Jordan DANSET

5A 2I

TP – mongoDB

Mise en place d'une base de données MongoDB :

MongoDB est une base de données NoSQL.

Pour fonctionner sur un OS windows, on doit procéder comme suit :

-installation du serveur

-création d'un répertoire C:\data\db pour stocker les données

-lancement du serveur avec le .exe mongod disponuble sur $MONGO/bin

Interface utilisateur :

L'interface utilisateur utilisée est Robo3T. Une nouvelle connexion à MongoDB est établie avec « localhost » et le port « 27017 ». Une nouvelle base de données est importée sous le nom « New York », elle contient un document au format json : restaurants.json.

Interrogation des données :

Dans la base de données chaque restaurant a un nom, un quartier (*borough*), le type de cuisine, une adresse (document imbriqué, avec coordonnées GPS, rue et code postale), et un ensemble de notes (résultats des inspections).

Filtrage et projection

La commande de base pour une requête de filtrage est la suivante :

```
db.restaurants.find( { "borough" : "Manhatan" } )
```

Elle permet de renvoyer les restaurants dans le quartier de Manhatan.

On va par la suite combiner deux « clés/valeurs » afin de chercher parmi les restaurants de Manhatan ceux qui font de la cuisine Italienne :

```
db.restaurants.find(
    { "borough" : " Manhatan ",
      "cuisine" : "Italian" }
)
```

Dans le but de rechercher un mot en particulier on va rajouter la notation /lemot/i de manière à regrouper tous les mots correspondant sans prendre en compte la présence d'une minuscule ou d'une majuscule. On parle d'insensibilité à la casse :

```
db.restaurants.find(
```

```
{ "borough" : " Manhatan ",
  "cuisine" : "Italian",
  "address.street" : "21 jump Street",
  "name" : /burger/i }
)
```

Ce filtre permet donc de renvoyer les restaurants de Manhatan dans la rue définie donc le nom contient « burger ».

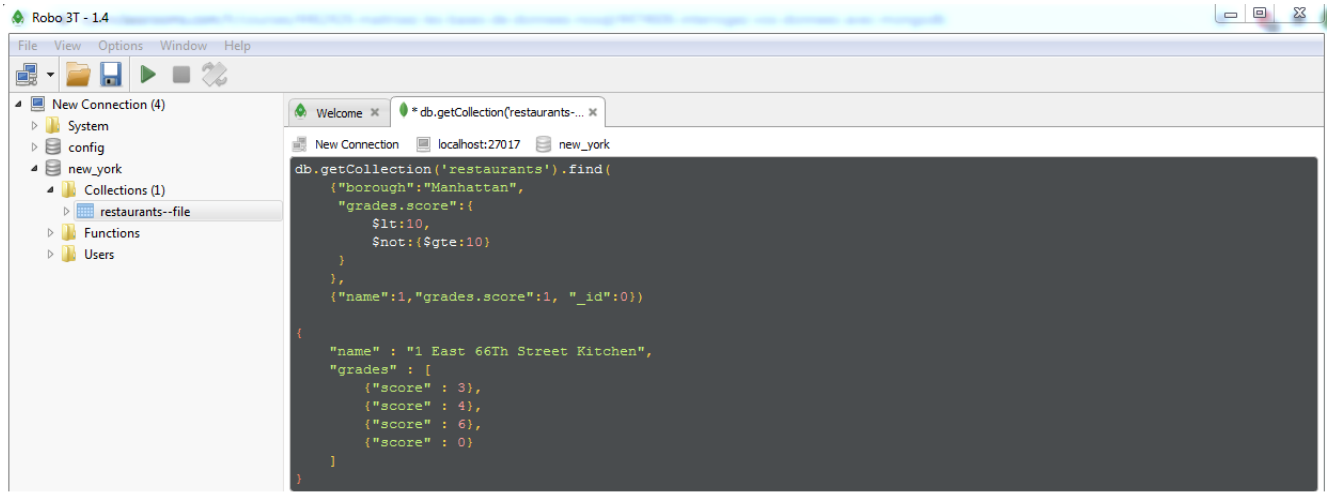
Pour afficher les résultats retournés par le filtrage d'une certaine manière on utilise le principe de Projection.

Pour projeter une valeur spécifique on va compléter le deuxième paramètre de la fonction « find » de cette manière {"name":1, "_id":0 }. Dans cet exemple, on va alors projeter la valeur selon le « nom » sans projeter « _id » du document, qui est normalement projeté par défaut.

Dans le but d'affiner davantage le filtrage, on peut utiliser des opérateurs arithmétiques sur les valeurs numériques des clés. Les différents opérateurs disponibles sont les suivants :

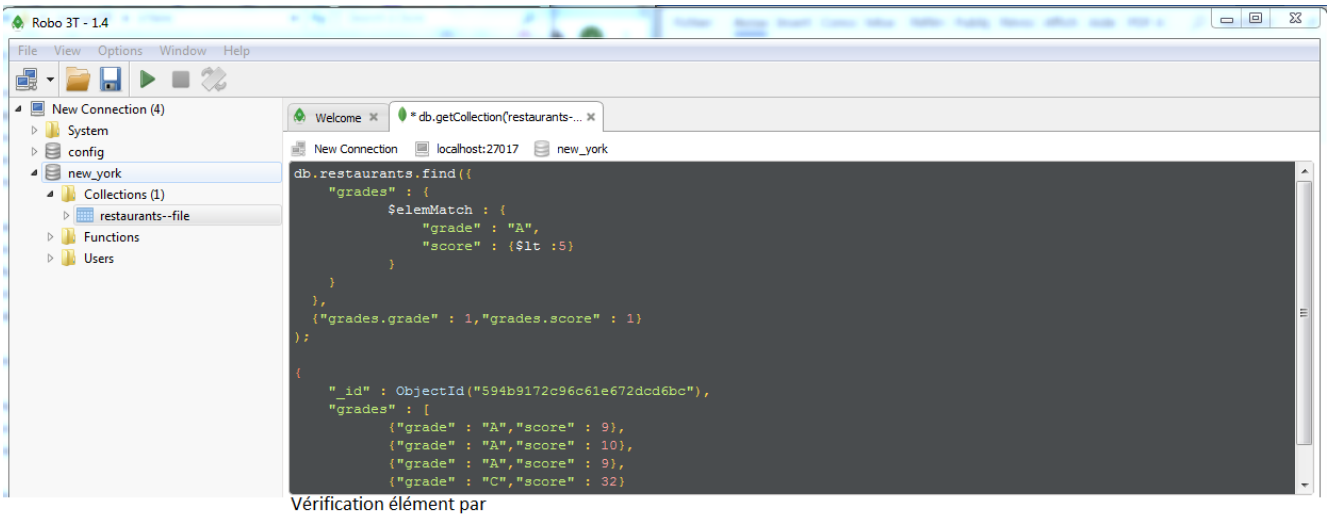
`$eq`	Matches values that are equal to a specified value.
`$gt`	Matches values that are greater than a specified value.
`$gte`	Matches values that are greater than or equal to a specified value.
`$lt`	Matches values that are less than a specified value.
`$lte`	Matches values that are less than or equal to a specified value.
`$ne`	Matches all values that are not equal to a specified value.
`$in`	Matches any of the values specified in an array.
`$nin`	Matches none of the values specified in an array.

Voici un exemple d'utilisation :



Nom et score des restaurants de Manhattan
ayant un score inférieur à 10 (lt --> less than)

Attention cependant à l'utilisation de ces opérateurs. En effet, si l'on souhaite combiner plusieurs conditions comme par exemple un score inférieur à 5 avec un grade A on doit utiliser la notation $elemMatch. Cette notation permet de faire en sorte que toutes les conditions soient vérifiées élément par élément :



Vérification élément par
élement avec $elemMatch

Aggrégation

La fonction « aggregate » est utilisée dans le cadre de manipulations plus complexes. En effet, celle-ci va pouvoir prendre en paramètre plusieurs opérateurs. Les principaux opérateurs sont les suivants :

- **{$match : {} }**
- **{$project : {} }**
- **{$sort : {} }**

Pour bien comprendre la différence, prenons tout d'abord une requête « find » qui permet de renvoyer le nom des restaurants dont la dernière inspection (la plus récente) a donné un grade B :

```
db.restaurants.find({
    "grades.0.grade":"B"
  },
  {"name":1, "_id":0}
);
```

La requête équivalente avec une aggrégation est la suivante :

```
db.restaurants.aggregate( [
    { $match : {
        "grades.0.grade":"B"
    }},
    { $project : {
        "name":1, "_id":0
    }}
] )
```

On voit que la structure dans ce cas est différente, en effet les opérateurs sont contenus dans un document et la valeur est elle aussi dans un autre document.