Index

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

Romain Tavenard

Pour ce TD, vous allez travailler sur une base nommée `large_db` et sa collection `users`.

1. Combien de documents contient cette collection ? Quels sont les attributs des documents de la collection ?

```
> db.users.count()
500000
> db.users.findOne()
{
    "_id" : ObjectId("6022496fbdccf915b948059b"),
    "i" : 3.0,
    "name" : "user3",
    "age" : 50.0,
    "created" : ISODate("2021-02-09T08:35:59.569Z")
}
```

1 Sans index

MongoDB permet d'obtenir un certain nombre d'informations sur le déroulement d'une requête à l'aide de la fonction `explain()` :

```
> db.nomDeLaCollection.find({...}).explain("executionStats")
```

2. Utilisez cette méthode pour obtenir (i) le temps d'exécution, (ii) le nombre de documents parcourus et (iii) le nombre de documents retournés par une simple requête demandant d'afficher les informations des utilisateurs dont la clé `"name"` vaut `"user101"`.

```
> db.users.find({name: "user101"}).explain("executionStats")
```

3. Même chose en ne demandant de retourner qu'un utilisateur. Que remarquez-vous ?

```
> db.users.find({name: "user101"}).limit(1).explain("executionStats")
```

Le nombre de documents parcourus a chuté : une fois un premier document trouvé, la requête s'arrête.

4. Répétez encore cette opération en cherchant maintenant l'utilisateur `"user499999"`. Que remarquez-vous ?

Dans ce cas, il faut parcourir tous les documents (ou presque) avant de finir par trouver le bon.

2 Avec un index

5. Passez maintenant à la base `large_db_with_index` ayant le même contenu que la base précédente, mais disposant d'index. Affichez les index existant sur la collection `users` de cette base : sur quel(s) attribut(s) portent-ils ?

```
> db.users.getIndexes()
[
    {
        "v" : 2,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_"
    },
    {
        "v" : 2,
        "key" : {
            "name" : 1.0
        },
        "name" : "name_1"
    },
    {
        "v" : 2,
        "key" : {
            "age" : 1.0,
            "name" : 1.0
        },
        "name" : "age_1_name_1"
    }
]
```

Il en existe 3. Ils portent respectivement sur les attributs `_id`, `name` et sur la paire d'attributs (`age`, `name`).

6. Répétez les opérations précédentes (questions 2 à 4). Que remarquez-vous ? Quel(s) index ont été utilisés par ces requêtes ?

Le nombre de documents parcourus pour effectuer la requête est maintenant égal à 1 pour chacune des trois requêtes : il a suffi de chercher les valeurs de `"name"` dans l'index correspondant (`name_1`) pour résoudre la requête.

3 Index composés

7. Effectuez les requêtes suivantes et examinez leur exécution (nombre de documents parcourus, index utilisés) :

— Afficher les utilisateurs âgés de 20 ans ;
— Afficher les utilisateurs âgés de 20 ans par ordre croissant de nom d'utilisateur ;
— Afficher les utilisateurs âgés de 20 ans par ordre croissant de date de création ;
— Afficher les utilisateurs âgés de 20 ans dont le nom d'utilisateur est compris entre `"user100000"` et `"user500000"`.

```
> db.users.find({"age": 20}).explain("executionStats")
> db.users.find({"age": 20}).sort({"name":1}).explain("executionStats")
> db.users.find({"age": 20}).sort({"created":1}).explain(
    "executionStats")
> db.users.find({"age": 20, "name":
    {$gt: "user100000", $lt: "user500000"}}).explain("executionStats")
```

Elles utilisent toutes l'index `age_1_name_1` car son préfixe `"age"` permet d'avoir accès aux documents par âge. Pour la requête triant sur la date de création, le temps d'exécution reste un peu plus important que pour les autres car il faut trier un nombre important de résultats.

4 Exercice de synthèse

Liens utiles :

— http://docs.mongodb.org/master/core/2dsphere/
— http://docs.mongodb.org/master/core/index-text/
— http://docs.mongodb.org/master/core/index-single/

10. Passez à la base `food`. Quels sont les index définis sur la collection `NYfood` et de quel type sont-ils ? Quelles commandes a-t-il fallu exécuter pour les créer ?

```
> db.NYfood.getIndexes()
[
    {
        "v" : 2,
        "key" : {
            "_id" : 1
```

```
        },
        "name" : "_id_"
    },
    {
        "v" : 2,
        "key" : {
            "address.loc" : "2dsphere"
        },
        "name" : "address.loc_2dsphere",
        "2dsphereIndexVersion" : 3
    },
    {
        "v" : 2,
        "key" : {
            "_fts" : "text",
            "_ftsx" : 1
        },
        "name" : "$**_text",
        "weights" : {
            "$**" : 1
        },
        "default_language" : "english",
        "language_override" : "language",
        "textIndexVersion" : 3
    },
    {
        "v" : 2,
        "key" : {
            "borough" : 1.0
        },
        "name" : "borough_1"
    },
    {
        "v" : 2,
        "key" : {
            "cuisine" : 1.0
        },
        "name" : "cuisine_1"
    }
]
```


Il s'agit (en plus de l'index par défaut sur l'attribut `_id`) de :

- `address.loc_2dsphere` : un index géospatial sur la localisation des restaurants ;
- `$**_text` : un index textuel couvrant tous les champs textuels de la collection ;

— `borough_1` : un index classique sur le quartier ;
— `cuisine_1` : un autre sur le type de cuisine.

```
> db.NYfood.createIndex({"address.loc" : "2dsphere"})
> db.NYfood.createIndex({"$**" : "text"})
> db.NYfood.createIndex({"borough" :1})
> db.NYfood.createIndex({"cuisine" :1})
```

11. Vérifiez que ces index sont effectivement utilisés à l'aide de quelques requêtes simples (éventuellement issues des TD précédents).

```
> db.NYfood.find({"borough": "Bronx"}).explain("executionStats")
```