



UNIVERSITE DE PARIS 8 SAINT-DENIS

UFR STN – MASTER 1 MATHEMATIQUES

PARCOURS CYBER SECURITE ET SCIENCES DES DONNEES

BASES DE DONNEES REPARTIES

RAPPORT TRAVAUX PRATIQUES SUR MONGODB

ENCADRE PAR L. BOUBCHIR

OCT 2020

**RALAINARIVO
N. T. THIERRY**

Contents

1	Introduction	2
2	Tutoriel pour installer MongoDB	2
3	PARTIE 1 - TP	5
3.1	Choix d'une base de données: LIBRARY	5
3.1.1	Création de document 1	5
3.1.2	Création de document 2	6
3.2	LES REQUÊTES	6
3.3	FONCTIONS: SORT, LIMIT ET SKIP	9
3.4	AGGREGATION: QUE FAIT CES COMMANDES?	10
3.5	DISTINCT(): QUE FAIT CES COMMANDES	11
3.5.1	Add a new record	11
3.5.2	Add more records	12
3.5.3	Insert with JavaScript	12
3.6	COMPARISON OPERATORS	13
3.6.1	Les opérateurs \$gt, \$lt, \$gte, \$lte, \$ne, \$in, \$nin (resp.>, <, >=, <=, !=, IN, NOT IN)	13
3.6.2	L'opérateur \$or	15
3.7	Opérateur: \$SLICE	15
3.8	Opérateurs: \$size et \$exists	16
3.9	INDEX CREATION	19
3.9.1	Ascending index	19
3.9.2	Descending index	19
3.9.3	Index for embedded objects	20
3.9.4	Force the index usage: hint()	20
3.10	DATA UPDATE	22
3.10.1	Update (condition, newObject, upsert, multi)	22
3.10.2	Add/delete an attribute	22
3.10.3	Delete	23
4	Partie 2: Base de données géographique	24
4.1	Import data	24
4.2	Quelques requêtes de base:	24
4.3	Modification des données	25
4.4	Nettoyage des données (DATA CLEANING)	27
4.5	Nettoyage des données (DATA CLEANING)	28
4.6	Requêtes	28
4.7	Indexation géographique	29
4.8	Index creation	30
4.9	Requêtes	30
4.10	Aggregate	30
4.11	Aggrégate: examples 1	31
4.11.1	Séquence:	31
4.11.2	\$unwind	31
4.12	Aggregate: examples 2	32
4.13	Administration	33
4.13.1	Backup	33
4.13.2	Restore	33
4.14	Security	34
4.14.1	Authentification	34
4.15	SECURITY: USER, PRIVILEGE,RESOURCE	36
5	Conclusion	37

Rapport Travaux Pratiques MongoDB

EC encadré par Mr Boubchir

Oct 2020

1 Introduction

Mongodb est une base de données distribuée non relationnelle qu'on désigne par NoSQL. Il permet de gérer et d'enregistrer les données sous forme de documents (ou table) en format JSON. A la différence du SQL, les tables NoSQL n'imposent pas de modèles de données strictes, il enregistre et organise des documents qu'il considère comme des collections d'objets. C'est ce qui le rend plus rapide qu'un système de gestion de base de données (SGBD) SQL relationnel qui fixe à l'avance la structure et les types de données avant d'implémenter la requête.

Voyons ensemble à travers ces travaux pratiques, les différentes requêtes de base et les fonctionnalités de MongoDB, pour essayer d'en tirer les avantages et les inconvénients. Mais quels sont alors ses spécificités et ses limites?

Nous allons voir comment faire pour installer MongoDB à travers un tutoriel, puis faire la partie 1 du TP pour apprendre les requêtes de bases et enfin nous allons appliquer les commandes de bases sur des données géospatiales.

2 Tutoriel pour installer MongoDB

1. Go on the following link: <https://www.mongodb.com>
2. Puis aller sur "Software" et "sur Community Server". Select the MongoDB version. If the one you're looking for is not in the list, go to all version binaries on the right menu.
3. Select your OS version
4. Select TGZ in Package pour Linux.
5. Click on Download

Extract it in a folder. We'll call the path to Mongodb folder: /path/to/mongo/

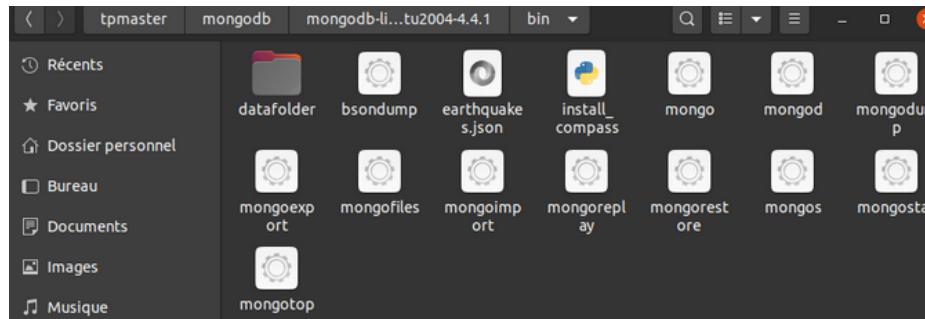
Launch Mongo Server: In a terminal (under Mac ou Linux), run these commands:

The screenshot shows the MongoDB website's download page for the Enterprise Server. At the top, there is a navigation bar with links for Cloud, Software, Pricing, Learn, Solutions, Docs, a search icon, Contact, Sign In, and a prominent green "Try Free" button. Below the navigation is a dropdown menu currently set to "MongoDB Enterprise Server". The main content area has two columns. The left column contains a heading "MongoDB Community Server" and a paragraph explaining that MongoDB offers both an Enterprise and Community version of its powerful distributed document database. It highlights the community version's flexible document model, ad hoc queries, indexing, and real-time aggregation. The right column is titled "Available Downloads" and includes dropdown menus for "Version" (set to 4.4.1 (current)), "Platform" (set to Ubuntu 20.04), and "Package" (set to tar). A large green "Download" button is located at the bottom of this section. To the right of the download section is a sidebar with links to "Current releases & packages", "Development releases", "Archived releases", "Changelog", and "Release Notes".

```

cd /path/to/mongo/
cd bin
mkdir dataFolder

```



On windows, use mongod.exe instead ./mongod

6. Launch the server:

```
./mongod --dbpath dataFolder
```

```

thierry@thierry-Inspiron-3793:~/Bureau/tpmaster/mongodb/mongodb-linux-x86_64-ubuntu
ntu2004-4.4.1/bin$ ./mongod --dbpath datafolder
{"t":{"$date":"2020-10-27T15:12:46.026+01:00"},"s":"I", "c":"CONTROL", "id":23
285, "ctx":"main","msg":"Automatically disabling TLS 1.0, to force-enable TLS
1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2020-10-27T15:12:46.028+01:00"},"s":"W", "c":"ASIO", "id":22
601, "ctx":"main","msg":"No TransportLayer configured during NetworkInterface
startup"}
{"t":{"$date":"2020-10-27T15:12:46.028+01:00"},"s":"I", "c":"NETWORK", "id":46

```

Don't close the terminal or you'll kill the server !

7. Launch Mongo Client: Open another terminal window. Launch following commands: cd /path/to/mongo/ cd bin ./mongo

```

thierry@thierry-Inspiron-3793:~/Bureau/tpmaster/mongodb/mongodb-linux-x86_64-ubuntu
ntu2004-4.4.1/bin$ ./mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName
=mongodb
Implicit session: session { "id" : UUID("1804bcce-9a59-419a-bfaa-3f62e5ae3593") }
MongoDB server version: 4.4.1
---
The server generated these startup warnings when booting:
2020-10-27T15:12:46.029+01:00: Using the XFS filesystem is strongly reco

```

On windows, use mongo.exe instead ./mongo

Note: You can also have client with graphical interface. i.e.: <https://stackoverflow.com/a/6691013>

8. LAUNCHING :

Server launching : ./mongod

Client launching : /bin/mongo

★Le processus qui permet à MongoDB de fonctionner s'appelle "mongod".
On se connecte en passant par le serveur depuis le terminal en exécutant la commande "mongo" .

To familiarize yourself with the environment try these few commands:

```
> help  
> db.help()
```

3 PARTIE 1 - TP

3.1 Choix d'une base de données: LIBRARY

Utilisons la collection "library" pour avoir un affichage sous forme de documents.

```
>use library  
>switched to db library
```

3.1.1 Création de document 1

```
>document = ( { Type : "Book", Title : "Definitive Guide to MongoDB",  
ISBN : "987-1-4302-3051-9", Publisher : "Apress", Author:  
"Membrey, Peter", "Plugge, Eelco", "Hawkins, Tim"
```

```
} )
```

Le document sera affiché comme suit:

```
> document = ( { Type : "Book", Title : "Definitive Guide to MongoDB", ISBN :  
... "987-1-4302-3051-9", Publisher : "Apress", Author: ["Membrey, Peter",  
... "Plugge, Eelco", "Hawkins, Tim"] } )  
{  
    "Type" : "Book",  
    "Title" : "Definitive Guide to MongoDB",  
    "ISBN" : "987-1-4302-3051-9",  
    "Publisher" : "Apress",  
    "Author" : [  
        "Membrey, Peter",  
        "Plugge, Eelco",  
        "Hawkins, Tim"  
    ]  
}  
"document":{}
```

3.1.2 Cr ation de document 2

```
> db.media.insert(document)
```

```
> db.media.insert( { Type : "CD" ,Artist : "Nirvana",Title : "Nevermind",
... Tracklist : [
... { Track : "1" , Title : "Smells like teen spirit" , Length : "5:02" } , { Track : "2" ,
... Title : "In Bloom" , Length : "4:15" }
... ]} )
WriteResult({ "nInserted" : 1 })
```

WriteResult("nInserted":1)

⇒ cela signifie que le document a  t  ins r  dans la collection "media". On a vu deux m thodes pour ins rer un document.

```
> db.media.insert(document)
WriteResult({ "nInserted" : 1 })
```

3.2 LES REQU TES

Que fait ces commandes?

```
> db.media.find()
```

⇒ donne la liste de tous les documents dans la collection media:

```
> db.media.find()
{ "_id" : ObjectId("5f947c357395ba497d541cf7") , "Type" : "Book" , "Title" : "Definitive
Guide to MongoDB" , "ISBN" : "987-1-4302-3051-9" , "Publisher" : "Apress" , "Author" :
[ "Membrey, Peter" , "Plugge, Eelco" , "Hawkins, Tim" ] }
{ "_id" : ObjectId("5f947c807395ba497d541cf8") , "Type" : "CD" , "Artist" : "Nirvana" ,
"Title" : "Nevermind" , "Tracklist" : [ { "Track" : "1" , "Title" : "Smells like teen
spirit" , "Length" : "5:02" } , { "Track" : "2" , "Title" : "In Bloom" , "Length" : "4:
15" } ] }
```

```
>db.media.find({Artist:"Nirvana" })
```

⇒ affiche le document qui contient la cl  Artiste et la valeur Nirvana.

```
> db.media.find ( { Artist : "Nirvana" } )
{ "_id" : ObjectId("5f947c807395ba497d541cf8") , "Type" : "CD" , "Artist" : "Nirvana" ,
"Title" : "Nevermind" , "Tracklist" : [ { "Track" : "1" , "Title" : "Smells like teen
spirit" , "Length" : "5:02" } , { "Track" : "2" , "Title" : "In Bloom" , "Length" : "4:
15" } ] }
```

```
>db.media.find({Artist:"Nirvana"}, {Title:1})
```

⇒ affiche le document AVEC la cl /valeur de "Title" uniquement.

```
> db.media.find ( {Artist : "Nirvana"}, {Title: 1} )
{ "_id" : ObjectId("5f947c807395ba497d541cf8") , "Title" : "Nevermind" }
```

```
>db.media.find({Artist:"Nirvana"},{Title:0})
```

⇒ donne toute les informations du document SANS la clé/valeur de "Title".

```
> db.media.find ( {Artist : "Nirvana"}, {Title: 0} )  
{ "_id" : ObjectId("5f947c807395ba497d541cf8"), "Type" : "CD", "Artist" : "Nirvana",  
"Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like teen spirit", "Length" : "5:  
02 " }, { "Track" : "2 ", "Title" : "In Bloom", "Length" : "4:15 " } ] }
```

```
>db.media.find({Tracklist.Title:"In Bloom"})
```

```
> db.media.find( { "Tracklist.Title" : "In Bloom" } )  
{ "_id" : ObjectId("5f947c807395ba497d541cf8"), "Type" : "CD", "Artist" : "Nirvana",  
"Title" : "Nevermind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like teen  
spirit", "Length" : "5:02 " }, { "Track" : "2 ", "Title" : "In Bloom", "Length" : "4:  
15 " } ] }
```

```
>db.media.findOne( )
```

⇒ affiche un document de la collection "media" au hasard

```
>db.media.find().pretty()
```

⇒ affiche les champs "media" de façon plus lisible en séparant les documents

```
] > db.media.find().pretty()
{
    "_id" : ObjectId("5f947c357395ba497d541cf7"),
    "Type" : "Book",
    "Title" : "Definitive Guide to MongoDB",
    "ISBN" : "987-1-4302-3051-9",
    "Publisher" : "Apress",
    "Author" : [
        "Membrey, Peter",
        "Plugge, Eelco",
        "Hawkins, Tim"
    ]
}
{
    "_id" : ObjectId("5f947c807395ba497d541cf8"),
    "Type" : "CD",
    "Artist" : "Nirvana",
    "Title" : "Nevermind",
    "Tracklist" : [
        {
            "Track" : "1",
            "Title" : "Smells like teen spirit",
            "Length" : "5:02"
        },
        {
            "Track" : "2",
            "Title" : "In Bloom",
            "Length" : "4:15"
        }
    ]
}
```

3.3 FONCTIONS: SORT, LIMIT ET SKIP

```
>db.media.find().sort({Title:1})
```

⇒ affiche tous les documents en triant par ordre alphabétique en commençant par la première lettre de l'alphabet la valeur de la clé "Title".

```
> db.media.find().sort( { Title: 1 })
{ "_id" : ObjectId("5f947c357395ba497d541cf7"), "Type" : "Book", "Title" : "Definitive Guide to MongoDB", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress", "Author" : [ "Membrey, Peter", "Plugge, Eelco", "Hawkins, Tim" ] }
{ "_id" : ObjectId("5f947c807395ba497d541cf8"), "Type" : "CD", "Artist" : "Nirvana", "Title" : "Nevermind", "Tracklist" : [ { "Track" : "1", "Title" : "Smells like teen spirit", "Length" : "5:02" }, { "Track" : "2", "Title" : "In Bloom", "Length" : "4:15" } ] }
>
```

```
>db.media.find().sort({Title:-1})
```

⇒ affiche tous les documents en triant par ordre alphabétique en commençant par la dernière lettre de l'alphabet.

```
> db.media.find().sort( { Title: -1 })
{ "_id" : ObjectId("5f947c807395ba497d541cf8"), "Type" : "CD", "Artist" : "Nirvana", "Title" : "Nevermind", "Tracklist" : [ { "Track" : "1", "Title" : "Smells like teen spirit", "Length" : "5:02" }, { "Track" : "2", "Title" : "In Bloom", "Length" : "4:15" } ] }
{ "_id" : ObjectId("5f947c357395ba497d541cf7"), "Type" : "Book", "Title" : "Definitive Guide to MongoDB", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress", "Author" : [ "Membrey, Peter", "Plugge, Eelco", "Hawkins, Tim" ] }
```

```
>db.media.find().limit(1)
```

⇒ la fonction limit affiche la quantité de document à afficher.

```
>db.media.find().skip(1)
```

⇒ affiche tous les prochains documents après le premier.

```
>db.media.find().limit(1)
```

⇒ la fonction limit affiche le nombre de document à afficher.

```
> db.media.find().limit(1)
{ "_id" : ObjectId("5f947c357395ba497d541cf7"), "Type" : "Book", "Title" : "Definitive Guide to MongoDB", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress", "Author" : [ "Membrey, Peter", "Plugge, Eelco", "Hawkins, Tim" ] }
>
```

```
>db.media.find().skip( 1 )  
⇒ la fonction skip affiche le prochain document.
```

```
> db.media.find().skip(1)  
{ "_id" : ObjectId("5f947c807395ba497d541cf8"), "Type" : "CD", "Artist" : "Nirvana",  
"Title" : "Nevermind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like teen  
spirit", "Length" : "5:02 " }, { "Track" : "2 ", "Title" : "In Bloom", "Length" : "4:  
15 " } ] }
```

```
>db.media.find().sort({Title:1}).limit(1).skip(1)  
⇒ affiche le prochain document après le premier document par ordre al-  
phabétique de la clé "Title".
```

```
> db.media.find().sort({Title:1}).limit(1).skip(1)  
{ "_id" : ObjectId("5f947c807395ba497d541cf8"), "Type" : "CD", "Artist" : "Nirvana",  
"Title" : "Nevermind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like teen  
spirit", "Length" : "5:02 " }, { "Track" : "2 ", "Title" : "In Bloom", "Length" : "4:  
15 " } ] }  
> []
```

3.4 AGGREGATION: QUE FAIT CES COMMANDES?

```
>db.media.count()  
⇒ affiche le nombre de document disponible.
```

```
> db.media.count()  
2
```

```
>db.media.find({Publisher:"Apress",Type:"Book"}).count()  
⇒ compte le nombre de document qui contient la clé et valeur choisis en  
tant que filtre.
```

```
> db.media.find({Publisher : "Apress",Type : "Book"}).count()  
1
```

```
>db.media.find({Publisher:"Apress",Type:"Book"}).skip(1).count(true) ⇒  
affiche le nombre de document qui contient les valeurs "Apress" et "Book"  
après le premier document et on lui demande de les compter.
```

```
> db.media.find( { Publisher: "Apress", Type: "Book" }).skip(2).count(true)  
0
```

3.5 DISTINCT(): QUE FAIT CES COMMANDES

3.5.1 Add a new record

```
>document = ( { Type : "Book", Title : "Definitive Guide to MongoDB", ISBN:  
"1- 4302-3051-7", Publisher : "Apress", Author :  
"Membrey, Peter", "Plugge, Eelco", "Hawkins, Tim"  
} )
```

```
>  
> document=({Type:"Book", Title:"Definitive Guide to MongoDB", ISBN:"1-4302-3051-7",P  
ublisher:"Apress",Author:["Membrey, Peter","Plugge,Eelco","Hawkins,Tim"]})  
{  
    "Type" : "Book",  
    "Title" : "Definitive Guide to MongoDB",  
    "ISBN" : "1-4302-3051-7",  
    "Publisher" : "Apress",  
    "Author" : [  
        "Membrey, Peter",  
        "Plugge,Eelco",  
        "Hawkins,Tim"  
    ]  
}  
> db.media.insert(document)  
WriteResult({ "nInserted" : 1 })  
>
```

```
>db.media.distinct("Title")  
⇒ affiche la liste des valeurs associés à la clé Title.
```

```
>  
> db.media.distinct("Title")  
[ "Definitive Guide to MongoDB", "Nevermind" ]
```

```
>db.media.group  
⇒La fonction group ne marche pas.
```

3.5.2 Add more records

```
>dvd = ( { Type : "DVD", Title : "Matrix, The", Released : 1999, Cast: ["Keanu Reeves","Carry-Anne Moss","Laurence Fishburne","Hugo Weaving","Gloria Foster","Joe Pantoliano"] } )
```

```
> dvd = ( { Type : "DVD", Title : "Matrix, The", Released : 1999, Cast: ["Keanu Reeves","Carry-Anne Moss","Laurence Fishburne","Hugo Weaving","Gloria Foster","Joe Pantoliano"] } )
{
    "Type" : "DVD",
    "Title" : "Matrix, The",
    "Released" : 1999,
    "Cast" : [
        "Keanu Reeves",
        "Carry-Anne Moss",
        "Laurence Fishburne",
        "Hugo Weaving",
        "Gloria Foster",
        "Joe Pantoliano"
    ]
}
```

```
>dvd = ( { "Type" : "DVD", "Title" : "Toy Story 3", "Released" : 2010 }
)
```

```
> dvd = ( { "Type" : "DVD", "Title" : "Toy Story 3", "Released" : 2010 } )
{ "Type" : "DVD", "Title" : "Toy Story 3", "Released" : 2010 }
```

```
>db.media.insert(dvd)
```

```
> db.media.insert(dvd)
WriteResult({ "nInserted" : 1 })
```

3.5.3 Insert with JavaScript

```
>function insertMedia( type, title, released ){db.media.insert({ "Type": type,"Title": title,"Released": released }); }
```

⇒ Il est important de noter que MongoDB comprend le langage sous JavaScript.
Il est surtout utilisé ici pour définir une fonction.

```
>function insertMedia( type, title, released ){db.media.insert({ "Type": type,"Title": title,"Released": released}); }
```

```
>insertMedia("DVD", "Blade Runner", 1982 )
```

```
> insertMedia("DVD", "Blade Runner", 1982 )
> db.media.insert(dvd)
WriteResult({ "nInserted" : 1 })

```

3.6 COMPARISON OPERATORS

3.6.1 Les opérateurs \$gt, \$lt, \$gte, \$lte, \$ne, \$in, \$nin (resp. >, <, >=, <=, !=, IN, NOT IN)

>db.media.find ({ Released : {gt : 2000} }, { "Cast" : 0 })

⇒ affiche toute les documents pour les valeurs de la clé "released" dont la valeur est supérieur à 2000 et sans afficher la valeur de la clé "Cast".

```
> db.media.find ( { Released : { $gt : 2000 } }, { "Cast" : 0 } )
{ "_id" : ObjectId("5f948ae67395ba497d541cfa"), "Type" : "DVD", "Title" : "Toy Story 3", "Released" : 2010 }
```

>db.media.find({Released : { \$gte: 1990, \$lt : 2010 }}, { "Cast" : 0 })

⇒ affiche toute les documents avec leur "type" et leur "Title" pour les valeurs de la clé "released" compris entre 1990 et 2010.

```
> db.media.find( {Released : { $gte: 1990, $lt : 2010 }}, { "Cast" : 0 } )
{ "_id" : ObjectId("5f90a8a1520df9859085a974"), "Type" : "DVD", "Title" : "Matrix, The", "Released" : 1999 }
```

```
>db.media.find( { Type : "Book", Author :{$ne : "Plugge, Eelco"} }).pretty()
⇒ affiche les 2 documents de type book et dont l'author est plugge eelco
```

```
> db.media.find( { Type : "Book", Author: {$ne : "Plugge, Eelco"} }).pretty()
{
  "_id" : ObjectId("5f9097d5520df9859085a971"),
  "Type" : "Book",
  "Title" : "Definitive Guide to Mongodb",
  "ISBN" : "987-1-4302-3051-9",
  "Publisher" : "Apress",
  "Author" : [
    "Membrey, Peter",
    "Plugge,Eelco",
    "Hawkins,Tim"
  ]
}
{
  "_id" : ObjectId("5f90a4ad520df9859085a973"),
  "Type" : "Book",
  "Title" : "Definitive Guide to Mongodb",
  "ISBN" : "1-4302-351-7",
  "Publisher" : "Apress",
  "Author" : [
    "Membrey, Peter",
    "Plugge,Eelco",
    "Hawkins,Tim"
  ]
}
db.media.find( {Released : {$in : ["1999", "2008", "2009"]}, {"Cast" : 0}})
```

```
>db.media.find( {Released : {$in : ["1999", "2008", "2009"] } }, { "Cast" : 0 })
⇒ ici ne nous renvoie rien.
```

```
>db.media.find( {Released : { $nin : ["1999", "2008", "2009"] }, Type : "DVD" }, {"Cast" : 0 } )
⇒ affiche les documents qui ne sont pas inclus dans le filtre.
```

```
> db.media.find( {Released : {$nin : ["1999", "2008", "2009"] },Type : "DVD" }, {
... "Cast" : 0 }
{
  "_id" : ObjectId("5f90a8a1520df9859085a974"), "Type" : "DVD", "Title" : "Matrix, The", "Released" : 1999
  "_id" : ObjectId("5f90a8f5520df9859085a975"), "Type" : "DVD", "Title" : "Toy Story 3", "Released" : 2010
  "_id" : ObjectId("5f90a9e0520df9859085a976"), "Type" : "DVD", "Title" : "Blade Runner", "Released" : 1982
}
> db.media.find( {Released : {$in : ["1999", "2008", "2009"] },Type : "DVD" }, { "Cast" : 0 })
```

3.6.2 L'opérateur \$or

What do these queries do?

```
>db.media.find({ $or:[{ "Title": "Toy Story 3" }, { "ISBN": "987-1-4302-3051-9" } ] })
```

⇒ affiche les 2 documents dans le filtre.

```
> db.media.find({ $or : [ { "Title" : "Toy Story 3" }, { "ISBN" : "987-1-4302-3051-9" } ... ] }).pretty()
{
  "_id" : ObjectId("5f9097d5520df9859085a971"),
  "Type" : "Book",
  "Title" : "Definitive Guide to MongoDB",
  "ISBN" : "987-1-4302-3051-9",
  "Publisher" : "Apress",
  "Author" : [
    "Membrey, Peter",
    "Plugge, Eelco",
    "Hawkins, Tim"
  ]
}
{
  "_id" : ObjectId("5f90a8f5520df9859085a975"),
  "Type" : "DVD",
  "Title" : "Toy Story 3",
  "Released" : 2010
}
```

```
>db.media.find({ "Type" : "DVD", $or : [ { "Title" : "Toy Story 3" }, { "ISBN" : "987-1-4302-3051-9" } ] })
```

```
> db.media.find({ "Type" : "DVD", $or : [ { "Title" : "Toy Story 3" }, { "ISBN" : "987-1-4302-3051-9" } ] }).pretty()
{
  "_id" : ObjectId("5f90a8f5520df9859085a975"),
  "Type" : "DVD",
  "Title" : "Toy Story 3",
  "Released" : 2010
}
```

3.7 Opérateur: \$SLICE

L'opérateur \$slice combine la fonction limit() et skip():

1. \$slice: [20, 10] : skip 20, limit 10

2. \$slice:-5 : The last 5

3. \$slice: 5 : The first 5

```
>db.media.find({ "Title" : "Matrix, The" }, { "Cast" : { $slice: 3 } })
```

⇒ affiche les 3 premiers éléments du champs "cast".

```
> db.media.find({ "Title" : "Matrix, The" }, { "Cast" : { $slice: 3 } })
{
  "_id" : ObjectId("5f90a8f5520df9859085a975"),
  "Type" : "DVD",
  "Title" : "Matrix, The",
  "Released" : 1999,
  "Cast" : [
    "Keanu Reeves",
    "Carrie-Anne Moss",
    "Laurence Fishburne"
  ]
}
{
  "_id" : ObjectId("5f90a8f5520df9859085a974"),
  "Type" : "DVD",
  "Title" : "Matrix, The",
  "Released" : 1999,
  "Cast" : [
    "Keanu Reeves",
    "Carrie-Anne Moss",
    "Laurence Fishburne",
    "Hugo Weaving",
    "Clark Gregg"
  ]
}
```

```
>db.media.find({ "Title" : "Matrix, The" }, { "Cast" : { $slice: -3 } })
```

⇒ affiche les 3 derniers éléments de "Cast" dans "title": "Matrix, The".

```

> db.media.find({ "_id": "Matrix_The" })
[{"_id": ObjectId("5f9eaba152bdf9859085a974"), "Type": "DVD", "Title": "Matrix, The", "Released": 1999, "Cast": ["Keanu Reeves", "Carrie-Anne Moss", "Laurence Fishburne", "Hugo Weaving", "Gloria Foster", "Joe Pantoliano"]},
 {"_id": ObjectId("5f9eaba152bdf9859085a974"), "Type": "DVD", "Title": "Matrix, The", "Released": 1999, "Cast": ["Hugo Weaving", "Gloria Foster", "Joe Pantoliano"]}]

```

3.8 Opérateurs: \$size et \$exists

>db.media.find ({ Tracklist : { \$size : 2 } })

⇒ l'opérateur coupe le champ "Tracklist" en plusieurs éléments et affiche les 2 premiers éléments uniquement.

```

> db.media.find ( { Tracklist : { $size : 2} } ).pretty()
{
    "_id" : ObjectId("5f909924520df9859085a972"),
    "Type" : "CD",
    "Artist" : "Nirvana",
    "Title" : "Nevermind",
    "Tracklist" : [
        {
            "Track" : "1",
            "Title" : "Smells like teen spirit",
            "Length" : "5:02"
        },
        {
            "Track" : "2",
            "Title" : "In Bloom",
            "Length" : "4:15"
        }
    ]
}
>
>
```

>db.media.find ({ Author : {\$exists : true } })
⇒ affiche les documents où la clé "Author" existe.

```
> db.media.find ( { Author : {$exists : true } } ).pretty()
{
    "_id" : ObjectId("5f9097d5520df9859085a971"),
    "Type" : "Book",
    "Title" : "Definitive Guide to Mongodb",
    "ISBN" : "987-1-4302-3051-9",
    "Publisher" : "Apress",
    "Author" : [
        "Membrey, Peter",
        "Plugge,Eelco",
        "Hawkins,Tim"
    ]
}
{
    "_id" : ObjectId("5f90a4ad520df9859085a973"),
    "Type" : "Book",
    "Title" : "Definitive Guide to Mongodb",
    "ISBN" : "1-4302-351-7",
    "Publisher" : "Apress",
    "Author" : [
        "Membrey, Peter",
        "Plugge,Eelco",
        "Hawkins,Tim"
    ]
}
```

```
> db.media.find ( {Author:{$exists : false} } )  
⇒ affiche les documents dont la clé "Author" n'existe pas.
```

```
> db.media.find ( { Author : {$exists : false} } ).pretty()  
{  
    "_id" : ObjectId("5f909924520df9859085a972"),  
    "Type" : "CD",  
    "Artist" : "Nirvana",  
    "Title" : "Nevermind",  
    "Tracklist" : [  
        {  
            "Track" : "1",  
            "Title" : "Smells like teen spirit",  
            "Length" : "5:02"  
        },  
        {  
            "Track" : "2",  
            "Title" : "In Bloom",  
            "Length" : "4:15"  
        }  
    ]  
}  
{  
    "_id" : ObjectId("5f90a8a1520df9859085a974"),  
    "Type" : "DVD",  
    "Title" : "Matrix, The",  
    "Released" : 1999,  
    "Cast" : [  
        "Keanu Reeves",  
        "Carry-Anne Moss",  
        "Laurence Fishburne",  
        "Hugo Weaving",  
        "Gloria Foster",  
        "Joe Pantoliano"  
    ]  
}  
{  
    "_id" : ObjectId("5f90a8f5520df9859085a975"),  
    "Type" : "DVD",  
    "Title" : "Toy Story 3",  
    "Released" : 2010  
}  
{  
    "_id" : ObjectId("5f90a9e0520df9859085a976"),  
    "Type" : "DVD",  
    "Title" : "Blade Runner",  
    "Released" : 1982  
}  
{  
    "_id" : ObjectId("5f944d407395ba497d541cf6"),  
    "Released" : {  
        "gt" : 2000  
    }  
}
```

3.9 INDEX CREATION

La création d'index est utilisé pour cibler des éléments du sous-ensembles d'un objet. Ils permettent de faire une sélection plus rapide des éléments sélectionnés et ainsi optimiser une requête.

3.9.1 Ascending index

```
>db.media.ensureIndex( { Title :1 } )
```

⇒ D'après la documentation MongoDB, la fonction "ensureIndex" a été remplacé par "**createIndex**" depuis la version 3.0.0. Ici, on a créé un index sur le champ "Title" dans l'ordre croissant de ses valeurs.

```
>
> db.media.ensureIndex( { Title :1 } )
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "ok" : 1
}
```

3.9.2 Descending index

```
>db.media.ensureIndex( { Title :-1 } )
```

⇒ on va créer un index pour les valeurs du champ "title" dans l'ordre décroissant.

```
>
> db.media.ensureIndex( { Title :-1 } )
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 2,
    "numIndexesAfter" : 3,
    "ok" : 1
}
```

3.9.3 Index for embedded objects

```
>db.media.ensureIndex({ "Tracklist.Title":1 } )
```

⇒ création d'un index pour des valeurs à l'intérieur d'un champ spécifique.

```
> db.media.ensureIndex( { "Tracklist.Title" : 1 } )
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 3,
    "numIndexesAfter" : 4,
    "ok" : 1
}
```

3.9.4 Force the index usage: hint()

```
>db.media.find({ ISBN: "987-1-4302-3051-9" } ).hint ({ISBN: -1} ) error: {
"$err" : "bad hint", "code" : 10113 }
```

⇒ ici, on force l'utilisation d'un index sur "ISBN" pour optimiser la requête sur celui-ci.

```
>db.media.ensureIndex({ISBN: 1})
```

```
> db.media.ensureIndex({ISBN: 1})
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 4,
    "numIndexesAfter" : 5,
    "ok" : 1
}
```

```
>db.media.find({ ISBN: "987-1-4302-3051-9" } ).hint ( { ISBN: 1 } )
```

```
> db.media.find( { ISBN: "987-1-4302-3051-9"} ) . hint ( { ISBN: 1 } )
{ "_id" : ObjectId("5f9097d5520df9859085a971"), "Type" : "Book", "Title" : "Definitive Guide to MongoDB", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress", "Author" :
[ "Membrey, Peter", "Plugge, Eelco", "Hawkins, Tim" ] }
```

```
>db.media.getIndexes()
```

⇒ on a créé des indexes et on peut tous les visualiser avec la fonction
getIndexes.

```
> db.media.getIndexes()
[
  {
    "v" : 2,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_"
  },
  {
    "v" : 2,
    "key" : {
      "Title" : 1
    },
    "name" : "Title_1"
  },
  {
    "v" : 2,
    "key" : {
      "Title" : -1
    },
    "name" : "Title_-1"
  },
  {
    "v" : 2,
    "key" : {
      "Tracklist.Title" : 1
    },
    "name" : "Tracklist.Title_1"
  },
  {
    "v" : 2,
    "key" : {
      "ISBN" : 1
    },
    "name" : "ISBN_1"
  }
]
```

3.10 DATA UPDATE

3.10.1 Update (condition, newObject, upsert, multi)

1. upsert = true : create the object if does not exist
2. Multi Specifies whether the change is made on a single object (default) or on all objects that meet the condition

>db.media.update({ "Title" : "Matrix, the"}, {"Type" : "DVD", "Title" : "Matrix, the", "Released" : "1999", "Genre" : "Action"}, true)
⇒ affiche la modification effectué avec succès en désignant par "nUpserted":1.

```
> db.media.update( { "Title" : "Matrix, the"}, {"Type" : "DVD", "Title" : "Matrix, the", "Released" : "1999", "Genre" : "Action"}, true)
WriteResult({
    "nMatched" : 0,
    "nUpserted" : 1,
    "nModified" : 0,
    "_id" : ObjectId("5f945e2a57ef23c9b368e929")
})
```

3.10.2 Add/delete an attribute

>db.media.update ({ "Title": "Matrix, the" }, {\$set : { Genre:"Sci-Fi" } })
⇒ mongodb affiche le résultat effectué en indiquant le nombre de correspondance selon le filtre ici "title" puis le nombre de modification effectué "nModified": 1

```
> db.media.update ( { "Title" : "Matrix, the" }, {$set : { Genre : "Sci-Fi" } } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

>db.media.update ({"Title": "Matrix, the"}, {\$unset: { "Genre" : 1 } })
⇒ affiche le nombre de résultat effectué ainsi que le nombre de changements effectué:

```
> db.media.update ( {"Title": "Matrix, the"}, {$unset: { "Genre" : 1 } } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

3.10.3 Delete

1. Documents meeting a condition: >db.media.remove({ "Title": "Different Title" })
⇒ **nremoved:0** car aucun document ne porte ce nom.

```
> db.media.remove( { "Title" : "Different Title" } )
WriteResult({ "nRemoved" : 0 })
```

2. All documents: >db.media.remove({ })
⇒ tous les documents seront effacés.

```
> db.media.remove({})
WriteResult({ "nRemoved" : 8 })
> db.media.find()
>
```

3. All the collection: >db.media.drop()

4 Partie 2: Base de données géographique

4.1 Import data

On copie et on colle le fichier earthquakes.json dans le bin avant de lancer l'importation suivant:

```
> ./bin/mongoimport --type json -d geodb -c earthquakes --file earthquakes.json
```

```
thierry@thierry-Inspiron-3793:~/Bureau/tpmaster/mongodb/mongodb-linux-x86_64-ubuntu20
04-4.4.1/bin$ ./mongoimport --type json -d geodb -c earthquakes --file earthquakes.json
on
2020-10-24T23:24:11.615+0200      connected to: localhost
2020-10-24T23:24:12.087+0200      imported 7669 documents
thierry@thierry-Inspiron-3793:~/Bureau/tpmaster/mongodb/mongodb-linux-x86_64-ubuntu20
04-4.4.1/bin$
```

La collection earthquakes est téléchargée avec succès.

4.2 Quelques requêtes de base:

1. Count the number of documents
2. Show first 5:

⇒ on regarde comment sont rangés les documents grâce à la fonction getIndexes.

```
> db.earthquakes.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
```

on trie les documents selon les index déjà attribués aux documents

```
> db.earthquakes.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
> db.earthquakes.find().sort({_id:1})
```

⇒ on affiche les 5 premiers avec la fonction LIMIT.

```
> db.earthquakes.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
> db.earthquakes.find().sort({_id:1})
```

3. View the 6th document:

```
> db.earthquakes.find().sort({_id:1}).limit(5)
```

4. How many separate statuses exist in the DB?

⇒ Il y a 4 différents valeur du champ "status".

```
> db.earthquakes.distinct("properties.status")
[ null, "AUTOMATIC", "PUBLISHED", "REVIEWED" ]
```

4.3 Modification des données

1. Combien de documents contient la propriété felt (property.felt! = Null)?

```
> db.earthquakes.find({"properties.felt": {$ne: null}}).pretty().count()
801
```

2. Supprimer ce champ pour les documents pour lesquels il est Null avant la suppression, il y a 6868 document avec felt=null.

```
> db.earthquakes.find({"properties.felt": null}).pretty().count()
6868
```

3. suppression:

```
> db.earthquakes.remove({"properties.felt": null})
WriteResult({ "nRemoved" : 6868 })
>
```

4. Ajouter une colonne iso date dont la valeur est la conversion de timestamp contenu dans "properties.time"

```
> db.earthquakes.find().forEach( function(eq) eq.properties.iso_date = new Date(eq.properties.time); db.ee
```

```
> db.earthquakes.findOne()
{
  "_id" : ObjectId("5f949b7b57ef23c9b368ea6c"),
  "type" : "Feature",
  "properties" : {
    "mag" : 2.6,
    "place" : "9km ENE of Yountville, California",
    "time" : NumberLong("1370234254500"),
    "updated" : NumberLong("1370243886482"),
    "tz" : -420,
    "url" : "http://earthquake.usgs.gov/earthquakes/eventpage/nc72001425"

    "detail" : "http://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/n
c72001425.geojson",
    "felt" : 6,
    "cdi" : 3.1,
    "mmi" : null,
    "alert" : null,
    "status" : "AUTOMATIC",
    "tsunami" : null,
    "sig" : 106,
    "net" : "nc",
    "code" : "72001425",
    "ids" : ",nc72001425,",
    "sources" : ",nc,",
    "types" : ",dyfi,focal-mechanism,general-link,geoserve,nearby-cities,
origin,phase-data,scitech-link,"
    "nst" : null,
    "dmin" : 0,
    "rms" : 0.2,
    "gap" : 46.8,
    "magType" : "Md",
    "type" : "earthquake",
    "iso_date" : ISODate("2013-06-03T04:37:34.500Z")
  },
  "geometry" : {
    "type" : "Point",
    "coordinates" : [
      -122.257,
      38.4373,
      7.1
    ]
  },
  "id" : "nc72001425"
}
```

4.4 Nettoyage des données (DATA CLEANING)

Convert the string from the properties.types field to an array and put it in a field types_as_array:

Use the function ch.split (",") to separate a string ch into several words according to the separator ','...

```
>db.earthquakes.find().forEach( function(eq){var str = new String(eq.properties.types); eq.properties.types_as_array = str.split(",");db.earthquakes.save(eq);});
```

Vérifier en montrant le premier document: il est affiché la mise en forme sous forme de liste le champ "properties.types".

```
"types_as_array" : [  
    "",  
    "dyfi",  
    "focal-mechanism",  
    "general-link",  
    "geoserve",  
    "nearby-cities",  
    "origin",  
    "phase-data",  
    "scitech-link",  
    ""  
]
```

4.5 Nettoyage des données (DATA CLEANING)

Clean the empty elements ("") from the array properties.types_as_array

```
>db.earthquakes.update({},{ $pullAll: "properties.types_as_array" : [""] }, { multi: true })
```

Vérifier en montrant le premier document: Cette fonction java string efface les espaces vides sous forme de (" ")de tous les documents.

```
>
> db.earthquakes.update(
... {},
... { $pullAll: { "properties.types_as_array" : [""] } },
... { multi: true }
...
WriteResult({ "nMatched" : 801, "nUpserted" : 0, "nModified" : 801 })
```

4.6 Requêtes

1. Indiquez le nombre de documents dont la liste de types (properties.type_as_array) contient "geoserve" et "tectonic-summary".
⇒ affiche les documents qui contiennent à la fois "geoserve" et "tectonic-summary".

```
> db.earthquakes.find({$and:[{"properties.types_as_array":"geoserve"}, {"properties.types_as_array":"tectonic-summary"}]}).count()
471
>
```

2. Indiquez le nombre de documents dont la liste de types (properties.type_as_array) contient "geoserve" ou "tectonic-summary".
⇒ affiche tous les documents qui contiennent soit "geoserve" soit "tectonic summary".

```
> db.earthquakes.find({$or:[{"properties.types_as_array":"geoserve"}, {"properties.types_as_array":"tectonic-summary"}]}).count()
801
>
```

4.7 Indexation géographique

Nous allons maintenant modifier les données afin d'adapter les coordonnées géographiques au format qui nous permettra de construire un index 2dsphere.

Normalisez les données en supprimant le dernier élément de la table "geometry.coordinates" et en le copiant dans un champ "profondeur".

```
        },
        "geometry" : {
            "type" : "Point",
            "coordinates" : [
                -122.257,
                38.4373
            ]
        },
        "id" : "nc72001425",
        "depth" : 7.1
    }
```

4.8 Index creation

Créer un index de type 2dsphere sur les attributs geometry.

```
db.earthquakes.createIndex({geometry:"2dsphere"})  
  "createdCollectionAutomatically" : false,  
  "numIndexesBefore" : 2,  
  "numIndexesAfter" : 3,  
  "ok" : 1
```

Requête: Exécuter une requête qui recherche à partir de "earthquakes" near -74, 40.74 (dans un rayon de 1000 m) (utilisez \$geoWithin and \$center) Documentation : <http://docs.mongodb.org/manual/reference/operator/query-geospatial/>

```
> db.earthquakes.find({"geometry":{$geoWithin:{$centerSphere:[[-74,40.74], 1000/3963.2]}}}).pretty()  
{
```

```
> db.earthquakes.find({"geometry":{$geoWithin:{$centerSphere:[[-74,40.74], 1000/3963.2]}}}).count()  
11
```

4.9 Requêtes

Trouvez les "earthquakes" qui sont autour de la place "8km NW of Cobb, California", avec une distance maximale de 500 km.

4.10 Aggregate

Séquence ordonnée des opérateurs:

Commande:

```
> db.earthquakes.aggregate( [ {$op1 : {},$op2 : ,...}]);
```

Opérateurs : //équivalent SQL

\$match : Simple filter // where

\$project : Projection //select

\$sort : Sorting //order by

\$unwind : normalisation 1NF //group by + fn

\$group : grouping + aggregate function

\$lookup : left join(from 3.2) //left outer-join

\$out : storing the result (from 3.2)

\$redact : conditional pruning (nested documents) + \$sample, \$limit, \$skip.

4.11 Aggrégate: examples 1

4.11.1 Séquence:

Le résultat d'une opération sert comme entrée pour la prochaine

```
> db.earthquakes.aggregate([{$match: { "properties.type" : "quarry"}}, {$project: { "_id" : 1, "geometry" : 1}}, {$sort: { "depth" : -1}}]);
```

```
> db.earthquakes.aggregate([{"$match": {"properties.type": "quarry"}},  
... {"$project": {"_id": 1, "geometry": 1}},  
... {"$sort": {"depth": -1}}]  
... ]);  
  
[{"_id": ObjectId("5f949b7b57ef23c9b368ebc9"), "geometry": {"type": "Point", "coordinates": [-120.5167, 49.4388]}},  
 {"_id": ObjectId("5f949b7b57ef23c9b368fb47"), "geometry": {"type": "Point", "coordinates": [-121.574, 39.662]}},  
 {"_id": ObjectId("5f949b7c57ef23c9b368fb0ae"), "geometry": {"type": "Point", "coordinates": [-122.1073, 37.3358]}},  
 {"_id": ObjectId("5f949b7c57ef23c9b368fb9bd"), "geometry": {"type": "Point", "coordinates": [-121.9482, 48.0883]}},  
 {"_id": ObjectId("5f949b7c57ef23c9b368fb9e2"), "geometry": {"type": "Point", "coordinates": [-120.5382, 49.4467]}},  
 {"_id": ObjectId("5f949b7c57ef23c9b3691067"), "geometry": {"type": "Point", "coordinates": [-119.1538, 47.8193]}},  
 {"_id": ObjectId("5f949b7c57ef23c9b3690378"), "geometry": {"type": "Point", "coordinates": [-123.4133, 46.1473]}},  
 {"_id": ObjectId("5f949b7c57ef23c9b369039d"), "geometry": {"type": "Point", "coordinates": [-120.3155, 47.667]}},  
 {"_id": ObjectId("5f949b7c57ef23c9b36903ab"), "geometry": {"type": "Point", "coordinates": [-122.1665, 37.32]}},  
 {"_id": ObjectId("5f949b7c57ef23c9b36904b2"), "geometry": {"type": "Point", "coordinates": [-120.5662, 49.4012]}}
```

4.11.2 \$unwind

Créer un document pour chaque instance >db.earthquakes.aggregate([{ \$unwind : "properties.types_as_array" }, { \$limit : 5 }])

⇒ affiche pour un même objetId, le champ Type_of_array sera reparti en 5 documents avec pour chaque document une valeur de type_as_array.

```
> db.earthquakes.aggregate([{$unwind : "$properties.types_as_array"}, {$limit:5} ]).pretty()
{
  "_id" : ObjectId("5f94fb7b57ef23c9b3d68ea6c"),
  "type" : "Feature",
  "properties" : {
    "mag" : 2.6,
    "place" : "9km ENE of Yountville, California",
    "time" : NumberLong("1370234254500"),
    "updated" : NumberLong("1370243886482"),
    "tz" : -428,
    "url" : "http://earthquake.usgs.gov/earthquakes/eventpage/nc72001425",
    "detail" : "http://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/nc72001425.geojson",
    "felt" : 6,
    "cdl" : 3.1,
    "mml" : null,
    "alert" : null,
    "status" : "AUTOMATIC",
    "tsunami" : null,
    "sig" : 106,
    "net" : "nc",
    "code" : "72001425",
    "ids" : "nc72001425",
    "sources" : "nc",
    "types" : "dyfl,focal-mechanism,general-link,geoserve,nearby-cities,origin,phase-data,scitech-link"
  }
},
```

"types as array" : "focal-mechanism"

```
        "types_as_array" : "general-link"
    },
}
```

```
"types_as_array" : "geoserve"  
"types_as_array" : "nearby-cities"
```

4.12 Aggregate: examples 2

Group : key (.id) + aggregate (\$sum / \$avg / ...)
No group: null > db.users.aggregate([{\$group : {"_id" : null, "res": {\$sum : 1}} }]);
Groupe by value : \$key >db.users.aggregate([{\$group:{"_id" : "\$age", "res": {"\$sum : 1}} }]);
Average: \$key >db.users.aggregate([{\$group:{"_id": "\$address.city", "moy": {\$avg: "\$age"} }}]);

Exemple: séquence

```
> db.movies.aggregate([  
  {$match: { "year" : {$gt : 1995}}},  
  {$unwind : "$genres_as_array"},  
  {$group : {"_id" : "$year", "count": {$sum: 1}}},  
  {$match : {"count" : {$gt : 2}},  
  {$sort : { "count" : -1}}]);
```

4.13 Administration

4.13.1 Backup

```
>mkdir testmongobackup >cd testmongobackup  
  >../mongodb/bin/mongodump -help  
  >../mongo/bin/mongodump  
  >../mongodump -db library -collection media  
  à ./dump/[databasename]/[collectionname].bson
```

4.13.2 Restore

```
>cd testmongobackup  
  >../mongo/bin/mongorestore -help  
  
1. Tout restaurer  
  >../mongo/bin/mongorestore -drop  
  
2. Restaurer une seule collection  
  >../mongo/bin/mongorestore -d library -c media -drop
```

4.14 Security

4.14.1 Authentification

⇒ Cette étape permet à MongoDB de prendre en charge l'authentification au lancement de celui-ci.

1. Client side

```
> use admin  
>db.addUser("admin", "adminpassword")
```

```
> db.addUser("admin", "adminpassword");  
uncaught exception: TypeError: db.addUser is not a function :  
@(shell):1:1
```

Par contre en essayant une autre commande comme la commande **"createUser"**, on a l'affichage suivante:

```
SWITCHED TO DB admin  
> db.createUser({ "user": "mongoroot", "pwd": "M0t2p4s5e!", "roles": [ "root" ] })  
Successfully added user: { "user" : "mongoroot", "roles" : [ "root" ] }
```

⇒ on a réussi à ajouter un autre utilisateur.

2. Arrêt de MongoDb:

```
⇒ on arrête MongoDB  
>db.shutdownServer()  
⇒ le serveur s'est arrêté.
```

```
> db.shutdownServer()  
server should be down...
```

3. Redémarrage de MongoDB On va redemarrer avec l'option **"auth"** qu'on va ajouter à la ligne de commande habituelle:

```
./mongod --dbpath datafolder --logappend --auth
```

4. Server side

```
> use admin  
>db.addUser("admin", "adminpassword")  
• Restart the server (commande shell)  
>sudo service mongodb restart or >db.shutdownServer()
```

```

585,    "ctx": "conn2", "msg": "now exiting"}
{"t":{"$date":"2020-10-27T14:05:48.665+01:00"},"s":"I",  "c":"CONTROL",  "id":23
138,    "ctx":"conn2","msg":"Shutting down","attr":{"exitCode":0}}


```

```

> db.shutdownServer()
server should be down...
> use admin
switched to db admin
> db.addUser("admin","adminpassword")
uncaught exception: TypeError: db.addUser is not a function :
@(shell):1:1
> db.auth("admin","adminpassword")
Error: Authentication failed.
0
> db.auth("mongoroot","M0t2p4s5e!")
1

```

- Authenticate

```

>use admin
>db.auth("admin","adminpassword")

>use library
>db.addUser("ronaldo", "ronaldopassword")
>db.addUser("messi", "messipassword",true) //read only

```

```

> db.createUser({"user":"ronaldo","pwd":"ronaldopassword","roles":["read"]})
Successfully added user: { "user" : "ronaldo", "roles" : [ "read" ] }
> db.createUser({"user":"messi","pwd":"messipassword","roles":["read"]})
Successfully added user: { "user" : "messi", "roles" : [ "read" ] }

```

on note qu'il faut absolument ajouter le champ "roles" sinon on a un message d'erreur.

```
>db.removeUser("ronaldo")
```

```

> db.removeUser("ronaldo")
WARNING: db.removeUser has been deprecated, please use db.dropUser instead
true
> db.dropUser("ronaldo")
false

```

4.15 SECURITY: USER, PRIVILEGE, RESOURCE

```
>db.createUser( { user: "reportsUser", pwd: "12345678", roles: [ { role: "read", db: "reporting" }, { role: "read", db: "products" }, { role: "read", db: "sales" }, { role: "readWrite", db: "accounts" } ] }) On note que:
```

1. **Read** autorise la lecture de tous les documents en formats JSON non utilisés par le système.
2. **readWrite** a le privilège de "read" et en plus autorise la modification des documents non-système.

```
Successfully added user: {
    "user" : "reportsUser",
    "roles" : [
        {
            "role" : "read",
            "db" : "reporting"
        },
        {
            "role" : "read",
            "db" : "products"
        },
        {
            "role" : "read",
            "db" : "sales"
        },
        {
            "role" : "readWrite",
            "db" : "accounts"
        }
    ]
}
```

```
>db.dropUser("reportsUser")
```

```
> db.dropUser("reportsUser")
true
```

```
>db.dropAllUsers( )
>db.createRole({ role: "< name >",
privileges: [ { resource: { < resource > } },
actions: [ "< action >", ... ] }, ... ],
roles: [ { role: "< role >", db: "< database >" } — "< role >", ... ] })
```

```
>db.updateRole("<rolename>", ...)  
⇒ modifie les roles attribués.
```

```
>db.dropRole("<rolename>")  
⇒ supprime les roles attribués.
```

```
>db.grantPrivilegesToRole( "<rolename>", [ {  
    resource: { <resource> },  
    actions: [ "<action>", ... ] } , ... ], )  
>db.grantRolesToRole( "<rolename>", [ <roles> ], )
```

⇒ donne une permission à un "rolename" qu'on défini.

Les "roles" représentent les différents niveaux d'autorisations d'accès aux données.

5 Conclusion

D'abord pour conclure, on peut noter que la commande de requêtes sur Mongodbd est facile à utiliser. En effet, il suffit de préciser la collection qui nous intéresse et puis poser les requêtes CRUD nécessaire pour obtenir un résultat. Les commandes CRUD pour créer, lire, modifier et effacer sont simples. Il suffit d'identifier les objets et les champs qui nous intéressent pour la requête.

Par contre, avant d'effacer les données, il faut être prudent car il n'est pas facile de faire le "backup" des données supprimées. Plusieurs manipulations seront à prévoir.

Ensuite, on a vu qu'il est possible de faire des requêtes sous javascript mais aussi d'exploiter d'autres types de documents sous formats JSON comme les bases de données géospatiales. Cela fait partie des principales avantages de MongoDB.

Finalement, on voit que la requête sur MongoDB est très stricte. En effet, le fait de ne pas respecter les règles de syntaxe empêchera l'exécution de la requête. Par exemple, le manque d'un guillemet à la fin d'une commande peut renvoyer une erreur. Il est donc nécessaire de bien vérifier la ligne de commande avant de la lancer. Aussi, on constate que pour des recherches plus fines, il est nécessaire d'indexer d'abord les données pour le rendre plus performante. Cela implique une manipulation supplémentaire. C'est un inconvénient important.

MongoDB est donc un outil de gestion de base de données performant et facile à utiliser mais qui a sa propre limite. Néanmoins, il fait partie des solutions NoSQL qui est la plus utilisée par les "data scientist" et "data analyst".

References

<https://docs.mongodb.com/manual/>