314 Exploration MongoDB

Moteur de base de données 'NoSQL'

Rédigé par

David ROUMANET
Professeur BTS SIO

Changement

Date	Révision
2021-08-07	Création
2022-12-06	Correction p9, } manquant + modification chemin d'environnement 'système'
2023-10-13	Correction lien téléchargement v5.21 et ajouts des objectifs.

Sommaire

A MongoDB, moteur de base de données NoSQL

A.1 Introduction

A.1.1 Généralités

Le moteur de base de données est le composant principal qui gère (lit, écrit, modifie) les informations contenues dans les bases de données.

Le modèle relationnel est le principe de fonctionnement permettant de lier les informations, afin de les retrouver plus rapidement, et de prendre en compte les contraintes nécessaires à la bonne modélisation des bases.

Dans ces moteurs utilisant les principes relationnels, on trouve PostgreSQL, Oracle, MySQL, SQLite, SQLServer (Microsoft), etc.

On abrège souvent l'ensemble permettant aux moteurs de base de données de fonctionner, sous le nom de **SGBD** (Système de Gestion de Base de Données). En effet, une base de données n'est qu'un fichier structuré, il faut un moteur pour travailler dessus et des outils d'administration pour gérer les méta-informations (les index, les schémas relationnels et autres informations de fonctionnement.

MongoDB (et quelques autres systèmes, dits NoSQL) ont fait le choix de permettre l'enregistrement de données moins cohérentes ou moins contraintes. Ici, les enregistrements peuvent ne pas contenir le même nombre de propriétés (de colonnes, dans un SGBD traditionnel).

A.1.2 Objectifs

Durant cette exploration, vous allez découvrir une alternative aux bases de données relationnelles. À la fin de l'activité :

- ☑ Vous connaîtrez quelques commandes pour accéder aux tables et aux données sous MongoDB
- ☑ Vous connaîtrez le format des données sous MongoDB (et généralement d'autres bases NoSQL)
- ☑ Vous saurez installer MongoDB sur un poste de développement
- ☑ Vous pourrez utiliser MongoDB dans une application MVC, à la place de MySQL

A.1.3 Prérequis

Le cycle Node.JS doit être terminé

A.2 Théorie et fonctionnement

A.2.1 Formats et différences

La description des données utilise le format JSON, qu'il est important de bien maîtriser pour utiliser MongoDB. De fait, MongoDB travaille sur les paires "clés – valeurs" (notion de dictionnaire).

D'autre part, MongoDB n'utilise pas le langage SQL pour le rendu des requêtes.

Le tableau suivant synthétise les différences entre un SGBD-R et un système No-SQL.

	Système SQL	Système NoSQL
Modèle de stockage de données	Tables avec des lignes et colonnes fixes	Document JSON utilisant les paires 'clé-valeur'. Format dynamique.
Période de développement	Dans les années 1970	Dans les années 2000
Exemple de SGBD	Oracle, MySQL, PostgreSQL	MongoDB, dynamoDB, Redis...
Schémas	Rigides	Flexibles
Jointures	Obligatoires	Non requises
Mappage Objet	Nécessite un ORM	Non requis

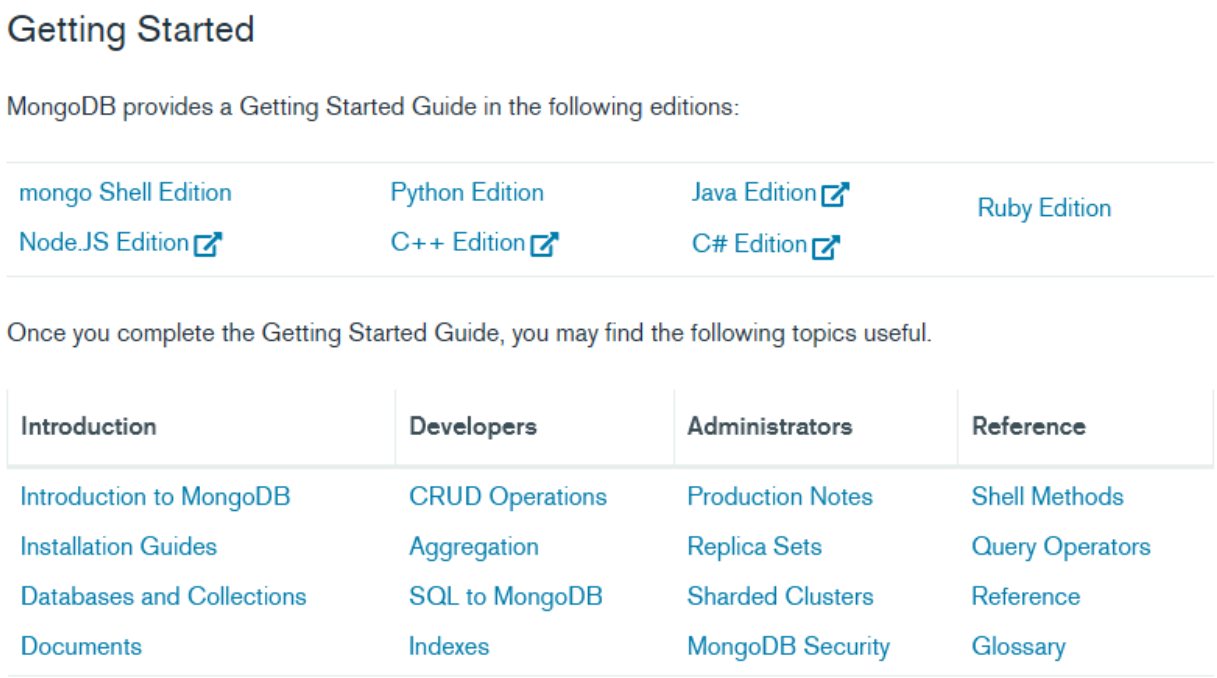
Si l'utilisation d'une base de type NoSQL semble plus pratique et simple, il ne faut pas croire que le modèle SGBD-R est déprécié : les deux modèles sont complémentaires et n'ont pas les mêmes raisons ou objectifs.

B Déploiement de MongoDB

B.1 Téléchargement et documentation

MongoDB est disponible en plusieurs versions mais pour nos besoins éducatifs, la version 'Community' est suffisante. La nouvelle version ne permet pas d'accéder par console directement à la base, il faut donc utiliser la version disponible depuis https://fastdl.mongodb.org/windows/mongodb-windows-x86_64-5.0.21-signed.msi.

D'autre part, la documentation est disponible sur le site officiel, elle permet de démarrer des projets avec ce moteur de base de données, mais aussi de l'utiliser avec différents langages.



Getting Started

MongoDB provides a Getting Started Guide in the following editions:

mongo Shell Edition	Python Edition	Java Edition	Ruby Edition
Node.JS Edition	C++ Edition	C# Edition	

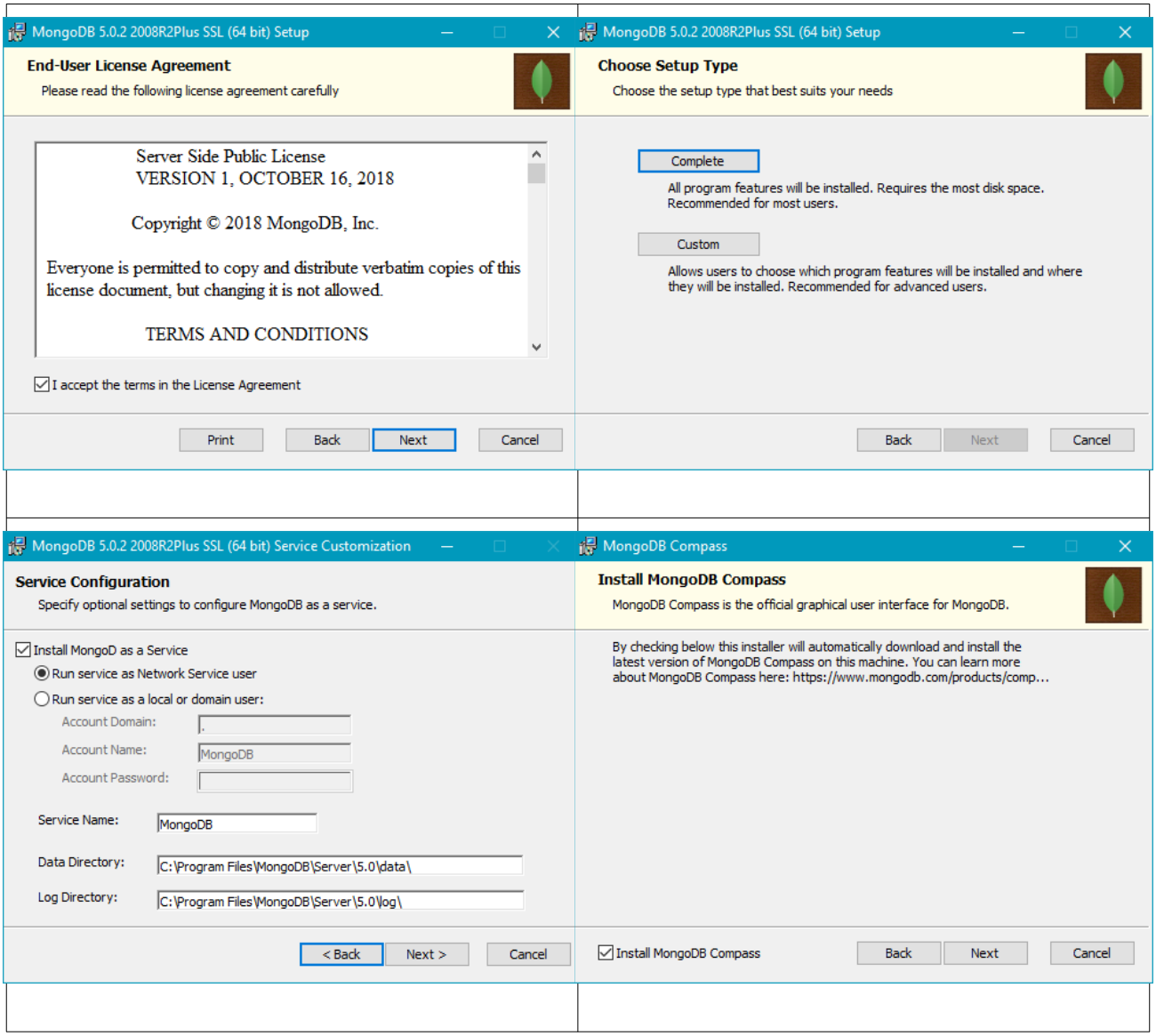
Once you complete the Getting Started Guide, you may find the following topics useful.

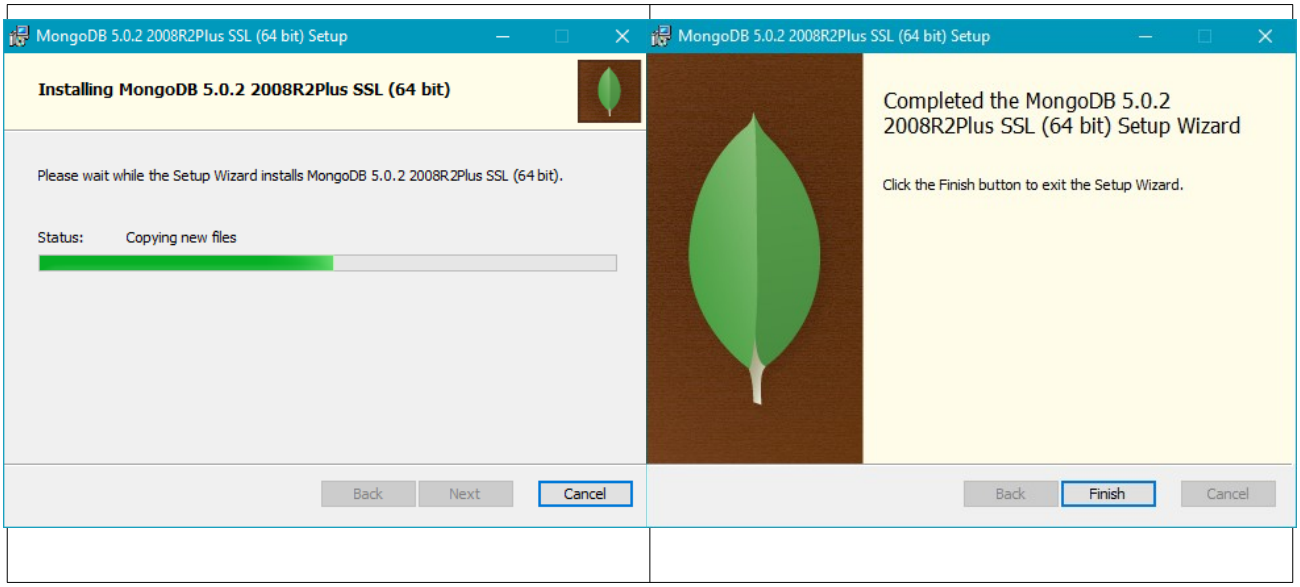
Introduction	Developers	Administrators	Reference
Introduction to MongoDB	CRUD Operations	Production Notes	Shell Methods
Installation Guides	Aggregation	Replica Sets	Query Operators
Databases and Collections	SQL to MongoDB	Sharded Clusters	Reference
Documents	Indexes	MongoDB Security	Glossary

Le lien vers la documentation du serveur est https://docs.mongodb.com/manual/.

B.2 Installation

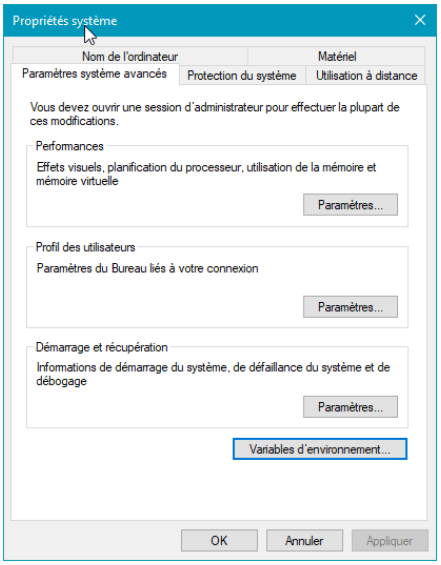
Les étapes de configuration de la base de données sont les suivantes :





Afin d'être tranquille lors de l'utilisation de commandes, vous devez inclure le chemin de MongoDB dans le 'PATH' de **Windows** (et pas de l'utilisateur)

C:\Program Files\MongoDB\Server\5.0\bin



N'oubliez pas de rouvrir votre session pour profiter de cet ajout, ou bien – pour éviter de redémarrer – dans l'invite de commandes, utilisez la commande :

```
setx PATH "%path%;C:\Program Files\MongoDB\Server\5.0\bin"
```

Note pour les sysops :
la commande `set` sous Windows permet de modifier une variable d'environnement directement dans la mémoire du système d'exploitation. Cependant, la commande `setx` s'exécute avec les droits d'administrateur.

B.3 Commandes d'administration

B.3.1 Connexion

Dans une invite de commande, vous pouvez lancer la commande 'mongo tutorial' qui vous connecte sur le lien mongodb://127.0.0.1:27017/tutorial?compressors=disabled&gssapiServiceName=mongodb

B.3.2 Commandes sur les bases

Une fois dans le programme d'administration Mongo, nous pouvons manipuler les bases existantes ou bien en créer de nouvelles.

B.3.2.a Affichage des bases existantes

Pour afficher les bases accessibles, il suffit de taper la commande `show dbs` :

```
> show dbs
admin     0.000GB
config    0.000GB
local     0.000GB
tutorial  0.000GB
```

B.3.2.b Utilisation d'une base

La commande `use` permet de sélectionner une des bases existantes. La commande `db`affiche quand à elle, la base actuellement utilisée.

```
> use tutorial
switched to db tutorial
> db
tutorial
```

Enfin, pour créer une base, il n'y a pas d'instruction propre, c'est mongoDB qui la créera lors de l'insertion d'une collection.

B.3.3 Commandes sur les collections

Nous allons donc créer une base de médecins que nous appellerons "Hopital" et qui contiendra une collection de médecins (l'équivalent d'une table, dans un SGBD-R).

B.3.3.a Création d'une collection

La commande `db.createCollection` permet de créer une collection vide. Notez l'écriture qui ne ressemble pas du tout à du SQL mais plutôt à une instruction en langage objet, avec une écriture respectant le "Camel Case".

```
> use Hopital
switched to db Hopital
> db.createCollection('medecins')
{ "ok" : 1 }
>
```

On peut renommer et effacer une collection, avec les commandes suivantes :

```
> show collections
medecins
> db.medecins.renameCollection('docteurs')
{ "ok" : 1 }
> db.      .renameCollection('     ')
> show collections
docteurs
>
db.docteurs.drop()
```

B.3.3.b Insertion de données

En réalité, MongoDB ne créé la collection que lorsqu'une donnée y est insérée. `Le format des données est JSON.

```
>
db.medecins.insertOne({nom:"ROUSSEL",specialite:"Généraliste",ville:"GRENOBLE",CP:38000}
)
{
        "acknowledged" : true,
        "insertedId" : ObjectId("610f065b2ffdd55d8e6f50d9")
}
>
```

Désormais, il existe une collection avec un document (un seul enregistrement).

Attention, MongoDB créé automatiquement un id pour chaque insertion : ainsi, le champ '_id' contiendra l'id présent lors de l'affichage, comme ci-dessus. Si vous décidez de générer vous-même cet ID, il faut bien évidemment qu'il soit unique.

Pour insérer plusieurs documents, il suffit d'utiliser la commande `db.medecins.insertMany()` dont on passera en paramètre un tableau d'objets au format JSON :

```
>
db.medecins.insert([{nom:"NIDJIAN",specialite:"Généraliste",ville:"GRENOBLE",CP:38100},
{nom:"BRAKARD",specialite:"Ophtalmologue",ville:"VOIRON",CP:38500}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

La réponse indique que 2 documents ont bien été insérés.

B.3.3.c Affichage des données (requêtes)

La commande `db.medecins.find()` est l'équivalent de `SELECT * FROM medecins`en SQL :

```
> db.medecins.find()
{ "_id" : ObjectId("610f065b2ffdd55d8e6f50d9"), "nom" : "ROUSSEL", "specialite" :
"Généraliste", "ville" : "GRENOBLE", "CP" : 38000 }
{ "_id" : ObjectId("610f0a212ffdd55d8e6f50da"), "nom" : "NIDJIAN", "specialite" :
"Généraliste", "ville" : "GRENOBLE", "CP" : 38100 }
{ "_id" : ObjectId("610f0a212ffdd55d8e6f50db"), "nom" : "BRAKARD", "specialite" :
"Ophtalmologue", "ville" : "VOIRON", "CP" : 38500 }
>
```

Une requête plus complexe comporte simplement les propriétés des objets JSON recherchés. La fonction find() accepte deux paramètres (la requête, la projection). Par exemple, pour obtenir la liste de tous les médecins de Grenoble, dans la partie sud (code postal est 38100) :

```
> db.medecins.find({ville : "GRENOBLE", CP : 38100})
{ "_id" : ObjectId("610f0a212ffdd55d8e6f50da"), "nom" : "NIDJIAN", "specialite" :
"Généraliste", "ville" : "GRENOBLE", "CP" : 38100 }
>
```

Les recherches sur les nombres peuvent utiliser les mots clés $gt (greater ou "plus grand"), $lt (less than ou "plus petit"), etc.

```
> db.medecins.find({ CP : {$gt: 38099}})
{ "_id" : ObjectId("610f0a212ffdd55d8e6f50da"), "nom" : "NIDJIAN", "specialite" :
"Généraliste", "ville" : "GRENOBLE", "CP" : 38100 }
{ "_id" : ObjectId("610f0a212ffdd55d8e6f50db"), "nom" : "BRAKARD", "specialite" :
"Ophtalmologue", "ville" : "VOIRON", "CP" : 38500 }
```

Et si on veut uniquement le nom du médecin ({nom : 1}) :

```
> db.medecins.find({ CP : {$gt: 38099}}, {_id: 0, nom: 1})
{ "nom" : "NIDJIAN" }
{ "nom" : "BRAKARD" }
```

```
>
```

B.4 CRUD et opérations liées

B.4.1 Insertion (Create)

Pour insérer des documents, MongoDB accepte 3 commandes :

commande	Commentaire
db.*collection*.insertOne({objet})	
db.*collection*.insert([{objet}, {objet}, {objet}...])	*Commande bientôt dépréciée*
db.*collection*.insertMany([{objet}, {objet}, {objet}...])	Le retour contient la liste des objets validés

B.4.2 Récupération (Read)

Pour la récupération, nous avons abordé la commande principale utilisant find(). Nous ajouterons la commande pretty() qui améliore la lisibilité humaine sans changer le contenu. Testez les deux commandes sur la collection 'medecins'.

commande	Commentaire
db.*collection*.find({requête}, {projection})	Le retour contient la liste brute des objets
db.*collection*.find({requête}, {projection}).pretty()	Le retour contient la liste formatée des objets

test sur la collection :

```
db.medecins.find()
db.medecins.find().pretty()
```

Il existe aussi une commande `findOne()`pour n'avoir qu'un seul document dans les recherches.

B.4.3 Modification (Update)

MongoDB propose deux commandes pour effectuer les modifications :

commande	Commentaire
db.*collection*.updateOne({filtre}, {update}, {option})	Le premier élément filtré est modifié
db.*collection*.updateMany({filtre}, {update}, {option})	Tous les éléments filtrés sont modifiés

Le filtre correspond à une requête de recherche (par exemple : nom : "NIDJIAN").

Le champ update commence par $set

Si le médecin BRAKARD se marie et change de nom, pour PELISSIER :

```
db.medecins.updateOne({nom: "BRAKARD"}, {$set: {nom: "PELISSIER"}})
```

B.4.4 Suppression (Delete)

Enfin, pour l'effacement d'un document ou plusieurs documents, voici les commandes :

commande	Commentaire
db.*collection*.deleteOne({filtre})	Le premier élément filtré est supprimé
db.*collection*.deleteMany({filtre})	Tous les éléments filtrés sont supprimés

B.5 JSON et documents

Il est possible de créer des champs et des sous-champs. Cette fonctionnalité proche de l'objet permet de gérer des champs présents seulement dans certaines fiches mais pas dans d'autres.

Prenons le cas de nos médecins : certains auront un téléphone fixe et d'autres donneront leur téléphone mobile. Le modèle du Docteur PICARD sera un téléphone fixe 0102030405 et un téléphone mobile 0607080910 :

```
db.medecins.insertOne({nom:"PICARD",specialite:"Généraliste",ville:"MEYLAN",CP:38240,
telephone:{fixe:"0102030405", mobile:"0607080910"}})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("611048be665042a5059d624d")
}
```

La fiche du docteur PICARD est le document JSON suivant :

```
{
        "_id" : ObjectId("611048be665042a5059d624d"),
        "nom" : "PICARD",
        "specialite" : "Généraliste",
        "ville" : "MEYLAN",
        "CP" : 38240,
        "telephone" : {
                "fixe" : "0102030405",
                "mobile" : "0607080910"
        }
}
```

Ainsi, pour faire une recherche sur le numéro de téléphone mobile, il faudra rédiger la requête suivante :

```
> db.medecins.find({"telephone.mobile":"0607080910"}).pretty()
{
        "_id" : ObjectId("611048be665042a5059d624d"),
        "nom" : "PICARD",
        "specialite" : "Généraliste",
        "ville" : "MEYLAN",
        "CP" : 38240,
        "telephone" : {
                "fixe" : "0102030405",
                "mobile" : "0607080910"
        }
}
```

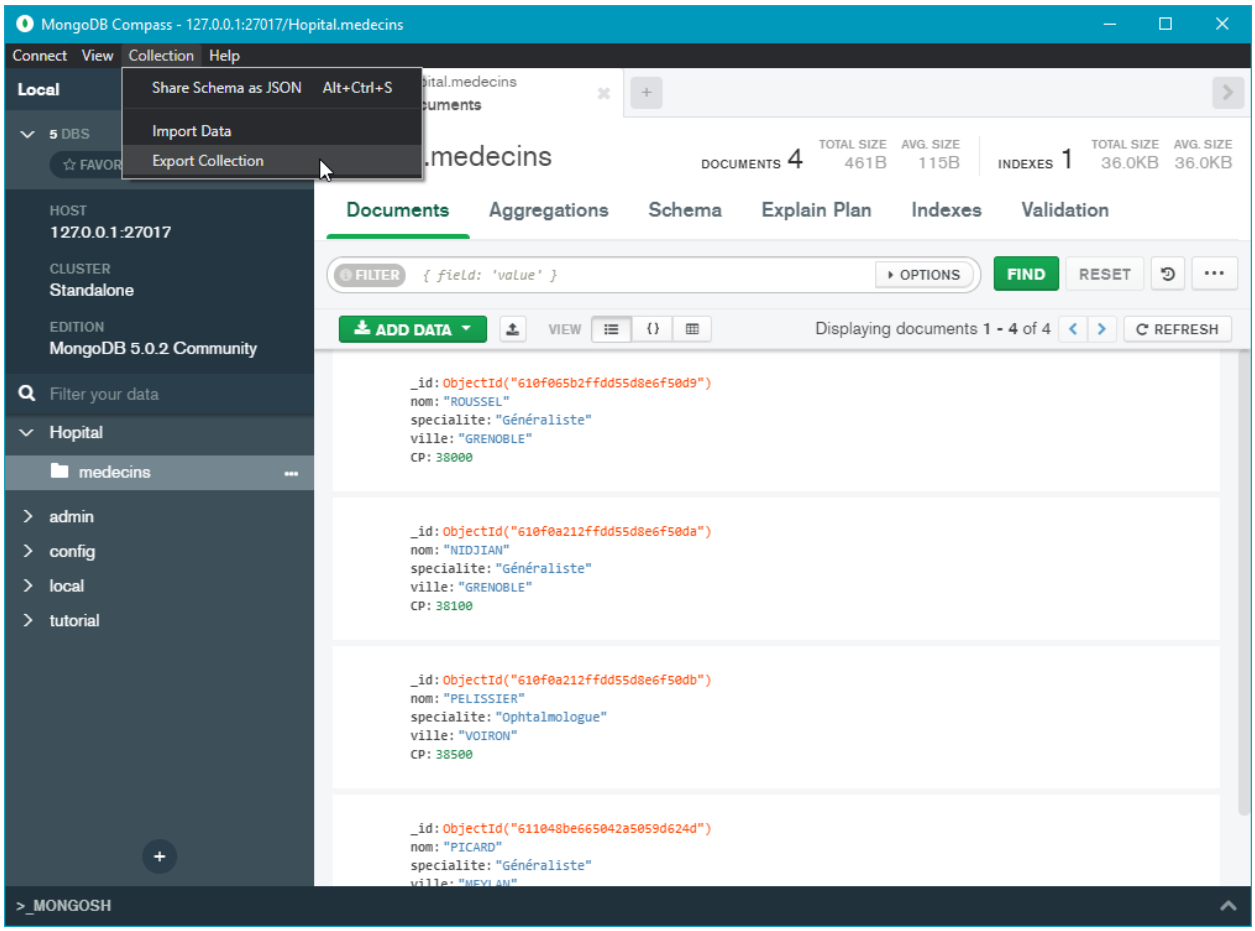
B.6 Outils

Nous terminons cette exploration avec l'outil graphique installé en même temps que la base MongoDB : Compass, puis l'ajout des outils d'administration mongodump et mongoexport.

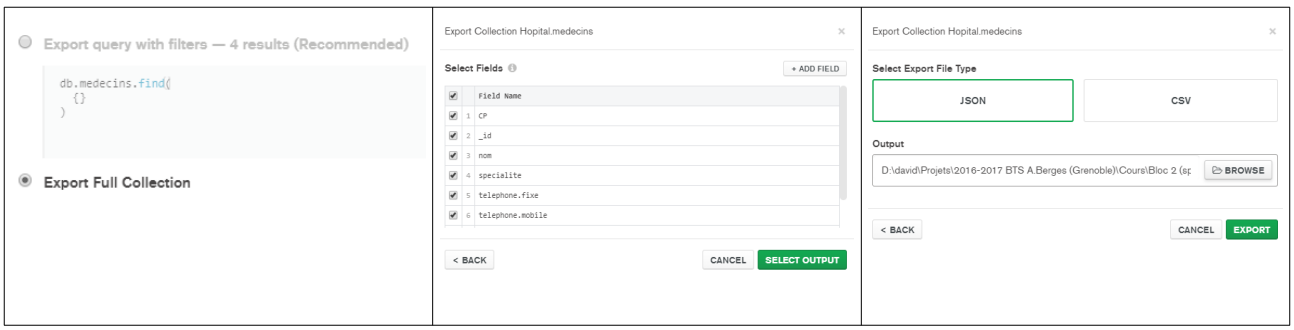
B.6.1 Exportation/importation de collections

Pour se connecter, il faut suivre l'URI donné au départ (au lancement de mongo, il est indiqué) :

mongodb://127.0.0.1:27017



En sélectionnant la collection à exporter, le menu change : collection > Export Collection puis :

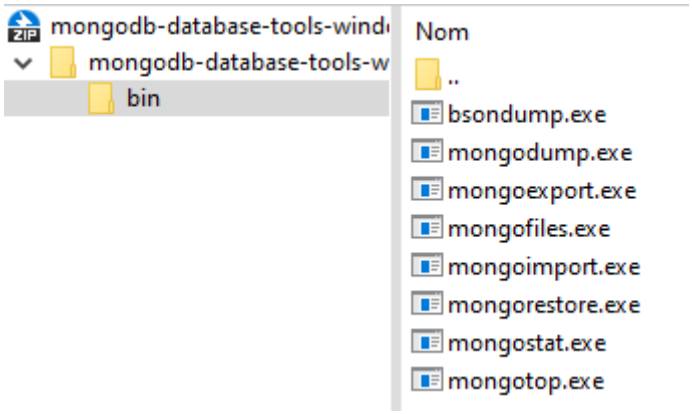


L'importation nécessite de se placer sur une collection vide dans une base, puis cliquer dans le menu Collection > Import et suivre les informations.

B.6.2 Importation/exportation complète

En réalité, il existe des outils pour importer/exporter une base complète, qu'il faut télécharger depuis https://www.mongodb.com/try/download/database-tools

La documentation se trouve sur https://docs.mongodb.com/database-tools/



L'installation de ces outils dans le répertoire de MongoDB permet de ne pas avoir à modifier le chemin d'accès (%PATH%)

Pour tester l'outil mongoexport, voici un jeu de test supplémentaire dans la base Hopital :

```
> db.salles.insertMany([{nom:"Vercors",type:"Réanimation"},{nom:"Chartreuse",type:"Bloc
opératoire"}, {nom:"Belledone",type:"Salle Réveil"}, {nom:"101",type:"chambre"} ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("61105aac665042a5059d624e"),
                ObjectId("61105aac665042a5059d624f"),
                ObjectId("61105aac665042a5059d6250"),
                ObjectId("61105aac665042a5059d6251")
        ]
}
```

La commande mongoexport prend les paramètres supplémentaires pour le serveur, l'utilisateur et le mot de passe dans le format :

```
mongoexport -h 127.0.0.1:27017 -u leNom -p leMotDePasse
```

Mais en local, on peut s'en passer… par contre, il faut lancer la commande pour autant de collections !

```
mongoexport --collection=medecins --db=Hopital --out=d:\temp\medecins.json
2021-08-09T00:31:08.158+0200    connected to: mongodb://localhost/
2021-08-09T00:31:08.184+0200    exported 4 records
mongoexport --collection=salles --db=Hopital --out=d:\temp\salles.json
2021-08-09T00:31:08.158+0200    connected to: mongodb://localhost/
2021-08-09T00:31:08.184+0200    exported 4 records
```

C Annexes

C.1 Les opérateurs de MongoDB

Source : https://geekflare.com/fr/getting-started-mongodb/

Opérateur	Utiliser	Exemple
$ eq	Vérifiez si la valeur est égale	{marques: {$ eq: 20}}
$ lt	Vérifiez si la valeur est inférieure à	{marques: {$ lt: 20}}
$ gte	Vérifiez si la valeur est supérieure ou égale à	{marques: {$ gte: 22}}
$ lte	Vérifiez si la valeur est inférieure à merci ou égale à	{marques: {$ lte: 22}}
$ ne	Vérifiez si la valeur n'est pas égale à	{marques: {$ ne: 22}}
$ en	Vérifiez si la valeur est égale à l'une des valeurs du tableau	{marques: {$ dans: [20,22]}}
nin $	Vérifiez si la valeur n'est égale à aucune valeur du tableau	{marques: {$ nin: [22,25]}}