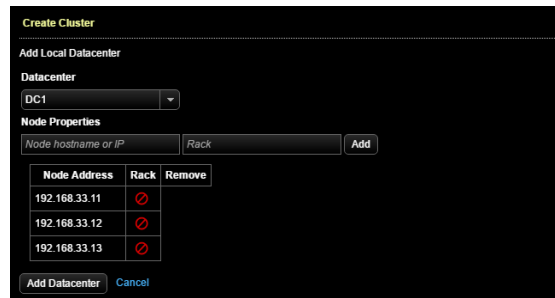
Property Name	Waarde
Cluster Name	BP Cluster
Type	local
Package	datastax community 2.1.11
Endpoint Snitch	GossipingPropertyFileSnitch
Username en password	vagrant/vagrant
Local Node Credentials	cassandra-node-1, cassandra-node-2, cassandra-node-3

Tabel 5.1: Configuratie van de Cassandra Cluster

The screenshot shows the 'Create Cluster' dialog box. The 'Cluster Name' is 'Test Cluster'. 'Provisioning Type' is set to 'Local'. The 'Package' is 'DataStax Community 2.1.11'. The 'Endpoint Snitch' is 'GossipingPropertyFileSnitch'. The 'Username' is 'vagrant' and the 'Password' is 'vagrant'. The 'Local Node Credentials' field contains 'cassandra-node-1', 'cassandra-node-2', and 'cassandra-node-3'. At the bottom, there are buttons for 'Add Datacenter', 'Build Cluster', 'Cancel', and 'View Advanced Options'.

Figuur 5.1: Cassandra: Instellingen deel 1

Hier moest een datacenter toegevoegd worden. Hierbij is er een vrije keuze voor de naam van het datacenter en zijn de node properties het ip-adres van de slave nodes (Figuur: 5.2).



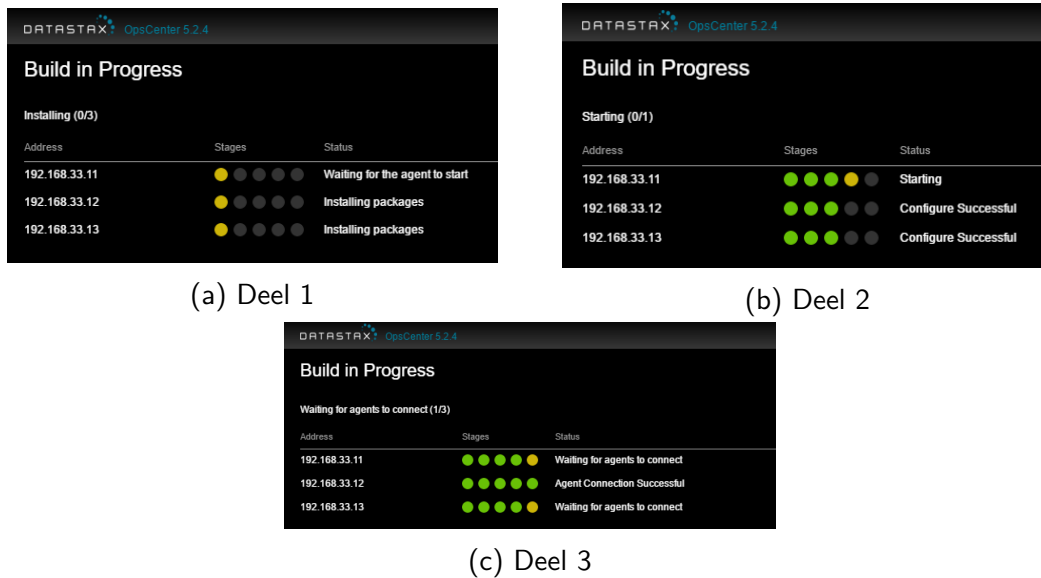
Node Address	Rack	Remove
192.168.33.11	X	
192.168.33.12	X	
192.168.33.13	X	

Figuur 5.2: Cassandra: Instellingen deel 2

Eenmaal de datacenters toegevoegd zijn, kan er verder gegaan worden. Bij het drukken op de knop 'build cluster' wordt verder nog gevraagd om de fingerprints van de nodes te accepteren.

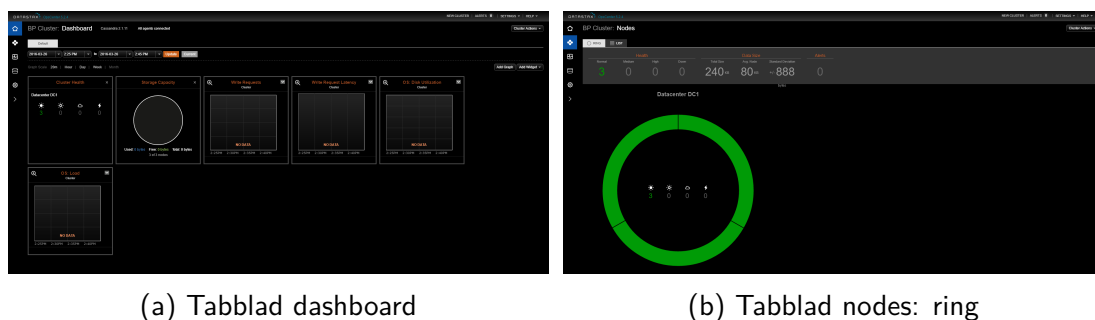
Hierna begint OpsCenter met de installatie van de Cassandra cluster (Figuur: 5.3). Deze installatie nam enige ogenblikken in beslag. De eerste keer kwam er een fout voor omdat er niet genoeg werkgeheugen was toegekend aan de slave nodes. In de setup die hier gebruikt werd, werd het minimum aanvaarde geheugen, nl 2GB, gegeven aan de slave nodes. Samen met Cassandra wordt ook de DataStax agent meegeïnstalleerd opdat het OpsCenter zou kunnen communiceren met iedere node.





Figuur 5.3: Installatie van Cassandra door OpsCenter

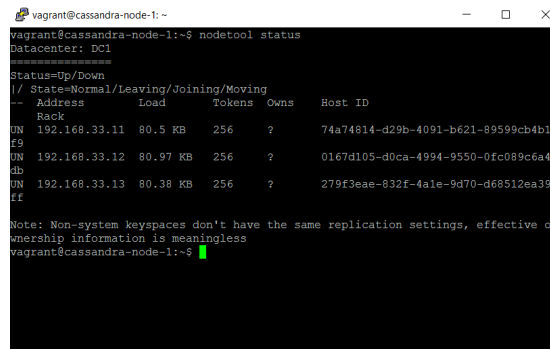
Na de installatie komt men terecht in het Dashboard van OpsCenter (Figuur 5.4a). Hierin worden een aantal zaken weergegeven zoals: de gezondheid van de cluster, het aantal write requests, de write request latency. . . Hier kunnen nog meer grafieken toegevoegd worden door de knop 'add graph' te gebruiken. In het tabblad 'nodes' kan de gezondheid van de cluster bekeken worden net zoals gezien kan worden hoe de data verdeeld zit over de cluster. Deze informatie kan in ringvorm, zoals in figuur 5.4b weergegeven wordt, of in een lijst weergegeven worden.



Figuur 5.4: Rondleiding in OpsCenter

Cassandra biedt zelf ook een tool aan die deze monitoring uitvoert: 'nodetool'. Om te bekijken of de installatie goed gelukt is, kan men via ssh inloggen op een van de nodes van de cluster en hier 'nodetool status' laten lopen (Figuur 5.5). Als men met nodetool

de status opvraagt, krijgt men eveneens alle nodes in de cluster te zien, samen met de hoeveelheid data die ze bevatten en hoeveel procent deze data voorstelt van alle data die op de cluster aanwezig is.



```
vagrant@cassandra-node-1:~$ nodetool status
Datacenter: DC1
=====
Status=Up/Down
// State=Normal/Leaving/Joining/Moving
-- Address      Load      Tokens   Owns    Host ID
--  -
UN 192.168.33.11  80.5 KB   256      ?       74a74814-d29b-4091-b621-89599cb4b1f9
UN 192.168.33.12  80.97 KB  256      ?       0167d105-d0ca-4994-9550-0fc089c6a4db
UN 192.168.33.13  80.38 KB  256      ?       279f3eae-832f-4a1e-9d70-d68512ea39ff

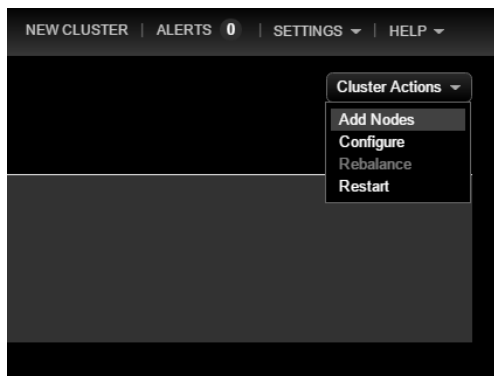
Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless
vagrant@cassandra-node-1:~$
```

Figuur 5.5: nodetool

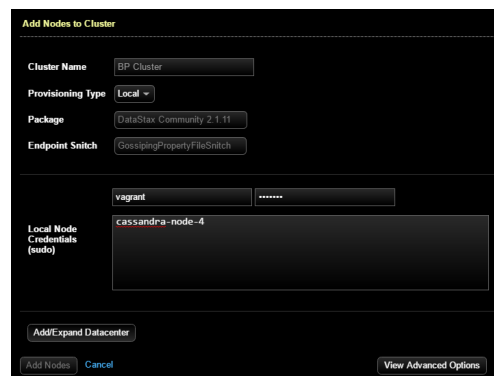
## 5.3 Toevoegen van een node

Het proces voor de toevoeging van een node is quasi gelijk aan het opzetten van de cluster. Hierbij dient men in het OpsCenter bij het menu 'cluster actions' voor de optie 'add node' te kiezen (Figuur 5.6a). Hier wordt een vierde node toegevoegd aan de cluster. Net zoals bij de installatie van de cluster wordt opnieuw het scherm getoond waarin de aanmeldgegevens voor de gegeven node gevraagd worden (Figuur 5.6b). Via de knop 'Add/Expand Datacenter' kan men op dezelfde manier nodes toevoegen aan het bestaande datacenter die tijdens de installatie werd aangemaakt. Bij het bevestigen wordt opnieuw gevraagd om de fingerprint van de node te accepteren en hierna begint de installatie van de software op de node. Waardoor de agent en Cassandra geïnstalleerd en geconfigureerd worden. Na dit alles kan men in het tabblad 'nodes', nu in een lijstvorm, zien dat de nieuwe node is toegevoegd aan de cluster (Figuur: 5.6c).

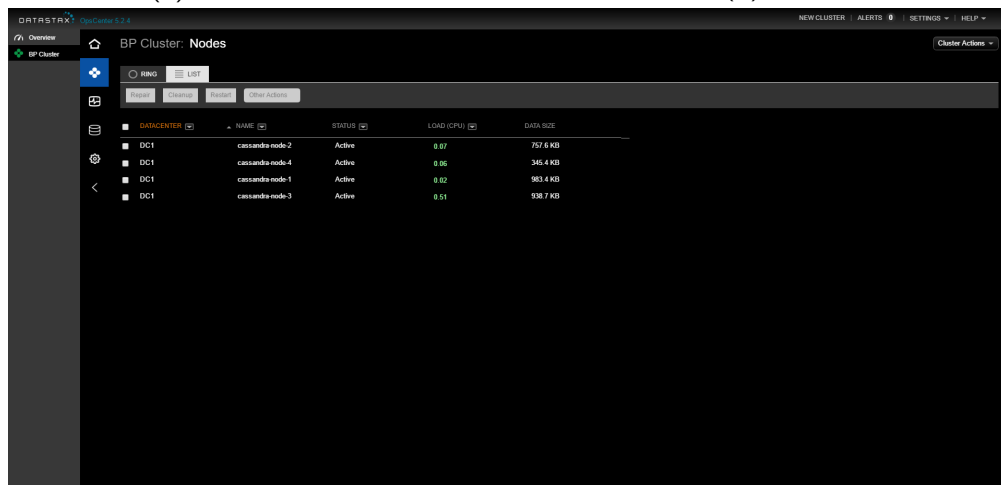
## HOOFDSTUK 5. OPZETTEN VAN DE CASSANDRA CLUSTER



(a) Add node



(b) Deel 2



(c) Tabblad node met 4 nodes

Figuur 5.6: Toevoegen van een node via OpsCenter

# Hoofdstuk 6

## Datamodellering in Cassandra

### 6.1 Primaire sleutel

Binnen de primaire sleutel in Cassandra worden de partitie kolommen (6.2) en clustering kolommen (6.3) vastgelegd voor een tabel. In tegenstelling tot relationele databases wordt de primaire sleutel hier niet gebruikt als unieke sleutel voor de rij, maar om snel te weten te komen waar de data zich binnen de cluster bevindt (Kan, 2014). Als een tabel slechts één kolom heeft als primaire sleutel, dan is deze kolom meteen ook de partitie kolom en zijn er geen clustering kolommen.

Door het feit dat de primaire sleutel in Cassandra niet uniek hoeft te zijn, gedragen het INSERT commando en het UPDATE commando zich op een identieke wijze. Een nadeel hiervan is dat er geen waarschuwing weergegeven wordt als een rij overschreven zou worden. Omdat dit veelal ongewenst gedrag is binnen applicaties komt men al snel terecht bij een samengestelde primaire sleutel, een primaire sleutel uit verschillende kolommen.

### 6.2 Partitie kolom

Het eerste deel van de primaire sleutel bestaat uit de partitie kolommen. Het doel van de partitie kolommen is om de data gebalanceerd over alle nodes te spreiden. Op deze manier kan de fysieke locatie van de data ook snel achterhaalt worden (Kan, 2014).

Bij het kiezen van de partitie kolommen dient er met een aantal zaken rekening gehouden te worden om de data evenwichtig over alle nodes te spreiden, maar er is slechts één fysieke restrictie. Iedere rij kan slechts 2 miljard kolommen bevatten (McFadin,

2013). In de meeste gevallen is dit ruim voldoende. Tijdsreeksen vormen hier een uitzondering op. Deze reeksen kunnen al gauw miljarden entries bevatten. Hier is het dus zeer belangrijk om goede partitie kolommen te kiezen of men komt hier al snel in de problemen.

## **6.3 Clustering kolom**

Het laatste deel van de primaire sleutel bestaat uit de clustering kolommen. Deze bepalen de volgorde van de data op de fysieke media (Strickland, 2014). De clustering kolommen hebben echter geen invloed over op welke node de data wordt opgeslagen. Toch zullen ze een belangrijke invloed hebben op welke query's er uitgevoerd kunnen worden.

## **6.4 De WHERE clause**

Zoals Lerer (2015) aanhaalt is een van de grootste verschillen tussen CQL en SQL de WHERE clause. In SQL kent de WHERE clause geen restricties. Bij CQL is dit anders. Hier worden restricties opgelegd door de partitie en clustering kolommen. Eveneens kan er enkel op de partitie en clustering kolommen gefilterd worden.

### **6.4.1 Restricties opgelegd door de partitie kolommen**

Een eerste restrictie die wordt opgelegd door de partitie kolommen is dat ofwel alle partitie kolommen worden opgenomen in de WHERE clause ofwel geen enkel. Dit is nodig omdat Cassandra anders de hash niet kan berekenen. Deze hash is echter nodig om te weten op welke node de data zich bevind.

Een volgende restrictie die wordt opgelegd door de partitie kolommen is het feit dat enkel de '=' en IN operator rechtstreeks gebruikt kunnen worden. Tot Cassandra 2.2 kon zelfs de IN operator enkel op de laatste gedefinieerde partitie kolom.

De laatste restrictie die wordt opgelegd door de partitie kolommen heeft te maken met de >, >=, < en <= operators. Deze zijn enkel rechtstreeks toepasbaar als de partitioner ingesteld op ByteOrderedPartitioner. Indien dit niet het geval is moet men een omweg maken via de token functie. Op het eerste zicht lijkt het gebruik van de token functie minder efficiënt voor het selecteren van data, maar dit weegt niet op

tegen de nadelen die ByteOrderedPartitioner heeft op gebied van de distributie van de data, zoals reeds eerder werd vermeld.

### 6.4.2 Restricties opgelegd door de clustering kolommen

Voor de restricties op de clustering kolommen uitgelegd kunnen worden, moet eerst uitgelegd worden hoe de data in Cassandra binnen een partitie opgeslagen wordt. Dit zal nu aan de hand van een voorbeeld uitgelegd worden (Lerer, 2015). Neem de onderstaande tabel:

```
CREATE TABLE NumberOfTwitterMessages (  
    userid bigint, date text, hour int, minute int,  
    nrOfTweets int,  
    PRIMARY KEY ((userid, date), hour, minute)  
);
```

Hier wordt de data nu als volgt opgeslagen binnen de partitie:

```
{hour: 20  
  {minute: 4 {nrOfTweets: 6}}  
  {minute: 7 {nrOfTweets: 1}}  
  {minute: 21 {nrOfTweets: 16}}  
  ...  
}
```

De eerste restrictie wordt hier opgelegd door de manier waarop de data opgeslagen zit. Als men een voorwaarde wil vastleggen voor een clustering kolom, dan dienen de clustering kolommen die voor deze komen in de primaire sleutel ook vastgelegd te worden. Dit is nodig omdat Cassandra de data anders niet efficiënt kan terugvinden.

Tot Cassandra 2.2 was de IN operator enkel toegelaten op de laatste clustering kolom. Sinds Cassandra 2.2 is deze restrictie vervallen en kan men nu zelf multi-kolom IN restricties opleggen.

De >, >=, < en <= operators ook enkel toegestaan op de laatste clustering kolom waar een restrictie is aan opgelegd. Toch is hier een oplossing voor aangezien men deze operators kan toepassen over meerdere kolommen. Bij deze multi-column slices is het echter wel belangrijk dat de restrictie van de WHERE clause met dezelfde kolom start.

## **6.5 Doelen bij datamodellering in Cassandra**

Zoals in bovenstaande sectie duidelijk werd, dient er bij Cassandra met heel wat rekening gehouden te worden als men een datamodel creëert. Hobbs (2015) definieerde wat de doelen zijn bij het opstellen van een datamodel in Cassandra. Zoals door Hobbs aangegeven wordt, lijkt de querytaal van Cassandra CQL sterk op SQL, maar kan het gebruik ervan zeer verschillend zijn.

Een eerste doel bij het opstellen van een datamodel in Cassandra is om de data evenwichtig over alle nodes te verspreiden. Dit kan bekomen worden door de partitie kolommen goed te kiezen. Zoals eerder vermeld is, worden de partitie kolommen bepaald door het eerste deel van de primaire sleutel.

Een tweede doel is om zo weinig mogelijk partitie reads te moeten doen. Doordat iedere partitie op een andere node kan staan is dit belangrijk. Als de partities effectief op verschillende nodes staan moet de afzonderlijke commando's naar elke node apart verstuurd worden en dit zorgt voor overhead. Het is zelfs zo dat als de data op dezelfde node staat, de partitie reads nog steeds inefficiënt zijn. Dit komt door de manier waarop Cassandra de rijen opslaat.

Zaken die bij relationele databanken belangrijk zijn zoals het aantal writes en data duplicatie minimaliseren zijn binnen Cassandra geen doelen. In Cassandra zijn write goedkoop omdat Cassandra hiervoor geoptimaliseerd is. Denormalisatie en duplicatie van data zijn ook zeer normaal binnen Cassandra. Dit komt door de architectuur van Cassandra. Hierbij gaat men ervan uit dat schijfruimte goedkoop is in vergelijking met andere resources zoals CPU, geheugen, netwerk ... Ook geeft Cassandra geen JOIN waardoor het ook hoogst inefficiënt en onpraktisch zou zijn om geen duplicate data te hebben.

## **6.6 Datamodel opstellen voor Cassandra**

Hobbs (2015) definieerde niet enkel de doelen van een datamodel in Cassandra, maar haalt ook aan hoe deze bekomen kunnen worden. Voor men begint aan het opstellen van een datamodel moet al nagedacht worden over de query's die ondersteunt moeten worden. Dit staat in schril contrast met relationele databanken waar het datamodel wordt bepaald door de objecten en hun relaties.

Door na te denken over de te ondersteunen query's kan het aantal partitie read al drastisch verminderd worden. Ook dient men rekening te houden met de restricties die de partitie en clustering kolommen opleggen aan de where clause.

Voor iedere query zou er slechts één partitie read uitgevoerd mogen worden. Hiervoor is het belangrijk dat de tabellen geoptimaliseerd zijn voor de reads die uitgevoerd gaan worden.

Een goed voorbeeld wordt ook gegeven door Hobbs (2015) waar alle zaken meteen duidelijk worden. In dit voorbeeld is het de bedoeling om users op het halen volgens hun email of username. De oplossing voor Cassandra zijn de volgende twee tabellen:

```
CREATE TABLE users_by_username (  
    username text PRIMARY KEY,  
    email text ,  
    age int  
)
```

```
CREATE TABLE users_by_email (  
    email text PRIMARY KEY,  
    username text ,  
    age int  
)
```

Bij dit datamodel is de krijg iedere user zijn eigen partitie. Cassandra kan zo de data evenwichtig verdelen over de nodes. Ook dient om een user op te zoeken via een email of username slechts één partitie gelezen te worden.

Stel dat men nu zou proberen om redundante data te verminderen, wat in Cassandra geen doel mag zijn, met de volgende tabellen:

```
CREATE TABLE users (  
    id uuid PRIMARY KEY,  
    username text ,  
    email text ,  
    age int  
)
```

```
CREATE TABLE users_by_username (  
    username text PRIMARY KEY,  
    id uuid  
)
```

```
CREATE TABLE users_by_email (  
    email text PRIMARY KEY,  
    id uuid  
)
```



De data zal met deze tabellen nog steeds evenwichtig gedistribueerd zijn over de nodes, maar nu moet men meer dan één partitie read doen. Dus door een niet-doel proberen te bereiken is hier een belangrijk doel verloren gegaan.

# Hoofdstuk 7

## Data importeren in Cassandra

### 7.1 Importeren via cqlsh

Om de data via cqlsh te kunnen importeren dient eerst een keyspace aangemaakt te worden. Bij het aanmaken van deze keyspace dient de replicatie strategie en de replicatie factor meegegeven te worden. Na het aanmaken dient de tabel waarin de data zal worden geïmporteerd worden aangemaakt te worden.

Nu kunnen we via het "COPY"commando, dat binnen cql voorzien is om data importeren in Cassandra (Cannon, 2012). Een aantal zaken dient hierbij opgemerkte te worden.

1. De volgorde van de kolommen kan gespecificeerd worden aangezien Cassandra de kolommen automatisch alfabetisch sorteert en dit niet noodzakelijk het geval is bij het csv bestand.
2. De scheidingstekens van de velden in het csv bestand kan gespecificeerd worden evenals de encapsulering van de velden.
3. Er kan specifiek meegegeven worden wat Cassandra moet aanvangen met de null waarde.

Dit is een zeer eenvoudige manier om data te importeren in Cassandra. Toch wordt dit deze methode niet aangeraden om te gebruiken bij het importeren van grote hoeveelheden data. Bij ca. 1 miljoen rijen, afhankelijk van het aantal kolommen, kan het zijn dat deze methode vast loopt. Dit kan men op een aantal manieren oplossen. De drie meest voorkomende zijn:

1. Het opsplitsen van één groot csv bestand in verschillende kleinere bestanden.

2. Het gebruik van de sstableloader die Cassandra voorziet.
3. Het gebruik van cassandra-loader

## 7.2 Importeren via sstableloader

Het importeren van data via sstableloader is eveneens een eenvoudig proces. Hier zijn reeds verschillende implementaties van, zoals cassandra bulkloader.

Enkele belangrijke nadelen van deze manier zijn:

- Men moet een aangepaste applicatie schrijven om dit te kunnen gebruiken.
- Om sstableloader te kunnen gebruiken dienen alle nodes van de cluster online te zijn.
- De SSTable dient aangemaakt te zijn vooraleer men deze methode kan gebruiken.

## 7.3 Importeren via cassandra-loader

Dit is een Java programma van Brain Hess. Dit programma maakt gebruik van de CQL driver die voorzien is door DataStax. Het ganse principe van dit programma is om asynchroon cql inserts te doen.

## 7.4 Een vergelijking van de drie methodes

## **Hoofdstuk 8**

### **Gedrag bij uitvallen van een node**

## **Hoofdstuk 9**

### **Back-ups in Cassandra**

# Hoofdstuk 10

## Conclusie

Cassandra can be simply described in a single phrase: a massively scalable, highly available open source NoSQL database that is based on peer-to-peer architecture. (Kan, 2014)

# Bibliografie

- Cannon, P. (2012). Simple data importing and exporting with cassandra. <http://www.datastax.com/dev/blog/simple-data-importing-and-exporting-with-cassandra>. Geraadpleegd op: 2016-03-04.
- Cantoni, B. (2016). MultiNode Template. <https://github.com/bcantoni/vagrant-cassandra/tree/master/2.MultiNode>. Geraadpleegd op 2016-03-25.
- Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4.
- DataStax (2016). Initializing a multiple node cluster (single data center). <https://docs.datastax.com/en/cassandra/2.1/cassandra/initialize/initializeSingleDS.html>. Geraadpleegd op 2016-02-26.
- DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Voshall, P., and Vogels, W. (2007). Dynamo: amazon's highly available key-value store. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 205–220. ACM.
- Fowler, M. (2012). Nosqldefinition. <http://martinfowler.com/bliki/NosqlDefinition.html>. Geraadpleegd op 2016/01/11.
- Fowler, M. (2013). Introduction to nosql. [https://www.youtube.com/watch?v=qI\\_g07C\\_Q5I](https://www.youtube.com/watch?v=qI_g07C_Q5I). [videofile], Geraadpleegd op 2016/01/11.
- Hobbs, T. (2015). Basic Rules of Cassandra Data Modeling. <http://www.datastax.com/dev/blog/basic-rules-of-cassandra-data-modeling>. Geraadpleegd op 2016-03-20.
- Kan, C. (2014). *Cassandra Data Modeling and Analysis*. Packt Publishing Ltd.

- Lakshman, A. and Malik, P. (2010). Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2):35–40.
- Lerer, B. (2015). A deep look at the cql where clause. <http://www.datastax.com/dev/blog/a-deep-look-to-the-cql-where-clause>. Geraadpleegd op 2016-04-25.
- McFadin, P. (2013). Cassandra 2.0 and timeseries. <http://www.slideshare.net/patrickmcfadin/cassandra-20-and-timeseries>. Geraadpleegd op 2016-04-25.
- Sadalage, P. (2014). Nosql databases: An overview. <https://www.thoughtworks.com/insights/blog/nosql-databases-overview>. Geraadpleegd op 2016/01/11.
- Strickland, R. (2014). *Cassandra High Availability*. Packt Publishing Ltd.



## Lijst van figuren

5.1	Cassandra: Instellingen deel 1 . . . . .	11
5.2	Cassandra: Instellingen deel 2 . . . . .	12
5.3	Installatie van Cassandra door OpsCenter . . . . .	13
5.4	Rondleiding in OpsCenter . . . . .	13
5.5	nodetool . . . . .	14
5.6	Toevoegen van een node via OpsCenter . . . . .	15

# Lijst van tabellen

5.1 Configuratie van de Cassandra Cluster . . . . .	11
---	----