MongoDB gebruiken in mongosh

Documenten en collecties

- Een document = als een rij uit een sql-db maar veel flexibeler
- We voegen ons document toe aan een collectie = als een tabel in een sql-db
- _id-veld is verplicht aanwezig in elk mongo document. Je laat best mongo eentje aanmaken
- Bevat een geëncodeerde datetime: gemakkelijk om te sorteren.

Documenten aanmaken

```
doc = {
    "title": "Chicken Tacos",
    "description": "Classic Mexican tacos",
    "cook_time": 20
};
db.tacos.insertOne(doc);
```

```
db.tacos.find();
[
  {
    _id: ObjectId('657c920f68ad34eb8c3dcad0'),
    title: 'Chicken Tacos',
    description: 'Classic Mexican tacos',
    cook_time: 20
  }
]
```

Documenten opvragen

- = select statement in SQL
- https://www.mongodb.com/docs/manual/tutorial/query-documents/
- https://www.mongodb.com/docs/manual/reference/method/db.collection.find/

```
db.recipes.find();
db.recipes.find({"title": "tacos"});
db.recipes.find({"title": "tacos"}).pretty();
db.recipes.find({"title": "tacos", "cook_time": 20});
db.recipes.find({"title": "tacos"}, {"title": 1}); // enkel t
db.recipes.find({"title": "tacos"}, {"title": 0}); // alles b
db.recipes.find({}, {"title": 1}); // alle documenten, enkel
db.recipes.find({"title": { $regex: /taco/i}}, {"title": 1});
```

Wat kunnen we opslaan in een document?

```
doc = {
  "title": "Apple Pie",
  "directions": [
    "Roll the pie crust",
    "Make the filling",
    "Bake"
  ],
```

```
  "ingredients": [
     {
       "amount": {
         "quantity": 2,
           "unit": null
         },
        "name": "pie crusts"
    }, {
       "amount": {
         "quantity": 1,
           "unit": "tbsp"
        },
        "name": "cinnamon"
    }
  ]
};
```

Collecties

Met collecties houden we gerelateerde documenten samen: eenvoudig te organiseren en zorgt voor betere prestaties.

```
db.recipes.find({}, {"title": 1}); // geeft alle documenten i
show dbs; // alle db in onze mongodb instance
show collections; // alle collecties in onze huidige db
db.getName(); // huidige db
db.cool_newStuff.insertOne({}); // nieuwe collectie aanmaken
```

Queries schrijven

Sort, limit, skip

```
db.recipes.find().count()
db.recipes.find({}, {"title": 1}).limit(2); // 2 eerste resul
db.recipes.find({}, {"title": 1}).sort({"title": 1}); // 1=AS
```

```
db.recipes.find({}, {"title": 1}).sort({"title": 1}).skip(1);
db.recipes.find({}, {"title": 1}).sort({"title": 1}).skip(1).
```

Operatoren en arrays

Vergelijkingsoperatoren: $gt (groter dan), $lt (kleiner dan), $lte (kleiner dan of gelijk aan)

```
db.recipes.find({"cook_time": {$lte: 30}}, {"title": 1});
db.recipes.find({"cook_time": {$lte: 30}, "prep_time": {$lte:
db.recipes.find({$or: [{"cook_time": {$lte: 30}}, {"prep_time
    {"title": 1});
```

Wat met arrays in objecten?

```
db.recipes.find({ "tags": "easy"}, {"title": 1, "tags": 1});
db.recipes.find({ "tags": { $in: ["easy", "quick]}}, {"title"
db.recipes.find({ "ingredients.name": "egg"}, {"title": 1};
```

Documenten updaten

$set, $unset en $inc

```
db.examples.find({}, {"title": 1});
db.examples.updateOne({"title": "Pizza"}, {$set: {"title": "T
db.examples.updateOne({"title": "Thin crust pizza"}, {$set: {
db.examples.updateOne({"title": "Thin crust pizza"}, {$unset:
db.examples.updateOne({"title": "Tacos"}, {$inc: {"likes_coun
db.examples.updateOne({"title": "Tacos"}, {$inc: {"likes_coun
```

Arrays updaten

```
db.examples.updateOne({"title": "Tacos"}, {$push: {"likes": 6
db.examples.updateOne({"title": "Tacos"}, {$pull: {"likes": 6
```

Documenten verwijderen

```
db.examples.deleteOne({_id: ObjectId('657c920f68ad34eb8c3dcad
```

ps: een database verwijderen doe je met db.dropDatabase().

Data en schema modeling

Document model = anders dan bij SQL-data met hun one-to-many en many-to-many relaties gaan we hier zoveel mogelijk data samen stockeren in een document. Door samen op te slaan vereenvoudigen we de zaken en reduceren we het aantal queries.

Soms hebben references meer nut. Data die bvb zelden moet gelezen of geschreven worden op dezelfde tijd als de andere data in ons document. Denk in dat geval aan unique indexen in beide documenten (= one-to-one relaties)

One-to-many: bvb recepten en comments. Indien het er heel veel zijn, laat de UX zijn ding doen en sla bvb top comments op in het recepten document.

Indexen

“Index early and often” is een waarheid hier. Indexen ondersteunen queries. Veel opties op indexen, zoals unique, compound , of geospatial indexen. Belangrijke is de queries die je maakt in de gaten te houden en de frequente queries te ondersteunen met indexen.

```
db.recipes.find({"cook_time": 10}, {title, 1});
db.recipes.find({"cook_time": 10}, {title, 1}).explain("execu
// totalDocsExamined = 1 indien er een index is
db.recipes.find({"title": "Tacos"}, {title, 1}).explain("exec
db.recipes.getIndexes();
db.recipes.createIndex({"cook_time": -1});
db.recipes.dropIndex("cook_time_-1");
```

Verschillende collectie-types

Capped collecties: zetten een cap op hoeveelheid data of aantal documenten. Eens de cap bereikt worden documenten verwijderd. Volgorde van toevoegen documenten wordt hier gehanteerd. Interessant voor bvb log-gegevens. FIFO!

```
db.createCollection("error_log", {capped: true, size: 10000,
```

Time series collecties: bvb aandelen data. bvb documenten met 1 waarde die niet verandert, bvb symbool aandeel, en dan andere waarden die wel veranderen, zoals waarde vh aandeel.