MongoDB

MongoDB est un SGBD NoSQL de type document. Il a été conçu pour être facilement mis à l’échelle (*scalability*) et c’est probablement le plus populaire des SGBD de ce type.

Il supporte les transactions ACID.

## Installation

1. Installer MongoDB (via download du site Web)
	1. Mettre “C:\Program Files\MongoDB\Server\3.2\bin » dans le path
2. Créer le dossier C:/data/db
3. Démarrer mongodb. Dans CMD, faire : « mongod.exe »
4. Installer python 3.x (en mode administrateur)
5. pymongo : (dans le dossier …/Python/Scripts/) pip install pymongo
	1. S’installe dans “C:\Users\Prof\AppData\Local\Programs\Python\Python35-32”
6. Mettre python dans le path
	1. Pointer vers le dossier du exe

Idéalement, le dossier MongoDB/Server/../bin devrait être dans votre variable d’environnement PATH afin de pouvoir appeler les exécutables de ce dossier à partir de n’importe où sur votre ordinateur.

## Le format JSON

Le format JSON est de plus en plus répandu et est inévitable dans le domaine du Web, principalement lors de la conception ou l’utilisation d’API (ou services Web). Il est également utilisé pour représenter les documents MongoDB.

JSON permet d’exprimer une structure en utilisant une syntaxe simple mais expressive.

Voici un exemple de représentation d’une liste :

["Fred", "Antoine", "Marc-André", "Autre"]

Pour représenter un objet :

{

 nom : "Fred",

 role : ["Prof", "DBA"],

 lieu : "Montreal,

 no : 233,

 actif : true,

 cours : ["DBA", "Web", "Linux"]

}

## Les commandes de base

Pour ouvrir une console et interroger MongoDB, ouvrir un terminal et écrire mongo.exe (dans le dossier …/Scripts/ si le chemin n’est pas dans le PATH).

Pour savoir dans quelle base de données on se trouve :

> db

Pour créer ou pour se déplacer vers une nouvelle base de données :

> use profDB

Pour ajouter un document dans une nouvelle collection (ou une collection existante) :

> db.profList.insertOne({ nom : "Fred", role : ["Prof", "DBA"] })

Pour afficher les collections disponibles dans la base de données:

> db.getCollectionNames()

Pour afficher les éléments de la collection :

> db.profList.find()

Voici quelques exemples où l’on cherche seulement certains éléments de la collection :

> db.profList.find({nom:"Fred"})

> db.profList.find({no:{$gt:200}}); 🡨 plus grand que

> db.profList.find({no:{$gt:5}}).limit(5);

> db.profList.find({no:{$gt:5}}).limit(5).sort({no:-1}) 🡨 1 = asc, -1 = desc

> db.profList.find({no:{$gt:5}}).limit(5).sort({no:-1}).skip(2)

> db.profList.find({$or: [ { role: "DBA", role: "Prof" } ] } )

Pour faire des mises à jour:

> db.profList.updateOne({nom:"Fred", {$set : {…} } }) 🡨 Le 1er argument correspond au WHERE, l’autre

 Le nouveau document JSON

Pour supprimer :

> db.profList.deleteOne({nom : "Fred"})

Pour supprimer une collection:

> db.profList.drop()

## Les index

Il existe plusieurs sortent d’index pour MongoDB. En voici 3 :

Pour créer un index sur un seul champ :

> db.profList.createIndex({nom : 1})

La valeur « 1 » signifie que l’index est que l’ordre des items est ascendant, -1 est descendant. Dans les index sur un seul champ, la valeur n’est pas vraiment importante.

Pour créer un index multiple :

> db.profList.createIndex({nom : 1, role : 1})

Dans ce cas-ci, il est possible d’ordonner les résultats en ascendant ou en descendant, mais pas l’un ascendant et l’autre descendant.

Pour les index sur les tableaux (MultiKey indexes)

> db.profList.createIndex({role : 1})

C’est donc pareil à la création d’index sur un seul champ.

## Créer une copie de sauvegarde

Un peu comme MySQL, on se sert d’un utilitaire :

mongodump --out C:/

Ceci fera un dossier par base de données présente dans MongoDB. La base de données est exportée en format BJSON, JSON binaire.

Pour faire une copie de sauvegarde en format lisible :

mongoexport --db profDB --collection profList --type=json --out ma-bd.json

## L’autre type de collections

Dans MongoDB, il existe un deuxième type de collections qui peut s’avérer fort utile, les collections dites « Capped ». L’idée est de créer une collection avec une grandeur fixe. Qu’arrive-t-il lorsque la collection est remplie? Les données les plus vieilles sont supprimées.



Voici la syntaxe qui permet de créer une collection « capped » de taille 10000 bytes.

> db.createCollection("ma\_col", {"capped" : true, "size" : 10000});

## MapReduce

MapReduce permet d’interroger une base de données NoSQL volumineuse. Un MapReduce se fait en 2 étapes (ou 3 s’il y a un premier filtre de défini). La première partie correspond regroupement des données (map). La deuxième partie permet de formater un résultat, comme un count, une somme, … (reduce).

**Map**

Une Map est une fonction qui possède en entrée un document/valeur et qui retourne des informations. Comme une map s’exécute sur un document à la fois, c’est facilement opérable en parallèle par plusieurs processus.

Exemple de MapReduce, faisant la somme des commandes faites par usager.



Source : <https://docs.mongodb.org/manual/core/map-reduce/>

Depuis quelque temps, mapReduce est maintenant remplacé par « aggregate », une forme plus simple et performante :

db.etudiants.aggregate(

 { $match : { matricule : {$gt : 0 } }},

 {

 $group : {

 \_id : "$genre",

 total : { $sum : "$solde" }

 }

 }

)