MongoDB

De Praktijk

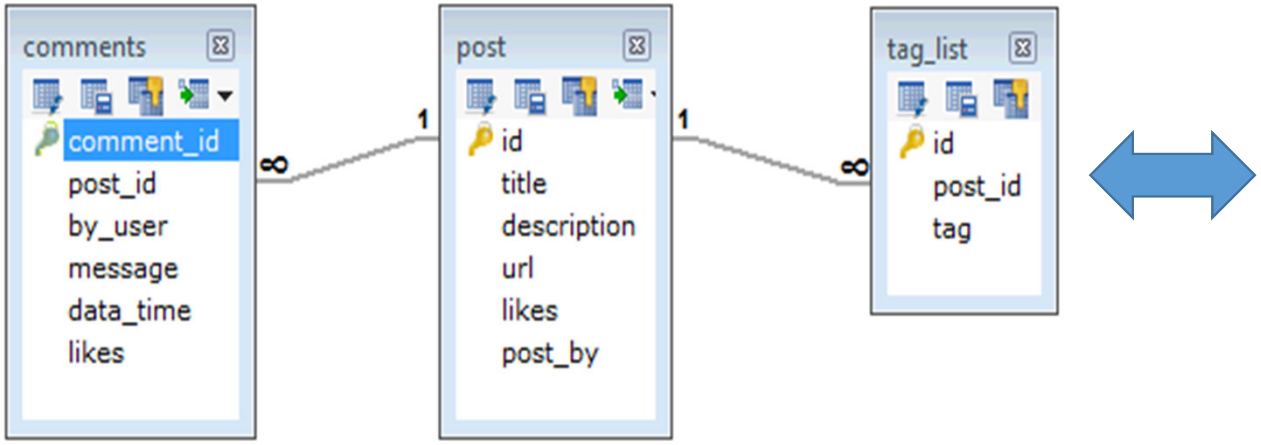
HOGESCHOOL PXL

MongoDB: Document based

- Documenten worden opgeslagen in collecties
- Collecties worden opgeslagen in DB
- Aantal velden, inhoud en grootte documenten kan verschillen
- Eenvoudig uitbreidbaar
- Geen joins, wel embedding
- Max. grootte document: 16MB → opletten bij embedding

- https://www.tutorialspoint.com/mongodb/mongodb_overview.htm
- https://docs.mongodb.com/manual/reference/operator/

RDBMS vs MongoDB




```
_id: POST_ID

title: TITLE_OF_POST,

description: POST_DESCRIPTION,

by: POST_BY,

url: URL_OF_POST,

tags: [TAG1, TAG2, TAG3],

likes: TOTAL_LIKES,

comments: [

   {

      user:'COMMENT_BY',

      message: TEXT,

      dateCreated: DATE_TIME,

      like: LIKES

   },

   {

      user:'COMMENT_BY',

      message: TEXT,

      dateCreated: DATE_TIME,

      like: LIKES

   }

]
```

Voorbeeld database

- Database: video
- Collectie: movies
- Collectie: movieDetails

Collectie movies


```
{
    "_id" : ObjectId("56918f5e24de1e0ce2dfcccf"),
    "title" : "Star Wars: Episode VI - Return of the Jedi",
    "year" : 1983,
    "imdb" : "tt0086190",
    "type" : "movie"
}
{
    "_id" : ObjectId("56918f5e24de1e0ce2dfccd0"),
    "title" : "Star Wars: Episode I - The Phantom Menace",
    "year" : 1999,
    "imdb" : "tt0120915",
    "type" : "movie"
}
{
    "_id" : ObjectId("56918f5e24de1e0ce2dfccd1"),
    "title" : "Star Wars: Episode III - Revenge of the Sith",
    "year" : 2005,
    "imdb" : "tt0121766",
    "type" : "movie"
}
{
    "_id" : ObjectId("56918f5e24de1e0ce2dfccd2"),
    "title" : "Star Trek",
    "year" : 2009,
    "imdb" : "tt0796366",
    "type" : "movie"
}
```


HOGESCHOOL PXL

Collectie movieDetails


```
        "_id" : ObjectId("569190ce24de1e0ce2dfcd6d"),
        "title" : "2001: A Space Odyssey",
        "year" : 1968,
        "rated" : "G",
        "released" : ISODate("1968-05-15T04:00:00Z"),
        "runtime" : 149,
        "countries" : [
                "USA",
                "UK"
        ],
        "genres" : [
                "Mystery",
                "Sci-Fi"
        ],
        "director" : "Stanley Kubrick",
        "writers" : [
                "Stanley Kubrick",
                "Arthur C. Clarke"
        ],
        "actors" : [
                "Keir Dullea",
                "Gary Lockwood",
                "William Sylvester",
                "Daniel Richter"
        ],
        "plot" : "Humanity finds a mysterious, obviously artificial object buried beneath the
Lunar surface and, with the intelligent computer H.A.L. 9000, sets off on a quest.",
        "poster" : "http://ia.media-imdb.com/images/M/MV5BNDYyMDgxNDQ5Nl5BMl5BanBnXkFtZTcwMjc1
DDg3OA@@._V1_SX300.jpg",
        "imdb" : {
                "id" : "tt0062622",
                "rating" : 8.3,
                "votes" : 396824
        },
        "tomato" : {
                "meter" : 96,
                "image" : "certified",
                "rating" : 9.2,
                "reviews" : 70,
                "fresh" : 67,
                "consensus" : "One of the most influential of all sci-fi films -- and one of t
he most controversial -- Stanley Kubrick's 2001 is a delicate, poetic meditation on the ingenu
ity -- and folly -- of mankind.",
                "userMeter" : 89,
                "userRating" : 3.8,
                "userReviews" : 294447
        },
        "metacritic" : 86,
        "awards" : {
                "wins" : 13,
                "nominations" : 7,
                "text" : "Won 1 Oscar. Another 13 wins & 7 nominations."
```

Data types

- String → tussen single of double quotes
- Integer – double → hoeft niet tussen quotes; decimale punt
- Boolean (true of false) → geen quotes
- Datum
  - Date() → toont huidige datum en tijd
  - new Date() → toont huidige ISOdatum en tijd
  - new Date(“yyyy-mm-dd”) → geeft ISODate met opgegeven datum
  - new Date("YYYY-mm-ddTHH:MM:ss") →geeft ISOdatumtijd in tijdzone client

Commando's - algemeen

use test	switch naar DB test; indien niet bestaande creatie DB test
show dbs	toon alle DB
db	toon huidige DB
db.dropDatabase()	drop huidige DB
db.CreateCollection(Name, options)	creatie collectie
show collections	toon alle collecties in de huidige DB
db.collectienaam.drop()	drop collectienaam
db.collectienaam.insert()	voegt een doc toe aan een collectie; indien collectie nog niet bestaat, creëert hij deze
db.collectienaam.update({SELECTION_CRITERIA}, {$set:{field1:{value1}}})	update het eerste document dat voldoet aan selectiecriteria van de collectie
db.collectienaam.update({SELECTION_CRITERIA}, {$push:{field1:{value1, ...}}})	voegt value1 toe aan array field1; array MOET BESTAAN
db.collectienaam.update(…, {multi:true})	update alle documenten die voldoen aan selectiecriteria van de collectie
db.collectienaam.save({_id:ObjectId(),NEW_DATA})	vervangt het document met opgegeven _id door de nieuwe data
db.collectienaam.remove({})	verwijdert alle documenten van de collectie
db.collectienaam.remove({DELETION_CRITERIA})	verwijdert alle documenten die voldoen aan selectiecriteria

Commando's – overzicht query

db.collectienaam.find()	geeft alle info weer vd collectie, alles achter elkaar
db.collectienaam.find().pretty()	geeft alle info weer vd collectie mooi met tabs
db.collectienaam.findOne()	geeft alle info weer van één document (het eerste)
db.collectienaam.find().sort({key:1})	sorteert oplopend op key (-1 voor dalende volgorde)
db.collectienaam.find().skip(getal)	geeft alle info na het overslaan van 'getal' documenten
db.collectienaam.find().limit(getal).pretty()	geeft alle info weer voor de eerste 'getal' documenten
db.collectienaam.find().count()	telt het aantal items
db.collectienaam.find({},{key:1})	PROJECTIE: geeft enkel alle waarden weer van key; deze is te kiezen
db.collectienaam.find({...}).pretty()	SELECTIE: geeft alle info weer vd collectie die voldoet aan de voorwaarde tussen {}
db.collectienaam.find({},{key:0})	geeft alle info weer behalve van de vermelde key

Projectie

- db.collectienaam.find({},{key:1})
- Geef welke velden je wil zien
- Meerdere velden mogelijk
- Volgorde weergave velden niet mogelijk !
- Voorbeeld:
  - db.movies.find({}, {title:1, year:1}).pretty()

  → geeft _id ook weer!!!
  - db.movies.find({}, {title:1, year:1, **_id:0**})

Sorteren - SORT

- db.collectienaam.find().sort({key:1})
- 1 → oplopend (alfabetisch)
- -1 → dalend
- Voorbeeld:
  - db.movies.find({}, {title:1, year:1, _id:0}).sort({year:1})

  → toont de titel en het jaar van de films gesorteerd volgens het jaar, de oudste films eerst

Beperken aantal documenten - LIMIT

- db.collectienaam.find().limit(getal)
- Voorbeeld:
  - db.movies.find({}, {title:1, year:1, _id:0}).limit(10)

  → toont de titel en het jaar van de eerste 10 films

Overslaan aantal documenten - SKIP

- db.collectienaam.find().skip(getal)
- Voorbeeld:
  - db.movies.find({}, {title:1, year:1, _id:0}).skip(10)

  → toont de titel en jaar van de films vanaf document 11

Volgorde SORT – LIMIT - SKIP

- db.collectienaam.find().sort({key:1}).skip(getal).limit(getal)
- MongoDB voert ALTIJD eerst de sort uit, vervolgens de skip en dan pas de limit; ongeacht de volgorde van syntax
- Voorbeeld:
  - db.movies.find({},{title:1, year:1, _id:0}). skip(9).sort({year:1}).limit(1)

  → toont de tiende titel en jaar van de gesorteerde films volgens jaartal

Aantal documenten tellen - COUNT

- db.collectienaam.count()
- Voorbeeld:
  - db.movies.count()

  → geeft het aantal films weer

Selectie

- db.collectienaam.find({...}).pretty()
- Voorbeeld:
  - db.movies.find({year:1982}).pretty()
- Strings tussen quotes (enkele of dubbele)
  - db.movieDetails.find({actors: "James Doohan"}).pretty()
  - db.movieDetails.find({actors: ‘James Doohan’}).pretty()

RDBMS WHERE clause → MongoDB

Operation	Syntax	Example	RDBMS Equivalent
Equality	{<key>:<value>}	db.mycol.find({"by":"tutorials point"}).pretty()	where by = 'tutorials point'
Less Than	{<key>:{$lt:<value>}}	db.mycol.find({"likes":{$lt:50}}).pretty()	where likes < 50
Less Than Equals	{<key>:{$lte:<value>}}	db.mycol.find({"likes":{$lte:50}}).pretty()	where likes <= 50
Greater Than	{<key>:{$gt:<value>}}	db.mycol.find({"likes":{$gt:50}}).pretty()	where likes > 50
Greater Than Equals	{<key>:{$gte:<value>}}	db.mycol.find({"likes":{$gte:50}}).pretty()	where likes >= 50
Not Equals	{<key>:{$ne:<value>}}	db.mycol.find({"likes":{$ne:50}}).pretty()	where likes != 50

Selectie met *in* of *nin* of *exists*

- Kolom gelijk of niet gelijk aan meerdere waarden
  - Voorbeeld:
    - db.movieDetails.find({year:{$in:[2012,2015]}}).pretty()

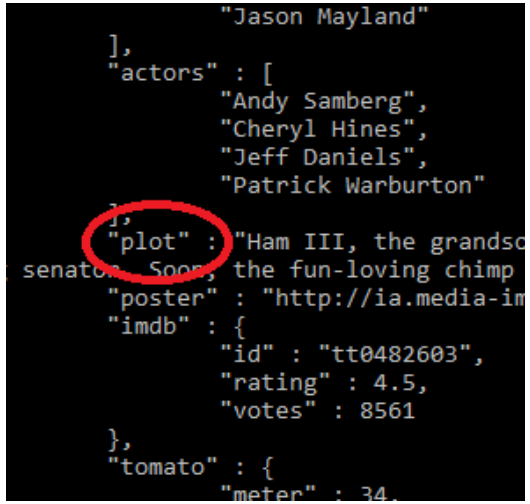
    → Geeft de films weer gemaakt in 2012 of gemaakt in 2015

    - db.movieDetails.find({year:{$nin:[2012,2015]}}).pretty()

    →Geeft de films weer NIET gemaakt in 2012 of 2015

- Bestaat er een waarde voor een kolom → exists
  - Voorbeeld
    - db.movieDetails.find({plot:{$exists:true}}).pretty()

    → geeft de films weer waarvoor er een plot staat beschreven



HOGESCHOOL PXL

Zoeken op tekst

- db.movies.find({title: /West/}, {title:1, _id:0}).pretty()

→ alle films waarin het woord West voorkomt

- db.movies.find({title: /^West/i}, {title:1, _id:0}).pretty()

→alle films die beginnen met West of west

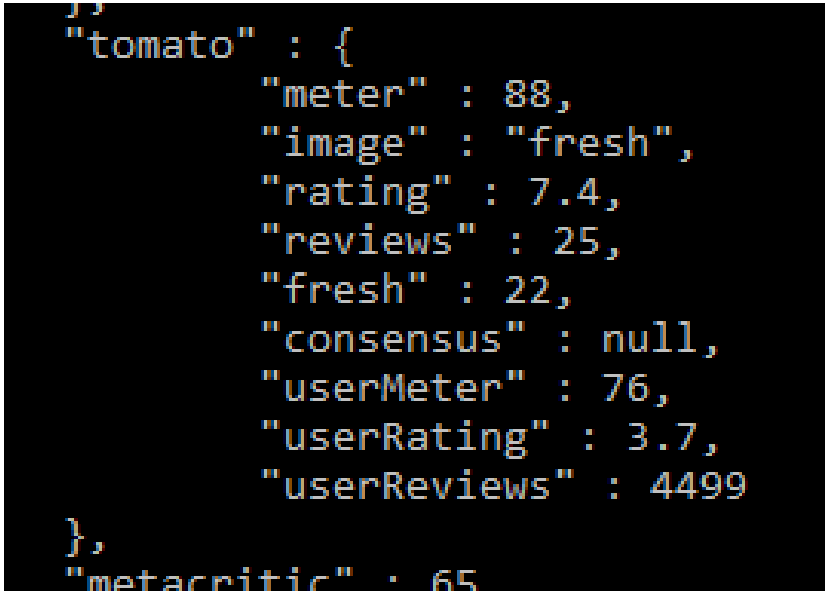
- tekst moet tussen / staan
- ^ vooraan → moet beginnen met
- $ achteraan → moet eindigen met
- i → maakt de zoekstring case insensitive

Subdocumenten

- → document embedded in een document
- Fields staan gegroepeerd tussen { }
- Aanroepen met punt en tussen ‘ ‘ of “ “
- Voorbeeld:

db.movieDetails.find({'tomato.meter' : 88}).pretty()

→ Alle films die 88 hebben op de tomato meter.



Arrays

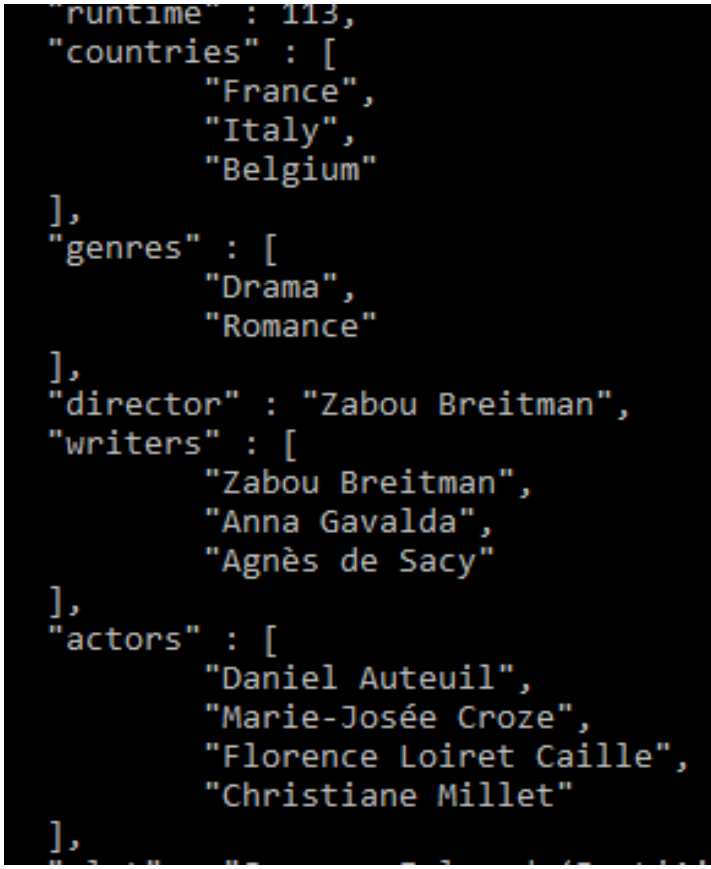
- → oplijsting van waarden
- waarden staan gegroepeerd tussen [ ]
- Aanroepen:
  - db.movieDetails.find({countries: 'Italy'}).pretty()

  → alle films die ook in Italy opgenomen zijn.
  - db.movieDetails.find({genres:['Drama','Romance']}).pretty()

  → alle films die als genre exact de 2 elementen Drama en Romance hebben.
  - Db.movieDetails.find({genres:{$all:[‘Romance’,’Drama’]}}).pretty()

  →alle films die als genre de elementen Romance en Drama hebben, ongeacht de volgorde of de aanwezigheid van nog andere genres.
- Aanroepen positie:
  - db.movieDetails.find({'countries.1': 'Italy'}).pretty()

  → alle films die als tweede filmlocatie het land Italy hebben.



Meerdere voorwaarden: AND

- db.collectienaam.find({$and:[{key1: value1}, {key2: value2}]}).pretty()
- Voorbeeld:
  - db.movies.find({$and:[{title:/^West/},{year:2012}]}).pretty()

  → alle films die beginnen met West en in 2012 uitgebracht zijn

  → 1 film
- AND is default bij meerdere selecties
  - db.movies.find({title:/^West/,year:2012}).pretty()
- Meerdere voorwaarden op zelfde key
  - Db.movieDetails.find({$and:[{metacritic:{$ne:null}}, {metacritic:{$exists:true}}]}, {_id:0, title:1, metacritic:1})

Eén van de voorwaarden: OR

- db.collectienaam.find({$or:[{key1: value1}, {key2: value2}]}).pretty()
- Voorbeeld:
  - db.movies.find({$or:[{title:/^West/},{year:2012}]}).pretty()

  → alle films die beginnen met West en/of in 2012 uitgebracht zijn

  → 151 films

AND en OR samen

- db.movieDetails.find({title:/^West/,$or:[{'tomato.meter':{$gte:80}},
  {year:2012}]}).pretty()

→alle films die beginnen met West EN
die minstens 80 hebben op de tomato meter of in 2012 zijn uitgebracht.

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

→ alle films die tussen 1915 en 1920 (inclusief) zijn uitgebracht OF/EN als titel Western hebben.