Université Abderrahmane Mira de Bejaia
Faculté des Sciences Exactes
Département d'informatique
Année Universitaire : 2023/2024

2ème Année Master IA (S3)

Module : **Big Data**

TP 1 : Manipulation d'un SGBD NoSql

MongoDB est un système de gestion de base de données ou SGBD, comme Mysql ou PostgreSQL, mais dont le mécanisme est complètement différent.. Grâce à MongoDB vous allez pouvoir stocker vos données un peu comme vous le feriez dans un fichier JSON. C'est à dire, une sorte de dictionnaire géant composé de clés et de valeurs. Ces données peuvent ensuite être exploitées par du JavaScript, directement intégré dans MongoDB, mais peuvent également être exploitées par d'autres langages comme le python.

Terminologie SQL	Terminologie MangoDB
Base de données	Base de données
Table	Collection
Ligne	Document
Colonne	Champ
Index	Index
Jointure	Imbrication ou Référence
Clef Primaire (peut être multiple)	Clef Primaire (une seule, et représentée par le champ _id)

Quelques commandes :

- Après avoir installé MangoDB, pour lancer le moteur de base de données : `mongod`

- Afficher toutes les bases de données : `show dbs`

Normalement vous devriez avoir une base **local** propre à mongo et une base **test**:

- Création d'une base de données : `use <nom de la base de données>`

Vous pouvez faire **db** pour voir la base de données courante. Attention, si vous faites **show dbs**, vous ne verrez pas encore votre base. En effet, mongo attend d'avoir du contenu pour créer votre base.

- Pour créer une collection, il suffit simplement d'ajouter un patient. Par exemple pour :

```
{
    "nom":"Dupond",
    "prenom":"Jean Claude",
    "ddn": new Date('May 18, 1984')
}
```

On fait : `db.patients.insert({"nom":"jay gould", "prenom":"stephen", new Date('May 18, 1984')})`

La collection ***patients*** se crée automatiquement lors de la première utilisation.

- voir le document : **`db.patients.find()`**

Notez que MongoDB ajoute automatiquement un **_index** si rien n'est spécifié.

- On va remplir notre collection en répétant cette procédure 50 fois.

```
for ( var i = 0 ; i&lt;50; i++){
    db.patients.insert({"nom":"jay gould" , "prenom":"stephen", "age": i})
```

```
}
```

- Vérifier le nombre de patients : **`db.patients.count()`**
- Retourner toute la liste de la collection patients : `db.patients.find()`
- Récupérer les patients dont l'âge = 5 : `db.patients.find({age:5})`
- Afficher tous les prénoms commençant par "*j*" : `db.patients.find({prenom: /^j*/})`
- Récupérer les patients dont l'âge est supérieur à 40 : `db.patients.find({age:{$gt:40}})`

$gt est un mot clef de mongo qui veut dire *greater than* (*supérieur à*).

- Récupérer les patients dont l'âge est 5 ou 10 : `db.patients.find({age:{$in:[5,10]}})`
- Récupérer uniquement les noms des patients dont l'âge est supérieur à 40 :

```
db.patients.find({age:{$gt:40}},{"nom":true})
```

Pour limiter le nombre de résultat à 3 : `db.patients.find().limit(3)`

- Pour ordonner la liste par âge décroissant. -1 pour décroissant et 1 pour croissant :

```
db.patients.find().sort(age:-1)
```

- Modifier la collection : `update(query, update, options)`
- Remplacer tous les prénoms *stephen* par *boby :*

```
db.patients.update({"prenom":"stephen"},{$set:{"prenom":"boby"}},{multi:true})
```

- Ajoute une clé *sexe* à tous les patients :

```
db.patients.update({prenom:"boby"}, {$set:{sexe:"male"}}, {multi:true})
```

- Ajouter un patient *olivier* s'il n'existe pas :

```
db.patients.update({prenom:"olivier"}, {$set:{sexe:"male"}}, {upsert:true})
```

- save(document, writeConcern)

La différence avec **insert** est que **save**, fait un **update** du document s'il existe déjà :

```
db.patient.save({"prenom":"jean claude", "nom":"Van Damme"})
```

- Suppression : `remove(query,justOne)`
- Supprimer tous les patients qui s'appellent olivier **:** `db.patients.remove({prenom:"olivier"})`
- Supprimer la collection : `db.patients.drop()`
- Supprimer la base de donnée :

```
use medical
db.runCommand({dropDatabase: 1});
```

<u>Consignes pour exécuter les requêtes:</u>

1. Se connecter à WINDOWS
2. Lancer la commande MongoDB
3. Copiez votre base de donnée (vous pouvez créer votre propre base de données)

<u>Activités</u>

Ecrire les requêtes **MongoDB** qui permettent de :

1. Afficher toutes les collections de la base :
2. Afficher tous les documents de la base
3. Compter le nombre de documents de la collection employés
4. Insérer de deux manières différentes deux employés avec les champs nom, prénom et soit prime soit ancienneté
5. Afficher la liste des employés dont le prénom est David
6. Afficher la liste des employés dont le prénom commence ou se termine par D

7. Afficher la liste des personnes dont le prénom commence par D et contient exactement 5 lettres
8. Afficher la liste des personnes dont le prénom commence et se termine par une voyelle
9. Afficher la liste des personnes dont le prénom commence et se termine par une même lettre
10. Afficher le noms et prénom de chaque employé ayant une ancienneté > 10
11. Afficher les nom et adresse complète des employés ayant un attribut rue dans l'objet adresse
12. Incrémenter de 200 la prime des employés ayant déjà le champ prime
13. Afficher les trois premières personnes ayant la plus grande valeur d'ancienneté
14. Regrouper les personnes dont la ville de résidence est Toulouse (afficher nom, prénom et ancienneté)
15. Afficher les personnes dont le prénom commence par M et la ville de résidence est soit Foix soit Bordeaux
16. Mettre à jour l'adresse de Dominique Mani : nouvelle adresse ({ numero : 20, ville : 'Marseille',codepostal : '13015' }). Attention, il n'y aura plus d'attribut rue dans adresse
17. Attribuer une prime de 1 500 à tous les employés n'ayant pas de prime et dont la ville de résidence est différente de Toulouse, Bordeaux et Paris.
18. Remplacer le champ tel, pour les documents ayant un champ tel), par un tableau nommé téléphone contenant la valeur du champ tel (le champ tel est à supprimer)
19. Créer un champ prime pour les documents qui n'en disposent pas et de l'affecter à 100 * nombre de caractère du nom de la ville
20. Créer un champ mail dont la valeur est égale soit à nom.prenom@formation.fr pour les employés ne disposant pas d'un champ téléphone, soit à prenom.nom@formation.fr (nom et prénom sont à remplacer par les vraies valeurs de chaque employé)
21. Calculer et afficher la somme de l'ancienneté pour les employés disposant du même prénom