Le shell Mongodb, premières requêtes

Corrigé de TD pour un cours dispensé à l'université de Rennes 2

Romain Tavenard

1 Avant-propos

Lors de ce TD, vous allez utiliser le *shell* MongoDB sur votre poste. Ce *shell* est une interface d'interrogation de bases de données MongoDB. Dans le cadre de ce TD, le serveur MongoDB sera lui aussi lancé depuis votre poste.

1. Pour cela, ouvrez une fenêtre de terminal et exécutez la commande suivante, en choisissant pour `port_num` un entier entre 1001 et 9999 :

```
mongod --port port_num --dbpath /export/db/
```

Cette fenêtre devra rester ouverte pendant toute la durée du TD pour pouvoir utiliser le *shell* MongoDB correctement. Vous pouvez remarquer parmi les informations affichées par `mongod` le port utilisé ainsi que le nom de l'hôte (votre machine) sur le réseau (qui sera notée `host_name` dans la suite de cet énoncé).

2. Vous allez maintenant démarrer un *shell* MongoDB se connectant au serveur de bases de données accessible *via* le port `port_num` de l'hôte `host_name` (dans la suite vous pourrez utiliser `localhost` comme valeur pour `host_name`) :

```
mongo host_name:port_num
```

Par défaut, cette commande charge dans la variable `db` la base de données `test`, vous pouvez le vérifier à l'aide de la commande :

```
> db
test
```

Si vous souhaitiez vous connecter à une autre base de données, appelée par exemple `myDB`, la commande à utiliser pour lancer le *shell* serait :

```
mongo host_name:port_num/myDB
```

Un autre moyen de charger la base `myDB` dans la variable `db` est d'utiliser la commande `use` du *shell* :

```
> use myDB
switched to db myDB
```

Pour connaître la liste des collections contenues dans une base de données, on utilise la commande :

```
> db.getCollectionInfos()
[...]
```

Enfin, pour connaître le nombre de documents dans une collection, la syntaxe à utiliser est :

```
> db.nomDeLaCollection.count()
[...]
```

Remarquez que la base de données initiale ne contient aucune collection (liste vide retournée par `getCollectionInfos`). Pour pallier à cela, importez les bases fournies sur CURSUS depuis un nouveau terminal :

```
mongorestore --host host_name:port_num -d test /chemin/vers/test/
mongorestore --host host_name:port_num -d test_singlecollec /chemin/vers/test_singlecollec/
mongorestore --host host_name:port_num -d etudiants /chemin/vers/etudiants/
```

3. Combien de collections contiennent respectivement les bases `test` et `test_singlecollec` ? Pour le savoir il faudra d'abord changer de base de données active pour l'une des bases `test` ou `test_singlecollec`.

```
> use test
> db.getCollectionInfos()
> use test_singlecollec
> db.getCollectionInfos()
```

2 Premières requêtes

Il existe en MongoDB deux types de requêtes simples, retournant respectivement toutes les occurrences d'une collection ou la première :

```
> db.nomDeLaCollection.find()
> db.nomDeLaCollection.findOne()
```

De plus, lorsqu'on utilise la méthode `find()`, on peut rendre le résultat plus lisible à l'aide de `pretty()` :

```
> db.nomDeLaCollection.find().pretty()
```

4. Chargez la base de données `test` et affichez le contenu de chacune de ses collections.

```
> use test
> db.blog.authors.find()
> db.blog.posts.find()
```

Nous allons maintenant changer de base de données de travail. Pour cela, téléchargez le fichier `NYfood.json` sur CURSUS et importez son contenu dans une base appelée `food` à l'aide de la commande suivante (entrée dans un shell quelconque, pas dans MongoDB) :

```
mongoimport --db food --file NYfood.json --jsonArray --host host_name:port_num
```

Si l'on souhaite fixer des contraintes sur les documents à retourner, il suffit de passer en argument d'une de ces fonctions un document masque contenant les valeurs souhaitées. Par exemple, la requête suivante retourne tous les documents ayant un champ `"x"` dont la valeur est `"y"` :

```
> db.nomDeLaCollection.find({"x": "y"})
```

5. Utilisez cette syntaxe pour n'afficher que les documents de la collection `NYfood` correspondant à des boulangeries (pour lesquels le champ `"cuisine"` vaut `"Bakery"`). Comptez le nombre de résultats (la méthode `count()` accepte un document masque elle aussi).

```
> db.NYfood.find({"cuisine": "Bakery"})
```

6. Affichez maintenant la liste des boulangeries du Bronx. Combien y a-t-il de restaurants chinois (`"Chinese"`) à Brooklyn ?

```
> db.NYfood.find({"cuisine": "Bakery", "borough": "Bronx"})
> db.NYfood.find({"cuisine": "Chinese", "borough": "Brooklyn"}).count()
```

Les résultats que vous avez obtenus jusqu'à présent sont assez indigestes, notamment parce que toutes les clés sont retournées pour tous les documents. Il est possible de limiter cela en spécifiant les clés à retourner comme second argument de `find()`, ici la clé `"z"`:

```
> db.nomDeLaCollection.find({"x": "y"}, {"z": true})
```

7. Utilisez cette astuce pour n'afficher que les noms des boulangeries du Bronx.

```
> db.NYfood.find({"cuisine": "Bakery", "borough": "Bronx"}, {"name": true})
```

8. Affichez les nom et notes (mais pas les dates de naissance) des étudiants prénommés Marc dans la base `etudiants`.

```
> db.notes.find({"nom": "Marc"}, {"nom": true, "notes": true})
```

3 Opérateurs de comparaison

Pour effectuer des requêtes plus complexes, impliquant des opérateurs de comparaison, on utilisera la syntaxe suivante :

```
> db.nomDeLaCollection.find({"x": {operateur: valeur}})
```

Il est à noter que le sous-document contenant l'opérateur peut en fait contenir plusieurs opérateurs et ainsi ne seront retournés que les documents vérifiant toutes les conditions.

Les opérateurs les plus classiques sont les suivants :

Opérateur logique	Mot-clé MongoDB
$<$	`$lt`
$>$	`$gt`
$<=$	`$lte`
$>=$	`$gte`
"est dans la liste"	`$in`
"n'est pas dans la liste"	`$nin`

Enfin, le mot-clé `$exists` permet de ne retourner que les documents dans lesquels la clé spécifiée existe.

9. Affichez la liste des étudiants ayant au moins une note au-dessus de 10.

```
> db.notes.find({"notes": {$gte: 10}})
```

10. Que retourne la requête suivante ? Vérifiez votre intuition dans le *shell* MongoDB.

```
> db.notes.find({"notes": {$gt: 12, $lte: 10}})
```

11. Affichez le nom de tous les restaurants chinois du Bronx et de Brooklyn dans une même liste. Vous pourrez utiliser la méthode `count()` pour vérifier que la liste obtenue est bien la fusion des deux listes issues de chaque quartier.

```
> db.NYfood.find({$or: [{"cuisine": "Chinese", "borough": "Brooklyn"},
                        {"cuisine": "Chinese", "borough": "Bronx"}]})
```

12. Affichez le nom de toutes les boulangeries du Bronx commençant par la lettre `"P"`.

```
> db.NYfood.find({"cuisine": "Bakery", "borough": "Bronx",
                  "name": {$gte: "P", $lt: "Q"}})
```

Lorsque l'on souhaite effectuer un test sur une date, on utilisera toujours des opérateurs de comparaison. Toutefois, pour comparer les dates d'une base à des dates de référence, il faudra :

— que les dates de la base soient bien codées au format date (ISODate en MongoDB) ;
— que les dates de référence soient des variables au format date, déclarées à l'aide d'une commande du type :

```
> d1 = new Date("AAAA-MM-JJ")
```

13. Dans la base de données `etudiants`, affichez tous les étudiants nés après le 1er Janvier 1995.

```
> date_avant = new Date("1995-01-01")
> db.notes.find({"ddn": {$gte: date_avant}})
```

14. Même question en ne sélectionnant que les étudiantes.

```
> date_avant = new Date("1995-01-01")
> db.notes.find({"ddn": {$gte: date_avant}, "sexe": "F"})
```

15. Dans la base de données `test`, affichez la liste des posts en date du 26 Août 2015.

```
> date_avant = new Date("2015-08-26")
> date_apres = new Date("2015-08-27")
> db.blog.posts.find({"date": {$gte: date_avant, $lt: date_apres}})
```

Enfin, il est possible de combiner plusieurs conditions à l'aide de l'opérateur `$or` :

```
> db.nomDeLaCollection.find({$or: [{"x": 1}, {"y": "zzz"}]})
```

La commande précédente retournera ainsi l'ensemble des documents vérifiant au moins une des deux conditions suivantes :

- `"x" = 1`
- `"y" = "zzz"`

De la même façon, l'opérateur `$nor` permet de ne retourner que les documents ne vérifiant aucune des conditions spécifiées.

16. Affichez la liste des étudiants qui vérifient l'une des deux conditions suivantes :

- de sexe féminin ;
- dont le prénom commence par la lettre `"M"`

```
> db.notes.find({$or: [{"sexe": "F"},
                       {"nom": {$gte: "M", $lt: "N"}}]})
```

Lorsque l'on travaille sur des listes, il peut être utile de tester leur longueur, ce qui se fait avec le mot-clé `$size` :

```
> db.nomDeLaCollection.find({"cleDeLaListe": {$size: 12}})
```

ne retournera ainsi que les documents pour lesquels la valeur associée à la clé `cleDeLaListe` est une liste de taille 12.

17. En utilisant la base `test`, affichez la liste des posts n'ayant pas reçu de commentaire.

```
> db.blog.posts.find({$or: [{"comments": {$exists: false}},
                            {"comments": {$size: 0}}])
```

18. Affichez la liste des étudiants ayant obtenu exactement deux notes.

```
> db.notes.find({"notes": {$size: 2}})
```

19. Affichez la liste des étudiants ayant obtenu une ou deux notes.

```
> db.notes.find({$or: [{"notes": {$size: 1}},
                       {"notes": {$size: 2}}]})
```

20. Affichez la liste des étudiants ayant obtenu au moins deux notes.

```
> db.notes.find({$nor: [{"notes": {$size: 1}},
                        {"notes": {$size: 0}},
                        {"notes": {$exists: false}}]})
```