M. AZZOUZ

Bases de Données Avancées

TD/TP : NoSQL- MongoDB

– USTHB Master 01 IL–
M. AZZOUZ
Dernière mis à jour :Mai 2020

Exercice 01

a. **Créer une nouvelle base de données nommée info et vérifiez qu'elle est sélectionnée.**

❑Syntaxe de création de base de données MongoDB est la suivante: **use DATABASE_NAME.**

❑La clause **use**: Si la base de données n'existe pas le SGBD va créer une base de données. Sinon le SGBD va se connecter à la base de données spécifiée. D'où use permet de créer et se connecter à une BD.

>use info
switched to db info
> db
info

❑Si vous voulez voir toutes les bases de données, vous pouvez utiliser la commande **show dbs**.

Exercice 01

- La syntaxe de suppression d'une base de données MongoDB est la suivante: **db.dropDatabase()**.

- L'instruction permet la supprimer de la base de données actuellement connectée. Le SGBD connecte par défaut la base de données test.

- **Exemple:**

```
> use mydb
switched to db mydb
> db.dropDatabase()
{ "ok" : 1 }
```

Exercice 01

Dans MongoDB, comme nous le verrons par la suite, nous utilisons un formalisme de type **db.collection.fonction()** :

- **db** représente la base de données choisie grâce à la commande use (ce mot clé est non modifiable).
- **collection** représente la collection dans laquelle nous allons effectuer l'opération, et doit donc correspondre à une des collections présentes dans la base.
- **fonction()** détermine l'opération à effectuer sur la collection.

Exercice 01

b. Créer une nouvelle collection nommée produits et y insérer le document suivant:

- La syntaxe de création de la collection est **db.createCollection("CollectionName")**

> db.createCollection("produit")
{ "ok" : 1 }

- Pour afficher les collections d'une BD, on utilise : **show collections**
- Plusieurs fonctionnalités sont disponibles pour insérer des documents: **Insert**, **InsertOne** et **InsertMany**.
- Il est possible d'insérer directement le document ou le définir puis l'insérer.

db.produit.insert({"nom": "Macbook Pro", "fabriquant": "Apple", "prix": 11435.99, "options": ["Intel Core i5", "Retina Display", "Long life battery"]})

Exercice 01

c. et d. Ajout d'autres documents:

- document1=({"nom" : "Macbook Air",
"fabriquant" : "Apple",
"prix" : 125794.73, "ultrabook" : true,
"options" : ["Intel Core i7", "SSD", "Long life battery"]})

db.produit.insert(document1)

- document2=({"nom" : "Thinkpad X230",
"fabriquant" : "Lenovo",
"prix" : 114358.74, "ultrabook": true,
options: ["Intel Core i5", "SSD", "Long life battery"]})

db.produit.insert(document2)

Exercice 01

e. Requêtes:

- Récupérer tous les produits.

➢Pour visualiser tous les documents de la collection on utilise la fonction **find()**.

db.produit.find()

➢L'affichage rendu par find() est compact et peu lisible directement. On peut ajouter la fonction pretty() pour avoir une présentation propre.

db.prouduit.find().pretty()

Exercice 01

e. Requêtes:

La fonction find() sans paramètre, elle renvoie l'ensemble des documents. Mais celle-ci peut aussi prendre deux paramètres :

- Les critères de sélection des documents (condition)
- Les choix d'items des documents à afficher(projection)

Ces deux paramètres doivent être écrits sous la forme d'objets JSON.

Exercice 01

e. Requêtes:

- Récupérer le premier produit.
  - db.produit.**findOne**()
  - ou bien db.produit.find()[0]

- Une autre fonction très utile pour mieux appréhender les données est de lister les valeurs prises par les différents items de la collection sans les doublons, grâce à **distinct**().

  - db.produit.**distinct**("fabriquant")

```
[   "Apple",
    "Lenovo"]
```

Exercice 01

e. Requêtes:

❑Trouver l'id du **Thinkpad X230** et faire la requête pour récupérer ce produit avec son id.

➢ db.produit.find(
{ "nom": "Thinkpad X230" }, ← Critères de sélection
{"_id":1}) ← Projection: les items à garder

Si l'on désire n'afficher que certains éléments, il est possible d'ajouter un deuxième argument spécifiant les items que l'on veut (avec 1) ou qu'on ne veut pas (avec 0).

- Résultat:{"_id":ObjectId("5ea74afa390a5ba392b3c7fd")}

-db.produit.find(
{"_id": ObjectId("5ea74afa390a5ba392b3c7fd")})

Exercice 01

e. Requêtes:

❑Récupérer les produits dont le prix est supérieur à 13723 DA.

➢Pour les comparaisons, nous disposons des opérateurs **$eq(equal)**, **$gt (greater than)**, $gte (greater than or equal), **$lt (less than**), **$lte (less than or equal)** et **$ne (not equal).**

-db.produit.find(
{"prix": {$gt: 13723}},
{"_id":0})

➢En plus de ces comparaisons simples, nous disposons d'opérateurs de comparaisons à une liste : **$in** (présent dans la liste) et **$nin** (non présent dans la liste).

Exercice 01

e. Requêtes:

- Récupérer le premier produit ayant le champ ultrabook à true.

Pour limiter le nombre de documents renvoyés par la fonction find() en lui ajoutant la fonction **limit()**, comme ici où nous nous restreignons au premier résultat.

```
-db.produit.find(
{"ultrabook" : true},

{"nom":1, "_id":0}

).limit(1)
```

Exercice 01

e. Requêtes:

- Une autre opération classique est le tri des résultats, réalisable avec la fonction **sort()**. On doit indiquer les items de tri et leur attribuer une valeur de 1 pour un tri ascendant et une valeur de -1 pour un tri descendant. On affiche ici les produits dans l'ordre croissant de leur prix.

db.produit.find().**sort**({ "prix" : **1** })

Idem que précédemment, mais dans l'ordre décroissant

db.produit.find().**sort**({ "prix" : **-1** })

Exercice 01

e. Requêtes:

- Utilisation des expressions régulières
- Récupérer le premier produit dont le nom contient Macbook.

db.produit.find({"nom": **/Macbook/**}).limit(1)

- Récupérer les produits dont le nom commence par Macbook

db.produit.find({"nom":**/^Macbook/**},
{"_id":0, "nom":1, "prix":1}
).sort({ "nom" : 1 })

- Récupérer les produits dont le nom se termine par Macbook: db.produit.find({"nom": **/Macbook$/**}).limit(1)

Exercice 01

- **Mise à jour d'un document dans une collection**

Dans cet exemple on modifié le prix de produit de nom Macbook Air.

db.produit.**updateOne**({"nom" : "Macbook Air"}, **{$set**:{"prix" : 125795}}).

- **Mise à jour de plusieurs documents dans une collection**

Dans cet exemple on modifié le fabricant des produit dont le nom commence par Macbook.

db.produit.**updateMany**({"nom":/^Macbook/}, **{$set**:{"fabriquant" : "Apple Entreprise"}})

Exercice 01

❑**Opération de mise à jour avec remplacement d'un document dans une collection:**

db.produit.**replaceOne**(
{"nom" : "Macbook Pro"},
{"nom" : "Macbook Pro",
"fabriquant" : "Apple",
"prix" : 11436,
"options" : [
"Intel Core i5",
"Retina Display",
"Long life battery"
]})

- Le document remplacé conserve le même identifiant de l'objet **_id**.

Exercice 01

e. Requêtes:

- Supprimer les deux produits dont le fabricant est Apple

-db.produit.**remove**({"fabriquant": "Apple"})

- Supprimer le Thinkpad X230 en utilisant uniquement son id
  - db.produit.find({"nom":"Thinkpad X230"}, {"_id":1, "nom":1})
  - **Résultat**:

{ "_id" : ObjectId("**5ea74afa390a5ba392b3c7fd**"),
"nom" : "Thinkpad X230"}

  - db.produit.remove({"_id": ObjectId("**5ea74afa390a5ba392b3c7fd**")})