TP2 : MongoDB
Saison III, épisode 2

Objectifs
Utilisation de javascript sur une base noSQL
Apprentissage de mongoDB
Utilisation du format Json
Réalisation de petits programmes simples utilisant mongoDB.

Section
R5_10: Département Informatique
IUT CAEN Campus 3

Auteur
E.Porcq

Date
08/09/2025 durée 4h00

L'objectif de ce TP est d'apprendre les bases du Big Data et de MongoDB
Rendre un compte-rendu par binôme à la fin de la séance

- nommé " TP2_<nom1>_<nom2>.pdf " (sans accent ni espace).
- L'étudiant qui envoie le CR met son binôme en copie.
- Il faut des commandes, des résultats (captures, ...), [des observations] [et des conclusions]
- les captures sont éditables **obligatoirement** (au moins les commandes)
- le compte-rendu concerne le 1.3 au 2.1

Si plusieurs binômes sont constituables et non constitués, tirage au sort.
Dans le TD, les fichiers XXX sont fournis et les fichiers YYY sont à créer

1 Initiation à MongoDB

1.1 Présentation

On peut créer des collections qui contiendront des documents.
Un document est 1 enregistrement de type json {} .
Un document JSON ne comprend que deux types d'éléments structurels :

- Des ensembles de paires "nom" (alias "clé") / "valeur";
- Des listes ordonnées de valeurs.
- Ces mêmes éléments représentent trois types de données :
  - Des objets ;
  - Des tableaux ;
  - Des valeurs génériques de type tableau, objet, booléen, nombre, chaîne ou null.

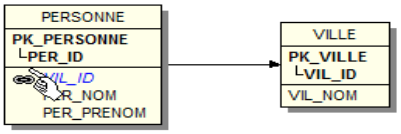
#	title	stuff	moar
1	Bla bla	Mdr	xD
3	TEST	Lmfao	XML
4	Azerty	GUI	Lol

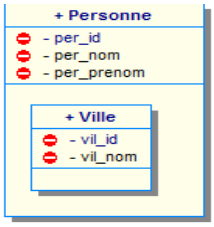
```
{ title: "Bla bla", stuff: "Mdr", moar: "xD" }
{ title: "TEST", stuff: "Lmfao", moar: "XML" }
{ title: "Azerty", stuff: "GUI", moar: "Lol" }
```

Sur disque, il est enregistré en binaire (Bson).
Dans une même collection, les documents peuvent avoir un format différent.

La façon d'associer plusieurs collections est assez différente aux SGBDR mais pas tant que cela. D'un point de vue UML, on peut parler

1. d'association simple (normalisé au sens BD)
2. de composition (dénormalisé)





Normalisé

Dénormalisé

Si la taille maximale allouée à ce document est dépassée, ce qui est généralement limité à 16Mo à cause de BSON, MongoDB va déplacer le fichier sur le disque-dur. Ceci peut être un facteur de décision pour le choix d'un modèle de données plutôt normalisé ou dénormalisé (**dupliquer** l'information, la **répéter**, autant de fois que nécessaire afin de faire « descendre » les attributs au plus près des données : de cette façon on simplifie les relations et on optimise l'accès aux données)

Il est possible d'utiliser une Collection Plafonnée ; une collection plafonnée est un type spécial de collection qui a soit des éléments fixes, soit un nombre d'éléments fixes. Une fois que la collection est pleine, les éléments les plus anciens seront effacés lors de l'ajout de nouveaux éléments.

De plus, si une application exécute de nombreuses opérations de lecture, on peut ajouter des Index.

En cas de " jointure " fréquente, mieux vaut choisir des compositions ; mais en cas de répétitions trop fréquentes, mieux vaut choisir l'association.

On peut aussi organiser la structure en arbre.

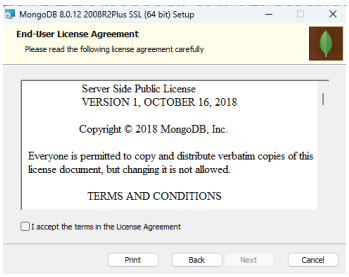
ObjectID représente l'identifiant d'un document, et est toujours contenu dans la propriété _id. Lorsque l'on insère un nouveau document dans une collection, MongoDB lui définit automatiquement une propriété _id. On peut définir soit-même un id à un document.

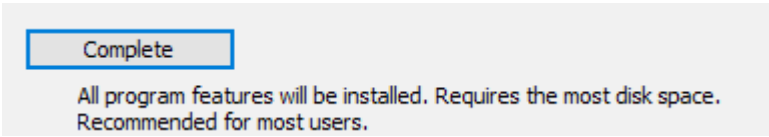
1.2 Installation (facultattif)

http://harry-wanki.developpez.com/tutoriels/mongodb/debuter-mongodb-introduction-base-donnees-nosql/
https://www.tutorielsenfolie.com/tutoriels-113-Installation-et-configuration-de-MongoDB-sous-Windows.html
https://www.mongodb.com/try/download/community?tck=docs_server

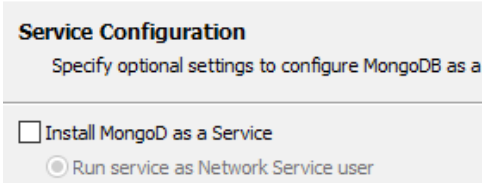
En 2236, mongoDB est déjà installé.

- https://www.mongodb.com/try/download/community
- https://www.mongodb.com/download-center/community/releases/archive
- Version installée 6.0.8 2008R2Plus



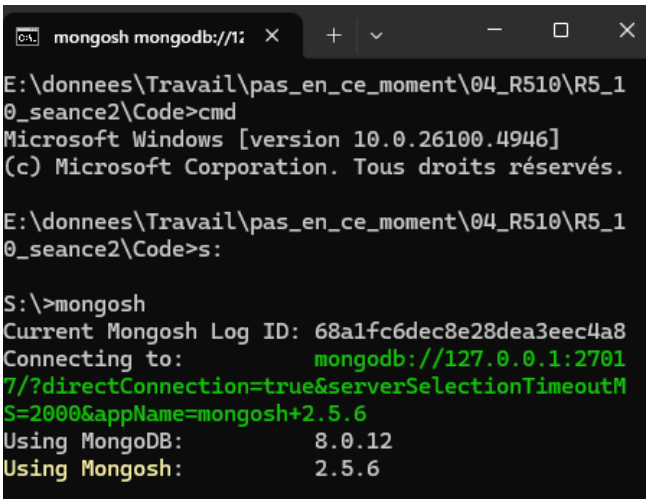


- décocher l'installation en tant que service

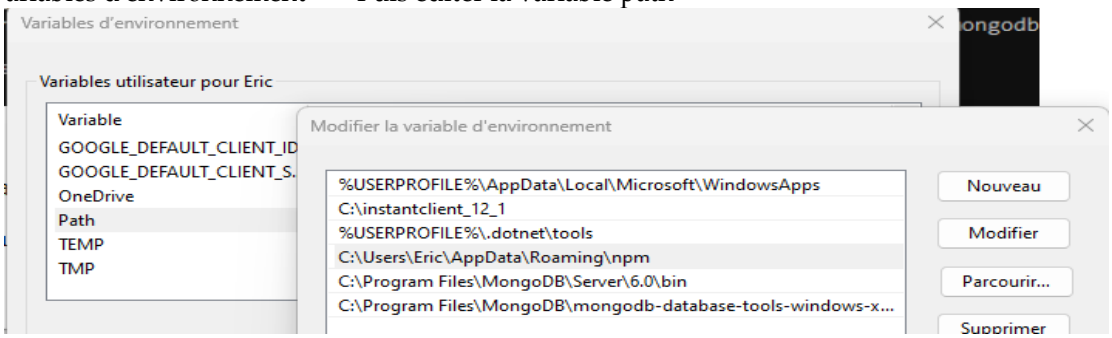


- Depuis la version 4.4 de mongoDB, certains outils sont à installer à part
  - https://www.mongodb.com/try/download/database-tools
  - https://www.mongodb.com/try/download/shell
    - copier le contenu des bin dans le bin de mongodb

- Vérifiez que vous parvenez à le lancer à partir d'un interprète de commande. Ce n'est pas grave si la connexion échoue.



- si cela ne fonctionne pas, ajoutez-le chemin dans le path (il faudra peut-être redémarrer)
  - paramètres--> Informations systèmes --> paramètres avancés du système
  - Variables d'environnement --> Puis éditer la variable path

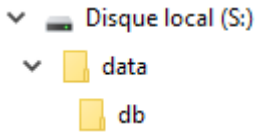


- Il peut être nécessaire de redémarrer

1.3 tests du serveur

- dans l'interprète de commande, créer un lecteur S dans le dossier où seront déposés les scripts de l'exercice et la base (à la racine de " S ")

```
Exemple :
Z:\fait_ici\M4101C_2_TP2>subst S: /D
Z:\fait_ici\M4101C_2_TP2>subst S: .
Z:\fait_ici\M4101C_2_TP2>S:
S:\>
```



- Fabriquer le dossier data/db dans son dossier de travail (donc dans S:\). Ce dossier :
  - contiendra toutes les bases de données.
  - est géré par mongodb. Ne rien y enregistrer directement.
- Lancer mongod

```
mongod.exe --dbpath S:\data\db --bind_ip 127.0.0.1
```

le serveur est lancé

- Ouvrir une autre console pour exécuter le client
- lancer mongosh

le client est lancé

- Tester les commandes suivantes :
  - db
  - show dbs ou show databases
  - use maBDDTP2
  - db
  - show dbs
  - show collections
  - db.createCollection("maCollection1")
  - show dbs
  - db.createCollection("maCollection2")
  - db.createCollection("maCollection3")
  - show collections
  - db.maCollection3.insertOne({ "_id" : "1", "nom" : "nom1" })
  - db.maCollection3.insertOne({ "_id" : "35", ville : "Moult" , pays : "France", pop : 2200 , loc : [05,05] })
  - db.maCollection3.insertOne ({ "nom" : "nom2" })
  - db.maCollection3.find()
  - db.maCollection3.find({},{_id:1})
  - db.maCollection3.find({"nom":"nom1"},{_id:0, nom:1})
  - db.maCollection3.drop()
  - config.set("displayBatchSize", 300)
  - db.dropDatabase()
  - show databases

1.4 Création d'une collection

https://openclassrooms.com/courses/guide-de-demarrage-pour-utiliser-mongodb
https://docs.mongodb.com/manual/reference/method/

1.4.1 Utiliser une base de données nommée maBDDTP2

1.4.2 Créer une collection nommée maCollec1. Vérifier que la création a fonctionné.

1.4.3 Etudier les fichiers fournis (p14_X.json) et indiquer l'erreur du p14_2_faux.json.

1.4.4 Y insérer les documents suivants. 3 solutions :

- On peut insérer ligne par ligne avec `db.<nom de la collection>.insert`
- On peut utiliser l'utilitaire mongoimport (ce n'est pas une commande du shell mongo). Le fichier associé contient tous les " db.<nom de la collection>.insert " nécessaires.
- On peut utiliser la commande load. Le fichier associé contient les requêtes « `db.<collection>.insert` ».

1.4.5 Analyser la collection. Quelle est la différence avec les données d'une BDDR ?

				telephone									
_id	nom	prenom	age	portable	fixe	ville	pays	pop	loc	type	couleur	matiere	fruit
	"Dubois"	"Maurice"	25										
"1"	"Dubois"	"Mauricette"	23										
2	"Dupont"	"Bertrand"	28										
3	"Dulong"	"sylvie"	34										
"4"	"Dubois"	"Sylvian"											
"5"	"Delalune"	"Claire"											
80	"Frere"	"Jacques"	22										
11										"casserolle"	"rouge"	"fonte"	
12										"casserolle"	"verte"	"fer"	
5	"Troj"			"06 85 45 48 10"									
6	"Super"			"06 85 45 48 10"	"02 45 38 45 33"								
7	"Super"	"veronique"		"06 80 23 48 12"	"02 32 41 22 16"								
8	"orange"	"Enpierre"	36										
21													["orange","banane","pomme"]
22													["kiwi","banane","pomme"]
23													["kiwi","orange","poire"]
24													["cerise","peche","citron"]
25													["cerise","banane","orange"]
26													["orange","pomme","raisin"]
"27"													["orange","prune","cerise"]

1.5 Requêtes sur maCollec1

L'objectif est d'apprendre à effectuer des projections, des restrictions et des mises à jour de base
Conserver toutes les requêtes (pas de rollback possible)

1.5.1 Afficher les documents avec et sans la méthode pretty

1.5.2 Afficher pour tous les documents la valeur (attribut) nom

```
{ nom: 'Dubois' }, { nom: 'Dubois' },
  { nom: 'Dubois' }, { nom: 'Dupont' },
  { nom: 'Dupont' }, { nom: 'Delalune' },
  { nom: 'Frere' },  {},
  {},                { nom: 'orange' },
  { nom: 'Troj' },   {},
  {},                {},
  {},                {},
  {},                { nom: 'Super' },
  {},                { nom: 'Super' }
```

1.5.3 Afficher l'attribut nom des documents qui en contiennent (utiliser $exists)

```
  { nom: 'Dubois' },
  { nom: 'Dubois' },
  { nom: 'Dubois' },
  { nom: 'Dupont' },
  { nom: 'Dupont' },
  { nom: 'Delalune' },
  { nom: 'Frere' },
  { nom: 'orange' },
  { nom: 'Troj' },
  { nom: 'Super' },
  { nom: 'Super' }
```

1.5.4 Afficher l'attribut nom sans doublons

```
'Delalune', 'Dubois',  'Dupont',   'Frere',  'Super',    'Troj',  'orange'
```

1.5.5 Afficher tous les documents sans afficher l'attribut nom

```
{ "_id" : ObjectId("5c8a21aa88037310c73168de"), "prenom" : "Maurice", "age" : 25 }
{ "_id" : "1", "prenom" : "Mauricette", "age" : 23 }
{ "_id" : 2, "prenom" : "Bertrand", "age" : 28 }
{ "_id" : 3, "prenom" : "Sylvie", "age" : 34 }
{ "_id" : "4", "prenom" : "Sylvian" }
{ "_id" : 11, "type" : "casserolle", "couleur" : "rouge", "matière" : "fonte" }
{ "_id" : 12, "type" : "casserolle", "couleur" : "verte", "matière" : "fer" }
{ "_id" : 5, "telephone" : { "portable" : "06 85 45 48 10" } }
{ "_id" : 6, "telephone" : { "portable" : "06 85 45 48 10", "fixe" : "02 45 38 45 3
{ "_id" : 7, "prenom" : "veronique", "telephone" : { "portable" : "06 80 23 48 12",
{ "_id" : 21, "fruit" : [ "orange", "banane", "pomme" ] }
```

1.5.6 Afficher les documents sans l'attribut nom, les données qui n'ont pas de nom

```
{ _id : 11, "type" : "casserolle", "couleur" : "rouge", "matière" : "fonte" }
{ _id : 12, "type" : "casserolle", "couleur" : "verte", "matière" : "fer" }
{ _id : 21, "fruit" : [ "orange", "banane", "pomme" ] }
{ _id : 22, "fruit" : [ "kiwi", "banane", "pomme" ] }
{ _id : 23, "fruit" : [ "kiwi", "orange", "poire" ] }
{ _id : 24, "fruit" : [ "cerise", "peche", "citron" ] }
{ _id: 25, "fruit" : [ "cerise", "banane", "orange" ] }
{ _id: 26, "fruit" : [ "orange", "pomme", "raisin" ] }
{ _id: '27', fruit: [ 'orange', 'prune', 'cerise' ] }
```

1.5.7 Afficher les documents dont le nom est Dubois

1.5.8 Afficher le nom et prénom des documents dont le nom est Dubois

1.5.9 Afficher les documents dont le nom est Dubois et le prénom Mauricette

1.5.10 Afficher les documents dont le nom contient un " r "

1.5.11 Afficher les documents dont le nom ne commence pas par un " D "

```
{ _id: 80, nom: 'Frere', prenom: 'Jacques' },
{ _id: 5, nom: 'Troj' },
{ _id: 6, nom: 'Super' },
{ _id: 7, nom: 'Super', prenom: 'veronique' },
{ _id: 8, nom: 'orange', prenom: 'Enpierre' }
```

1.5.12 Afficher les documents d'_id > 23. Justifier les résultats

1.5.13 Afficher les documents d'_id < "32". Justifier les résultats

1.5.14 Afficher les documents dont l'age est soit 22, soit 25 ans

1.5.15 Afficher les documents dont les numéros de téléphones sont "06 85 45 48 10" et "02 45 38 45 33"

1.5.16 Afficher les documents dont le numéro de mobile est "06 85 45 48 10"

```
  { _id: 5, nom: 'Troj', telephone: { portable: '06 85 45 48 10' } },
  { _id: 6, nom: 'Super', telephone: { portable: '06 85 45 48 10', fixe: '02 45 38 45 33' }   }
```

1.5.17 Afficher les documents contenant le fruit" orange ".

1.5.18 Afficher les documents dont le premier fruit est " orange ".

1.5.19 Donner l'age de 34 ans à tous les Dubois.

1.5.20 Donner l'age de 36 ans au premier Dubois.

1.5.21 Donner l'age de 19 ans au second Dubois (merci chatGPT)

```
const second = db.maCollec1.find().skip(1).limit(1).next()
db.maCollec1.updateOne(  { _id: second._id },  { $set: { age: 19 } })

ou

db.maCollec1.updateOne(  { _id: { $in: [ db.maCollec1.find().skip(1).limit(1).next()._id ] } },  [ { $set: {
age: 19 } } ] )
```

1.5.22 Donner l'age de 35 ans à Dudouit. S'il n'existe pas, le créer.

1.5.23 Remplacer le document 24 par celui-ci {"_id" : 24 , fruit:["mandarine","citron","prune"] }).

- Quelle est la différence entre update et replace ?

1.5.24 Supprimer le document 4

1.5.25 Supprimer le premier Dubois

1.6 Requêtes sur maCollec2

L'objectif est d'apprendre à effectuer des mises à jour plus complexes

1.6.1 Créer une nouvelle collection : maCollec2.

1.6.2 Créer 7 documents à partir d'un fichier p16_X.Json :

```
{ "_id" : "1", "titre" : "Les misérables", "auteur" : "Victor Hugo", "nbExemplaires" : 100 }
{ "_id" : "2", "titre" : "Notre-Dame de Paris", "auteur" : "Victor Hugo", "nbExemplaires" : 30 }
{ "_id" : "3", "titre" : "La Bête humaine", "auteur" : "Emile Zola", "nbExemplaires" : 19 }
{ "_id" : "4", "titre" : "Germinal", "auteur" : "Emile Zola", "nbExemplaires" : 100 }
{ "_id" : "6", "titre" : "Notre-Dame de Paris le retour", "auteur" : "Victor Hugo", "nbExemplaires" : 20 }
{ "_id" : "7", "titre" : "Notre-Dame de Paris contre les misérables", "auteur" : "Victor Hugo","nbExemplaires" : 40 }
{ "_id" : "9", "titre" : "Le Dernier Jour d'un confiné", "auteur" : "Victor Hugo", "nbExemplaires" : 64 }
```

1.6.3 En utilisant la fonction findAndModify (voir annexe), retirer un exemplaire du livre de Victor Hugo comportant le moins d'exemplaires.

1.6.4 En utilisant la fonction findAndModify (voir annexe), ajouter un exemplaire au livre de Victor Hugo " les travailleurs de la mer" . L'exemplaire sera créé s'il n'existe pas. Répéter l'opération et tester.

1.6.5 En utilisant la fonction findAndModify (voir annexe), supprimer le livre de Victor Hugo comportant 19 exemplaires.

1.6.6 Afficher les enregistrements triés du plus grand au plus petit nombre d'exemplaires.

1.6.7 Donner le nombre de documents de la collection.

1.7 Requêtes sur maCollec3

L'objectif est d'apprendre à utiliser les fonctions d'agrégation

1.7.1 Créer une nouvelle collection : maCollec3

1.7.2 Charger la collection p17_X.json

_id"	"ville"	"pays"	"pop"	"loc"
"31"	"Paris"	"France"	2187526	[15 , 20]
"32"	"Marseille"	"France"	862211	[16 , 28]
"33"	"Rouen"	"France"	515695	[16 , 28]
32	"Caen"	"France"	110000	[10 , 25]
"34"	"Ifs"	"France"	1111500	[1 , 5]
"35"	"Moult"	"France"	3200	[0 , 0]
"36"	"Berlin"	"Allemagne"	3748148	[16 , 30]
"37"	"Hambourg"	"Allemagne"	1645095	[20 , 30]
"38"	"Munich"	"Allemagne"	1298941	[20 , 35]
"42"	"Glasgow"	"Royaume Uni"	765030	[0 , 0]
"43"	"Birmingham"	"Royaume Uni"	1013395	[0 , 0]
"44"	"Londres"	"Royaume Uni"	6574009	[55 , 7]
"45"	"Rome"	"Italie"	2855397	[16 , 15]
"46"	"Milan"	"Italie"	1378689	[34 , 10]
"47"	"Naples"	"Italie"	957075	[8 , 7]

1.7.3 Afficher les documents

1.7.4 Tester et commenter les requêtes suivantes :

1.7.4.1 db.maCollec3.aggregate({$group:{_id:null,population:{ $max:"$pop"} } })

1.7.4.2 db.maCollec3.aggregate({$group:{_id:"$pays",population:{ $max:"$pop"} } })

1.7.4.3 db.maCollec3.aggregate({$group:{_id:"$pays",population:{ $avg:"$pop"} } })

1.7.4.4 db.maCollec3.aggregate({$group:{_id:"$pays",population:{ $sum:"$pop"} } })

1.7.4.5 db.maCollec3.aggregate({$group:{_id:"$pays",population:{$sum:"$pop"}}},{$sort:{population:1}})

1.7.4.6 db.maCollec3.aggregate({$match:{pays:"France"} })

1.7.5 En déduire la requête affichant la ville française ayant le moins d'habitants.

2 Accès à une base mongodb avec un script

2.1 Accès à la base en mongoDB-Script

https://docs.mongodb.com/manual/tutorial/write-scripts-for-the-mongo-shell/

2.1.1 Avec un autre interprète de commande lancer mongosh p21_1.js . ~~Pourquoi ce script ne fonctionne pas ?~~

2.1.2 Tester et étudier de la même façon p21_2A.js et p21_2B.js

2.1.3 En utilisant `printjson`, n'afficher que les documents possédant un champ " nom " (p21_3.js).

2.1.4 Compléter ce programme pour n'afficher que l'id et le nom (p21_4.js)

```
S:\>mongosh p21_5.js
Current Mongosh Log ID: 64ef550fe76396e9d8881a75
Connecting to:          mongodb://127.0.0.1:27017/?
directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.10.3
Using MongoDB:          6.0.8
Using Mongosh:          1.10.3

Loading file: p21_5.js
{  _id: '1',  nom: 'Dubois'}
{  _id: 2,  nom: 'Dupont'}
{  _id: 3,  nom: 'Dupont'}
{  _id: '4',  nom: 'Dubois'}
{  _id: '5',  nom: 'Delalune'}
{  _id: 80,  nom: 'Frere'}
{  _id: 5,  nom: 'Troj'}
{  _id: 6,  nom: 'Super'}
{  _id: 7,  nom: 'Super'}
```

2.1.5 En utilisant insert, ajouter dans macollec4, 10 enregistrements composés uniquement de i variant de 1 à 10 (p21_5.js). Vérifier avec le client connecté (mongosh)

```
switched to db maBDDTP2
db.maCollec4.find( { } )
{ _id: ObjectId("64ef57b0b7332cd15a58a569"), i: 1 },
{ _id: ObjectId("64ef57b0b7332cd15a58a56a"), i: 2 },
{ _id: ObjectId("64ef57b0b7332cd15a58a56b"), i: 3 },
{ _id: ObjectId("64ef57b0b7332cd15a58a56c"), i: 4 },
{ _id: ObjectId("64ef57b0b7332cd15a58a56d"), i: 5 },
{ _id: ObjectId("64ef57b0b7332cd15a58a56e"), i: 6 },
{ _id: ObjectId("64ef57b0b7332cd15a58a56f"), i: 7 },
{ _id: ObjectId("64ef57b0b7332cd15a58a570"), i: 8 },
{ _id: ObjectId("64ef57b0b7332cd15a58a571"), i: 9 },
{ _id: ObjectId("64ef57b0b7332cd15a58a572"), i: 10 }
```

2.1.6 Supprimer les 8 premiers enregistrements (p21_6.js)

```
maBDDTP2> db.maCollec4.find( { } )
[
  { _id: ObjectId("64ef57b0b7332cd15a58a571"), i: 9 },
  { _id: ObjectId("64ef57b0b7332cd15a58a572"), i: 10 },
  { _id: ObjectId("64ef57b0b7332cd15a58a573"), i: 11 },
  { _id: ObjectId("64ef57b0b7332cd15a58a574"), i: 12 }
]
```

Annexe :

https://docs.mongodb.org/v3.0/reference/method/db.collection.findAndModify/

db.collection.findAndModify()

Description
Modifies and returns a single document. By default, the returned document does not include the modifications made on the update. To return the document with the modifications made on the update, use the `new` option. The `findAndModify()` method is a shell helper around the `findAndModify` command.

```
db.collection.findAndModify({
  query: <document>,
  sort: <document>,
  remove: <boolean>,
  update: <document>,
  new: <boolean>,
  fields: <document>,
  upsert: <boolean>
});
```

Parameter	Type	Description
query	document	Optional. The selection criteria for the modification. The query field employs the same query selectors as used in the db.collection.find() method. Although the query may match multiple documents, findAndModify() will only select one document to modify.
sort	document	Optional. Determines which document the operation modifies if the query selects multiple documents. findAndModify() modifies the first document in the sort order specified by this argument.
remove	boolean	Must specify either the remove or the update field. Removes the document specified in the query field. Set this to true to remove the selected document . The default is false.
update	document	Must specify either the remove or the update field. Performs an update of the selected document. The update field employs the same update operators or field: value specifications to modify the selected document.
new	boolean	Optional. When true, returns the modified document rather than the original. The findAndModify() method ignores the new option for remove operations. The default is false.
fields	document	Optional. A subset of fields to return. The fields document specifies an inclusion of a field with 1, as in: fields: { <field1>: 1, <field2>: 1, ... }. See projection.
upsert	boolean	Optional. Used in conjunction with the update field. When true, findAndModify() creates a new document if no document matches the query, or if documents match the query, findAndModify() performs an update. To avoid multiple upserts, ensure that the query fields are uniquely indexed. The default is false.