

SHAKE THE FUTURE



Bases de Données

NoSQL avec MongoDB

JY Martin

Plan

- 1 Généralités
- 2 Un peu de syntaxe
- 3 Utilisation de MongoDB
- 4 Programmation
- 5 Conclusion

MongoDB

- Base de données noSQL.
- Mise en œuvre, par 10gen en 2007, à l'origine comme un PAAS (Plateforme As A Service)
- C'est aujourd'hui une base de données distribuées assez répandue pour les appli-web basée Javascript.

<https://www.mongodb.com>

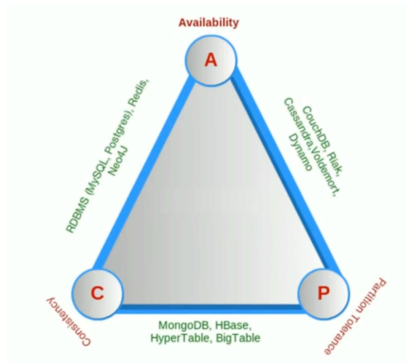
Aperçu

- **NoSQL format document.**
 - Modèle riche
 - Elastique
- Utilisable en standalone ou en PAAS (MongoDB Atlas)
Disponibles sur Google Cloud, AWS, ou Azure
- S'appuie sur BSON, pour simplifier du JSON binarisé

Le logiciel

- Sous licence SSPL (Server Side Public Licence)
<https://www.mongodb.com/licensing/server-side-public-license>
- Les pilotes sont sous licence Apache (Creative Commons)
- Ecrit en C++

Par rapport au CAP



Le modèle document

MongoDB utilise le modèle Document.

- MongoDB gère un ensemble de **Bases de Données**
- Chaque Base de données contient des **Collections**
- une Collection contient des **Documents**
- Un **Document** est un ensemble de champs et de valeurs, et est identifié par une clé
- Les documents n'ont pas obligatoirement tous la même structure.

BSON

BSON = Binary JSON.

La syntaxe est donc très proche de JSON, et mise en binaire pour le stockage.

MongoDB utilise la syntaxe BSON pour mémoriser les documents.
1 document = 1 objet BSON

BSON : Quelques éléments de syntaxe

- `{ ... }` : objet
- `Champ :valeur`
- `[...]` : tableau

Exemples :

- `{ nom : "valeur", prenom : "valeur" }`
- `{ personne : { nom : "valeur", prenom : "valeur" } }`
- `["Bleu", "Rouge", "Vert"]`

Plan

- 1 Généralités
- 2 Un peu de syntaxe
- 3 Utilisation de MongoDB
- 4 Programmation
- 5 Conclusion

Lancer les commandes

2 façon de travailler :

- En mode terminal, via le logiciel mongo
- Dans un programme via une API

En mode terminal

Logiciel mongo

```
[mac-informat04:~ kwyhr$ mongo [1] 65843
MongoDB shell version v4.0.10 mac-informat04:etc kwyhr$ 2019-07-01T11:44:00.330+0200
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("b4a9b615-1585-436c-8b37-19673d7376a9") }/usr/
MongoDB server version: 4.0.10 [mac-informat04:etc kwyhr$ ps aux | grep mongo
```

Quelques commandes de base

- Liste des bases de données existantes

```
show dbs
```

```
db
```

- Créer / Se connecter à une base de données

```
use nomBase
```

- Supprimer une base de données (à laquelle on est connecté)

```
db.dropDatabase()
```

Remarques :

- si la base de données ne contient pas de document, elle ne sera pas listée par “show dbs”.
- db permet d’avoir la liste des bases, mais c’est aussi le mécanisme qui désigne la base courante

Création / Suppressions

- Voir les collections d'une base de données

```
show collections
```

- Créer une collection

```
db.createCollection(name)  
db.createCollection(name, options)
```

Attention : name est une chaîne de caractères entre guillemets

- Supprimer une collection (maCollection)

```
db.maCollection.drop()
```

Autres commandes

Liste des utilisateurs

```
show users
```


CRUD : la création

```
db.nomDeLaCollection.insertOne( objetBSON )
```

objetBSON est un objet au format BSON.

Vous pouvez également définir une variable et l'insérer.

```
var objetBSON = { ... }  
db.nomDeLaCollection.insertOne( objetBSON )
```

CRUD : la création

A chaque document, mongoDB ajoute un champ : `_id` qui sert à gérer le document en interne.
L'objet correspondant peut également être référencé via la fonction `ObjectId(valeur_de_l_ID)`

Exemple :

```
ObjectId("542310906694ce357ad2a1a9")
```

CRUD : la création

Pour une insertion multiple, il suffit d'insérer un tableau de valeurs.

```
var objetBSON = [ { ... }, { ... } ]  
db.nomDeLaCollection.insertOne( objetBSON )
```

CRUD : l'accès

Lister le contenu d'une collection :

```
db.nomDeLaCollection.find( )
```

... et pour des questions de présentation

```
db.nomDeLaCollection.find( ).pretty()
```

CRUD : l'accès

Avec un critère de recherche

```
db.nomDeLaCollection.find( critère )
```

critère est un **object BSON** définissant les critères de recherche.

CRUD : la modification

```
db.nomDeLaCollection.update( critere_recherche, elements_mis_a_jour,  
OPTIONS)
```

- critere_recherche est un objet BSON définissant le/les objets à modifier, comme pour une recherche
- elements_mis_a_jour définit la mise à jour à réaliser
- OPTIONS est optionnel et donne des indications complémentaires de mise à jour

Attention, par défaut update ne met à jour qu'un document. Pour mettre à jour tous les documents du critère de recherche, il faut ajouter une option.

CRUD : la modification

- `elements_mis_a_jour` est un objet BSON de la forme
 - **`modif`** pour remplacer complètement un document
 - avec des commandes particulières pour des opérations de modification de certains champs
 - `{commande :modif, commande :modif, ...}`**
 - **`$set :modif`** pour modifier des champs
 - **`$inc :modif`** pour incrémenter un champ
 - **`$unset :modif`** pour supprimer des champs
 - **`$rename :modif`** pour renommer des champs
- `modif` est un objet BSON définissant les modifications apportées

CRUD : la modification

Les options sont sous forme d'un objet BSON.

L'objet BSON peut comporter les éléments suivants :

- **upset** : faux par défaut. Si vrai, un nouvel élément est créé si aucun document ne correspond au critère de recherche.
- **multi** : met à jour tous les documents respectant le critère de sélection
- **writeConcern** : gère la façon dont MongoDB duplique l'information sur les serveurs, et en particulier l'acquiescement (=j'ai fait l'opération)
- **collation** : pour indiquer l'encodage
- **arrayFilters** : filtre pour les opérations sur plusieurs documents

CRUD : la suppression

```
db.nomDeLaCollection.delete( critere_suppression )
```

critere_suppression est un objet BSON définissant le/les objets à supprimer

```
db.nomDeLaCollection.delete( )
```

supprime tous les documents de la collection

CRUD exemple

```
db.createCollection("personne")
db.personne.insertOne({
  "id" : 1,
  "email" : "Jean-Yves.Martin@ec-nantes.fr"
})
db.personne.find( { "id" : 1 } ).pretty()
db.personne.update( { "id" : 1 }, { $set : { "bureau" : "E217" } } )
db.personne.delete( "id" : 1 )
```

Les index

Pour être efficace lors d'une recherche, il faut indexer les données pour retrouver plus rapidement les informations.

Pour avoir la liste des index définis sur une collection :

```
db.nomDeLaCollection.getIndexes()
```

Normalement, il y a un index défini par défaut et ajouté à chaque document.

Ajouter des index

Si vous faites fréquemment des recherches sur certains champs des documents, il peut être intéressant d'ajouter un index sur ces champs.

```
db.nomDeLaCollection.createIndex( indexBSON )
```

indexBSON est un objet BSON qui contient un champ et une valeur

- le champ est celui sur lequel vous créez un index
- la valeur est le sens du tri (1 = croissant, -1 = décroissant)

Plan

- 1 Généralités
- 2 Un peu de syntaxe
- 3 Utilisation de MongoDB**
- 4 Programmation
- 5 Conclusion

Installation

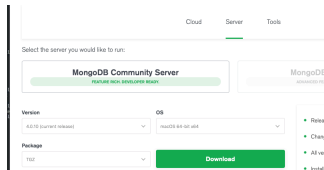
Suivant votre système, il existe plusieurs modes d'installation

<https://docs.mongodb.com/manual/installation/#tutorial-installation>

détaille les différents modes d'installation

Le site offre également le téléchargement direct :

<https://www.mongodb.com/download-center>



Le fichier de configuration

Le fichier `mongod.conf` est le fichier de configuration de MongoDB. Il vous sera utile pour démarrer le serveur. Si vous passez par un installateur de paquets, une version par défaut du fichier de configuration sera créée.

```
systemLog :  
  destination : file  
  path : /usr/local/var/log/mongodb/mongo.log  
  logAppend : true  
storage :  
  dbPath : /usr/local/var/mongodb  
net :  
  bindIp : 127.0.0.1
```

Installation Windows

- Téléchargez le package directement
- Démarrez l'installateur
- Si vous voulez installer "Compass", décochez puis recochez la case. Il y a un petit bug.
- Installez le comme un service

Installation MacOS

<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/>

- Si vous avez HomeBrew (ou que vous l'installez)
 - `brew tap mongodb/brew`
 - `brew install mongodb-community@4.0`
Et suivez les instructions pour le démarrer temporairement ou comme un service
- `mongod -config /usr/local/etc/mongod.conf &`

Installation MacOS

- Si vous préférez le faire manuellement
 - Téléchargez le package directement et décompressez le
 - Créez un chemin d'accès export
PATH=<mongodb-install-directory >/bin :\$PATH
 - Créez un fichier de configuration : mongod.conf
 - Lancez le serveur mongod –config mongod.conf &

Installation Linux

L'installation passe par votre installateur de paquets.

Par exemple sur Debian :

- Ajoutez la clé
`sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com :80
--recv 9DA31620334BD75D9DCB49F368818C72E52529D4`
- Mettez à jour puis installez MongoDB
`sudo apt-get update
sudo apt-get install -y mongodb-org`

Utilisation en ligne de commande

Vérifiez au préalable que votre serveur mongo est bien lancé.

Prenez votre terminal / invite de commande

```
mongo
```

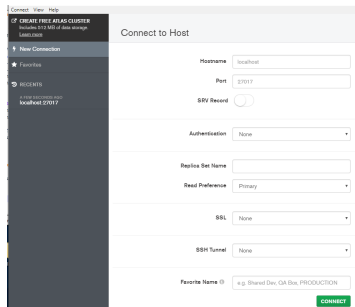
Exemple de connexion

Vérifiez que le serveur est bien lancé. Lancez le logiciel mongo.

```
use maBase
db.createCollection("personne")
db.personne.insertOne({
  "id" : 1,
  "name" : [
    { "lastName" : "MARTIN" },
    { "firstName" : "Jean-Yves" } ],
  "email" : "Jean-Yves.Martin@ec-nantes.fr",
  "designation" : "Responsable Opt Info"
})
db.personne.find( { "id" : 1 } ).pretty()
db.personne.update( { "id" : 1 }, { $set : { "bureau" : "E217" } }
```

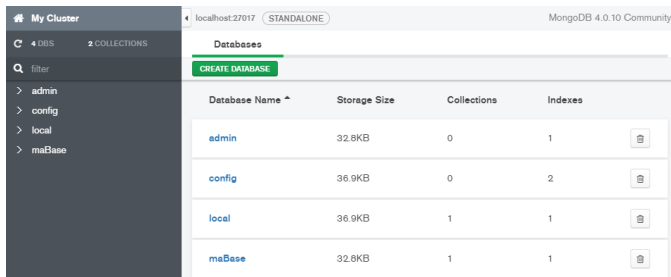
MongoDB Compass

Compass est un outil de connexion et d'analyse d'informations pour les bases MongoDB.
Pas indispensable mais peut tenir lieu de logiciel d'administration.







MongoDB Compass

Entrez les informations de connexion au serveur et connectez vous.



The screenshot shows the MongoDB Compass interface. On the left, a sidebar displays the cluster name 'My Cluster', the number of databases (4 DBS), and the number of collections (2 COLLECTIONS). Below this is a search filter and a list of databases: admin, config, local, and maBase. The main panel shows the 'Databases' section with a 'CREATE DATABASE' button. Below the button is a table listing the databases:

Database Name ^	Storage Size	Collections	Indexes	
admin	32.8KB	0	1	
config	36.9KB	0	2	
local	36.9KB	1	1	
maBase	32.8KB	1	1	

Vous pouvez ensuite examiner vos bases, vos collections, effectuer des requêtes ou des diagnostics de connexion.

Un peu de sécurité ...

La documentation de MongoDB fournit une liste d'éléments à mettre en oeuvre pour la sécurisation de MongoDB.

<https://docs.mongodb.com/manual/administration/security-checklist/>

A minima, vous devriez activer le contrôle d'accès via login/mot de passe et définir des rôles.

Un peu de sécurité ...

Les utilisateurs sont créés dans une base de données, avec le rôle qu'ils utiliseront.

Vous pouvez ainsi créer le même utilisateur dans autant de base que vous voulez, avec éventuellement des rôles différents.

Lors de la connexion, un utilisateur devra indiquer avec quel login et avec quelle base d'identification il souhaite se connecter.

Un peu de sécurité ...

Pour activer la sécurité, connectez-vous à votre base (celle pour vous identifier) et définissez l'utilisateur et son mot de passe. Vous devrez également définir son rôle pour chacune des bases.

```
use maBase
db.createUser(
  {
    user : "myUserAdmin",
    pwd : "myPassword",
    roles : [ { role : "userAdminAnyDatabase", db : "maBase" },
              { role : "readWrite", db : "test" },
              { role : "read", db : "reporting" },
              "readWriteAnyDatabase" ]
  }
)
```

Les rôles

<https://docs.mongodb.com/manual/reference/built-in-roles/>

- dbAdmin : administrateur
- dbOwner : possesseur d'une base
- userAdminAnyDatabase : admin de toutes les bases
- userAdmin : admin de la base indiquée
- readWriteAnyDatabase : peut lire et écrire dans n'importe quelle base
- readWrite : peut lire et écrire dans la base indiquée
- read : peut lire dans la base indiquée
- ...

Plan

- 1 Généralités
- 2 Un peu de syntaxe
- 3 Utilisation de MongoDB
- 4 Programmation**
- 5 Conclusion

MongoDB et Java

La connexion d'un programme JAVA à MongoDB se fait comme d'habitude via un driver JDBC.

Vous pouvez le télécharger directement sur ...
Bon courage. Passez par Maven.

ou via Maven :

```
<dependencies>
  <dependency >
    <groupId >org.mongodb </groupId >
    <artifactId >mongo-java-driver </artifactId >
    <version >3.7.1 </version >
  </dependency >
</dependencies>
```

MongoDB et Java

Créez un projet d'application Java avec Maven sous Netbeans.

Ajoutez la dépendance du driver JDBC de MongoDB

Attention, la première fois, ça peut prendre un peu de temps, le temps que Maven aille chercher les documents....

MongoDB et Java

Exemple : se connecter et lister les collections disponibles

```
MongoClient mongoClient = new MongoClient("localhost", 27017);  
MongoDatabase database = mongoClient.getDatabase("maBase");  
System.out.println("Liste des collections");  
for (String name : database.listCollectionNames()) {  
    System.out.println(name);  
}  
mongoClient.close();
```

MongoDB et Java

Exemple : se connecter et ajouter un document à la collection "personne".

```
MongoClient mongoClient = new MongoClient("localhost", 27017);  
MongoDatabase database = mongoClient.getDatabase("personne");  
MongoCollection <Document> collection = database.getCollection("personne");  
Document person = new Document("id", "2")  
    .append("name", new Document("lastName", "MAGNIN")  
        .append("firstName", "Morgan"))  
    .append("email", "Morgan.Magin@ec-nantes.fr")  
    .append("mission", "TICE");  
collection.insertOne(person);  
mongoClient.close();
```


MongoDB et Java

Exemple : se connecter et lister les documents de la collection "personne".

```
MongoClient mongoClient = new MongoClient("localhost", 27017);
MongoDatabase database = mongoClient.getDatabase("maBase");
MongoCollection<Document> personnes = database.getCollection("personne");
System.out.println("Liste des personnes");
MongoCursor<Document> cur = personnes.find().iterator();
while (cur.hasNext()) {
    Document newPersonne = cur.next();
    for (String key : newPersonne.keySet()) {
        System.out.println(key + "=" + newPersonne.get(key));
    }
}
mongoClient.close();
```

MongoDB et Python

En python, on utilise la bibliothèque PyMongo.
Elle fonctionne sous Python 3.

```
python -m pip install pymongo
```

ou

```
python3 -m pip install pymongo
```

MongoDB et Python

Le programme python se connecte à la base (maBase), la collection (personne), puis peut faire une requête

```
import pymongo
import pprint
from pymongo import MongoClient
client = MongoClient('localhost', 27017)
db = client.maBase
collection = db.personne
for person in collection.find():
    pprint.pprint(person)
```

MongoDB et Javascript

MongoDB utilisant BSON, le langage de prédilection pour communiquer avec MongoDB est Javascript. Et le plus fréquemment on utilise le serveur NodeJS qui gère les scripts Javascript.

Le driver Javascript de MongoDB peut être trouvé ici :
<https://mongodb.github.io/node-mongodb-native/>

Techniquement, il est conseillé d'utiliser npm pour installer le driver :

```
npm install mongodb --save
```

MongoDB et Javascript

Comme pour les autres langages, pour communiquer avec MongoDB, il faut établir une connexion, sélectionner la collection avec laquelle on travaille et envoyer les requêtes.

Attention, la syntaxe javascript peut sembler parfois perturbante à cause de l'imbrication des fonctions

MongoDB et Javascript

```
var MongoClient = require("mongodb").MongoClient ;
MongoClient.connect("mongodb://localhost/maBase", function(error, db) {
  db.collection("personne").find().toArray(function (error, results) {
    if (error) throw error ;
    results.forEach(function(i, obj) {
      console.log(
        "ObjectID : " + obj._id.toString() + "\n"
        "ID : " + obj.id.toString() + "\n"
        "Email : " + obj.email + "\n"
        "Nom : " + obj.name.lastName + "\n"
        "Prénom : " + obj.name.firstName + "\n"
      );
    });
  });
});
```

MongoDB et Javascript : Les opérations

- Selection/Recherche : **find**(critere, fonctionCallBack),
findOne(critere, fonctionCallBack)
- Insertion : **insert**(objetInséré, options, , fonctionCallBack)
ou **save** (objetInséré, options, fonctionCallBack)
- Mise à jour : **save**(objetModifié, options, fonctionCallBack)
objetModifié doit contenir l' _id de l'objet (celui de Mongo)
- Suppression : **remove** (objetSupprimé, options,
fonctionCallBack)

NB :

- les options et fonction de CalBack ne sont pas obligatoires
- les objets sont exprimés en JSON

MongoDB et Javascript

```
var MongoClient = require("mongodb").MongoClient ;
MongoClient.connect("mongodb://localhost/maBase", function(error, db) {
    db.collection("personne").update(
        { id : "1" },
        { $set : { email : "jean-yves.martin@ec-nantes.fr" } }
    );
});
```


Plan

- 1 Généralités
- 2 Un peu de syntaxe
- 3 Utilisation de MongoDB
- 4 Programmation
- 5 Conclusion

Conclusion

- MongoDB est une base de données NoSQL répartie
- Elle représente les informations sous forme de documents indépendants
- Elle est facile à manipuler et la représentation des documents en BSON offre une grande flexibilité

Mais ...

- si vous manipulez de grandes masses de documents disparates, vos requêtes risquent de ne pas être efficaces
- si vous faites du Big Data, la notion de BIG risque d'être limitée.

