LAB MongoDB

Professeur: BOUBCHIR, Larbi

Module: ADI

Table des matières

I. Part 1 : INSERT DATA

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

Use library permet de sélectionner la base de données nommée "library" pour les opérations ultérieures. db affiche le nom de la base de données actuellement sélectionnée, qui est "library".

```
> 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"
        ]
}
```

Initialiser une variable nommée document avec un objet JSON qui représente un livre, incluant le type, le titre, l'ISBN, l'éditeur et une liste des auteurs.

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

db.media.insert(document) insère l'objet document dans la collection media de la base de données courante et retourne un objet WriteResult indiquant qu'une insertion a eu lieu avec succès.

```
> db.media.find()
{ "_id" : ObjectId("65a93857d31ad2619ab6dd7a"), "Type" : "Book", "Title" : "Definitive Guide to MongoD
B", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress", "Author" : [ "Membrey, Peter", "Plugge, Eelc
o", "Hawkins, Tim" ] }
>
```

db.media.find() est utilisée pour rechercher et afficher tous les documents dans la collection media de la base de données courante, ici montrant un document représentant un livre avec des détails tels que le type, le titre, l'ISBN, l'éditeur et les auteurs.

AUTRE METHODE qui réalise la même chose que précédemment :

```
> 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 })
```

db.media.insert() insère un nouveau document dans la collection media représentant un CD avec des informations sur l'artiste, le titre de l'album et une liste de pistes, puis retourne un objet WriteResult indiquant qu'une insertion a été effectuée avec succès.

```
> db.media.find()
{ "_id" : ObjectId("65a93857d31ad2619ab6dd7a"), "Type" : "Book", "Title" : "Definitive Guide to MongoD
B", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress", "Author" : [ "Membrey, Peter", "Plugge, Eelc
o", "Hawkins, Tim" ] }
{ "_id" : ObjectId("65a93875d31ad2619ab6dd7b"), "Type" : "CD", "Artist" : "Nirvana", "Title" : "Neverm
ind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like teen spirit", "Length" : "5:02 " }, { "
Track" : "2 ", "Title" : "In Bloom", "Length" : "4:15 " } ] }
>
```

db.media.find() est utilisée pour récupérer tous les documents de la collection media. Elle montre ici deux documents : un livre avec ses détails et un CD avec des informations sur l'artiste, le titre de l'album et les pistes.

1. QUERIES: WHAT DO THESE COMMANDS DO?

```
> db.media.find()
{ "_id" : ObjectId("65a93857d31ad2619ab6dd7a"), "Type" : "Book", "Title" : "Definitive Guide to MongoD
B", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress", "Author" : [ "Membrey, Peter", "Plugge, Eelc
o", "Hawkins, Tim" ] }
{ "_id" : ObjectId("65a93875d31ad2619ab6dd7b"), "Type" : "CD", "Artist" : "Nirvana", "Title" : "Neverm
ind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like teen spirit", "Length" : "5:02 " }, { "
Track" : "2 ", "Title" : "In Bloom", "Length" : "4:15 " } ] }
```

db.media.find() a récupéré deux documents de la collection media : le premier est un livre et le second un CD, chacun avec son identifiant unique et des détails comme le type, le titre, et d'autres métadonnées spécifiques.

```
> db.media.find ( { Artist : "Nirvana" } )
{ "_id" : ObjectId("65a93875d31ad2619ab6dd7b"), "Type" : "CD", "Artist" : "Nirvana", "Title" : "Neverm
ind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like teen spirit", "Length" : "5:02 " }, { "
Track" : "2 ", "Title" : "In Bloom", "Length" : "4:15 " } ] }
```

db.media.find({ Artist: "Nirvana" }) cherche dans la collection media pour les documents où l'artiste est "Nirvana", et ici elle affiche un document qui est un CD avec des détails sur les pistes.

```
> db.media.find ( {Artist : "Nirvana"}, {Title: 1} )
{ "_id" : ObjectId("65a93875d31ad2619ab6dd7b"), "Title" : "Nevermind" }
```

db.media.find({ Artist: "Nirvana"}, {Title: 1}) recherche dans la collection media pour les documents où l'artiste est "Nirvana", mais ne renvoie que le titre du document, en excluant les autres champs à l'exception de l'ID par défaut.

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

Cette commande semble essayer d'exécuter deux requêtes find sur la collection media en utilisant une opération logique, probablement pour vérifier si les deux conditions sont vraies. Cependant, la syntaxe utilisée n'est pas valide en MongoDB 5.0.23 pour combiner des requêtes de cette manière, d'où le résultat false.

```
> db.media.findOne();
{
        "_id" : ObjectId("65a93857d31ad2619ab6dd7a"),
        "Type" : "Book",
        "Title" : "Definitive Guide to MongoDB",
        "ISBN" : "987-1-4302-3051-9",
        "Publisher" : "Apress",
        "Author" : [
                "Membrey, Peter",
                "Plugge, Eelco",
                "Hawkins, Tim"
        ]
}
```

db.media.findOne() retourne le premier document trouvé dans la collection media. Ici, elle a retourné un document qui est un livre avec ses détails comme le type, le titre, l'ISBN, l'éditeur et les auteurs.

```
> db.media.find().pretty();
{
        "_id" : ObjectId("65a93857d31ad2619ab6dd7a"),
        "Type" : "Book",
        "Title" : "Definitive Guide to MongoDB",
        "ISBN" : "987-1-4302-3051-9",
        "Publisher" : "Apress",
        "Author" : [
                "Membrey, Peter",
                "Plugge, Eelco",
                "Hawkins, Tim"
        ]
}
{
        "_id" : ObjectId("65a93875d31ad2619ab6dd7b"),
        "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().pretty() est utilisée pour rechercher et afficher de manière formatée (pour une meilleure lisibilité) tous les documents dans la collection media. Cette sortie montre deux documents : un livre et un CD, chacun avec des informations détaillées comme l'ID, le type, l'artiste, le titre, etc.

2. FUNCTIONS: SORT, LIMIT AND SKIP :

```
> db.media.find().sort({Title:1});
{ "_id" : ObjectId("65a93857d31ad2619ab6dd7a"), "Type" : "Book", "Title" : "Definitive Guide to MongoD
B", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress", "Author" : [ "Membrey, Peter", "Plugge, Eelc
o", "Hawkins, Tim" ] }
{ "_id" : ObjectId("65a93875d31ad2619ab6dd7b"), "Type" : "CD", "Artist" : "Nirvana", "Title" : "Neverm
ind", "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}); effectue une recherche dans la collection media et trie les documents retournés en ordre croissant basé sur le champ Title.

```
> db.media.find().sort( { Title: -1 });
{ "_id" : ObjectId("65a93875d31ad2619ab6dd7b"), "Type" : "CD", "Artist" : "Nirvana", "Title" : "Neverm
ind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like teen spirit", "Length" : "5:02 " }, { "
Track" : "2 ", "Title" : "In Bloom", "Length" : "4:15 " } ] }
{ "_id" : ObjectId("65a93857d31ad2619ab6dd7a"), "Type" : "Book", "Title" : "Definitive Guide to MongoD
B", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress", "Author" : [ "Membrey, Peter", "Plugge, Eelc
o", "Hawkins, Tim" ] }
```

db.media.find().sort({Title: -1}); recherche dans la collection media et trie les documents par ordre décroissant basé sur le champ Title.

```
> db.media.find().limit( 10 )
{ "_id" : ObjectId("65a93857d31ad2619ab6dd7a"), "Type" : "Book", "Title" : "Definitive Guide to MongoD
B", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress", "Author" : [ "Membrey, Peter", "Plugge, Eelc
o", "Hawkins, Tim" ] }
{ "_id" : ObjectId("65a93875d31ad2619ab6dd7b"), "Type" : "CD", "Artist" : "Nirvana", "Title" : "Neverm
ind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like teen spirit", "Length" : "5:02 " }, { "
Track" : "2 ", "Title" : "In Bloom", "Length" : "4:15 " } ] }
```

db.media.find().limit(10) est utilisée pour rechercher dans la collection media et limiter les résultats retournés à 10 documents. Cependant, ici seulement deux documents sont retournés car il n'y a probablement que deux documents dans la collection.

```
> db.media.find().skip( 20 )
>
```

db.media.find().skip(20) est utilisée pour passer (ignorer) les 20 premiers documents dans la collection media et retourne les documents suivants. La sortie montre qu'il n'y a pas de documents retournés après avoir sauté les 20 premiers, ce qui suggère qu'il y a moins de 20 documents dans la collection.

```
> db.media.find().sort ( { Title : -1 } ).limit ( 10 ).skip ( 20 )
>
```

db.media.find().sort({Title: -1}).limit(10).skip(20) cherche dans la collection media, trie les documents par ordre décroissant selon le titre, limite les résultats à 10 documents après avoir sauté les 20 premiers. La sortie vide indique qu'il n'y a pas de documents à afficher après l'application de ces critères.

3. AGGREGATION :

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

db.media.count() retourne le nombre total de documents présents dans la collection media. Ici, la sortie indique qu'il y a 2 documents dans la collection.

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

db.media.find({ Publisher: "Apress", Type: "Book" }).count() compte le nombre de documents dans la collection media qui ont "Apress" comme éditeur et dont le type est "Book". La sortie indique qu'il y a 1 document correspondant à ces critères.

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

db.media.find({ Publisher: "Apress", Type: "Book" }).skip(2).count(true) cherche les documents où l'éditeur est "Apress" et le type est "Book", saute les deux premiers documents trouvés, puis compte le reste. La méthode .count(true) est obsolète et l'argument true n'est pas nécessaire; cependant, cela indique l'intention de compter tous les documents qui correspondent au critère sans tenir compte du .skip(). La sortie 0 suggère qu'il n'y a pas de documents après avoir sauté les deux premiers, ou qu'il y a eu une erreur dans l'utilisation de la méthode count.

4. DISTINCT() :

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

Déclaration de variable en MongoDB, où document est assigné un objet qui représente un livre. Cet objet contient des champs pour le type, le titre, l'ISBN, l'éditeur et une liste d'auteurs.

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

db.media.insert(document) insère l'objet document dans la collection media de la base de données MongoDB et retourne un objet WriteResult indiquant qu'un document a été inséré avec succès.

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

db.media.distinct("Title") récupère une liste de toutes les valeurs distinctes du champ Title dans la collection media. La sortie montre que les titres distincts sont "Definitive Guide to MongoDB" et "Nevermind".

```
> db.media.distinct ("ISBN")
[ "1-4302-3051-7", "987-1-4302-3051-9" ]
```

db.media.distinct("ISBN") retourne toutes les valeurs distinctes du champ ISBN de tous les documents dans la collection media. La sortie indique qu'il y a deux ISBN uniques dans la collection.

```
> db.media.distinct ("Tracklist.Title")
[ "In Bloom", "Smells like teen spirit" ]
```

db.media.distinct("Tracklist.Title") récupère toutes les valeurs distinctes pour le champ Title à l'intérieur du tableau Tracklist de la collection media. La sortie montre que "In Bloom" et "Smells Like Teen Spirit" sont les titres distincts de pistes dans la collection.

5. AGGREGATION :

```
>db.media.group ( { key: {Title : true},
initial: {Total : 0},
reduce : function (items,prev) {
prev.Total += 1 }
})
```

- Key: grouping parameter
- Initial: initial value (0 by default)
- Reduce: takes 2 arguments, the document (items) and the counter (prev) and
- performs aggregation
- Cond: condition that the attributes of the document must respect

Ne fonctionne pas. Voici une commande équivalente :

```
> db.media.group({
...    key: { Title: 1 },
...    initial: { Total: 0 },
...    reduce: function (curr, result) { result.Total += 1; }
... })
```

Effectue une opération de groupement sur la collection media. Pour chaque groupe unique déterminé par le Title, il initialise un compteur Total à 0, puis incrémente ce compteur de 1 pour chaque document trouvé dans ce groupe. Le résultat final est un ensemble de documents où chaque document représente un titre unique dans la collection media, accompagné du nombre total d'occurrences de ce titre.

6. ADD MORE RECORDS :

```
> dvd = ( { Type : "DVD", Title : "Matrix, The", Released : 1999, Cast: ["KeanuReeves","Carry-Anne Mos
s","Laurence Fishburne","Hugo Weaving","GloriaFoster","Joe Pantoliano"] } )eanuReeves","Carry-Anne Mos
{
        "Type" : "DVD",
        "Title" : "Matrix, The",
        "Released" : 1999,
        "Cast" : [
                "KeanuReeves",
                "Carry-Anne Moss",
                "Laurence Fishburne",
                "Hugo Weaving",
                "GloriaFoster",
                "Joe Pantoliano"
        ]
}
>
> db.media.insert(dvd)
WriteResult({ "nInserted" : 1 })
```

La première partie crée un objet dvd qui représente un DVD du film "The Matrix", sorti en 1999, avec une liste de membres de la distribution.

La deuxième partie est l'exécution d'une commande insert pour ajouter cet objet dvd à la collection media. La sortie indique que l'insertion a été effectuée avec succès avec un document inséré.

```
> dvd = ( { "Type" : "DVD", "Title" : "T
{ "Type" : "DVD", "Title" : "Toy Story 3
>
> db.media.insert(dvd)
WriteResult({ "nInserted" : 1 })
```

Crée un objet dvd avec des informations sur un DVD de type "DVD", de titre "Toy Story 3" et de publication 2010, puis l'insère dans la collection media de la base de données MongoDB. La sortie WriteResult({ "nInserted" : 1 }) indique que l'insertion a été réalisée avec succès.

Insert with JavaScript :

```
> function insertMedia( type, Ytle, released ){
... db.media.insert({
... "Type": type,
... "Title": Ytle,
... "Released": released
... }); }
>
> insertMedia("DVD", "Blade Runner", 1982 )
```

Définition d'une fonction insertMedia en JavaScript, qui prend trois paramètres (type, title, released) et insère un nouveau document avec ces valeurs dans la collection media de la base de données MongoDB.

Appel de la fonction insertMedia avec les arguments "DVD", "Blade Runner", et 1982 pour insérer un DVD intitulé "Blade Runner" sorti en 1982 dans la collection media.

7. COMPARISON OPERATORS :

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

```
> db.media.find ( { Released : {$gt : 2000} }, { "Cast" : 0 } )
{ "_id" : ObjectId("65a942d3d31ad2619ab6dd7e"), "Type" : "DVD", "Title" : "Toy Story 3", "Released" :
2010 }
```

db.media.find({ Released: {$gt: 2000} }, { Cast: 0 }) recherche dans la collection media pour les documents où le champ Released est supérieur à l'année 2000 et ne retourne pas le champ Cast dans les résultats. La sortie montre un document correspondant à un DVD de "Toy Story 3" qui a été publié en 2010.

```
> db.media.find( {Released : {$gte: 1990, $lt : 2010}}, { "Cast" : 0 })
{ "_id" : ObjectId("65a942b0d31ad2619ab6dd7d"), "Type" : "DVD", "Title" : "Matrix, The", "Released" :
1999 }
```

Recherche dans la collection "media" les documents où le champ "Released" est supérieur ou égal à 1990 et inférieur à 2010, et ne retourne que les champs "Cast" (avec des données probablement omises ici pour la brièveté), "Type", "Title" et "Released" de chaque document correspondant.

```
> db.media.find( { Type : "Book", Author: {$ne : "Plugge, Eelco"}})
{ "_id" : ObjectId("65a93d85d31ad2619ab6dd7c"), "Type" : "Book", "Title" : "Definitive Guide to MongoD
B", "ISBN" : "1-4302-3051-7", "Publisher" : "Apress", "Author" : [ "Membrey, Peter", "Plugge,Eelco", "
Hawkins, Tim" ] }
```

Recherche dans la collection "media" tous les documents où le champ "Type" est égal à "Book" et le champ "Author" ne contient pas la valeur "Plugge, Eelco", et retourne tous les champs des documents correspondants.

```
> db.media.find( {Released : {$in : ["1999","2008","2009"] } }, { "Cast" : 0})
>
```

Récupère tous les documents de la collection "media" dont le champ "Released" contient l'une des valeurs "1999", "2008" ou "2009", mais ne retourne que le champ "Cast" pour ces documents.

Rien n'est renvoyé, c'est normal car la date de sortie des enregistrements a été saisie avec un type numérique et non une chaîne.

Voici la requête réécrit :

```
> db.media.find({ Released: { $in: [1999, 2008, 2009] } }, { "Cast": 0 }).pretty()
{
        "_id" : ObjectId("65a97d830566f0df46eeacf3"),
        "Type" : "DVD",
        "Title" : "Matrix, The",
        "Released" : 1999
}
```

```
> db.media.find( {Released : {$nin : ["1999","2008","2009"] },Type : "DVD" }, {
... "Cast" : 0 } )
{ "_id" : ObjectId("65a942b0d31ad2619ab6dd7d"), "Type" : "DVD", "Title" : "Matrix, The", "Released" :
1999 }
{ "_id" : ObjectId("65a942d3d31ad2619ab6dd7e"), "Type" : "DVD", "Title" : "Toy Story 3", "Released" :
2010 }
{ "_id" : ObjectId("65a94310d31ad2619ab6dd7f"), "Type" : "DVD", "Title" : "Blade Runner", "Released" :
 1982 }
```

Recherche dans la collection "media" les documents où le champ "Released" est soit "1999", "2008", ou "2009", et le champ "Type" est "DVD", en excluant le champ "Cast" des résultats retournés.

$or :

```
> db.media.find({ $or : [ { "Title" : "Toy Story 3" }, { "ISBN" : "987-1-4302-3051-9" }
... ] } )
{ "_id" : ObjectId("65a93857d31ad2619ab6dd7a"), "Type" : "Book", "Title" : "Definitive Guide to MongoD
B", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress", "Author" : [ "Membrey, Peter", "Plugge, Eelc
o", "Hawkins, Tim" ] }
{ "_id" : ObjectId("65a942d3d31ad2619ab6dd7e"), "Type" : "DVD", "Title" : "Toy Story 3", "Released" :
2010 }
```

Recherche dans la collection "media" des documents qui correspondent soit au titre "Toy Story 3" soit à l'ISBN "987-1-4302-3051-9" et retourne tous les champs des documents correspondants.

```
> db.media.find({"Type":"DVD",$or:[{"Title":"Toy Story 3"},{"ISBN":"987-1-4302-3051-9"}]})
{ "_id" : ObjectId("65a942d3d31ad2619ab6dd7e"), "Type" : "DVD", "Title" : "Toy Story 3", "Released" :
2010 }
```

Recherche dans la collection "media" les documents qui sont de type "DVD" et qui ont soit un titre "Toy Story 3", soit un ISBN "987-1-4302-3051-9", et retourne tous les champs de ces documents.

8. $SLICE :

$slice: combines limit() and skip()

– $slice: [20, 10] // skip 20, limit 10

– $slice: 5 // The first 5

– $slice:-5 //The last 5

```
> db.media.find({"Title" : "Matrix, The"}, {"Cast" : {$slice: 3}})
{ "_id" : ObjectId("65a942b0d31ad2619ab6dd7d"), "Type" : "DVD", "Title" : "Matrix, The", "Released" :
1999, "Cast" : [ "KeanuReeves", "Carry-Anne Moss", "Laurence Fishburne" ] }
```

Recherche dans la collection "media" les documents avec le titre "The Matrix" et utilise l'opérateur $slice pour retourner seulement les trois premiers éléments du tableau "Cast".

```
> db.media.find({"Title" : "Matrix, The"}, {"Cast" : {$slice: -3}})
{ "_id" : ObjectId("65a942b0d31ad2619ab6dd7d"), "Type" : "DVD", "Title" : "Matrix, The", "Released" :
1999, "Cast" : [ "Hugo Weaving", "GloriaFoster", "Joe Pantoliano" ] }
```

Recherche dans la collection "media" les documents avec le titre "The Matrix" et utilise l'opérateur $slice pour retourner les trois derniers éléments du tableau "Cast".

9. $SIZE AND $EXISTS :

```
> db.media.find ( { Tracklist : {$size : 2} } )
{ "_id" : ObjectId("65a93875d31ad2619ab6dd7b"), "Type" : "CD", "Artist" : "Nirvana", "Title" : "Neverm
ind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like teen spirit", "Length" : "5:02 " }, { "
Track" : "2 ", "Title" : "In Bloom", "Length" : "4:15 " } ] }
```

Recherche dans la collection "media" les documents où le tableau "TrackList" contient exactement deux éléments, pour les documents de type "CD" de l'artiste "Nirvana" avec le titre "Nevermind", et retourne ces documents.

```
> db.media.find ( { Author : {$exists : true } } )
{ "_id" : ObjectId("65a93857d31ad2619ab6dd7a"), "Type" : "Book", "Title" : "Definitive Guide to MongoD
B", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress", "Author" : [ "Membrey, Peter", "Plugge, Eelc
o", "Hawkins, Tim" ] }
{ "_id" : ObjectId("65a93d85d31ad2619ab6dd7c"), "Type" : "Book", "Title" : "Definitive Guide to MongoD
B", "ISBN" : "1-4302-3051-7", "Publisher" : "Apress", "Author" : [ "Membrey, Peter", "Plugge,Eelco", "
Hawkins, Tim" ] }
```

Recherche dans la collection "media" tous les documents qui possèdent le champ "Author", c'est-à-dire où "Author" existe, et retourne ces documents.

```
> db.media.find ( { Author : {$exists : false } } )
{ "_id" : ObjectId("65a93875d31ad2619ab6dd7b"), "Type" : "CD", "Artist" : "Nirvana", "Title" : "Neverm
ind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like teen spirit", "Length" : "5:02 " }, { "
Track" : "2 ", "Title" : "In Bloom", "Length" : "4:15 " } ] }
{ "_id" : ObjectId("65a942b0d31ad2619ab6dd7d"), "Type" : "DVD", "Title" : "Matrix, The", "Released" :
1999, "Cast" : [ "KeanuReeves", "Carry-Anne Moss", "Laurence Fishburne", "Hugo Weaving", "GloriaFoster
", "Joe Pantoliano" ] }
{ "_id" : ObjectId("65a942d3d31ad2619ab6dd7e"), "Type" : "DVD", "Title" : "Toy Story 3", "Released" :
2010 }
{ "_id" : ObjectId("65a94310d31ad2619ab6dd7f"), "Type" : "DVD", "Title" : "Blade Runner", "Released" :
 1982 }
```

Recherche dans la collection "media" tous les documents qui ne possèdent pas le champ "Author", c'est-à-dire où le champ "Author" n'existe pas, et retourne ces documents.

10. INDEX CREATION :

Dans la version 5.0.23 de MongoDB, ensureIndex ne fonctionne pas :

```
> db.media.ensureIndex( { Title :1 } )
uncaught exception: TypeError: db.media.ensureIndex is not a function :
@(shell):1:1
```

Il faut donc utiliser createIndex :

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

Crée avec succès un index sur le champ "Title" dans la collection "media", augmentant le nombre total d'indexes de 1 à 2, et confirme que la collection existait déjà avant la création de l'index (pas créée automatiquement) avec un résultat "ok" qui indique que l'opération a réussi.

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

Crée un index décroissant sur le champ "Title" dans la collection "media", augmentant le nombre d'indexes de 2 à 3, sans créer automatiquement de collection, comme indiqué par le résultat "ok" qui signifie que l'opération a été exécutée avec succès.

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

Crée un index croissant sur le champ "Title" au sein du tableau "Tracklist" dans la collection "media", augmentant le nombre d'indexes de 3 à 4, et confirme que l'opération a réussi sans création automatique de collection.

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

Erreur car aucun index a été créé.

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

Crée un index sur le champ ISBN dans la collection media de MongoDB, et le résultat montre qu'avant la création il y avait 4 index, après il y en a 5, et qu'aucune nouvelle collection n'a été créée automatiquement lors de l'opération.

```
> db.media.find( { ISBN: "987-1-4302-3051-9"} ).hint( { ISBN: 1 } ).pretty()
{
        "_id" : ObjectId("65a944230566f0df46eeacf1"),
        "Type" : "Book",
        "Title" : "Definitive Guide to MongoDB",
        "ISBN" : "987-1-4302-3051-9",
        "Publisher" : "Apress",
        "Author" : [
                "Membrey, Peter",
                "Plugge, Eelco",
                "Hawkins, Tim"
        ]
}
```

Recherche dans la collection media un document avec un ISBN spécifique, en utilisant l'index sur le champ ISBN pour la requête, et affiche le résultat de manière lisible (formatée).

```
> 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"
        }
]
```

Montre le résultat de la commande db.media.getIndexes() dans MongoDB, qui liste tous les indexes existants sur la collection "media". Il y a un index sur le champ "_id" par défaut, un index croissant sur "Title", un index décroissant sur "Title", et un index croissant sur "Tracklist.Title".

11. DATA UPDATE :

Update (condition,newObject,upsert,multi):

− upsert=true //create the object if does not exist

− 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" : "Ac'on"}, true)
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : ObjectId("65a9475bde184fc882d39cdb")
})
```

Tente de mettre à jour un document dans la collection "media" avec le titre "Matrix, the" et le type "DVD" sorti en "1999", pour ajouter le genre "Action". Aucun document correspondant n'a été trouvé (nMatched: 0), donc un nouveau document a été inséré (nUpserted: 1) puisque l'option upsert était vraie (true).

Add/delete an aQribute :

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

Met à jour le genre d'un document existant dans la collection "media" avec le titre "Matrix, the" pour être "Sci-Fi". Elle a trouvé un document correspondant (nMatched: 1), rien n'a été inséré (nUpserted: 0), et un document a été modifié (nModified: 1).

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

Utilise l'opération $unset pour supprimer le champ "Genre" d'un document dans la collection "media" avec le titre "Matrix, the". Un document correspondant a été trouvé (nMatched: 1), aucun nouveau document n'a été inséré (nUpserted: 0), et le champ a été retiré avec succès (nModified: 1).

Delete:

− Documents meeting a condition: >db.media.remove({ "Title" : "Different Title" })

− All documents: >db.media.remove({})

```
> db.media.remove({})
WriteResult({ "nRemoved" : 7 })
```

db.media.remove({}) a été exécutée pour supprimer tous les documents de la collection "media", avec le résultat indiquant que 7 documents ont été supprimés (nRemoved: 7).

− All the collec'on: >db.media.drop()

```
> db.media.drop()
true
```

db.media.drop() a été utilisée pour supprimer la collection "media" entièrement de la base de données, et la commande a retourné true, ce qui indique que l'opération a été exécutée avec succès.

II. Part 2: GEOGRAPHIC DATABASE

```
PS C:\Users\Morgan> cd D:\Téléchargements\mongodb-windows-x86_64-5.0.23\mongodb-win32-x86_64-windows-5.0.23\bin
```

Nous utilisons cd pour accéder au dossier bin de notre dossier mongodb.

```
PS D:\Téléchargements\mongodb-windows-x86_64-5.0.23\mongodb-win32-x86_64-windows-5.0.23\bin> mongoimport --type json -d
geodb -c earthquakes --file D:\Bureau\NOSQL\Cours-TP-MongoDB-20240118\earthquakes.json
2024-01-18T20:51:39.726+0100    connected to: mongodb://localhost/
2024-01-18T20:51:39.972+0100    7669 document(s) imported successfully. 0 document(s) failed to import.
```

Nous exécutons la commande nous permettant d'importer la base de données désiré.

```
> show dbs
Advanced_Data_Base   0.000GB
admin                0.000GB
config               0.000GB
geodb                0.004GB
local                0.000GB
persons              0.000GB
```

Une fois fait, nous retournons sur le shell MongoDB pour voir si geodb a bien été importé.

geodb, à bien été importé, nous pouvons dès à présent l'utiliser.

```
> use geodb
switched to db geodb
> db
geodb
```

1. Some basic queries:

Count the number of documents:

```
> db.earthquakes.countDocuments({})
15338
>
```

Show first 5:

```
> db.earthquakes.find().limit(5)
{ "_id" : ObjectId("65a98128cf99fae65da85517"), "type" : "Feature", "properties" : { "mag" : 1.8, "place" : "9km NW of Greenville, California", "time" : NumberLong("137024
9916100"), "updated" : NumberLong("1370255651319"), "tz" : -420, "url" : "http://earthquake.usgs.gov/earthquakes/eventpage/nc72001575", "detail" : "http://earthquake.usgs.
gov/earthquakes/feed/v1.0/detail/nc72001575.geojson", "felt" : null, "cdi" : null, "mmi" : null, "alert" : null, "status" : "AUTOMATIC", "tsunami" : null, "sig" : 50, "net
" : "nc", "code" : "72001575", "ids" : ",nc72001575,", "sources" : ",nc,", "types" : ",general-link,geoserve,nearby-cities,origin,phase-data,scitech-link,", "nst" : null,
"dmin" : 0.34135981, "rms" : 0.05, "gap" : 172.8, "magType" : "Md", "type" : "earthquake" }, "geometry" : { "type" : "Point", "coordinates" : [ -121.0372, 40.189, 4.9 ] },
 "id" : "nc72001575" }
{ "_id" : ObjectId("65a98128cf99fae65da85518"), "type" : "Feature", "properties" : { "mag" : 1.4, "place" : "14km NE of Fritz Creek, Alaska", "time" : NumberLong("13702564
08000"), "updated" : NumberLong("1370256968290"), "tz" : -480, "url" : "http://earthquake.usgs.gov/earthquakes/eventpage/ak10729205", "detail" : "http://earthquake.usgs.go
v/earthquakes/feed/v1.0/detail/ak10729205.geojson", "felt" : null, "cdi" : null, "mmi" : null, "alert" : null, "status" : "AUTOMATIC", "tsunami" : null, "sig" : 30, "net"
: "ak", "code" : "10729205", "ids" : ",ak10729205,", "sources" : ",ak,", "types" : ",general-link,geoserve,nearby-cities,origin,tectonic-summary,", "nst" : null, "dmin" :
null, "rms" : 0.46, "gap" : null, "magType" : "Ml", "type" : "earthquake" }, "geometry" : { "type" : "Point", "coordinates" : [ -151.1041, 59.8294, 41.6 ] }, "id" : "ak107
29205" }
{ "_id" : ObjectId("65a98128cf99fae65da85519"), "type" : "Feature", "properties" : { "mag" : 0.9, "place" : "63km ENE of Talkeetna, Alaska", "time" : NumberLong("137025209
4000"), "updated" : NumberLong("1370252564508"), "tz" : -480, "url" : "http://earthquake.usgs.gov/earthquakes/eventpage/ak10729194", "detail" : "http://earthquake.usgs.gov
/earthquakes/feed/v1.0/detail/ak10729194.geojson", "felt" : null, "cdi" : null, "mmi" : null, "alert" : null, "status" : "AUTOMATIC", "tsunami" : null, "sig" : 12, "net" :
 "ak", "code" : "10729194", "ids" : ",ak10729194,", "sources" : ",ak,", "types" : ",general-link,geoserve,nearby-cities,origin,tectonic-summary,", "nst" : null, "dmin" : n
ull, "rms" : 0.96, "gap" : null, "magType" : "Ml", "type" : "earthquake" }, "geometry" : { "type" : "Point", "coordinates" : [ -149.0512, 62.6141, 58 ] }, "id" : "ak107291
94" }
{ "_id" : ObjectId("65a98128cf99fae65da8551a"), "type" : "Feature", "properties" : { "mag" : 2.1, "place" : "82km ENE of Cantwell, Alaska", "time" : NumberLong("1370250728
000"), "updated" : NumberLong("1370251862751"), "tz" : -480, "url" : "http://earthquake.usgs.gov/earthquakes/eventpage/ak10729187", "detail" : "http://earthquake.usgs.gov/
earthquakes/feed/v1.0/detail/ak10729187.geojson", "felt" : null, "cdi" : null, "mmi" : null, "alert" : null, "status" : "AUTOMATIC", "tsunami" : null, "sig" : 68, "net" :
"ak", "code" : "10729187", "ids" : ",ak10729187,", "sources" : ",ak,", "types" : ",general-link,geoserve,nearby-cities,origin,tectonic-summary,", "nst" : null, "dmin" : nu
ll, "rms" : 0.53, "gap" : null, "magType" : "Ml", "type" : "earthquake" }, "geometry" : { "type" : "Point", "coordinates" : [ -147.3555, 63.5948, 0.1 ] }, "id" : "ak107291
87" }
{ "_id" : ObjectId("65a98128cf99fae65da8551b"), "type" : "Feature", "properties" : { "mag" : 1.3, "place" : "76km NNW of Talkeetna, Alaska", "time" : NumberLong("137024991
3000"), "updated" : NumberLong("1370251256124"), "tz" : -480, "url" : "http://earthquake.usgs.gov/earthquakes/eventpage/ak10729181", "detail" : "http://earthquake.usgs.gov
/earthquakes/feed/v1.0/detail/ak10729181.geojson", "felt" : null, "cdi" : null, "mmi" : null, "alert" : null, "status" : "AUTOMATIC", "tsunami" : null, "sig" : 26, "net" :
 "ak", "code" : "10729181", "ids" : ",ak10729181,", "sources" : ",ak,", "types" : ",general-link,geoserve,nearby-cities,origin,tectonic-summary,", "nst" : null, "dmin" : n
ull, "rms" : 0.87, "gap" : null, "magType" : "Ml", "type" : "earthquake" }, "geometry" : { "type" : "Point", "coordinates" : [ -150.7789, 62.9376, 78.6 ] }, "id" : "ak1072
9181" }
>
```

View the 6th document:

```
> db.earthquakes.find().skip(5).limit(1)
{ "_id" : ObjectId("65aa31f4e6bbb84c753a4149"), "type" : "Feature", "properties" : { "mag" : 0.8, "place" : "34km WSW of
 North Nenana, Alaska", "time" : NumberLong("1370258440000"), "updated" : NumberLong("1370259002731"), "tz" : -480, "url
" : "http://earthquake.usgs.gov/earthquakes/eventpage/ak10729207", "detail" : "http://earthquake.usgs.gov/earthquakes/fe
ed/v1.0/detail/ak10729207.geojson", "felt" : null, "cdi" : null, "mmi" : null, "alert" : null, "status" : "AUTOMATIC", "
tsunami" : null, "sig" : 10, "net" : "ak", "code" : "10729207", "ids" : ",ak10729207,", "sources" : ",ak,", "types" : ",
general-link,geoserve,nearby-cities,origin,", "nst" : null, "dmin" : null, "rms" : 0.06, "gap" : null, "magType" : "Ml",
 "type" : "earthquake" }, "geometry" : { "type" : "Point", "coordinates" : [ -149.7232, 64.4106, 13.2 ] }, "id" : "ak107
29207" }
>
```

How many separate statuses exist in the DB?:

```
> db.earthquakes.aggregate([
...     { $group: { _id: "$properties.status" } },
...     { $count: "distinctStatusCount" }
... ])
{ "distinctStatusCount" : 4 }
>
```

2. MODIFICATION OF DATA

− How many documents contain the property felt (propriety.felt! = Null) ? :

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

− Delete this field for documents for which it is null:

```
> db.earthquakes.updateMany({ "properties.felt": null }, { $unset: { "properties.felt": "" } })
{ "acknowledged" : true, "matchedCount" : 6868, "modifiedCount" : 6868 }
>
```

− Add an iso_date column whose value is the conversion of the timestamp contained in properties.time:

```
> db.earthquakes.updateMany({}, { $set: { "properties.iso_date": { $toDate: "$properties.time" } } })
{ "acknowledged" : true, "matchedCount" : 7669, "modifiedCount" : 7669 }
>
```

Réalisation de la même chose mes avec une commande moins performante :

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

Verification que les champs iso_date ont été correctement ajoutés :

```
> db.earthquakes.find({}, { "properties.iso_date": 1 }).limit(5)
{ "_id" : ObjectId("65aa31f4e6bbb84c753a4144"), "properties" : { "iso_date" : ISODate("2013-06-03T11:46:08Z") } }
{ "_id" : ObjectId("65aa31f4e6bbb84c753a4145"), "properties" : { "iso_date" : ISODate("2013-06-03T11:45:41.100Z") } }
{ "_id" : ObjectId("65aa31f4e6bbb84c753a4146"), "properties" : { "iso_date" : ISODate("2013-06-03T11:26:52.900Z") } }
{ "_id" : ObjectId("65aa31f4e6bbb84c753a4147"), "properties" : { "iso_date" : ISODate("2013-06-03T11:41:03Z") } }
{ "_id" : ObjectId("65aa31f4e6bbb84c753a4148"), "properties" : { "iso_date" : ISODate("2013-06-03T10:36:10.900Z") } }
>
```

3. 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);
... });
>
```

db.earthquakes.find().forEach(func]on(eq){

var str = new String(eq.proper]es.types);

eq.proper]es.types_as_array = str.split(",");

db.earthquakes.save(eq); });

```
> db.earthquakes.findOne({}, { "properties.types_as_array": 1 })
{
        "_id" : ObjectId("65aa31f4e6bbb84c753a4144"),
        "properties" : {
                "types_as_array" : [
                        "",
                        "general-link",
                        "geoserve",
                        "nearby-cities",
                        "origin",
                        "phase-data",
                        "scitech-link",
                        ""
                ]
        }
}
>
```

Check by showing the 1st document:

```
> db.earthquakes.updateMany({}, { $set: { "properties.types_as_array": { $split: ["$properties.types", ","] } } })
{ "acknowledged" : true, "matchedCount" : 7669, "modifiedCount" : 7669 }
>
```

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

```
> db.earthquakes.updateMany(
...   {},
...   { $pull: { "properties.types_as_array": "" } }
... )
WriteError({
        "index" : 0,
        "code" : 2,
        "errmsg" : "Cannot apply $pull to a non-array value",
        "op" : {
                "q" : {

                },
                "u" : {
                        "$pull" : {
                                "properties.types_as_array" : ""
                        }
                },
                "multi" : true,
                "upsert" : false
        }
}) :
WriteError({
        "index" : 0,
        "code" : 2,
        "errmsg" : "Cannot apply $pull to a non-array value",
        "op" : {
                "q" : {

                },
                "u" : {
                        "$pull" : {
                                "properties.types_as_array" : ""
                        }
                },
                "multi" : true,
                "upsert" : false
        }
})
```

```
WriteError@src/mongo/shell/bulk_api.js:465:48
mergeBatchResults@src/mongo/shell/bulk_api.js:871:49
executeBatch@src/mongo/shell/bulk_api.js:940:13
Bulk/this.execute@src/mongo/shell/bulk_api.js:1182:21
DBCollection.prototype.updateMany@src/mongo/shell/crud_api.js:690:17
@(shell):1:1
>
```

L'erreur indique que l'opération $pull ne peut pas être appliquée car elle rencontre une valeur qui n'est pas un tableau. Cela signifie que dans au moins un document de notre collection, le champ properties.types_as_array n'est pas un tableau. Pour résoudre ce problème, nous devons nous assurer que l'opération $pull est appliquée uniquement aux documents ou properties.types_as_array est un tableau.

Une solution consiste à ajouter un critère de sélection pour ne mettre à jour que les documents où properties.types_as_array est un tableau.

Voici les commandes à utiliser :

```
> db.earthquakes.updateMany(
...   {},
...   [{ $set: { "properties.types_as_array": { $split: ["$properties.types", ","] } } }]
... )
{ "acknowledged" : true, "matchedCount" : 7669, "modifiedCount" : 7669 }
>
```

Convertir la chaîne de caractères dans properties.types en un tableau et le stocker dans properties.types_as_array.

```
> db.earthquakes.updateMany(
...   { "properties.types_as_array": { $exists: true, $type: "array" } },
...   { $pull: { "properties.types_as_array": "" } }
... )
{ "acknowledged" : true, "matchedCount" : 7669, "modifiedCount" : 7669 }
>
```

Nettoyer les éléments vides ("") du tableau properties.types_as_array.

```
> db.earthquakes.findOne({}, { "properties.types_as_array": 1 })
{
        "_id" : ObjectId("65aa31f4e6bbb84c753a4144"),
        "properties" : {
                "types_as_array" : [
                        "general-link",
                        "geoserve",
                        "nearby-cities",
                        "origin",
                        "phase-data",
                        "scitech-link"
                ]
        }
}
>
```

Vérifié le premier document pour confirmer que le champ types_as_array ne contient plus d'éléments vides.

4. QUERIES

- Give the number of documents whose type list (properties.type_as_array)

contains "geoserve" and "tectonic- summary":

```
> db.earthquakes.countDocuments({
...   "properties.types_as_array": { $all: ["geoserve", "tectonic-summary"] }
... })
2954
```

- Give the number of documents whose type list (properties.type_as_array)

contains "geoserve" or "tectonic- summary":

```
> db.earthquakes.countDocuments({
...   "properties.types_as_array": { $in: ["geoserve", "tectonic-summary"] }
... })
7669
```

5. GEOGRAPHIC INDEXING

We will now modify the data in order to adapt the geographical coordinates to the

format that will allow us to build a 2dsphere index.

Normalize the data by removing the last element from the 'geometry.coordinates'

table and copy it to a 'depth' field.

```
> db.earthquakes.updateMany(
...   { "geometry.coordinates": { $exists: true, $type: "array" } },
...   [
...     {
...       $set: {
...         depth: { $arrayElemAt: ["$geometry.coordinates", -1] },
...         "geometry.coordinates": { $slice: ["$geometry.coordinates", 2] }
...       }
...     }
...   ]
... )
{ "acknowledged" : true, "matchedCount" : 7669, "modifiedCount" : 7669 }
>
```

```
> db.earthquakes.findOne({}, { geometry: 1, depth: 1 })
{
        "_id" : ObjectId("65aa31f4e6bbb84c753a4144"),
        "geometry" : {
                "type" : "Point",
                "coordinates" : [
                        -122.7955,
                        38.8232
                ]
        },
        "depth" : 3
}
>
```

6. INDEX CREATION

Create a 2dsphere type index on « geometry » a_ributes

Queries:

Execute a query that looks for earthquakes near -74, 40.74 (within a radius of

1000m) (use $geoWithin and center):

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

```
> db.earthquakes.find({
...   "geometry": {
...     $geoWithin: {
...       $centerSphere: [[-74, 40.74], 1000 / 6378.1]
...     }
...   }
... })
{ "_id" : ObjectId("65aa31f4e6bbb84c753a53b6"), "type" : "Feature", "properties" : { "mag" : 2.3, "place" : "9km SSW of Louisa, Virginia", "time" : NumberLo
ng("1368615708200"), "updated" : NumberLong("1369307853810"), "tz" : -240, "url" : "http://earthquake.usgs.gov/earthquakes/eventpage/se051513b", "detail" :
"http://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/se051513b.geojson", "felt" : 55, "cdi" : 3.2, "mmi" : null, "alert" : null, "status" : "REVIEWED",
"tsunami" : null, "sig" : 99, "net" : "se", "code" : "051513b", "ids" : ",usc000gx7i,se051513b,", "sources" : ",us,se,", "types" : ",cap,dyfi,general-link,g
eoserve,impact-text,nearby-cities,origin,p-wave-travel-times,phase-data,scitech-link,tectonic-summary,", "nst" : 6, "dmin" : 0.09881468, "rms" : 0.27, "gap"
 : 154.8, "magType" : "MbLg", "type" : "earthquake", "iso_date" : ISODate("2013-05-15T11:01:48.200Z"), "types_as_array" : [ "cap", "dyfi", "general-link", "
geoserve", "impact-text", "nearby-cities", "origin", "p-wave-travel-times", "phase-data", "scitech-link", "tectonic-summary" ] }, "geometry" : { "type" : "P
oint", "coordinates" : [ -78.0063, 37.9541 ] }, "id" : "se051513b", "depth" : 0.2 }
```

Documentaton :

http://docs.mongodb.org/manual/reference/operator/query- geospatial/

7. QUERY

Find the earthquakes that are around the square '8km NW of Cobb, California'

with a maximum distance of 500km:

```
> db.earthquakes.find({
...   "geometry": {
...     $near: {
...       $geometry: {
...         type: "Point",
...         coordinates: [-122.730, 38.820]
...       },
...       $maxDistance: 500000
...     }
...   }
... })
{ "_id" : ObjectId("65aa31f4e6bbb84c753a4c53"), "type" : "Feature", "properties" : { "mag" : 0.5, "place" : "1km WSW of Cobb, California", "time" : NumberLo
ng("1369191866800"), "updated" : NumberLong("1369192811347"), "tz" : -420, "url" : "http://earthquake.usgs.gov/earthquakes/eventpage/nc71995736", "detail" :
 "http://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/nc71995736.geojson", "cdi" : null, "mmi" : null, "alert" : null, "status" : "AUTOMATIC", "tsunami"
 : null, "sig" : 4, "net" : "nc", "code" : "71995736", "ids" : ",nc71995736,", "sources" : ",nc,", "types" : ",general-link,geoserve,nearby-cities,origin,ph
ase-data,scitech-link,", "nst" : null, "dmin" : 0.02694946, "rms" : 0.03, "gap" : 176.4, "magType" : "Md", "type" : "earthquake", "iso_date" : ISODate("2013
-05-22T03:04:26.800Z"), "types_as_array" : [ "general-link", "geoserve", "nearby-cities", "origin", "phase-data", "scitech-link" ] }, "geometry" : { "type"
: "Point", "coordinates" : [ -122.74, 38.8142 ] }, "id" : "nc71995736", "depth" : 4.3 }
{ "_id" : ObjectId("65aa31f4e6bbb84c753a5e9e"), "type" : "Feature", "properties" : { "mag" : 0.9, "place" : "1km SW of Cobb, California", "time" : NumberLon
g("1367741117300"), "updated" : NumberLong("1367743332260"), "tz" : -420, "url" : "http://earthquake.usgs.gov/earthquakes/eventpage/nc71985751", "detail" :
"http://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/nc71985751.geojson", "cdi" : null, "mmi" : null, "alert" : null, "status" : "AUTOMATIC", "tsunami"
: null, "sig" : 12, "net" : "nc", "code" : "71985751", "ids" : ",nc71985751,", "sources" : ",nc,", "types" : ",general-link,geoserve,nearby-cities,origin,sc
itech-link,", "nst" : null, "dmin" : 0.00898315, "rms" : 0.04, "gap" : 86.4, "magType" : "Md", "type" : "earthquake", "iso_date" : ISODate("2013-05-05T08:05
:17.300Z"), "types_as_array" : [ "general-link", "geoserve", "nearby-cities", "origin", "scitech-link" ] }, "geometry" : { "type" : "Point", "coordinates" :
 [ -122.7367, 38.8117 ] }, "id" : "nc71985751", "depth" : 2.9 }
```

8. AGGREGATE

Orderly sequence of operators

• Command :

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

• Operators : //equivalent 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,

```
> db.earthquakes.aggregate([
...   { $match: { "properties.mag": { $gt: 5 }, "properties.time": { $gt: new Date("2000-01-01").getTime() } } },
...   { $project: {
...       year: { $year: { $toDate: "$properties.time" } },
...       type: "$properties.type"
...   } },
...   { $unwind: "$type" },
...   { $group: {
...       _id: { year: "$year", type: "$type" },
...       count: { $sum: 1 }
...   } },
...   { $sort: { "_id.year": 1, count: -1 } },
...   { $limit: 10 }
... ]);
{ "_id" : { "year" : 2013, "type" : "earthquake" }, "count" : 99 }
>
```

Requête d'agrégation sur la collection earthquakes filtre les séismes d'une magnitude supérieure à 5 et survenus après le 1er janvier 2000, groupe les résultats par année et type, compte leur nombre, trie les groupes par nombre décroissant et limite les résultats aux 10 premiers groupes.

9. AGGREGATE: EXAMPLES

Sequence: result of one opera]on serves as input for the next

> db.earthquakes.aggregate([{$match : { "proper]es.type" : "quarry"}},

{$project : { "_id" : 1, "geometry" : 1}},

{$sort : { "depth" : -1}}

]);

$unwind

Create a document for each instance

>db.earthquakes.aggregate([{$unwind :"$proper]es.types_as_array"},{$limit:5}])

10. AGGREGATE: EXAMPLES

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"}}}]);

Example: sequence

> db.movies.aggregate([

{$match: { "year" : {$gt : 1995}}},

{$unwind : "$genres_as_array"},

{$group : {"_id" : "$year", "count": {$sum: 1}}},

{$match : {"count" : {$gt : 2}}},

{$sort : { "count" : -1}]);

11. ADMINISTRATION

Backup

>mkdir testmongobackup

>cd testmongobackup

>../mongodb/bin/mongodump --help

>../mongo/bin/mongodump

>../mongodump --db library --collecYon media

à ./dump/[databasename]/[collecYonname].bson

Restore

>cd testmongobackup

>../mongo/bin/mongorestore –help

• Tout restaurer

>../mongo/bin/mongorestore –drop

• Restaurer une seule collecYon

>../mongo/bin/mongorestore –d library -c media –drop

12. SECURITY

Authentication

• Client side

> use admin

>db.addUser(“admin”, “adminpassword”)

• Server side

> use admin

>db.addUser(“admin”, “adminpassword”)

• Shell (Restart the server)

>sudo service mongodb restart

or

>db.shutdownServer()

• Authenticate

>use admin

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

>use library

>db.addUser(“ronaldo”, “ronaldopassword”)

>db.addUser(“messi”, “messipassword”,true) //read only

>db.removeUser(“ronaldo”)

13. 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" }
]
})
db.dropUser("reportsUser")
db.dropAllUsers( )
```

14. CREATE ROLE

```
db.createRole({
role: "<name>",
privileges: [ { resource: { <resource> }, acIons: [ "<acIon>", ... ] }, ... ],
roles: [ { role: "<role>", db: "<database>" } | "<role>", ... ] })
db.updateRole("< rolename >",
...)
db.dropRole("< rolename >")
db.grantPrivilegesToRole( "< rolename >",
[
{ resource: { <resource> }, acIons: [ "<acIon>", ... ] },
... ],
)
db.grantRolesToRole( "<rolename>", [ <roles> ],)
```

15. FOR MORE INFORMATION:

Resource	Location
Case Studies	mongodb.com/customers
Presentations	mongodb.com/presentations
Free Online Training	university.mongodb.com
Webinars and Events	mongodb.com/events
Documentation	docs.mongodb.org
MongoDB Downloads	mongodb.com/download
Additional Info	info@mongodb.com