MongoDB CRUD

Une fois les documents ajoutés aux collections, il est possible de les manipuler à travers le CRUD (création, lecture, mise à jour et suppression).

Ajouter un document

Le travail commence par définir une base. Avec MongoDb, il suffit de déclarer la base à utiliser et si elle n'existe pas, le moteur la créée :

```
use ecole
```

Donc pas de table en NoSQL, mais des collections qui vont rassembler des documents de même nature. La création de collections est implicite à l'ajout du premier document. Pour ça il faut passer par la commande : **db.<nom de collection>.insert(<document>)**.

```
db.eleves.insert({nom : "Granger", prenom : "Hermione"})
```

Pour vérifier que le document a bien été ajouté à votre collection, passez par la commende :

```
db.eleves.find()
```

Vous pouvez constater que MongoDb à ajouter un id, automatiquement, au document. Il le fait quand ce dernier n'est pas spécifié à l'insertion.

Deux choses à retenir sur l'insertion :

- MongoDb est *shemaless*, c'est-à-dire qu'il n'y a pas de format à respecter pour tous les documents d'une même collection. Par exemple, on peut faire **db.eleves.insert({nom : "Malefoy", age : 16})**. Cela rend la base évolutive. C'est pratique pour démarrer un projet lorsque toutes les données ne sont pas encore connues.
- Même s'il n'y a pas de structure, il est recommandé de respecter les collections. Il faut créer des collections différentes par type de documents. Par exemple, n'allez pas mélanger les élèves et les classes dans une même collection.

A Noter que MongoDB limite la taille des documents à 16MB, ce qui est une taille important pour un document JSON. Si vos documents sont plus importants, vous pouvez les découper en plusieurs collections (si vous ne l'avez pas fait avant). Comme dans une base relationnelle, on utilise les _id pour faire le lien entre plusieurs collections. La différence est qu'il n'y a pas de jointure. Il vous faudra alors faire 2 requêtes.

Javascript

L'interpréteur supporte, en plus des commandes MongoDb, le javascript. Il est donc possible de travailler sur les tables avec du code, comme dans l'exemple ci-dessous :

```
for(i=1; i<=100000; i++)
{
	var dixmill = i%10000;
	var mill = i%1000;
	var cent = i%100;
	db.produits.insert({compteur:i, dixmill:dixmill, mill:mill, cent:cent });
};
```

Cette boucle permet d'ajouter 100000 documents à la collection *produits*. Puisque la base n'a pas été changée, la collection est ajoutée à *ecole*. Pour pouvoir executer ce code dans la console, faites toutes les instructions sur une seule ligne, sans retour chariot.

Les recherches

Faites la commande pour afficher les documents de la collection *produits* :
```
db.produits.find()
```

Par défaut, MongoDB n'affiche que les 20 premiers documents, c'est la capacitée d'un page de résultat. Pour vous aidez dans la lecture des documents vous pouvez utiliser la commande **it**, pour passer de page en page.

L'instruction **find()** prend des paramètres pour affiner les recherche. Par exemple, vous pouvez utiliser un objet JSON comme critère de recherche :
```
db.produits.find({dixmill:9999})
```

Il est possible de préciser la recherche en y ajoutant d'autres critères, dans le même objet :
```
db.produits.find({compteur:19999, dixmill:9999})
```

Vous pouvez compter le nombre de documents en ajoutant la commande **count()** après le **find()** :
```
db.produits.find({dixmill:9999}).count()
```

Il existe aussi des opérateurs de recherches. Il est possible de les utiliser avec des chaînes de caractères et dans ce dernier cas, c'est l'ordre alphabétique qui est appliqué.

- **$and** : cumul des critères,
- **$gt** : plus grand que,
- **$gte** : plus grand ou égal à,
- **$in** : inclus dans un tableau de valeurs,
- **$lt** : plus petit que,
- **$lte** : plus petit ou égal à,
- **$or** : récupère les documents qui correspondent à 2 critères différents,
- **$ne** : différent du critère,
- **$nin** : non inclus dans un tableau de valeurs.

Exemple :
```
db.produits.find({dixmill:{$gte:9997, $lt:9999}})
```

Lorsque le retour de la recherche est un unique document, utilisez **findOne()**. Cette fonction ne renvoie qu'un seul et unique document (le premier s'il y en a plusieurs).
```
db.produits.findOne({dixmill:{$gte:9997, $lt:9999}})
```

Enfin, pour afficher les collections d'une base, vous pouvez passez par l'instruction **show** :
```
show collections
```

Les filtres

find() prend un deuxième paramètre qui permet de récupérer que certaines propriétés et pas le document dans son ensemble. Dans ce cas, le paramètre est un objet littéral qui précise les propriétés voulues avec une valeur **1**.
```
db.produits.find({dixmill:525}, {compteur:1})
```

Par défaut, l'id est toujours retourné dans le résultat. Si vous ne le voulez pas, il faut le préciser en donnant la valeur **0** à la propriété **_id**.

```
db.produits.find({dixmill:525}, {_id:0,compteur:1})
```

Dans ce dernier cas, les id des documents ne sont plus retournés.

Les tris

Le tri se fait avec la commande **sort()** qui prend aussi un objet JSON en paramètre, contenant le critère de tri et une valeur pour le sens du tri.
Pour classer dans l'ordre croissant les valeurs de compteur.

```
db.produits.find({dixmill:525}).sort({compteur:1})
```

Pour classer dans l'ordre décroissant les valeurs de compteur.

```
db.produits.find({dixmill:525}).sort({compteur:-1})
```

Mettre à jour

La commande **update()** permet la mise à jour. Le premier objet sert de critère de recherche pour cibler le document à modifier et le deuxième contient la mise à jour à faire.

```
db.produits.update({dixmill:525}, {$set : {ajout:"nouveau"}})
```

Mais cette commande ne met à jour que le premier document trouvé. Si vous voulez modifier tous les documents correspondant au critère, faites comme ceci :

```
db.produits.update({dixmill:525}, {$set : {ajout:"nouveau"}}, {multi:true})
```

Il est aussi possible de supprimer une propriété.

```
db.produits.update({dixmill:525}, {$unset : {ajout:"nouveau"}}, {multi:true})
```

La mise à jour peut aussi s'appliquer à un tableau, en lui ajoutant un élément :

```
db.produits.update({dixmill:525}, {$push : {tab : 'd'}})
```

il existe aussi un opérateur pour ajouter plusieurs éléments, **$pushAll**, et un pour ajouter sans doublons **$addToSet**. Et pour supprimer un élément, c'est **$pop** :

```
db.produits.update({dixmill:525}, {$pop : {tab : 1}})
```

Cette commande supprime le dernier élément du tableau. Pour le premier, il faut la valeur **-1**.

Supprimer

La commande **remove()** supprime tous les documents correspondant au critère donné :

```
db.produits.remove({mill: 600})
```

La suppression d'une collection complète se fait avec l'instruction suivante :

```
db.produits.drop()
```

Recherche sur les propriétés

Puisque MongoDb est *shemaless*, tous les documents d'une collection n'ont pas forcément tous les mêmes propriétés. Il est donc possible de faire des recherches sur les propriétés, et non sur les valeurs, avec l'opérateur **$exists**.

```
db.eleves.find({age: {$exists:1}})
```

Dans cet exemple, la recherche se fait sur tous les documents de la collection eleves qui contient la propriété **age**. Si il faut, au contraire, trouver tous les documents sans propriété **age**, **$exists** doit être à **0**.

Vous pouvez aussi faire une recherche sur le type des propriétés, avec l'opérateur **$type**. Il prend pour valeur, soit une chaine, soit un entier, répertoriés dans le tableau suivant :

Type	Valeur
Double	1
String	2
Object	3
Array	4
Binaire	5
Objet id	7
Booleen	8
Date	9
Null	10
Timestap	17

Par exemple :

```
db.eleves.find({age: {$type:1}}) //retourne tous les élèves dont l'age est un double

db.eleves.find({nom: {$type:"string"}}) //retourne tous les élèves dont le nom est un string
```

Exercice

Sur la collection eleves :

- Ajouter un éléve Ronald Weasley en classe de seconde,
- Ajouter un élève Neville Londubat en classe de première et agé de 17 ans,
- Ajouter Hermione en classe de première,
- Modifier l'age de Neville, il a 18 ans,
- Trouver tous les éléves qui ont 16 ans ou plus,
- Trouver tous les élèves en première.

Document officiel

https://docs.mongodb.com/manual/