

Data Advanced

MongoDB

De Praktijk



**DE HOGESCHOOL
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be



Voor we starten, een klein beetje achtergrond...

RDBMS vs MongoDB - What's The Difference



C1	C2	C3	C4
—	—	—	—
—	—	—	—
—	—	—	—
—	—	—	—

Relational data model



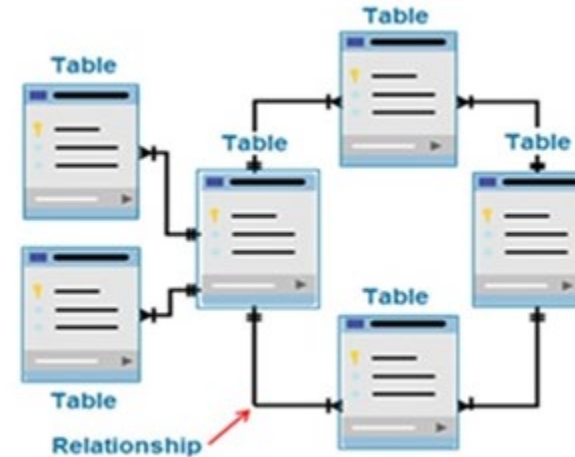
Document data model

RDBMS =
Relationeel database management system

MongoDB = NoSQL databank
er zijn 4 types NoSQL databanken, waarvan
document based er eentje is

Relationele databank versus NoSQL databank

- **Relationele databank** (Vb Oracle, MySQL, SQL Server,...)
 - Genormaliseerde data → meerdere tabellen met relaties (foreign keys)
 - Gestructureerde data: elk record heeft dezelfde structuur
- **NoSQL databank** (Vb MongoDB, CouchDB, Cassandra,...)
 - Data zit (meestal) in 1 groep gegevens
 - Ongestructureerde data: elk 'lijn' heeft een eigen structuur



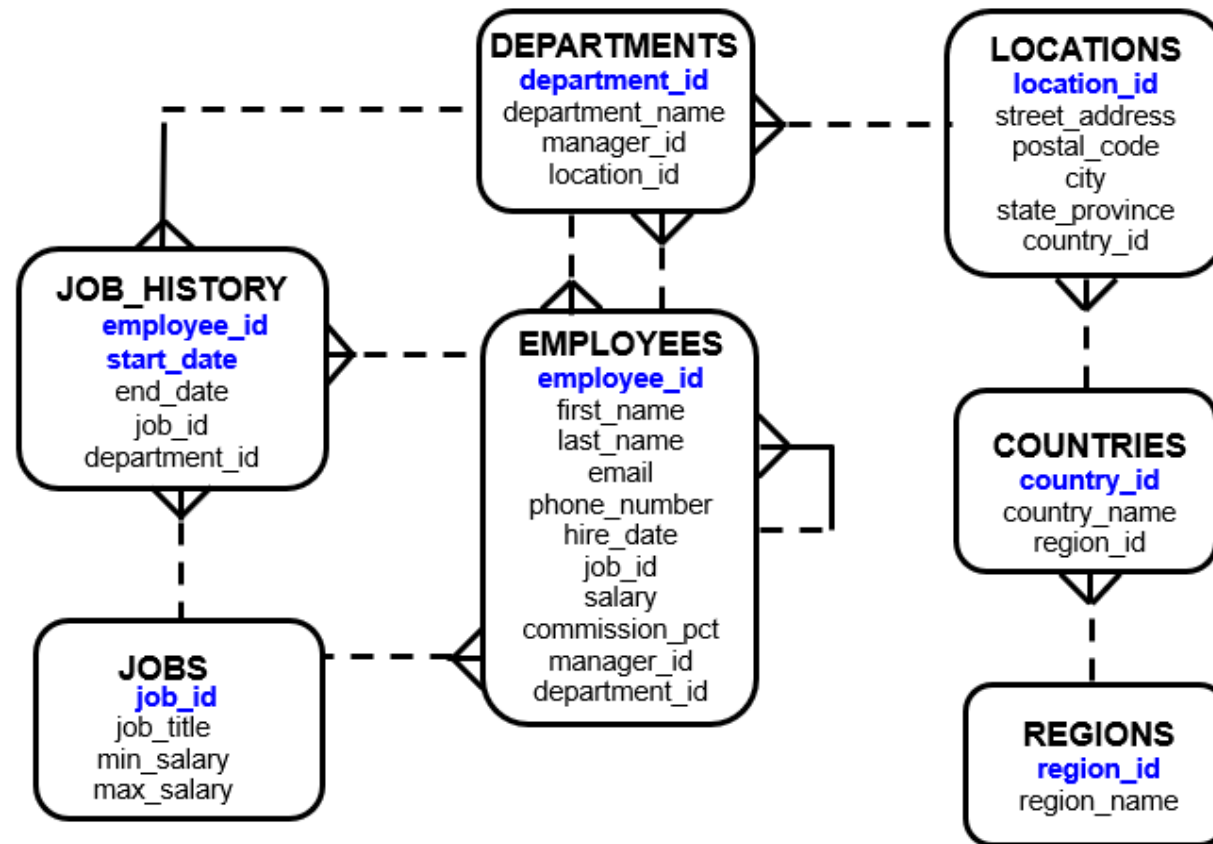
RDBMS

Relational Databases

NOSQL
Not Only

Relationele databank voorbeeld **ORACLE®**

- Genormaliseerde data → meerdere tabellen met relaties (foreign keys)



- Gestructureerde data: elk record heeft dezelfde structuur

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	(null)	1700

NoSQL databank voorbeeld



- Data zit (meestal) in 1 groep
 - Dit noemen we een collection
- Ongestructureerde data: elk 'lijn' heeft een eigen structuur
 - MongoDB = document based databank
 - Een lijn komt overeen met een document

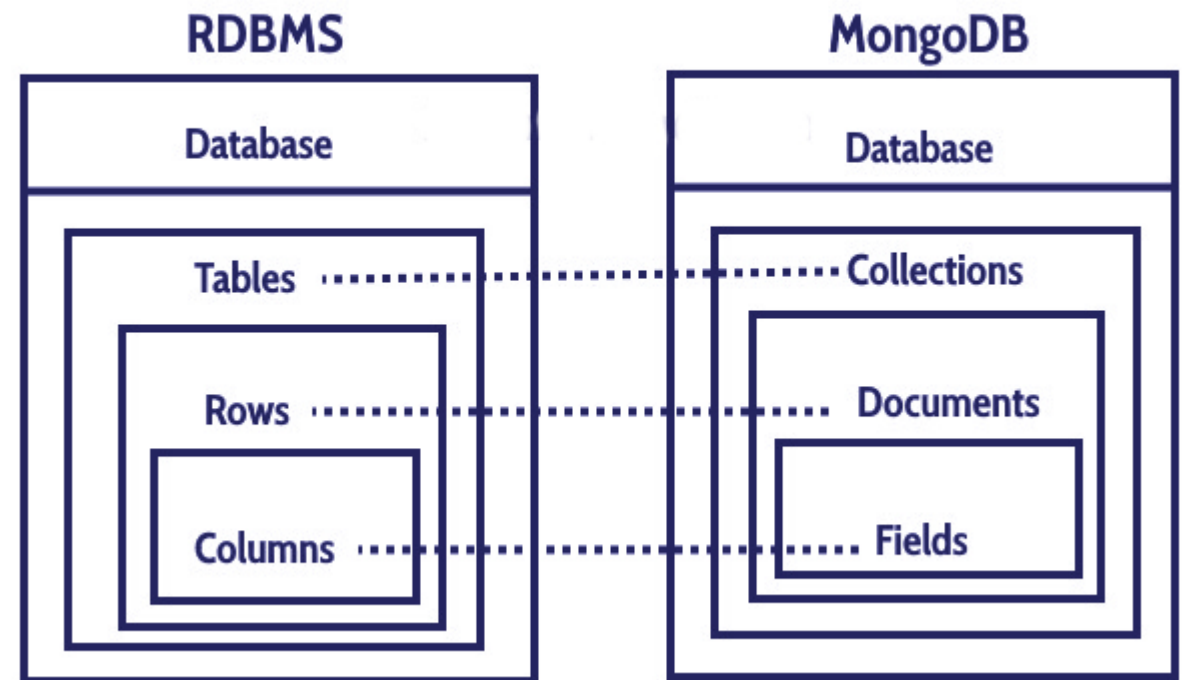
➤ Vb collection pokemon: 3 documents

- { "_id" : ObjectId("5ad885fb5875726cba326ea0"), "Name" : "Pikachu", "Type" : "electric", "Weakness" : "ground" }
- { "_id" : "004", "Name" : "Charmander", "Type" : "fire" }
- { "_id" : "005", "Name" : "Charmeleon", "Type" : "fire", "Sort" : "flame", "Weakness" : ["ground", "rock", "water"] }

Een beetje terminologie

Definitions

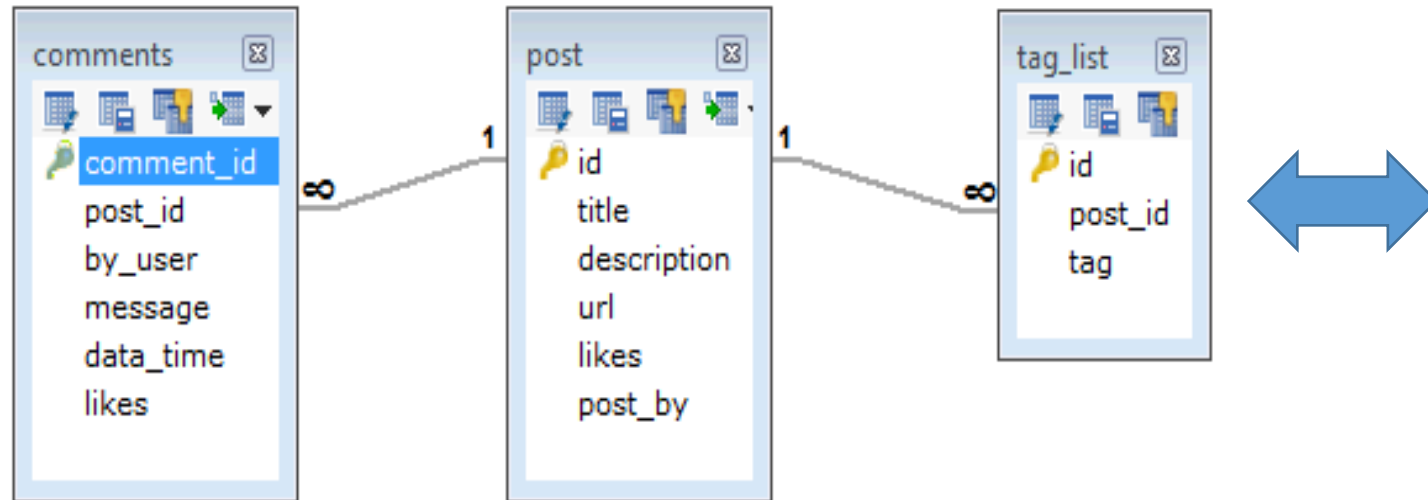
RDBMS	MongoDB
Table	Collection
Row	Document
Column	Field
Index	Index
JOIN	Embedded Document or Reference



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://www.mongodb.com/docs/manual/>

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"
```

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/MV5BNBYyMDgxNDQ5N15BM15BanBnXkFtZTcwMjc1ODg3OA@@._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 the most controversial -- Stanley Kubrick's 2001 is a delicate, poetic meditation on the ingenuity -- 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.insertOne(); db.collectienaam.insertMany()	voegt een doc toe aan een collectie; indien collectie nog niet bestaat, creëert hij deze
db.collectienaam.updateOne({SELECTION_CRITERIA}, {\$set:{field1:value1}})	update het eerste document dat voldoet aan selectiecriteria van de collectie
db.collectienaam.updateMany({SELECTION_CRITERIA}, {\$set:{field1:value1}})	update alle documenten die voldoen aan selectiecriteria van de collectie
db.collectienaam.updateOne({SELECTION_CRITERIA}, {\$unset:{field1:value1}})	maakt een set ongedaan
db.collectienaam.updateOne({SELECTION_CRITERIA}, {\$push:{field1:value1}})	voegt value1 toe aan array field1; array MOET BESTAAN
db.collectienaam.updateOne({SELECTION_CRITERIA}, {\$push:{field1:{\$each:['value1','value2', ...]}}})	voegt value1, value2, ... toe aan array field1; array MOET BESTAAN
db.collectienaam.updateOne({SELECTION_CRITERIA}, {\$pull:{field1:value1}})	verwijdert value1 uit array field1
db.collectienaam.deleteOne({}); db.collectienaam.deleteMany({})	verwijdert documenten van de collectie
db.collectienaam.deleteOne({DELETION_CRITERIA}); db.collectienaam.deleteMany({DELETION_CRITERIA})	verwijdert documenten die voldoen aan selectiecriteria
db.collectienaam.save({_id:ObjectId(),NEW_DATA})	vervangt het document met opgegeven _id door de nieuwe data

Commando's – overzicht query

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

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})`
 - → 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.countDocuments()`

Voorbeeld:

`db.movies.countDocuments()`

→ geeft het aantal films weer

- `db.collectienaam.find({kolom: waarde}).count()`

Voorbeeld:

`db.movieDetails.find({year: 2000}).count()`

→ geeft het aantal films weer die in 2000 zijn verschenen

Selectie

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

RDBMS WHERE clause → MongoDB

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

db.movieDetails.find({runtime:{\$gt:120}}, {title:1, runtime:1, _id:0})

db.movieDetails.find({runtime:{\$gt:120,\$lt:180}}, {title:1, runtime:1, _id:0})

Zoeken op tekst

- `db.movies.find({title: /West/}, {title:1, _id:0})`
→ alle films waarin het woord West voorkomt
- `db.movies.find({title: /^West/i}, {title:1, _id:0})`
→ 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

Oefeningen Query: basis

Opgave 1 - 17

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

- Kolom gelijk of niet gelijk aan meerdere waarden

Voorbeeld:

```
db.movieDetails.find({year:{$in:[2012,2015]}})
```

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

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

→ 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}})
```

→ geeft de films weer waarvoor er iets ingevuld staat bij plot

```
      "Jason Mayland"
    ],
    "actors" : [
      "Andy Samberg",
      "Cheryl Hines",
      "Jeff Daniels",
      "Patrick Warburton"
    ],
    "plot" : "Ham III, the grandso
senator. Soon, the fun-loving chimp
    "poster" : "http://ia.media-im
    "imdb" : {
      "id" : "tt0482603",
      "rating" : 4.5,
      "votes" : 8561
    },
    "tomato" : {
      "meter" : 34
```

Subdocumenten

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

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

→ Alle films die 88 hebben op de tomato meter.

```
    "tomato" : {  
      "meter" : 88,  
      "image" : "fresh",  
      "rating" : 7.4,  
      "reviews" : 25,  
      "fresh" : 22,  
      "consensus" : null,  
      "userMeter" : 76,  
      "userRating" : 3.7,  
      "userReviews" : 4499  
    },  
    "metacritic" : 65
```


Arrays

- → olijsting van waarden
- waarden staan gegroepeerd tussen []
- `db.movieDetails.find({countries: 'Italy'})`
→ alle films die ook in Italy opgenomen zijn.
- `db.movieDetails.find({genres:['Drama','Romance']}, {_id: 0, title: 1, genres:1})`
→ alle films die als genre **exact** de 2 elementen Drama en Romance hebben.
- `db.movieDetails.find({genres:{$all:['Romance','Drama']}})`
→ alle films die als genre de elementen Romance en Drama bevatten, ongeacht de volgorde of de aanwezigheid van nog andere genres.

```
{  
  "title" : "Love Actually",  
  "countries" : [  
    "UK",  
    "USA",  
    "France"  
  ],  
  "genres" : [  
    "Comedy",  
    "Drama",  
    "Romance"  
  ]  
}
```

Arrays

- `db.movieDetails.find({'countries.2': 'France'})`
→ alle films die als derde filmlocatie het land France hebben
(teller start met 0)
- `db.movieDetails.find({countries:{$size:1}})`
→ alle films die maar op 1 locatie opgenomen werden
→ Size telt aantal in de array

```
    "title" : "Love Actually",  
    "countries" : [  
      "UK",  
      "USA",  
      "France"  
    ],  
    "genres" : [  
      "Comedy",  
      "Drama",  
      "Romance"  
    ]  
  ]  
}
```

Meerdere voorwaarden: and

- `db.collectienaam.find({$and:[{key1: value1}, {key2: value2}]})`
- Voorbeeld:
 `db.movies.find({$and:[{title:/^West/},{year:2012}]})`
 → 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})`
- Meerdere voorwaarden op zelfde key → AND verplicht
 `db.movieDetails.find({$and:[{year:{$gte:2012}},{year:{$lte:2015}}]}).count()`
 `db.movieDetails.find({$and:[{title:/once/i},{title:/west/i}]})`

Eén van de voorwaarden: or

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

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

- `db.movieDetails.find({$or:[{$and:[{title:/once/i},{title:/west/i}]},{year:2015}]},
{title: 1, _id: 0, year:1})`

→ alle films die zowel de woorden “once” als “west” case insensitive in de titel hebben EN alle films van het jaar 2015

Null-waarden

```
"_id" : ObjectId("5692a14...")
"title" : "A Cruel Romanc...
"year" : 1984,
"rated" : null,
"released" : ISODate("198...")
"runtime" : 142,
"countries" : [
```

- `db.movieDetails.find({'tomato.consensus': null}, {title:1, _id:0, 'tomato.consensus':1})`
 - geeft weer als er null staat bij de tomato consensus of als dat veld niet bestaat
- `db.movieDetails.find({rated:null}).count() → 1599`
- `db.movieDetails.find({rated:{$exists:true}}).count() → 2295`
- `db.movieDetails.find({rated:{$exists:true}, title:"Ma vie en l'air"})`
 - bij rated staat het woordje null
 - bij exists kan MongoDB null-waarden niet aan; kijkt enkel of er iets staat
- Eenvoudiger: `db.movieDetails.find({rated:{$ne:null}}).count() → 696`

Oefeningen Query: uitbreiding

Opgave 18 - 44