TP3 - À la découverte de MongoDb

1 Étape 1 : installation de MongoDb

La première étape consiste à installer MongoDb. Cela pourra être fait au choix sur une des machines virtuelles créées précédemment ou sur votre machine physique. Pour cela, rendez-vous sur le site de MongoDb . Nous ne détaillons pas les étapes d'installation de MongoDb car elles sont parfaitement décrites dans la documentation et dépendent de votre configuration matérielle.

Pour vérifier que MongoDb est correctement installé, ouvrez un terminal et tapez `mongo`. Si vous obtenez quelque chose de similaire à la capture d'écran présentée dans la Figure 1 vous pouvez continuer. Sinon, à vous de voir ce qui ne va pas avant d'appeler un de vos professeurs.

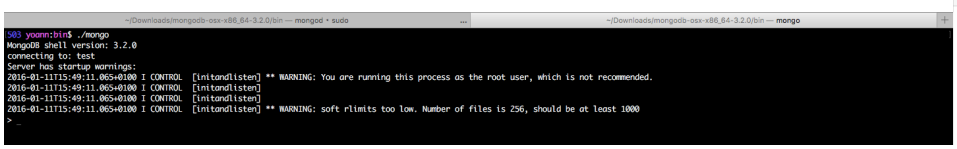

Figure 1 – Ce que vous devez voir après une installation correcte de MongoDb

2 Étape 2 : Création et peuplement d'une base de données

Une fois MongoDb installé, aucune base (et donc aucune donnée) n'existe. C'est ce à quoi nous allons nous atteler maintenant. Il y a deux manières de faire cela :

1. Création de la base et insertion manuelle des enregistrements

2. Création de la base et utilisation d'un fichier de données au format Json pour importer des données

Schématiquement, un serveur MongoDb peut abriter plusieurs bases de données qui elles-mêmes peuvent contenir plusieurs collections. Pour créer une base de données, il suffit d'utiliser la commande `use <nomBd>`. L'exécution de cette commande crée si besoin la base et la sélectionne comme base courante.

Nous allons maintenant voir les deux manières de procéder pour peupler une collection.

2.1 Création et peuplement manuel

La création d'une collection est en quelque sorte automatique dès lors que l'on insère un enregistrement dans une collection qui n'existe pas encore. La commande permettant d'insérer un enregistrement est :

```
db.<nomBd>.insert({<enregistrement Json>})
```

Consigne. Insérez les enregistrements suivants dans la base `sid1` :

```
{
    first: 'matthew',
    last: 'setter',
    dob: '21/04/1978',
    gender: 'm',
    hair_colour: 'brown',
    occupation: 'developer',
    nationality: 'australian'
}
{
    first: 'james',
    last: 'caan',
    dob: '26/03/1940',
    gender: 'm',
    hair_colour: 'brown',
    occupation: 'actor',
    nationality: 'american'
}
{
    first: 'arnold',
    last: 'schwarzenegger',
```

```
    dob: '03/06/1925',
    gender: 'm',
    hair_colour: 'brown',
    occupation: 'actor',
    nationality: 'american'
}
{
    first: 'tony',
    last: 'curtis',
    dob: '21/04/1978',
    gender: 'm',
    hair_colour: 'brown',
    occupation: 'developer',
    nationality: 'american'
}
{
    first: 'jamie lee',
    last: 'curtis',
    dob: '22/11/1958',
    gender: 'f',
    hair_colour: 'brown',
    occupation: 'actor',
    nationality: 'american'
}
{
    first: 'michael',
    last: 'caine',
    dob: '14/03/1933',
    gender: 'm',
    hair_colour: 'brown',
    occupation: 'actor',
    nationality: 'english'
}
{
    first: 'judi',
    last: 'dench',
    dob: '09/12/1934',
    gender: 'f',
    hair_colour: 'white',
    occupation: 'actress',
```

```
    nationality: 'english'
}
```

Pour vérifier que l'insertion s'est bien passée, exécutez la commande `db.sid1.find()`. Cette commande renvoie tout le contenu de la base `sid1`.

2.2 Création et peuplement automatique

Nous allons utiliser un jeu de données classique pour aborder cette partie. Celui-ci décrit quelques restaurants ainsi que des notes soumises par les clients. Voici un extrait du jeu de données :

```
{
  "address": {
     "building": "1007",
     "coord": [ -73.856077, 40.848447 ],
     "street": "Morris Park Ave",
     "zipcode": "10462"
  },
  "borough": "Bronx",
  "cuisine": "Bakery",
  "grades": [
     { "date": { "$date": 1393804800000 }, "grade": "A",
        "score": 2 },
     { "date": { "$date": 1378857600000 }, "grade": "A",
        "score": 6 },
     { "date": { "$date": 1358985600000 }, "grade": "A",
        "score": 10 },
     { "date": { "$date": 1322006400000 }, "grade": "A",
        "score": 9 },
     { "date": { "$date": 1299715200000 }, "grade": "B",
        "score": 14 }
  ],
  "name": "Morris Park Bake Shop",
  "restaurant_id": "30075445"
}
```

Effectuez les opérations suivantes :

1. Téléchargez les données à l'adresse

2. Ouvrez un terminal et exécutez la commande :

```
mongoimport --db sid --collection restaurants --drop --file data.json
```

3 Étape 3 : interrogation d'une collection

Dans la section précédente, nous avons vu que la fonction `find` sans argument retourne l'ensemble des enregistrements d'une collection. Nous allons maintenant voir comment ajouter des conditions pour filtrer les résultats retournés par la fonction `find`.

Le format du paramètre à spécifier pour retourner un résultat qui matche les conditions spécifiées est le suivant :

```
{ <field1>: <value1>, <field2>: <value2>, ... }
```

Si le champ `field` n'est pas dans un document imbriqué, les guillemets (simples ou doubles) ne sont pas obligatoires. Par contre, si `field` est dans un objet imbriqué ou dans un tableau, il faudra passer par la notation objet (le point). À ce moment-là, les guillemets sont obligatoires.

3.1 Filtrer via un champ non imbriqué

Consigne. Retournez les documents tels que le champ `borough` vaille `Manhattan`.

3.2 Filtrer via un champ imbriqué

Consigne. Retournez les documents tels que le champ `zipcode` dans le sous-document `address` vaille `10075`.

3.3 Filtrer via un champ dans un tableau

Consigne. Retournez les documents tels que le champ `grade` dans le tableau `grades` vaille `B`.

3.4 Opérateurs

MongoDb fournit des opérateurs pour spécifier les conditions, tels que des opérateurs de comparaison. Ces opérateurs sont généralement de la forme :

```
{ <field1>: { <operator1>: <value1> } }
```

3.4.1 L'opérateur > : $gt

Consigne. Retournez les documents qui contiennent un document-imbriqué avec un score > 30.

3.4.2 L'opérateur < : $lt

Consigne. Retournez les documents qui contiennent un document-imbriqué avec un score < 10.

3.5 Combiner les conditions

Il est possible de combiner les critères avec les classiques `ET` et `OU`.

3.5.1 Le ET logique

Il suffit de séparer les conditions par une virgule.

Consigne. Retournez les restaurants italiens dont le zip code est 10075.

3.5.2 Le OU logique

Le OU logique s'exprime avec l'opérateur `$or`. Par exemple, pour retourner les restaurants dont les spécialités sont italiennes OU dont le zip code est 10075, il faudra saisir la requête :

```
db.restaurants.find(
   { $or: [ { "cuisine": "Italian" }, { "address.zipcode":
       "10075" } ] }
)
```

4 Étape 4 : trier les enregistrements

Dans cette section, nous allons voir comment trier le résultat d'une requête. A cette fin, MongoDb introduit la commande/fonction `sort`. Nous allons étudier son fonctionnement. Tout comme la commande `find`, la commande `sort` accepte une liste d'options pour, notamment, définir sur quels champ trier les documents et dans quel ordre (1 pour ascendant et -1 pour descendant).

Voici un exemple qui retrouve tous les acteurs masculins, anglais ou américains, et qui les ordonne selon leur nationalité, dans l'ordre alphabétique inverse.

```
db.sid1.find({gender: 'm', $or: [{nationality: 'english'},
    {nationality: 'american'}]}).sort({nationality: -1});
```

Pour trier sur deux champ, il faut procéder comme pour la conjonction de conditions dans la commande `find`.
Consigne. Modifiez la requête précédente pour trier d'abord sur la nationalité (descendant) puis selon leur prénom (ascendant).

5 Étape 5 : limiter les enregistrements retournés

Si nous voulons limiter le nombre d'enregistrements retournés, MongoDb introduit la commande `limit` dont voici un exemple :

```
db.sid1.find({gender: 'm', $or: [{nationality: 'english'},
    {nationality: 'american'}]}).limit(2);
```

Si, en plus, nous souhaitons ignorer les deux premiers enregistrements et ne retourner que les troisième et quatrième, il faudra alors utiliser la commande `skip` dont voici un exemple :

```
db.sid1.find({gender: 'm', $or: [{nationality: 'english'},
    {nationality: 'american'}]}).limit(2).skip(2);
```

6 Étape 6 : mise à jour d'enregistrements

MongoDb fournit bien évidemment une commande pour mettre à jour des enregistrements, la commande `update`. En voici la syntaxe :

```
db.<collection>.update({<conditions>},{$set:{<field>:<value>}});
```

où `<conditions>` sont des conditions spécifiées comme pour la commande `find`.
Consigne. Mettez à jour la couleur de cheveux de l'acteur *James Caan* pour la faire passer de *marron* à *grise*.

7 Étape 7 : suppression d'enregistrements

MongoDb a évidemment une commande pour supprimer des enregistrements, la commande `remove`. Ici aussi, il faudra spécifier des conditions sous peine de supprimer tous les enregistrements. Voici la syntaxe associée à cette commande :

```
db.<collection>.remove({<conditions>});
```

où `<conditions>` sont des conditions spécifiées comme pour la commande `find`.

Consigne. Supprimez l'acteur *James Caan* de la collection et vérifiez que la suppression a été effective.