HARMANTEPE MELIS

TP1 - MongoDB

Partie 1:

use : si la base de donnée existe, il va l'ouvrir, si elle n'existe pas il va la créer

Automatiquement db est égal à library. C'est un alias pour faciliter l'écriture de nos requêtes car écrire le vrai nom de la base de données à chaque fois peut être fastidieux.

On peut directement insérer un document dans la collection "media" avec **db.media.insert()**

```
> db.media.find()
{ "_id" : ObjectId("63c7c67dabc3b8666b366a70"), "Type" : "Book", "Title" : "Defi
nitive Guide to MongoDB", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress",
"Author" : [ "Membrey, Peter", "Plugge, Eelco", "Hawkins, Tim" ] }
{ "_id" : ObjectId("63c7c707abc3b8666b366a71"), "Type" : "CD", "Artist" : "Nirva
na", "Title" : "Nevermind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells
like teen spirit", "Length" : "5:02 " }, { "Track" : "2 ", "Title" : "In Bloom",
 "Length" : "4:15 " } ] }
>
```

QUERIES: WHAT DO THESE COMMANDS DO?

On peut directement chercher une donnée par son ou ses field(s) grâce à la méthode **.find()**

On accède au field "Title" de la collection "Tracklist" avec Tracklist.Title

```
> db.media.find ( { Artist : "Nirvana" } )
{ "_id" : ObjectId("63c7c707abc3b8666b366a71"), "Type" : "CD", "Artist" : "Nirvana",
 "Title" : "Nevermind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like tee
n spirit", "Length" : "5:02 " }, { "Track" : "2 ", "Title" : "In Bloom", "Length" :
"4:15 " } ] }
> db.media.find ( { Artist : "Nirvana" }, {Title:1} )
{ "_id" : ObjectId("63c7c707abc3b8666b366a71"), "Title" : "Nevermind" }
> db.media.find ( { Artist : "Nirvana" }, {Title:0} )
{ "_id" : ObjectId("63c7c707abc3b8666b366a71"), "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" } )
{ "_id" : ObjectId("63c7c707abc3b8666b366a71"), "Type" : "CD", "Artist" : "Nirvana",
 "Title" : "Nevermind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like tee
n spirit", "Length" : "5:02 " }, { "Track" : "2 ", "Title" : "In Bloom", "Length" :
"4:15 " } ] }
> db.media.findOne()
{
        "_id" : ObjectId("63c7c67dabc3b8666b366a70"),
        "Type" : "Book",
        "Title" : "Definitive Guide to MongoDB",
        "ISBN" : "987-1-4302-3051-9",
        "Publisher" : "Apress",
        "Author" : [
                "Membrey, Peter",
                "Plugge, Eelco",
                "Hawkins, Tim"
        ]
}
```

HARMANTEPE MELIS

La méthode **.findOne()** retourne le premier document de la base de données.

Comparé au **db.media.find(),** l'ajout de la méthode **.pretty()** rend le document plus lisible.

```
> db.media.find().pretty()
{
        "_id" : ObjectId("63c7c67dabc3b8666b366a70"),
        "Type" : "Book",
        "Title" : "Definitive Guide to MongoDB",
        "ISBN" : "987-1-4302-3051-9",
        "Publisher" : "Apress",
        "Author" : [
                "Membrey, Peter",
                "Plugge, Eelco",
                "Hawkins, Tim"
        ]
}
{
        "_id" : ObjectId("63c7c707abc3b8666b366a71"),
        "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 "
                }
        ]
}
>
```

FUNCTIONS: SORT, LIMIT AND SKIP

db.media.find().sort({ Title: 1 }) => sorte le contenu de la base de donnée dans l'ordre de la création des collections. Alors qu'avec { Title: -1 } on affiche les collections dans l'ordre inverse de création.

```
> db.media.find().sort( { Title: 1 }).pretty()
{
        "_id" : ObjectId("63c7c67dabc3b8666b366a70"),
        "Type" : "Book",
        "Title" : "Definitive Guide to MongoDB",
        "ISBN" : "987-1-4302-3051-9",
        "Publisher" : "Apress",
        "Author" : [
                "Membrey, Peter",
                "Plugge, Eelco",
                "Hawkins, Tim"
        ]
}
{
        "_id" : ObjectId("63c7c707abc3b8666b366a71"),
        "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 }).pretty()
{
        "_id" : ObjectId("63c7c707abc3b8666b366a71"),
        "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("63c7c67dabc3b8666b366a70"),
        "Type" : "Book",
        "Title" : "Definitive Guide to MongoDB",
        "ISBN" : "987-1-4302-3051-9",
        "Publisher" : "Apress",
```

.limit() méthode limite le nombre de documents qu'on souhaite afficher. Donc si on met limit(10) on va afficher tous les documents car on en a que 2 dans notre base de données.

.skip() prend en paramètre le nombre de document à sauter. Donc si on souhaite "skip" 20 documents, ça veut dire qu'on va rien afficher car notre base de données contient seulement 2 documents.

```
> db.media.find().limit( 10 )
{ "_id" : ObjectId("63c7c67dabc3b8666b366a70"), "Type" : "Book", "Title" : "Definiti
ve Guide to MongoDB", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress", "Author"
 : [ "Membrey, Peter", "Plugge, Eelco", "Hawkins, Tim" ] }
{ "_id" : ObjectId("63c7c707abc3b8666b366a71"), "Type" : "CD", "Artist" : "Nirvana",
 "Title" : "Nevermind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like tee
n spirit", "Length" : "5:02 " }, { "Track" : "2 ", "Title" : "In Bloom", "Length" :
"4:15 " } ] }
> db.media.find().skip( 20 )
> db.media.find().sort ( { Title : -1 } ).limit ( 10 ).skip ( 20 )
>
```

AGGREGATION

.count() compte le nombre de documents dans notre base de données.

```
> db.media.count()
2
> db.media.find( { Publisher : "Apress", Type: "Book" } ).count()
1
> db.media.find( { Publisher: "Apress", Type: "Book" }).skip(2).count(true)
0
>
```

DISTINCT()

```
> db.media.distinct( "Title")
[ "Definitive Guide to MongoDB", "Nevermind" ]
> db.media.distinct ("ISBN")
[ "1- 4302-3051-7", "987-1-4302-3051-9" ]
> db.media.distinct ("Tracklist.Title")
[ "In Bloom", "Smells like teen spirit" ]
>
```

.distinct() trouve les valeurs distinctes d'un champ (field) donné.

AGGREGATION
La commande .group() ne fait plus partie de la version 5 du Mongo.

ADD MORE RECORDS
On peut écrire une fonction en java qui prend en paramètre les attributs de nos colonnes et appeler cette fonction pour ajouter un document avec les valeurs qu'on souhaite.

COMPARISON OPERATORS

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

HARMANTEPE MELIS

db.media.find ({ Released : {$gt : 2000} }, { "Cast" : 0 }) => cherche le document qui contient une date de “Released” plus grande que 2000 et en ne prenant pas le champ “Cast” car on utilise "Cast" : 0. Si on avait mit "Cast" : 1, cela pourrait nous afficher les données de cast.
db.media.find({Released : {$gte: 1990, $lt : 2010}}, { "Cast" : 0 }) => cherche un document qui contient une date de “Released” plus grande ou égal à 1990 et plus petit que 2010 en ne prenant pas le champ “Cast”.

db.media.find({ Type : "Book", Author: {$ne : "Plugge, Eelco”}}) => retourne le(s) document(s) qui ne contiennent pas la valeur "Plugge, Eelco” dans leur champ Author et qui sont de type “Book”.

db.media.find({Released : {$in : ["1999","2008","2009"] } }, { "Cast" : 0}) => retourne le(s) document(s) qui contient des valeurs dans son champ “Released” soit 1999, soit 2008 ou soit 2009. On n’affiche pas les information de cast.

db.media.find({Released : {$nin : ["1999","2008","2009"] },Type : "DVD" }, { "Cast" : 0 }) => retourne le(s) document(s) qui ne contient pas des valeurs dans son champ “Released” comme 1999, 2008 ou 2009. On n’affiche pas les information de cast.

db.media.find({ $or : [{ "Title" : "Toy Story 3" }, { "ISBN" : "987-1-4302-3051-9" }]}) => retourne le(s) document(s) qui a soit un Title égal à “Toy Story 3”, soit qui a ISBN égal à “987-1-4302-3051-9".

db.media.find({"Type": "DVD", $or : [{ "Title" : "Toy Story 3" }, { "ISBN" : "987-1-4302-3051-9" }]}) => retourne le(s) document(s) qui a le Type DVD obligatoirement, et parmi ces documents elle retourne le document qui a un title égal à toy story 2 ou un isbn égal à 987-1-4302-3051-9.

$SLICE

$slice: combines **limit() and skip()**

db.media.find({"Title" : "Matrix, The"}, {"Cast" : {$slice: 3}}) => retourne le(s) document(s) qui ont le title "Matrix, The” en ne prenant que les 3 premiers elements de cast.

db.media.find({"Title" : "Matrix, The"}, {"Cast" : {$slice: -3}}) => retourne le(s) document(s) qui ont le title "Matrix, The” en ne prenant que les 3 derniers elements de cast.

$SIZE AND $EXISTS

```
> db.media.find ( { Tracklist : {$size : 2} } )
{ "_id" : ObjectId("63c7c707abc3b8666b366a71"), "Type" : "CD", "Artist" : "Nirvana",
 "Title" : "Nevermind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like tee
n spirit", "Length" : "5:02 " }, { "Track" : "2 ", "Title" : "In Bloom", "Length" :
"4:15 " } ] }
> db.media.find ( { Author : {$exists : true } } )
{ "_id" : ObjectId("63c7c67dabc3b8666b366a70"), "Type" : "Book", "Title" : "Definiti
ve Guide to MongoDB", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress", "Author"
 : [ "Membrey, Peter", "Plugge, Eelco", "Hawkins, Tim" ] }
{ "_id" : ObjectId("63c7d679abc3b8666b366a72"), "Type" : "Book", "Title" : "Definiti
ve Guide to MongoDB", "ISBN" : "1- 4302-3051-7", "Publisher" : "Apress", "Author" :
[ "Membrey, Peter", "Plugge, Eelco", "Hawkins, Tim" ] }
> db.media.find ( { Author : {$exists : false } } )
{ "_id" : ObjectId("63c7c707abc3b8666b366a71"), "Type" : "CD", "Artist" : "Nirvana",
 "Title" : "Nevermind", "Tracklist" : [ { "Track" : "1 ", "Title" : "Smells like tee
n spirit", "Length" : "5:02 " }, { "Track" : "2 ", "Title" : "In Bloom", "Length" :
"4:15 " } ] }
{ "_id" : ObjectId("63c7f363abc3b8666b366a73"), "Type" : "DVD", "Title" : "Matrix, T
he", "Released" : 1999, "Cast" : [ "Keanu Reeves", "Carry-Anne Moss", "Laurence Fish
burne", "Hugo Weaving", "Gloria Foster", "Joe Pantoliano" ] }
{ "_id" : ObjectId("63c7f373abc3b8666b366a74"), "Type" : "DVD", "Title" : "Toy Story
 3", "Released" : 2010 }
{ "_id" : ObjectId("63c7f3d7abc3b8666b366a75"), "Type" : "DVD", "Title" : "Blade Run
ner", "Released" : 1982 }
>
```

HARMANTEPE MELIS

db.media.find ({ Tracklist : {$size : 2} }) => retourne le(s) document(s) qui avec une Tracklist de taille 2.

db.media.find ({ Author : {$exists : true } }) =>retourne le(s) document(s) qui contient un champ nommé “Author”.

db.media.find ({ Author : {$exists : false } }) =>retourne le(s) document(s) qui ne contient pas de champ nommé “Author”.

INDEX CREATION

Dans la version 5 du mongo, .ensureIndex() est changé par .createIndex().

.createIndex() pour minimiser l'impact de la création d'un index sur les jeux de réplicas et les clusters partitionnés.

```
> db.media.createIndex( { Title :-1 } )
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
> db.media.createIndex( { Title :1 } )
{
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
> db.media.createIndex( { "Tracklist.Title" :1 } )
{
        "numIndexesBefore" : 3,
        "numIndexesAfter" : 4,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
```

db.media.createIndex({ Title : -1 }) => permet de classer les données en fonction des titre en ordre inverse en créant les index sur le champ Title.

db.media.createIndex({ Title : 1 }) => permet de classer les données en fonction des titre en ordre ascendant en créant les index sur le champ Title.

db.media.createIndex({ "Tracklist.Title" :1 }) => permet de classer les données en fonction des titre en ordre ascendant en créant les index sur le champ Title du champ Tracklist.

```
> db.media.find( { ISBN: "987-1-4302-3051-9"} ).hint( { ISBN: -1 } )
Error: error: {
        "ok" : 0,
        "errmsg" : "error processing query: ns=library.mediaTree: ISBN $eq \"987-1-4
302-3051-9\"\nSort: {}\nProj: {}\n planner returned error :: caused by :: hint provi
ded does not correspond to an existing index",
        "code" : 2,
        "codeName" : "BadValue"
}
> db.media.createIndex({ISBN: 1})
{
        "numIndexesBefore" : 4,
        "numIndexesAfter" : 5,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
> db.media.find( { ISBN: "987-1-4302-3051-9"} ) . hint ( { ISBN: 1 } )
{ "_id" : ObjectId("63c7c67dabc3b8666b366a70"), "Type" : "Book", "Title" : "Definiti
ve Guide to MongoDB", "ISBN" : "987-1-4302-3051-9", "Publisher" : "Apress", "Author"
 : [ "Membrey, Peter", "Plugge, Eelco", "Hawkins, Tim" ] }
```

.hint() sert à trouver une donnée à partir de son indexe (si une indexe est déjà assigné à cette donnée).

db.media.find({ ISBN: "987-1-4302-3051-9"}).hint({ ISBN: -1 }) => retourne une erreur car la requête trouve la donnée avec cet ISBN mais elle ne le trouve pas dans cette indexe car cette donnée n'est pas encore indexé. On a pas encore assigné une indexe à cette donnée.

db.media.createIndex({ISBN: 1}) => on cree une indexe pour cette donnée.

db.media.find({ ISBN: "987-1-4302-3051-9"}).hint({ ISBN: 1 }) => on peut enfin trouver cette valeur de cet ISBN à partir de son indexe car on vient de créer et assigner une indexe à cette donnée. La requête retourne l'information de cette donnée.

db.media.getIndexes() => retourne toutes indexes assignées aux attributs.

DATA UPDATE

db.media.update({ "Title" : "Matrix, the"}, {"Type" : "DVD", "Title" : "Matrix, the", "Released" : "1999", "Genre" : "Action"}, true) => mets à jour ce document qui contient ces données, l'option "true" permet si aucun document ne correspond au paramètres { "Title" : "Matrix, the"}, {"Type" : "DVD", "Title" : "Matrix, the", "Released" : "1999", "Genre" : "Action"}, l'opération de mise à jour insère un document avec uniquement le document de remplacement.

db.media.update ({ "Title" : "Matrix, the" }, {$set : { Genre : "Sci-Fi" } }) => mets à jour cette query en changeant le champ Genre par Sci-Fi.

db.media.update ({"Title": "Matrix, the"}, {$unset : { "Genre" : 1 } }) => mets à jour cette query en enlevant l'élément contenu dans le champ Genre. Ne fait rien si Genre : 1 n'existe pas.

```
> db.media.remove( { "Title" : "Different Title" } )
WriteResult({ "nRemoved" : 0 })
> db.media.remove({})
WriteResult({ "nRemoved" : 7 })
> db.media.drop()
true
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
persons  0.000GB
>
```

db.media.remove({ "Title" : "Different Title" }) => supprime le champ qui contient la valeur "Different Title" dans le champ Title.
db.media.remove({}) => supprime tous les documents
db.media.drop() => supprime toute la collection