ONDERZOEKSVERSLAG MONGODB

Een onderzoek naar het verschil tussen MongoDB en MySQL in de Spotitube applicatie

Jordy Veldhuizen 547555

27-10-2020

Abstract

In dit document wordt er gekeken naar de verschillen tussen MySQL en MongoDB bij een Java applicatie. Dit wordt gedaan aan de hand van drie onderzoeksmethoden. Deze drie onderzoeksmethoden hebben betrekking tot drie deelvragen. Deze deelvragen zijn: (Bieb) Wat is MongoDB? (Lab) Hoe is MongoDB te gebruiken bij een Java applicatie? En (Werkplaats) Wat veranderd er aan de 'Spotitube backend' applicatie wanneer er gebruik word gemaakt van MongoDB i.p.v. MySQL? MongoDB is een non relationele NoSQL database die gebruikt maakt van collections en documents voor opslag. Om MongoDB te gebruiken bij een Java applicatie is er een Driver nodig die toe te voegen is via een Maven dependency. Wanneer MongoDB MySQL vervangt in de 'Spotitube backend' applicatie veranderen voornamenlijk de POJO's. Zo moeten er nieuwe aangemaakt worden omdat deze niet passen bij de structuur van Mongo. Zoals benoemd zitten de grootste verschillen in het verschil van opslaan van de data, het gebruik van POJO's en het kunnen wisselen tussen databases.

Inhoudsopgave

1. Inleiding

Dit document bevat een onderzoek over de verschillen en overeenkomsten tussen MySQL en MongoDB bij een Java applicatie. Om deze hoofdvraag te ondersteunen zijn er drie deelvragen opgesteld. Deze zijn:

- Wat is MongoDB?
- Hoe is MongoDB te gebruiken bij een Java applicatie?
- Wat veranderd er aan de 'Spotitube backend' applicatie wanneer er gebruik word gemaakt van MongoDB i.p.v. MySQL?

Om deze deelvragen te beantwoorden wordt gebruik gemaakt van meerdere methoden van de Methodenkaart (“HBO-i-methoden-toolkit”, 2016). Deze methoden zijn: Bieb, lab en werkplaats. Deze methoden zitten gekoppeld aan de drie deelvragen. De resultaten van deze deelvragen zullen per deelvraag weergegeven worden. Uit deze resultaten zullen conclusies worden getrokken en zo ook de hoofdvraag beantwoorden.

2. Resultaten

In dit hoofdstuk worden de deelvragen die de hoofdvraag ondersteunen beantwoord.

Wat is MongoDB?

Om er achter te komen waarin MySQL en MongoDB verschillen word er eerst gekeken naar wat MongoDB precies is. In de huidige uitwerking van Spotitube, die veranderd zal worden bij deelvraag 3, word gebruik gemaakt van MySQL. Om er achter te komen wat MongoDB nou precies is, is er gebruik gemaakt van twee bronnen. De eerste bron is van MongoDB zelf. Namenlijk: What is MongoDB? (MongoDB, 2020). Omdat MongoDB ook de maker is van deze database heb ik er voor gekozen om ook gebruik te maken van een afhankelijke bron. Dit is: Definition: MongoDB (Rouse, 2020). Mijn bevindingen van deze twee bronnen zijn hier onder te vinden.

In tegenstelling tot MySQL is MongoDB een NoSQL database. NoSQL wordt gebruikt als een alternatief voor traditionele relationele databases. MongoDB is dan ook een non relationele database. Waar bij relationele databases gebruik wordt gemaakt van tables en rows wordt er bij MongoDB gebruik gemaakt van collections en documents. De data wordt dus opgeslagen in documents aan de hand van key-value paren. Deze manier van opslaan doet denken aan JSON, zo word er ook gebruik gemaakt van een variant van JSON. Deze variant heet BSON (Binary JSON). Het voordeel van deze variant is dat het ruimte geeft voor meer datatypes. Een groep documents wordt een collection genoemd. Deze collections zijn te vergelijken met de tables die gebruikt worden in relationele databases.

Net als andere NoSQL databases maakt MongoDB geen gebruik van schemas. Zo kunnen alle datatypen opgeslagen worden. Dit geeft meer flexibiliteit en maakt het makkelijker om een database groter te maken.

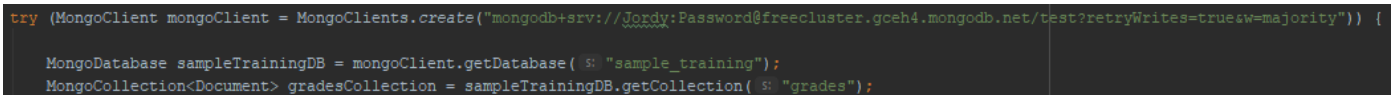
Hoe is MongoDB te gebruiken bij een Java applicatie?

Om MongoDB toe te kunnen passen bij de Spotitube backend applicatie moet er onderzocht worden hoe MongoDB te gebruiken is bij een Java applicatie. Om hier achter te komen heb ik een tutorial gevolgd op de website van MongoDB. De eerste stap van deze tutorial is: MongoDB & Java – CRUD Operations Tutorial (Beugnet, 2020b). Omdat er in de spotitube applicatie ook gebruik wordt gemaakt van POJO's zal ik ook een tutorial hier over volgen: Java – Mapping POJO's (Beugnet, 2020a). Om weer te geven hoe deze tutorials verliepen zal ik mijn ervaring weergeven met een aantal codevoorbeelden waar nodig. Om niet te veel de tutorials te herhalen zal ik voornamenlijk samenvatten in plaats van bij iedere actie een voorbeeld te geven. Het uiteindelijke project zal te vinden zijn in Bijlage 1: MongoDB tutorials: Java Quickstart.

CRUD Operations Tutorial

Voordat ik bezig kon met deze tutorial moest ik eerst een cluster aanmaken op MongoDB Atlas. Dit is een redelijk nieuwe cloud oplossing van MongoDB. Zo kan ik deze clusters gebruiken voor test databases of om te leren hoe MongoDB werkt. Deze clusters geven een optie om test data in te voeren in verschillende databases. Deze test data wordt gebruikt bij deze tutorial. Bij het aanmaken van deze cluster heb ik gebruik gemaakt van een andere tutorial: Getting your free MongoDB atlas cluster (Beugnet, 2020c).

Wanneer je een project aanmaakt met maven moet je natuurlijk een dependency toevoegen in je pom.xml. Hiernaast wordt er ook gebruik gemaakt van een plugin. Verbinding maken met een database is redelijk simpel. Je maakt een Mongo client aan en die geef je de connection string van de eerder aangemaakte Cluster. Wanneer je op zoek bent naar een specifieke database op deze cluster kan je hier op filteren. Een voorbeeld uit de tutorial is hier onder te vinden.

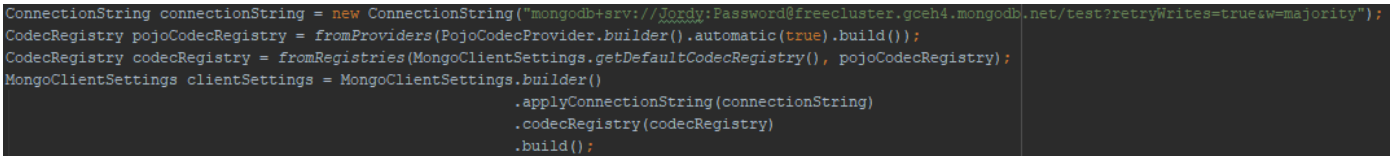


Afbeelding 1: Codevoorbeeld CRUD tutorial, Connecting to MongoDB Cluster

Net zoals bij de MySQL oplossing zullen er alleen CRUD acties worden gedaan op de database. Na het uitproberen van de connectie en het ophalen van de test data gaan we verder met de CRUD acties. Het grootste verschil wat gelijk opvalt is dat er gebruik gemaakt word van documents. Naast een paar kleine dingen zijn er niet veel dingen die er echt tussen uit springen. Deze tutorial geeft een duidelijke voorbeelden bij alle CRUD acties. Bij deze tutorial haal je documenten uit de database op. Bij de volgende tutorial krijg ik hopelijk te zien hoe ik deze kan mappen naar POJO's.

Java – Mapping POJOs

Deze kort maar krachtige tutorial laat zien hoe je met behulp van de MongoDB drivers documents naar POJO's kan mappen. Naast een redelijk ingewikkeld stukje code over de MongoClientSettings is deze tutorial heel duidelijk. Dit 'ingewikkeld stukje code' is een CodecRegistry die er voor zorgt dat de documents worden gemapt naar POJO's. Zo wordt dus eigenlijk alles voor je gedaan. Wanneer je een POJO goed hebt opgezet (indien nodig een BSONProperty toegevoegd) kost het heel weinig moeite om te mappen. Zo kan je bij CRUD acties gewoon de POJO meegeven. Het eerder genoemde 'ingewikkeld stukje code' is hier onder te vinden.



Afbeelding 2: Codevoorbeeld Mapping tutorial: 'Ingewikkeld stukje code'

Wat veranderd er aan de 'Spotitube backend' applicatie wanneer er gebruik word gemaakt van MongoDB i.p.v. MySQL?

Wanneer er gebruik wordt gemaakt van MongoDB in plaats van MySQL bij de spotitube applicatie zullen er meerdere dingen moeten veranderen. Bij deelvraag 2: Hoe is MongoDB te gebruiken bij een Java applicatie? Is onderzocht hoe dit gedaan kan worden. Bij deze deelvraag zal worden beschreven wat er allemaal precies is veranderd en hoe dit is gedaan.

Na twee duidelijke tutorials gevolgd te hebben is het nu tijd om deze nieuwe kennis toe te passen op de Spotitube backend applicatie. Om te beginnen heb ik nieuwe DAO's aangemaakt en de oude (MySQL) DAO's deprecated gemaakt. Om te kunnen testen of ik connectie kon maken met de database heb ik het voorbeeld uit de eerste tutorial overgenomen. Hier uit bleek dat ik verbinding kon maken met mijn nieuwe database. Deze database (genaamd Spotitube) bevat data die lijkt op de test data gebruikt in de MySQL database. Echter wordt er nu in plaats van een koppeltabel die bijhoudt welke nummers in welke playlist staan een document gebruikt met daar in een array van nummers. Een voorbeeld van een playlist is hier onder te zien.


```
_id: ObjectId("5f9b470616f7f9657525904b")
name: "TestPlaylist"
owner: "Jordy323"
playlistId: 1
tracks: Array
  0: Object
      _id: ObjectId("5f9b1d99fc9155bd7a8381a6")
      album: "The Black Album"
      description: " "
      duration: 423
      offlineAvailable: false
      performer: "Metallica"
      playcount: 2
      title: "One"
      trackId: 3
```


Afbeelding 3: Voorbeeld data Playlist

Voor het ontwikkelen zal ik net als bij de MySQL implementatie beginnen met het inloggen. Het inloggen betreft twee acties: het controleren van username en password en het toevoegen van een token. Omdat er niet wordt gedacht aan security zijn username en password gewoon beschikbaar als plain tekst. Voor het inloggen zal dus alleen gekeken hoeven worden of de gegevens bestaan in de database. Dit heb ik als volgt gedaan:

```
@Override
public void userLogin(UserLoginDTO dto) throws InvalidUserOrPasswordException {
    try {
        MongoCollection<Document> userCollection = database.getDatabase().getCollection( s: "user");
        Document user = userCollection.find(and(eq( fieldName: "username", dto.getUser()), eq( fieldName: "password", dto.getPassword()))).first();
        if (user == null){
            throw new InvalidUserOrPasswordException();
        }
    }catch (Exception e) {
        e.printStackTrace();
        throw new InvalidUserOrPasswordException();
    }
}
```

Afbeelding 4: Controleren username en password

Bij het maken van de login ben ik niet echt tegen problemen aan gelopen. De problemen kwamen pas bij het ophalen van een playlist. Dit was het moment dat ik er achter kwam dat de door mij gemaakte POJO's (DTO genoemd) niet geschikt waren voor het mappen van documenten. Als oplossing hier voor heb ik nieuwe POJO's aangemaakt die wel gemapt kunnen worden. Om toch de oude DTO's door te kunnen sturen naar de front end heb ik nieuwe constructors aangemaakt. Hier onder zal ik een voorbeeld weergeven van een constructor. Dit betreft de constructor van de PlaylistDTO.

```
public PlaylistDTO(PlaylistPOJO playlist, String currentUser) {
    this.id = playlist.getPlaylistId();
    this.name = playlist.getName();
    this.owner = currentUser.equals(playlist.getOwner());
    for (TrackPOJO trackPOJO : playlist.getTracks()) {
        this.tracks.add(new TrackDTO(trackPOJO));
    }
}
```

Afbeelding 5: Constructor PlaylistDTO

Hier is te zien dat deze constructor twee parameters heeft. Als eerst de POJO opgehaald uit de database, en als tweede de currentUser gekregen van de front end. De opgehaalde owner in de POJO is een string met de username van de Owner. De front end verwacht echter een boolean die weergeeft of de huidige gebruiker de Owner is. Omdat een playlist een lijst met tracks bevat zullen al deze tracks ook moeten worden omgezet van POJO naar DTO.

Nadat dit probleem was opgelost verliep het verder best soepel. Alle andere CRUD taken die ook beschikbaar waren in de MySQL uitwerking zijn aanwezig. Een van de dingen die ik hier nog weer wil geven is het verwijderen van een track uit een playlist.

```
@Override
public void removeTrackFromPlaylist(int playlistId, int trackId) throws DeletionException {
    try{
        MongoCollection<PlaylistPOJO> playlistCollection = database.getDatabase().getCollection( s: "playlist", PlaylistPOJO.class);
        PlaylistPOJO playlist = playlistCollection.find(eq( fieldName: "playlistId", playlistId)).first();

        if (playlist == null){
            throw new Exception();
        }
        playlist.getTracks().removeIf(trackPOJO -> trackPOJO.getTrackId() == trackId);
        Document filterByPlaylistId = new Document("_id", playlist.getId());
        playlistCollection.findOneAndReplace(filterByPlaylistId, playlist);
    }catch (Exception e){
        e.printStackTrace();
        throw new DeletionException("PlaylistID: " + playlistId + " TrackID: " + trackId);
    }
}
```

Afbeelding 6: Het verwijderen van een track uit een playlist

Doordat playlistId en trackId aanwezig zijn als parameter is er heel weinig informatie nodig om deze Delete actie uit te voeren. Hier is ook goed te zien dat het mappen naar POJO's veel werk scheelt.

De volledige uitwerking van Spotitube is te vinden in Bijlage 2.

3. Conclusie

Bij iedere deelvraag wordt een conclusie getrokken aan de hand van de resultaten te vinden in hoofdstuk 3. Resultaten. Aan de hand van deze conclusies wordt ook de hoofdvraag beantwoord.

Wat is MongoDB?

MongoDB is een non relationele, NoSQL database. Om de data op te slaan wordt er gebruik gemaakt van Collecties en Documenten. In een collectie kunnen meerdere documenten zitten. In deze documenten wordt de data opgeslagen in BSON, een JSON variant die ruimte biedt voor meer datatypes.

Hoe is MongoDB te gebruiken bij een Java applicatie?

Om gebruik te maken van MongoDB bij een Java applicatie moet er eerst een cluster gecreëerd worden. Dit kan worden gedaan via Mongo Atlas. Na het aanmaken van deze cluster moet er een maven dependency worden toegevoegd. Na het toevoegen van deze dependency is het mogelijk om verbinding te maken met de cluster. Op een cluster kunnen meerdere databases zitten. Het is dus gemakkelijk om tussen databases te wisselen. Na het ophalen van de database kunnen alle CRUD handelingen verricht worden.

Wanneer er gebruik wordt gemaakt van POJO's kunnen de documenten die worden opgehaald uit de database gelijk worden gebruikt als POJO's. Dit verminderd (hopelijk) werk en maakt het overzichtelijker om data te verwerken.

Wat veranderd er aan de 'Spotitube backend' applicatie wanneer er gebruik word gemaakt van MongoDB i.p.v. MySQL?

Wanneer de database veranderd zullen alle DAO's (Data Access Objects) veranderen. De interfaces zullen hetzelfde blijven aangezien de CRUD acties hetzelfde blijven. Omdat er bij het ontwikkelen van de applicatie geen rekening is gehouden met de opbouw van de POJO's (t.o.v. MongoDB) zullen deze aangemaakt moeten worden. Om de DTO's te kunnen versturen naar de front end zijn er meerdere constructors aangemaakt die het wisselen tussen POJO's en DTO's ondersteunen.

De opzet van alle CRUD acties is goed met elkaar te vergelijken. Vaak haal je iets op uit de database aan de hand van een al bekend ID. Hier wordt dan op gefilterd. Voornamelijk bij het updaten en deleten van documenten (en onderdelen van documenten) zit er een duidelijk verschil. Waar het bij MySQL veel draait om relaties krijg je, bijvoorbeeld een playlist, in een compleet document. Zo kan je dit gehele document aanpassen en/of verwijderen.

Hoofdvraag: Wat zijn de verschillen en de overeenkomsten tussen MySQL en MongoDB bij een Java applicatie?

Als je MongoDB en MySQL tegenover elkaar zet kan je zien dat er veel verschillen zijn. Echter zijn er ook een aantal overeenkomsten. Het duidelijkste verschil zat hem bij de spotitube applicatie in het gebruik van de POJO's. Bij de MySQL implementatie waren deze volledig gefocussed op de informatiebehoeften van de front end. Zo moest er veel veranderen aan deze structuur wanneer er gebruik werd gemaakt van MongoDB. De opzet van de DAO laag komt qua opzet veel overeen. Het maken van een connectie is bij beiden simpel. Bij MongoDB is het echter mogelijk om met een cluster te verbinden die toegang heeft tot meerdere databases. Zo is het veel makkelijker om hier tussen te wisselen. Bij MySQL is dit niet mogelijk en geld een connectionstring voor 1 database. Een ander verschil is het dat het bij MongoDB mogelijk is om de data uit de database te mappen naar POJO's. Dit scheelt als je het vergelijkt met de MySQL oplossing. Daar wordt namenlijk door de resultaten gelooped en stuk voor stuk opgeslagen in de eerder genoemde DTO's.

4. Literatuurlijst

HBO-i-methoden-toolkit. (2016, 14 maart). Geraadpleegd op 29 oktober 2020, van https://onderzoek.hbo-i.nl/index.php/Methoden_Toolkit_HBO-i

MongoDB. (2020). *What Is MongoDB?* Geraadpleegd op 29 oktober 2020, van https://www.mongodb.com/what-is-mongodb

Rouse, M. (2020, 28 augustus). *MongoDB*. Geraadpleegd op 29 oktober 2020, van https://searchdatamanagement.techtarget.com/definition/MongoDB

Beugnet, M. (2020a, 21 oktober). *Learn how to map POJOs using the MongoDB Java Driver*. Geraadpleegd op 29 oktober 2020, van https://developer.mongodb.com/quickstart/java-mapping-pojos/

Beugnet, M. (2020b, 21 oktober). *Learn how to use MongoDB with Java in this CRUD operations tutorial*. Geraadpleegd op 29 oktober 2020, van https://developer.mongodb.com/quickstart/java-setup-crud-operations/

Beugnet, M. (2020c, 2 maart). *Getting Your Free MongoDB Atlas Cluster*. Geraadpleegd op 29 oktober 2020, van https://developer.mongodb.com/quickstart/free-atlas-cluster/

5. Bijlagen

Bijlage 1: MongoDB tutorials: Java Quickstart

/Bijlagen/javaquickstart.zip

Deze code is naast de bijlagen ook te vinden op git: https://github.com/jordyebk/java-quick-start

Bijlage 2: Spotitube met MongoDB

/Bijlagen/Oose-Spotitube-MongoDB.zip

Deze uitwerking is naast de bijlagen ook te vinden op git: https://github.com/jordyebk/Oose-spotitube/tree/DEA-ONDERZOEK