MongoDB
Getting started (tutorial)

In dit document maak je kennis met de technische mogelijkheden van MongoDB. Neem het grondig door en experimenteer met de voorgestelde features alvorens aan DEEL 4 van het examenproject te beginnen.

Installatie

De installatieprocedure voor de verschillende platformen is terug te vinden op http://docs.mongodb.org/manual/installation. Neem best de meeste recente versie, en kies voor de Community Edition.
Hoewel dit handig kan zijn, hoef je MongoDB niet als een windows service te installeren, dus dat stuk kan je overslaan.

Management

De command line interface van MongoDB is gebaseerd op Javascript maar is niet echt gebruiksvriendelijk.

Uiteraard zijn er ook management *tools* beschikbaar:
- ***Robo 3T*** *(vroegere Robomongo)* (Windows, Mac, Linux) wordt veel gebruikt en heeft een handige ingebouwde shell.[1] De versie 1.2 van deze light-weigth MongoDB management tool is gratis. Studio 3T is dat niet en kan je dus best niet downloaden. 3T heeft Robomongo overkocht waardoor Robo 3T niet kan rekenen op nieuwe features.

- MongoDB Compass Community: De GUI tool van MongoDB zelf. Deze is vrij beperkt maar heeft wel een nuttige built in aggregations editor. Wel jammer dat je nergens gewoon een shell achtig venster hebt.

Aanmaken database in MongoDb

1. Om MongoDb op te kunnen starten, moet je lokaal een folderstructuur aanmaken waarin MongoDb zijn data bestanden gaat zetten. Standaard is dit path “C:\data\db” maar je kan dat zelf aanpassen.
2. Start nadien MongoDB ((<install_dir>/bin/mongod) op en experimenteer even met de Robo 3T interface en de shell/command line
    a. Als je het path waar je je datafiles in stockeert anders dat de standaard wil, moet je dit path wel als argument (--dbpath) meegeven aan het commando mongod.
    b. Als je mongod opgestart heb, toont hij ook op welke poort MongoDb staat. Die poortnummer heb je nodig om vanuit Robo 3T een connectie te leggen naar je MongoDb

[1] *Voor een ruim overzicht van MongoDB tools zie bv. http://mongodb-tools.com/ en https://scalegrid.io/blog/which-is-the-best-mongodb-gui/).*

3. Gebruik Robo 3T om in je MongoDb server een database en een collectie aan te maken. Voeg ook een JSON document toe en vraag dit via de interface terug op. Maak jezelf vertrouwt met de interface. Op het einde mag je terug alles verwijderen van inhoud.

4. Maak nu een database 'cinema' aan en importeer hierin de dataset films.csv (zie Canvas) via mongoimport. Zie http://docs.mongodb.org/manual/reference/program/mongoimport/ voor de juiste opties die je dient mee te geven aan dit commando.
   Let op: dit is een afzonderlijk programma en run je dus niet vanuit de mongoDB shell

Zoekoperaties (query'en) binnen MongoDb

Het opzoeken van documenten in collecties gebeurt met behulp van find(). Ter vergelijking:

```
SELECT *
FROM users
WHERE status = "A"
AND age = 50
```

VERSUS

```
db.users.find(
    { status: "A",
      age: 50 }
)
```

Een mooi overzicht van MongoDB document operaties versus de overeenkomstige SQL table commando's vind je hier: http://docs.mongodb.org/manual/reference/sql-comparison/

Opdracht:

- Geef het aantal films dat na 1990 is gemaakt en een award heeft gehaald.
- Geef het aantal films dat na 1990 is gemaakt of een award heeft gehaald.
- Geef het aantal films dat na 1990 is gemaakt of een award heeft gehaald, gesorteerd volgens title en beperkt tot de eerste 10 films.

Indexen

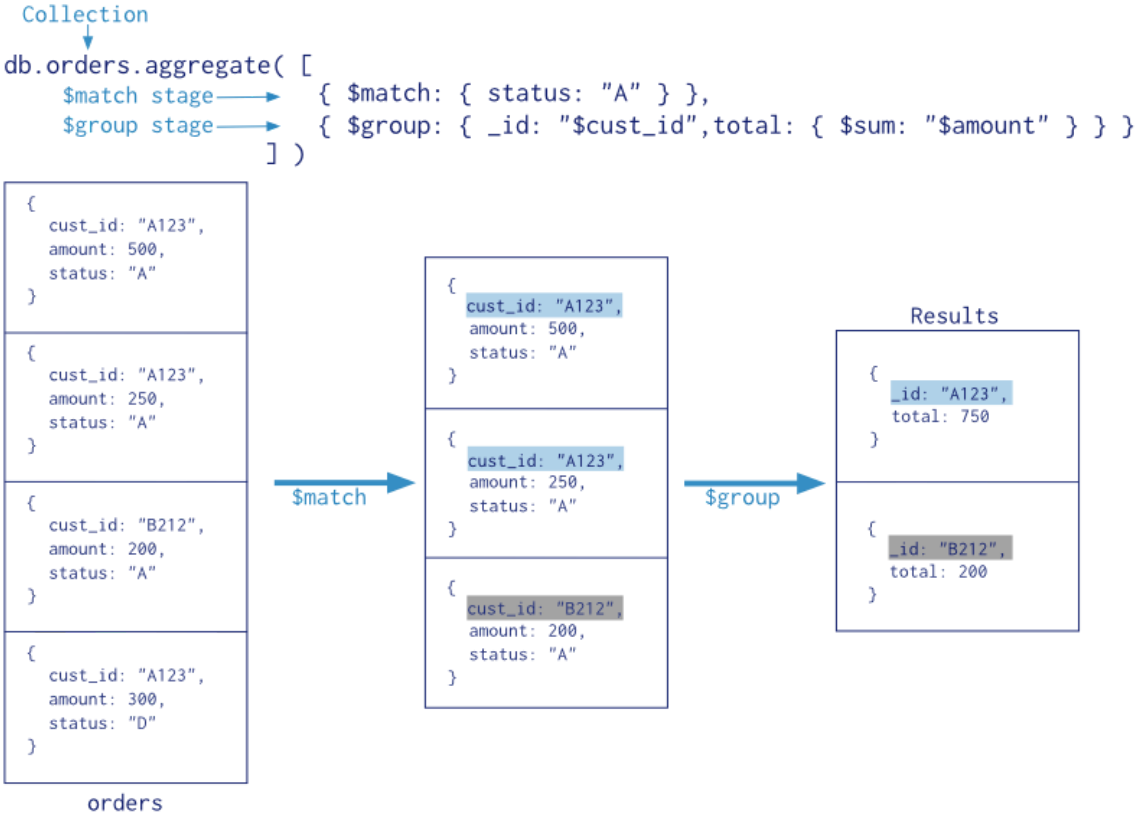
Net zoals op een relationele databank kan je indexen leggen op collecties.
Ze laten je toe om snel documenten te zoeken op basis van de waarde van een bepaalde property.
MongoDB heeft naar analogie ook een explain() commando waarmee je kan nagaan of een index ook effectief gebruikt wordt.

Opdracht:

- Geef het aantal films dat na 1990 is gemaakt en een award heeft gehaald (zie hoger)
- Roep explain aan en bestudeer de output.
- Leg een index op het veld Year in de films collectie.
- Roep explain aan en vergelijk de output.

Aggregation framework

Voor het uitvoeren van groeperingsacties op documenten ($sum, $group, $match,…) beschikt MongoDB over een uitgebreid aggregation framework.
Dit werkt met het concept van stages die in een pipeline achter elkaar worden uitgevoerd.
In onderstaand voorbeeld worden in de eerste stage alle orders gefilterd met de status “A”.
Vervolgens worden deze gegroepeerd per klant ($cust_id) met berekening van het totaal.



Naast $match en $group komen ook de commando’s $sort en $project (om een selectie te maken van velden) vaak voor in pipelines.
Je vindt gedetailleerde documentatie op http://docs.mongodb.org/manual/core/aggregation-pipeline/ en een aantal uitgewerkte voorbeelden op http://docs.mongodb.org/manual/applications/aggregation

Opdracht:

Produceer met behulp van de pipeline een nieuwe collectie ($out) waarin voor alle films gemaakt voor 1990 per subject (genre) een document zit met de naam van het genre en de gemiddelde populariteit. Deze collectie moet gesorteerd zijn volgens populariteit.

Opmerking

Naast het aggregation framework ondersteunt MongoDB ook Map-Reduce. Hiermee kan je vergelijkbare dingen doen, met nog meer opties dan het aggregation framework. Map-Reduce is in MongoDB momenteel wel trager omdat alle operaties in JS worden uitgevoerd tgv C++ voor het framework. We gaan hier niet verder op Map-Reduce in, dit gebeurt in het keuzevak Big Data.

Shards

MongoDB biedt uitgebreide support voor sharding. Om meer te weten hoe sharding werkt en hoe je een shard key moet kiezen, verwijzen we naar de slides rond NoSql_Praktisch.

Opdracht:
Shard de film collectie over 2 nodes op basis van een goed gekozen shard key. Voeg een aantal films toe en ga na in welke node deze terechtkomen.
De procedure die je dient te volgen, verloopt ongeveer als volgt. Deze is gebaseerd op http://docs.mongodb.org/manual/tutorial/deploy-shard-cluster. Je vindt daar meer details over sommige van de te gebruiken commando's.

- Stop eerste de mongod instantie die je voor de voorgaande oefeningen had opgestart. Je werkt best met een volledig gescheiden setup.

- Architectuur richtlijnen:
  - Zowel de shards, config server als de router (mongos) mogen allemaal op localhost gedraaid worden.
  - Aangezien het om een test setup gaat, volstaat 1 config server, één router en één node per shard i.p.v. telkens per 3.
  - Let op: sinds MongoDb 3.4 is het verplicht om je shard alsook je config server elks in een replica set te definiëren. Je kan echter een replica set met enkel een primary node definiëren en dat is voldoende voor deze opgave
  - Gebruik dus voor al je nodes andere poortnummers (--port argument telkens meegeven). Schrijf vooraf best even op welke poort nummers je gaat gebruiken voor de verschillende servers.
  - Denk er ook aan dat je een heel aantal terminal vensters open gaat moeten hebben staan.
  - Alle shard nodes alsook de config-server moeten een eigen data directory hebben die je zelf instelt (argument --dbpath).

- Shards aanmaken
  - Start voor elke shard een <install-dir>/bin/mongod instanties op. Elke shard moet als shardserver kunnen draaien, alsook de naam van zijn replica set waarvan hij deel uit maakt. Vergeet ook niet een eigen poortnummer en data directory mee te geven.
  - Stel voor elke shard via de mongo shell (mongo) de ReplicaSet in (TIP: rs.initialize)

- Config server aanmaken
  - Start vervolgens de config server, ook met een eigen data dir en poort. De config server zal de metadata van de shard cluster bevatten (op welke shards de chunks zitten).
  - Ook deze draait in een replica set, maar je hoeft enkel maar een primary node te voorzien

- Router opstarten
    - Start ten slotte mongos (de router) op zijn eigen poort.
    - De router heeft heeft geen data dir nodig want deze gaat zijn settings in de config server bijhouden. Dus zorg er wel voor dat de router je net opgestarte config server kan gebruiken

- Sharding instellen en data importeren
    - Open een mongo command prompt (install-dir>/bin/mongo commando) voor de mongos server
    - registreer de 2 shards via sh.addShard()
    - Zet de chunk size op 1MB of kleiner, anders moet je te veel film data importeren vooraleer de sharding uitgevoerd wordt.
        - use config (chunck size dient ingesteld te worden op de config database)
        - db.settings.save({ _id:"chunksize", value: 1 })
    - Maak de database ‘cinema’ aan en enable hierop sharding via sh.enableSharding(“cinema”)
    - Kies een shard key.
        - use cinema (de instellingen dienen te gebeuren op de cinema database)
        - Leg via mongos een index op de shard key (createIndex, zie hoger).
        - Enable vervolgens sharding op de collectie via sh.shardCollection(“cinema.films”, <de gekozen shard key>)
    - Importeer de films een aantal keer na elkaar (zeker 7x) in de cinema database in een collectie films via het <install-dir>/bin/mongoimport commando. Vergeet het poortnummer van de router (mongos) niet mee te geven als argument
    - Connecteer vanuit Robo 3T naar de 2 shards en naar mongos om te kunnen nagaan wat er gebeurt.
    - Ga na of de documenten verspreid worden over de verschillende shards. Importeer de films nog een aantal keer indien je vaststelt dat alle chuncks nog op 1 shard zouden zitten. (Het kan nodig zijn om meer dan 10 keer een import uit te voeren alvorens de shards effectief gebruikt worden. Blijf imports uitvoeren om te zien hoe de chunks verdeeld worden. Hou er rekening mee dat het herverdelen van de data niet onmiddellijk in orde is.)
      Via sh.status() kan je nagaan hoe de chunks verdeeld zijn. Interpreteer de output van dit commando.

Optioneel

Zet sharding op over je eigen laptop en deze van een medestudent in een lokaal netwerk.

Optionele opdrachten (verplicht voor teams van 3)

Replica Sets

Naast sharding zijn er ook voorziening voor high availability ingebouwd in MongoDB. Zie de NoSQL slides en http://docs.mongodb.org/manual/sharding/ voor meer info hierrond.

Opgave

Zet beide shards om in een 3-node replica set ipv een 1-node replica set. Breng eens de primary van één van de shard down en ga na of de cluster nog steeds functioneert en welke secondary nu tot primary is gepromoveerd. Zie http://docs.mongodb.org/manual/tutorial/deploy-replica-set-for-testing/ voor een duidelijke procedure.

Drivers

Voor alle grote talen en frameworks zijn drivers en integraties beschikbaar.
De Java driver is een jar file die je moet toevoegen aan het classpath in je IDE (bv. IntelliJ). Je kan deze driver downloaden vanaf http://mongodb.github.io/mongo-java-driver/ alsook volgende tutorial volgen: http://mongodb.github.io/mongo-java-driver/3.8/driver/tutorials/connect-to-mongodb/ . De huidige stabiele release van de Java driver is versie 3.8

Ook voor .NET kan je een driver downloaden en gebruiken om vanuit een .NET applicatie een connectie naar MongoDb te leggen. Meer info vind je op https://docs.mongodb.com/ecosystem/drivers/csharp/

Opgave

Gebruik de Java of .NET API (of andere) om een film toe te voegen aan de collectie films.

Succes!