# Loric Informatique

## MongoDB (NoSQL)

**Version 1.1 (créé le 14/03/2024, modifié le 04/04/2024)**

NoSQL (No Structured Query Language) est un langage servant à exploiter des bases de données non-relationnelles.

MongoDB est un système de gestion de base de données orienté documents qui permet de manipuler des objets structurés au format JSON.

### Outils nécessaires :

- MongoDB (contenant mongod)
- MongoSH (contenant mongosh)
- MongoDB Command Line Database Tools
- Compass (facultatif)

### Table des matières :

# I. Prise en main

## I.1. Installer MongoDB

Télécharger MongoDB, et créer un dossier nommé data dans le répertoire C:\ et ajouter un dossier db dans ce dossier.

Copier le contenu des dossiers bin de MongoSH et MongoDB Command Line Database Tools dans le dossier bin de MongoDB.

### I.2. Démarrer MongoDB

Ouvrir mongod.exe dans une console pour démarrer le serveur MongoDB.

Ouvrir ensuite mongosh.exe dans une autre console pour démarrer le client au serveur. Appuyer sur entrée pour utiliser le serveur local. C'est dans cette console que le code pourra être exécuté.

### I.3. Arrêter proprement MongoDB

Dans la console de mongosh.exe, tapez la commande : `db.shutdownServer()`

## II. Bases

### II.1. Syntaxe dans le fichier json

### II.2. Commentaires dans un fichier json

### II.3. Les conditions

| Condition | Signe équivalent |
|---|---|
| "*champ*" : *valeur* | == |
| "*champ*" : {$eq : *valeur*} | == |
| "*champ*" : {$lt : *valeur*} | ‹ |
| "*champ*" : {$lte : *valeur*} | ‹= |
| "*champ*" : {$gt : *valeur*} | › |
| "*champ*" : {$gte : *valeur*} | ›= |
| "*champ*" : {$ne : *valeur*} | != |
| "*champ*" : {$in : [*valeur1, valeur2…*]} | Contient une de ces valeurs |
| "*champ*" : {$nin : [*valeur1, valeur2…*]} | Ne contient pas une de ces valeurs |
| $and : [{*condition1*}, {*condition2*}…] | *condition1* et *condition2* |
| $or : [{*condition1*}, {*condition2*}…] | *condition1* ou *condition2* |
| $not : {*condition*} | !*condition* |

## III. Requêtes d'initialisation de la base de données

### III.1. Utiliser une base de données (ou la créer si elle n'existe pas) pour réaliser des

### requêtes

```
use maBaseDeDonnees
```

### III.2. Afficher les bases de données

```
show dbs
```

### III.3. Supprimer une base de données

```
use maBaseDeDonnees
db.dropDatabase()
```

### III.4. Afficher les collections d'une base de données

```
use maBaseDeDonnees
show collections
```

### III.5. Supprimer une collection d'une base de données

```
use maBaseDeDonnees
db.maCollection.drop()
```

### III.6. Importer une collection (avec mongoimport)

Ouvrir la console dans le dossier bin du MongoDB, et tapez :
`mongoimport fichier.json /db:maBaseDeDonnees --jsonArray --collection maCollection`

## IV. Requêtes de mise à jour de la collection

### IV.1. Insérer un document

```
db.maCollection.insertOne({"champ1" : valeur1, "champ2" : valeur2…})
```

> Si aucune valeur n'est renseignée pour le champ _id, la valeur est générée automatiquement (sous forme d'un Objet et non d'un entier)

### IV.2. Insérer plusieurs documents

```
db.maCollection.insertMany([{"champ1" : valeur1, "champ2" : valeur2…}, {"champ1" : valeur3, "champ2" : valeur4…}])
```

### IV.3. Mettre à jour un document

#### IV.3.1. Mise à jour simple

```
db.maCollection.updateOne({condition}, {$set : {"champ1" : valeur1}})
```

#### IV.3.2. Faire incrémenter un document

```
db.maCollection.updateOne({condition}, {$inc : {"champ1" : valeur1}})
```

#### IV.3.3. Mettre à jour un tableau

##### IV.3.3.1. Pour un rang du tableau

```
db.maCollection.updateOne({condition}, {$set : {"tableau.rang" : valeur1}})
```

##### IV.3.3.2. Pour ajouter une valeur dans le tableau sans redondance

```
db.maCollection.updateOne({condition}, {$addToSet : {"tableau" : valeur1}})
```

##### IV.3.3.3. Pour ajouter une valeur dans le tableau avec redondance

```
db.maCollection.updateOne({condition}, {$push : {"tableau" : valeur1}})
```

##### IV.3.3.4. Pour ajouter plusieurs valeurs dans le tableau (avec ou sans redondance)

```
db.maCollection.updateOne({condition}, {$addToSet : {"tableau" : {$each : [valeur1,
valeur2…]}}})
```

##### IV.3.3.5. Pour supprimer une valeur dans le tableau sans redondance

```
db.maCollection.updateOne({condition}, {$pull : {"tableau" : valeur1}})
```

##### IV.3.3.6. Pour mettre à jour une occurence du tableau des valeurs (respectant les conditions)

```
db.maCollection.updateOne({condition}, {$inc : {"tableau.$" : valeur1}})
```

##### IV.3.3.7. Pour mettre à jour toutes les occurences du tableau des valeurs

```
db.maCollection.updateOne({condition}, {$inc : {"tableau.$[]" : valeur1}})
```

### IV.3.4. Supprimer un champ

```
db.maCollection.updateOne({condition}, {$unset : "champ1"})
```

## IV.4. Mettre à jour plusieurs documents

### IV.4.1. Mise à jour simple

```
db.maCollection.updateMany({condition}, {$set : {"champ1" : valeur1}})
```

### IV.4.2. Faire incrémenter plusieurs documents

```
db.maCollection.updateMany({condition}, {$inc : {"champ1" : valeur1}})
```

### IV.4.3. Mettre à jour un tableau

#### IV.4.3.1. Pour un rang du tableau

```
db.maCollection.updateMany({condition}, {$set : {"tableau.rang" : valeur1}})
```

#### IV.4.3.2. Pour ajouter une valeur dans le tableau sans redondance

```
db.maCollection.updateOne({condition}, {$addToSet : {"tableau" : valeur1}})
```

#### IV.4.3.3. Pour ajouter une valeur dans le tableau avec redondance

```
db.maCollection.updateOne({condition}, {$push : {"tableau" : valeur1}})
```

#### IV.4.3.4. Pour ajouter plusieurs valeurs dans le tableau (avec ou sans redondance)

```
db.maCollection.updateOne({condition}, {$addToSet : {"tableau" : {$each : [valeur1,
valeur2…]}}})
```

#### IV.4.3.5. Pour supprimer une valeur dans le tableau sans redondance

```
db.maCollection.updateOne({condition}, {$pull : {"tableau" : valeur1}})
```

#### IV.4.3.6. Pour mettre à jour une occurence du tableau des valeurs (respectant les conditions)

```
db.maCollection.updateOne({condition}, {$inc : {"tableau.$" : valeur1}})
```

#### IV.4.3.7. Pour mettre à jour toutes les occurences du tableau des valeurs

```
db.maCollection.updateOne({condition}, {$inc : {"tableau.$[]" : valeur1}})
```

#### IV.4.3.8. Pour mettre à jour toutes les occurences du tableau des valeurs respectant la condition du tableau

```
db.maCollection.updateOne({condition}, {$inc : {"tableau.$[elem].champ1Tableau" :
valeur1}}, {arrayFilters: [{"elem.champ2Tableau" : valeur2}]})
```

### IV.4.4. Supprimer un champ

```
db.maCollection.updateMany({condition}, {$unset : "champ1"})
```

### IV.4.5. Mises à jour multiples

```
db.maCollection.updateMany({condition}, [{$set : {"champ1" : valeur1}, {$set : {"champ2" :
valeur2}}…])
```

## IV.5. Supprimer un document

```
db.maCollection.deleteOne({condition})
```

### IV.6. Supprimer plusieurs document

```
db.maCollection.deleteMany({condition})
```

## V. Requêtes de sélection de données

### V.1. find

La commande find permet d'interroger une base de données MongoDB et de renvoyer toutes les valeurs de la collection.

#### V.1.1. Afficher toute la collection

```
db.maCollection.find({})
```

#### V.1.2. Afficher certains champs

```
db.maCollection.find({}, {"_id" : 0, "champ1" : 1, "champ2" : 1…})
```

> Par défaut, le champ _id est toujours affiché, pour le masquer, définir la visibilité à 0. Pour afficher un champ, définir la visibilité à 1.

#### V.1.3. Afficher certains champs d'une liste

```
db.maCollection.find({}, {"_id" : 0, "liste1.champ1" : 1, "champ2" : 1…})
```

#### V.1.4. Afficher les documents respectant la condition

```
db.maCollection.find({condition}, {"_id" : 0, "champ1" : 1, "champ2" : 1…})
```

#### V.1.5. Limiter le nombre de documents à afficher

```
db.maCollection.find({condition}, {"_id" : 0, "champ1" : 1, "champ2" : 1…}).limit(n)
```

### V.2. distinct

La commande distinct permet d'afficher facilement différentes valeur que peut prendre un champ

Commande : `db.maCollection.distinct("champ")`

### V.3. aggregate

la commande aggregate permet de faire des requêtes plus avancées, notamment des calculs sur les attributs.

Commande : `db.maCollection.aggregate([{operateur1}, {operateur2}…])`

#### V.3.1. L'opérateur group

##### V.3.1.1. Grouper sur un champ

`$group : {_id : "$champ"}`

#### V.3.1.2. Grouper sur un champ et faire des calculs

##### V.3.1.2.1. Commande pour grouper un champ et faire des calculs

`$group : {_id : "$champ", nomChamp : {operateurCalcul}}`

> Note : Pour faire des calculs sur tous les documents, il est possible de faire :
> `$group : {_id : null, nomChamp : {operateurCalcul}}`

##### V.3.1.2.2. Les différents calculs possibles

| Opérateur de calcul | Utilité |
|---|---|
| `$sum : "$champ"` | Calculer la somme des valeurs |
| `$avg : "$champ"` | Calculer la moyenne des valeurs |
| `$min : "$champ"` | Calculer la valeur minimale |
| `$max : "$champ"` | Calculer la valeur maximale |
| `$count : {}` | Calculer le nombre d'documents (à partir de la version 5.0 de mongoDB) |
| `$sum : 1` | Calculer le nombre d'documents (pour toutes les versions) |

### V.3.2. L'opérateur match

Cet opérateur permet d'utiliser une condition (avant ou après un regroupement).

Commande : `$match : {condition}`

### V.3.3. L'opérateur unwind

Cet opérateur permet de transformer chaque élément d'un tableau en un document séparé contenant tous les champs.

Commande : `$unwind : "$tableau"`

### V.3.4. L'opérateur sort

Cet opérateur permet de trier les résultats dans l'ordre croissant (1) ou dans l'ordre décroissant (-1).

Commande : `$sort : {"champ1" : ordre1, "champ2" : ordre2…}`

### V.3.5. L'opérateur project

Cet opérateur permet de n'afficher que certains champs. C'est l'équivalent de la commande find.

Commande : `$project : {"_id" : 0, "champ1" : 1, "champ2" : 1…}`