## Veranderden van database waar je in het werken bent

use naamvandatabase

bv use test

als je dit commando intypt ga je werken met de database test als deze nog niet bestaat zal deze ook aangemaakt worden met dit commando.

## Tonen van alle databasen

show dbs

hier ga je enkel de databasen zien waar al reeds iets inzit

## Tonen van de huidige database

Db

## Tonen van alle collections

Show collections

## Aanmaken van een collection in een database

Omdat het big data is en dus niet als sql is kan elk record anders zijn dit collections die je moet aanmaken en daar een record in steken dit doe je door

db.createCollection(naamvancolletion)

bv

db.createCollection(“dbtest”)

dit word altijd gedaan in de huidige db waar je in aan het werken bent

# Insert

## Aanmaken van een record in een collection

Je kan dan ook een record invoegen in een collection dit kan je doen op veel verschillende manieren de eenvoudige manier bv:

db.studentenregatta.insert({\_id: '001', team: 'PXL', aantal\_likes:820, teamleden: 9})

Je kan best altijd een \_id meegeven omdat anders mongodb zelf een gegenereerde key aanmaken

## Als je meerdere records tegelijk wilt toevoegen kan je dit doen door [] te gebruiken bv:

db.studentenregatta.insert([{\_id: '002', team: 'UCLL', teamleden:8}, {\_id:'003', team 'Uhasselt', aantal\_likes: 400}])

# Update

## Een record aanpassen

Dit kan je op verschillende manieren opvatten of een naam van iets veranderen bv:

db.studentenregatta.update({team:'PXL'},{$set:{team: 'Hogeschool PXL'}})

## een record aanpassen met verschillende records

db.films.update({naam:'Iedreen beroemd'},{$set:{acteurs:['Josse De Pauw', 'Eva van Der Gucht','Werner De Smedt']}}})

## een record aanpassen met één extra waarde te willen toevoegen

als je een extra persoon bv wilt toevoegen aan acteurs moet je elke keer alle acteurs toevoegen omdat dit anders denkt dat die je niet schrijft weg moeten.

db.film.update({naam:'Loft'},$set:{acteurs:['Koen De bouw', 'Filip Peeters', 'Matthisas Schenaeerts']}})

## een record aanpappen met één extra waarde te willen toevoegen met $push

Als je dit niet wilt dit elke keer dit wilt typen kan je ook $push gebruiken dit commando zet een waarde bij een set.

Dus bv:

db.film.update({naam:’Loft’},{$push:{acteurs:’naam4’}})

zo kan je eenvoudig één extra record toevoegen ipv een hele lijst typen bij $set

## of aan een record een veld toevoegen. Bv:

db.studentenregatta.update({team:'Hogeschool PXL'}, {$set:{supporters:{personeel:50,studenten:150}}})

Hier word ook nog een subcategorie gemaakt van eerst supporters met daarin het aantal personeel supporters en het aantal studenten.

## Een record aanpassen waar een aan een voorwaarde voldaan is voor alle records

Volgende commando voegt iets toe waar de naam gelijk is aan “” en je moet multi op true zetten als je op alle records wilt.bv:

## db.films.update({director: 'Erik Van Looy'}, {$set:{genre: "thriller"}}, {multi:true})

## Een collection verwijderen

db.studentenregatta.drop()

## Tonen van de data in je collation

Als de data uit je collation wilt kijken kan je dit doen op twee manieren met find of met pretty extra bv:

db.films.find()

db.films.find()pretty

Dit tweede commando zal de data leesbaardere tonen dan gewoon .find()

## Tonen van bepaalde data

Als je niet alles wilt tonen kan je dit doen door de naam die je wilt gebruiken en deze op 1 te zetten LET OP als je enkel bv de naam wilt moet je de \_id ook wel op 0 zetten wat dit staat standaard altijd er bij. Bv:

db.movies.find({},{title:1,year:1,\_id:0})

## Sorteren

Je kan dit ook sorteren 1 = oplopend of alfabetisch en -1 is dalend of reverse alfabetisch dus bv:

db.movies.find({},{title:1,genres:1}).sort({genre:1})

## Skip en Limit

Dit zijn twee dingen die redelijke eenvoudig zijn. Skip skiped gewoon de zoveel eerste resultaten dus stel je wilt de 5 de dan kan je dit doen door skip van 4 en limit is gewoon laat maar een bepaald aantal resultaten zien dus als je enkel de 5de record wilt kan je dit doen door skip 4 en limit van 1 bv:

db.movies.find().sort({genre:1}).skip(82).limit(15)

## Count

Je kan natuurlijk ook je records die je bekomt gaan tellen met count bv:

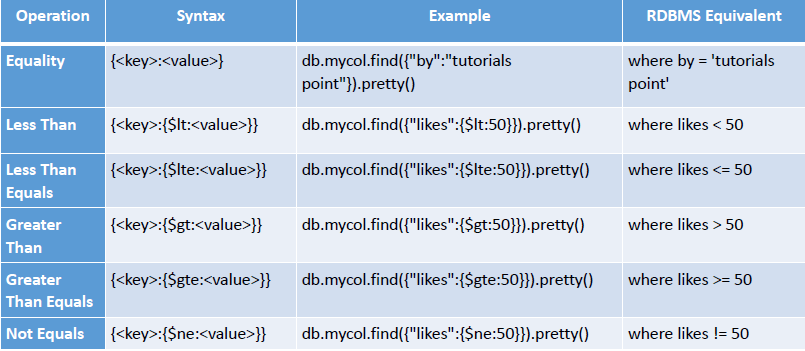
db.movieDetails.find().count()

of bv als je wilt weten hoeveel films er bv gemaakt zijn in een bepaald jaar

db.movies.find({year:1982}).count().pretty()

## Grotere dan, kleiner dan en niet gelijk aan

Je kan ook zoeken op groter dan kleiner dan, groter of gelijk aan, kleiner en gelijk aan en niet gelijk aan.



Bv:

db.movieDetails.find({"runtime":{$gt:120}}).count()

je kan natuurlijk ook werken met letters bij groter dan dus bv de titel begint met een letter tussen L en letter P dus:

db.movies.find( { $and: [ {title: {$gt: "L"} } ,  { title : { $lt : "Q"} }   ] } )

## Opzoeken op naam

Je kan natuurlijk ook zoeken op naam dit doe je door het keyword regex. Daarna zet je het woord dat je zoekt tussen /woord/.

Daarna zijn er nog 3 tekens dat je kan meegeven een ^,$ en i .

^ en $ zet je na de eerste /. ^ wilt zeggen dat het record moet beginnen met dit woord. En $ wilt zeggen dat dit moet eindigen met dit woord. i zet je na de tweede / dit zorgt er voor dat je zoekt op niet hoofdletter gevoelig dus zoekt die op zowel klein als groter letter van dit woord dus bv:

db.movieDetails.find({"actors":{$regex:/^Gary/i}},{title:1,\_id:0}).sort({title:1})

dus moet beginnen met Gary en toon ook als het begint met gary zonder hoofdletter.

Je kan natuurlijk ook perfect werken zonder regex bv:

db.movieDetails.find({'awards.text':/nominations/i},{title:1,'awards.nominations':1,'awards.text':1,\_id:0})

## AND

Als je wilt zoeken dat een waarde gelijk is aan zowel een eerste waarde als ook een tweede waarde kan je dit doen door AND. Dus bv jaar 2000 en actor = naam van actor. Maar je kan ook AND gebruiken als je wilt gaan kijken of het ligt tussen een bepaald jaar tal dan kan je dit doen door bv:

db.movies.find({$and:[{year:{$gte:2014}},{year:{lte:2017}}]},{title:1,year:1,\_id:0}).sort({year:1})

hier ga je de films nemen die liggen tussen 2014 en 2017 inclusief.

## OR

Als je wilt zoeken of het een is of het andere is kan je dit doen moet OR bv:

db.movies.find({$or:[{year:2000},{year:2010}]}).count()

## AND en OR

Je kan natuurlijk ook deze twee combineren in één grote lus. Waar je beide gebruikt.

Er zijn dan twee manieren of een AND met een OR of een OR met een AND.

Dus AND en OR

db.movies.find({title:{$regex:/^West/},$or:[{year:2008},{year:2012}]}).pretty()

Hier moet het dus de naam West zijn EN het jaar 2008 OF 2012.

AND moet niet geschreven worden omdat dit standaard is.

Dus OR en AND

db.movies.find({$or:[{$and:[{year:{$gte:1915}},{year:{$lte:1920}}]},{title:"Western"}]}).pretty()

Hier moet het dus of tussen 1915 en 1920 liggen of met den titel western zijn. Natuurlijk is dit een of dus mag het ook en de titel western omdat dit geen exclusieve of is.

## Subdocumenten

In big data zit je ook met subdocumeten hier kan je ook op gaan zoeken dan met je wel gebruik maken van ‘veldnaam’ als je dit wilt aanroepen bv:

db.movies.find({'awards.wins':20}},{title:1,\_id:0,'awards.nominations':1)

hier is awards een veld en wins zijn subdocument daar van .

zowel bij het zoeken als bij het tonen moet je ‘’ gebruiken.

## Arrays

Je kan zoeken altijd of een waarde in een veld zit dus stel je wilt weten of er in de array een bepaalde naam bevind kan je dit eenvoudig doen door een eenvoudige find.

Als je echt wilt weten op welke plaats dit bepaalde waarde moest staan dan kan je dit doen door een cijfer toe te voegen. Beginnend bij 0 voor de eerste plaats en 1 enzo verder.

Dus bv aantal films waar is opgenomen in Italië

db.movieDetails.find({countries: 'Italy'}).count()

als je wilt dat je enkel het tweede land Italië moet zijn moet je ‘countries.1’ doen dus:

db.movieDetails.find({'countries.1': 'Italy'}).count()

je kan ook bv hier $ne gebruiken dus bv zorg er voor dat het eerste land NIET USA is dus :

db.movieDetails.find({'countries.0':{$ne:'USA'}},{\_id:0,title:1,countries:1}).count()