Spotitube Onderzoeksrapport

ALEX CHENG (634967)
ITA-OOSE-A-F

02-04-2021 | DEA (Michel Portier)

Inhoud

Inleiding

Dit onderzoeksrapport is geschreven om te kijken of het mogelijk is om een bestaande applicatie met MySQL vervangen te kunnen worden met MongoDB. De huidige applicatie, Spotitube, is een Java applicatie dat zicht vooral verdiept in de back-end.

Op dit moment maakt Spotitube nog gebruik van een MySQL database, maar vanuit dit rapport wordt er uitgebreid gekeken of de schakel naar een MongoDB database realistisch is.

Om dit rapport te voltooien zal er een onderzoek plaatsvinden met de volgende vragen. De hoofdvraag van dit onderzoek is “Hoe kan ik de MySQL database vervangen door een MongoDB database?”. Gebaseerd op deze hoofdvraag zijn er nog enkele deelvragen:

- Wat precies is MongoDB?
- Wat zijn de verschillen tussen MySQL & MongoDB?
- Hoe kan ik MongoDB implementeren in een Java Applicatie?
- Wat moet er plaatsvinden om MongoDB te implementeren in de Spotitube applicatie?

De bovenstaande vragen zullen dan ook zorgvuldig beantwoord worden doormiddel van de onderzoeksmethoden beschreven in de ICT research methods[1].

Voor de eerste twee deelvragen wil ik gebruik maken van de Library research method. Aangezien dit onderzoeksgebied nog in de verkennende fase zit. Doormiddel van al uitgevoerde onderzoeken wordt er uitgezocht wat precies MongoDB is en hoe het verschilt van MySQL.

De derde deelvraag met de Lab research method beantwoord worden. Hiermee zal er een eenvoudige testapplicatie geschreven worden om mee te experimenteren.

De laatste deelvraag zal beantwoord worden met gebruik van de Workshop research method, aangezien hier de database van de al bestaande applicatie, Spotitube, daadwerkelijk aangepast wordt met MongoDB.

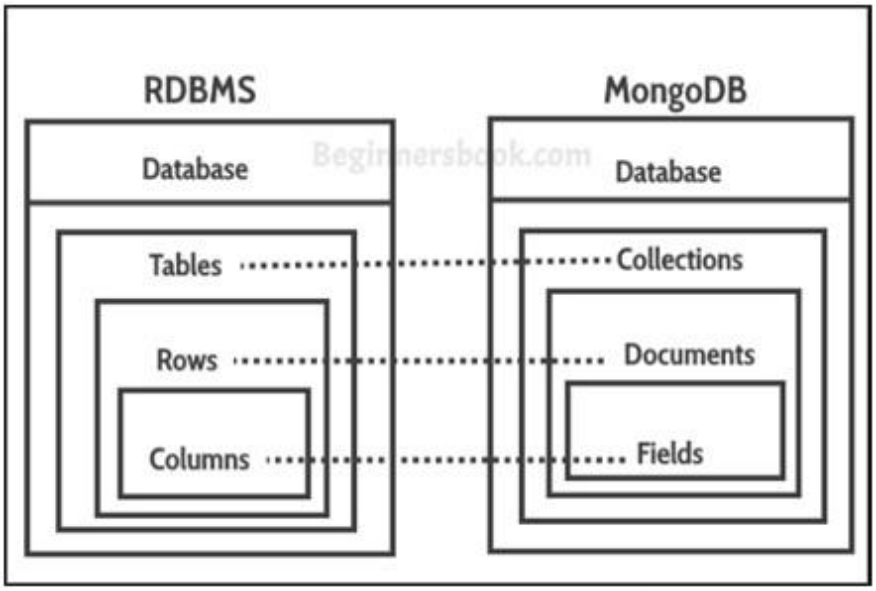
Het uiteindelijke doel van dit onderzoeksrapport is om aan te tonen of MongoDB gebruikelijk kan zijn voor een Java applicatie in plaats van MySQL en dat de lezer nu zelf eenvoudig MongoDB kan implementeren in een al bestaande applicatie.

[1] http://ictresearchmethods.nl/. (2018). Literature study - ICT research methods. Geraadpleegd op 1 April 2021, van http://ictresearchmethods.nl/Literature_study

1. Wat precies is MongoDB?

Hedendaags zijn er veel databases zoals MySQL, misschien wel een van de bekendste relationele databases. Een relationeel database[2] is opgebouwd volgens een relationeel model en bestaat uit tabellen en via een zogenaamde JOIN-clause worden deze tabellen aan elkaar verbonden.

Naast relationele databases bestaan er ook vele andere typen databases, zoals NoSQL databases[3]. Een database die gebruik maakt van deze NoSQL structuur is MongoDB. Wat deze NoSQL structuur zo uniek maakt is het feit dat deze database bestaat uit een of meerdere collecties. Deze collectie bevat dan weer documenten. En ten slotte bevatten deze documenten velden. In principe zijn deze velden de data die opgeslagen wordt.



Zoals hier aangegeven wordt zijn collecties identiek aan tabellen uit een SQL database. Documenten vergelijkbaar met rijen en velden vergelijkbaar met kolommen[4].

Daarnaast wordt de data in MongoDB documenten opgeslagen op een soortgelijke manier als JSON. Hierdoor hebben webdevelopers graag de voorkeur naar MongoDB aangezien de data makkelijk op te halen is met deze structuur.

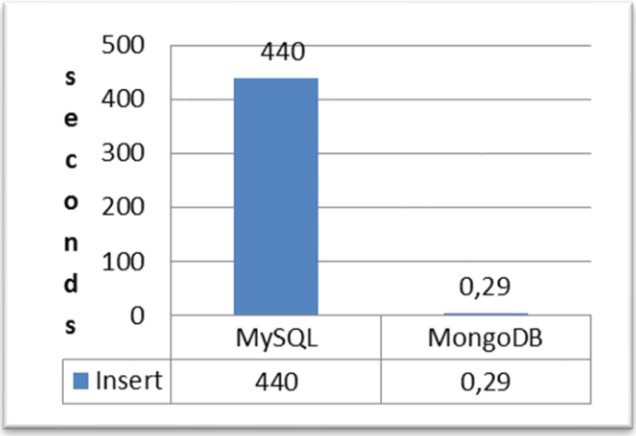
```
{
    "_id": "5f788a2a904ced85fe71231c",
    "username": "Kenerz",
    "firstname": "Ken",
    "password": "$2a$04$acaUVljoRAvazzj6YX7K2eEfUt9PHVVgr",
    "adress": {
        "street": "Stallstigen 93",
        "city": "Hargshamn",
        "state": "Sweden",
        "zip": "74073"
    },
    "games": ["Overwatch", "Valorant", "Minecraft"]
}
```

[2] Wikipedia-bijdragers. (2020b, 21 juni). Relationele database. Geraadpleegd op 1 april 2021, van https://nl.wikipedia.org/wiki/Relationele_database
[3] VdJ, E. (2020, 4 september). Welke soorten databases bestaan er en waarvoor dienen ze? Geraadpleegd op 1 april 2021, van https://www.ictportal.nl/ict-lexicon/database
[4] A deep dive into Mongo Database. (z.d.). Geraadpleegd op 1 april 2021, van https://www.includehelp.com/mongodb/a-deep-dive-into-mongo-database.aspx

2. Wat zijn de verschillen tussen SQL & Mongo?

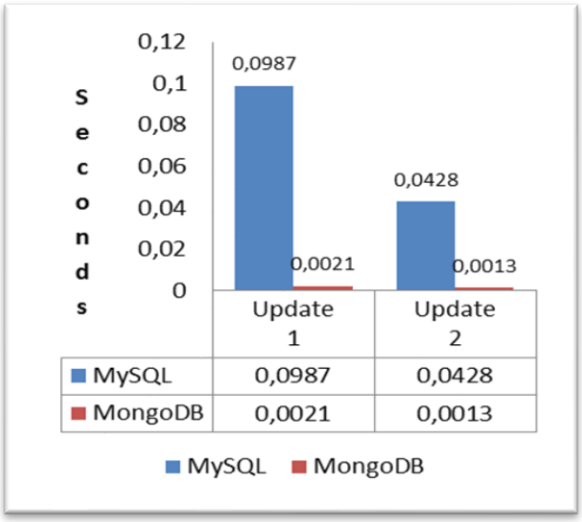
In het vorige hoofdstuk is er uitgelegd wat precies MongoDB is en wat het zo uniek maakt, nu wordt er verdiept wat de daadwerkelijke verschillen zijn tussen MongoDB en MySQL.

Het eerste verschil is dat MongoDB flexibelere data modellen heeft, waardoor gegevens makkelijker te beheren zijn en ook is er wetenschappelijk bewezen[5] dat MongoDB meestal veranderingen in de database sneller uitgevoerd. Althans het moeilijk is om precies te weten welke database sneller zijn er echter toch kleine testen geweest om te achterhalen welke database het snelst is. Hieronder staan dan ook enkele afbeeldingen van de resultaten, deze verschillen dus echt per functie.



Deze afbeelding laat zien hoelang elke database er over doet om 10,000 gebruikers te inserten, MySQL deed hier 440 seconden over terwijl,

MongoDB er maar 0,29 seconden over deed.



In de tweede afbeelding wordt getoond hoe twee updates worden uitgevoerd. Hieruit is zichtbaar dat Mongo al twee updates heeft uigevoerd voordat SQL er eentje heeft uitgevoerd.

Uit nog een aantal vergelijkbare tests kan er geconcludeerd worden dat bij een grote hoeveelheid van data het bij MongoDB sneller aangepast kan worden dan bij MySQL.

Daarnaast is er nog een groot verschil tussen de twee databases, namelijk het feit dat NoSQL databases zoals MongoDB horizontaal schaalbaar zijn, terwijl SQL databases verticaal schaalbaar zijn. Dit houdt in dat MongoDB meer apparaten, bijvoorbeeld servers, nodig heeft om het gehele systeem krachtiger te maken. Ondertussen krijgt SQL meer kracht door een al bestaand systeem te verbeteren (denk hierbij bijvoorbeeld aan een verbeterde processor).

[5] Győrödi, C., Győrödi, R., Pecherle, G., & Olah, A. (z.d.). A Comparative Study: MongoDB vs. MySQL. Geraadpleegd op 1 april 2021, van https://www.researchgate.net/publication/278302676_A_Comparative_Study_MongoDB_vs_MySQL

Het derde verschil tussen MySQL en Mongo zijn de query's.
Een simpele UPDATE query ziet er als volgt uit:

MySQL

```
UPDATE accounts SET firstname = "Alex"
WHERE username = "Kenerz"
```

MongoDB

```
db.accounts.update
(
    {
        "username" : "Kenerz"
    },
    {
        $set :
        {
            "firstname" : "Alex"
        }
    }
)
```

De meeste vinden dat een eenvoudige query bij MySQL er overzichtelijker uitziet dan bij MongoDB, dit klopt wel aangezien MySQL in dit voorbeeld leesbaarder is dan MongoDB door de globale woorden 'UPDATE', 'SET' & 'WHERE'.
In tegendeel, als we eenmaal uitgebreide query's krijgen waarin je complexere query's in wilt verwerken dan blijkt MongoDB toch gemakkelijker te schrijven te zijn.

Ten slotte, is uit een survey[6] te blijken welke databases er het meest gebruikt was in 2020. Hieruit is te zien dat MySQL nog steeds in 2020 bovenaan stond met een percentage van 53,5% van alle professionele developers, kort daarna komt MongoDB tevoorschijn met 26,7%. MongoDB de enige niet relationele database die voorkomt in de top5

[6] Romanowski, J. (2020, 18 augustus). The Most Popular Databases in 2020. Geraadpleegd op 1 april 2021, van https://learnsql.com/blog/most-popular-sql-databases-2020/

3. Hoe kan ik MongoDB implementeren in een Java Applicatie?

In dit hoofdstuk wordt er uitgelegd hoe MongoDB geïmplementeerd kan worden in een Java applicatie[7]. Er wordt verwacht dat de lezer al een kennis heeft van Maven, TomEE en Java.

Als eerste moet de juiste dependency toegevoegd worden in het *pom.xml* bestand. Hierdoor wordt het mogelijk dat MongoDB kan communiceren met Java.

```xml
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>4.2.2</version>
</dependency>
```

Daarnaast importeer je de volgende punten die nu aangeboden worden door de *MongoDB Driver* dependency.

```java
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoDatabase;
import com.mongodb.client.MongoCollection;

import org.bson.Document;
import java.util.Arrays;

import static com.mongodb.client.model.Filters.*;
import static com.mongodb.client.model.Updates.*;
```

Door deze imports is het nu mogelijk om een verbinding te maken met de MongoDB database door een MongoClient aan te maken. Daarna vermeld je de juiste database & collectie waaruit de data opgehaald moet worden.

```java
MongoClient mongoClient = MongoClients.create();
MongoDatabase database = mongoClient.getDatabase("test_database");
MongoCollection<Document> collection = database.getCollection("test_collection");
```

[7] A deep dive into Mongo Database. (z.d.). Geraadpleegd op 1 april 2021, van https://www.includehelp.com/mongodb/a-deep-dive-into-mongo-database.aspx

Nadat de connectie gemaakt is, is het eindelijke mogelijk om daadwerkelijke documenten toe te voegen aan de betreffende collectie. Dit wordt gedaan door een nieuw document te maken en deze mee te geven aan een *insertOne.* Vermeld hierbij de veldnaam met de waarde. In de afbeelding wordt er ook gebruik gemaakt van een *append*, dit is om extra velden toe te voegen aan het document.

```
Document document = new Document("name", "Alex")
        .append("age", 18);
collection.insertOne(document);
```

Om het gemaakte document nu op te zoeken kunnen wij gebruik maken van de *find* methode. Hierbij voegen we een *filter* toe om iets specifieks te zoeken en daarnaast gebruiken we ook nog een *first* zodat wij alleen het eerste resultaat terugkrijgen.

```
Document document = collection.find(eq("name", "Alex")).first();
```

Ook is het mogelijk om een al bestaand document aan te passen, dit wordt gedaan doormiddel van de *updateOne* methode. Ook hierbij wil je doormiddel van een *filter* specificeren welk document je precies wilt veranderen en met een *set* laten wij weten naar wat we het willen veranderen.

```
collection.updateOne(eq("name", "Alex"), set("name", "Alexandra"));
```

Natuurlijk is het ook mogelijk om een document te verwijderen uit de collectie, dit doen we met de *deleteOne* methode. Weer specificeren we met een *filter* welk document we willen verwijderen.

```
collection.deleteOne(eq("name", "Alexandra"));
```

De source code van deze voorbeeld applicatie is te vinden als bijlage. Hiermee kan er dus nog gespeeld worden om een beter begrip te krijgen voor MongoDB in een Java applicatie.

4. Wat moet er plaatsvinden om MongoDB te implementeren in de Spotitube applicatie?

Aangezien de huidige Spotitube applicatie al gebruik maakt van *dependency injection* kunnen we eenvoudig de MySQL database connectie naar een MongoDB connectie veranderen in de DAO, zonder dat we iets moeten aanpassen in de andere lagen. In ons geval zullen we de TokenDAO & UserDAO aanpassen zodat het werkt met MongoDB.

Voordat we MongoDB kunnen implementeren in de Spotitube applicatie moeten we net zoals bij het vorige hoofdstuk de *pom.xml* bestand aanpassen om hier de MongoDB dependency te importeren.

Nadat we de MongoDB driver hebben toegevoegd moeten we specificeren in de *beans.xml* file welke klassen gebruik moeten maken van een alternatieve implementatie.

```
<alternatives>
    <class>han.oose.dea.dao.UserDAOMongo</class>
    <class>han.oose.dea.dao.TokenDAOMongo</class>
</alternatives>
```

Nadat we hebben gespecificeerd welke klassen gebruik maken van een alternatieve implementatie kunnen we gebruik maken van de *@Alternative* annotatie bovenaan de klasse in plaats van de oude *@Default* annotatie.

```
@Alternative
public class UserDAOMongo implements IUserDAO {
```

Ook moeten we daarna bij de injection setter alleen nog de juiste klasse aanwijzen. Dit doen we eenvoudig door de correcte klasse mee te geven aan de setter als volgt.

```
@Inject
public void setUserDAO(IUserDAO userDAOMongo) {
    this.iUserDAO = userDAOMongo;
}
```

Als we aan al deze eisen hebben voldaan kunnen we de methode zelf schrijven, zo ziet dan de uiteindelijke klasse eruit. De uiteindelijke uitwerking kan ook gevonden worden als bijlage van dit document.

```java
@Alternative
public class UserDAOMongo implements IUserDAO {
    public static final String DB_NAME = "spotitube";
    public static final String CO_NAME = "users";

    MongoClient mongoClient = MongoClients.create();
    MongoDatabase database = mongoClient.getDatabase(DB_NAME);
    MongoCollection<Document> usersCollection = database.getCollection(CO_NAME);

    @Override
    public User checkAuthenticated(String username, String password) throws UnauthorizedException {
        try {
            Document result = usersCollection.find(and(eq("username", username), eq("password", password))).first();

            User user = new User();
            user.setUsername((String) result.get("username"));
            user.setToken((String) result.get("token"));

            return user;
        } catch (Exception e){
            throw new UnauthorizedException();
        }
    }
}
```

Een aantal belangrijke punten die we zien in de volledige uitwerking zijn:

- De annotatie van de klasse
- De geïmplementeerde klasse via de injection
- De database connectie doormiddel van de MongoClient met daarbij de initialisatie van de juiste database & collectie.
- De MongoDB find query

Conclusie

MongoDB is een erg geschikte database om te gebruiken wanneer je gebruik maakt van grote hoeveelheden data. Dit is wegens het feit dat MongoDB een NoSQL database is en de data opslaat op een JSON manier, hierdoor kunnen ze sneller opgehaald en verwerkt worden dan in een MySQL database.

Zoals aangetoond in hoofdstuk 4, kost het niet al teveel moeite om een Java applicatie met MySQL te veranderen naar MongoDB. Een paar simpele imports en de injecties, daarna is het vooral de query's waar even goed gekeken naar moet worden.

Uit de al bestaande onderzoeken en mijn eigen onderzoek kan er geconcludeerd worden dat MySQL makkelijker te gebruiker is bij simpele query's, maar MongoDB erg gepast is voor uitgebreidere query's aangezien het geen ingewikkelde joins heeft.

Tenslotte na het beantwoorden van alle deelvragen kan er ook de conclusie worden getrokken dat MongoDB een goede alternatief is voor MySQL, zeker voor developers, aangezien het eenvoudig is te implementeren. Hierdoor is het niet alleen gepast voor de Spotitube applicatie, maar ook voor toekomstige applicaties waar veel data uit de database bij wordt betrokken.

Bronnen

Dit is een lijst van alle gebruikte bronnen om dit onderzoeksrapport te voltooien:

- http://ictresearchmethods.nl/. (2018). Literature study - ICT research methods. Geraadpleegd op 1 April 2021, van http://ictresearchmethods.nl/Literature_study
- A deep dive into Mongo Database. (z.d.). Geraadpleegd op 1 april 2021, van https://www.includehelp.com/mongodb/a-deep-dive-into-mongo-database.aspx
- Győrödi, C., Győrödi, R., Pecherle, G., & Olah, A. (z.d.). A Comparative Study: MongoDB vs. MySQL. Geraadpleegd op 1 april 2021, van https://www.researchgate.net/publication/278302676_A_Comparative_Study_MongoDB_vs_MySQL
- Installation Guide. (z.d.). Geraadpleegd op 1 april 2021, van https://mongodb.github.io/mongo-java-driver/3.2/driver/getting-started/installation-guide/
- Romanowski, J. (2020, 18 augustus). The Most Popular Databases in 2020. Geraadpleegd op 1 april 2021, van https://learnsql.com/blog/most-popular-sql-databases-2020/
- VdJ, E. (2020, 4 september). Welke soorten databases bestaan er en waarvoor dienen ze? Geraadpleegd op 1 april 2021, van https://www.ictportal.nl/ict-lexicon/database
- Wikipedia-bijdragers. (2020b, 21 juni). Relationele database. Geraadpleegd op 1 april 2021, van https://nl.wikipedia.org/wiki/Relationele_database