ENSAM
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

TD/TP 5 :

INDEXATION ET PERFORMANCE

Le but de ce TP5 est la création d'index pour ensuite évaluer les performances et l'efficacité

Important : une préparation préalable (opération 1.1) de la base fictive prend un temps considérable ~ 40min

Les bases NoSQL, comme MongoDB, ont été créés pour répondre à des problématiques de requêtages sur une grosse masse de données manière beaucoup plus efficaces. Nous avons déjà vu dans le précédent TD/TP3 comment rechercher, insérer, modifier ou supprimer des documents. Nous allons voir maintenant comment faire cela rapidement avec des index

.

1. CREATION D'UNE BASE DE TEST (FICTIVE)

1.1. Créez une base **db_test** qui contient une collection personnes dans laquelle vous allez créer des 7000000 de documents à l'aide de la commande suivante :

```
for(i=1;i<=7000000;i++) db.personnes.save({nom:"nom "+i,pernom:"prenom_"+i, age : i, compteur : i})
```

Cette opération peut prendre un certain temps considérable.

1.2. Vérifiez qu'effectivement il y'a 7000000 de documents insérés :

```
db.personnes.count()
```

2. CREATION D'UN INDEXE

2.1. Afin de déterminer les indexes existants, on peut faire appel à la fonction **getIndexes()** qui va lister les index présents :

```
db.personnes.getIndexes()
```

Remarquez ici, on voit que la collection personnes contient un index sur le champ **_id.** En fait, c'est un index créé par défaut par le système sur l'identifiant des collections afin d'accélérer les recherches.

2.2. Pour créer un index spécifique, on utilise la fonction **createIndex().** Cette fonction prend comme argument un document JSON contenant un ou plusieurs champs avec pour chacun d'eux le sens du tri (1: croissant, -1: décroissant). Créez un index sur le champ **compteur.**

```
db.personnes.createIndex({"compteur" : 1 })
```

2.3. La création n'est pas rapide car la collection contient 7000000 documents. Vérifiez ensuite que l'index est bien créé.

```
db.personnes.getIndexes()
```

3. EVALUATION DE LA PERFORMANCE

3.1. Lançons maintenant une requête sur la collection avec un filtre sur le champ non indexé **age.** Par exemple, chercher les documents qui corresponde à l'âge 10 :

```
db.personnes.find({"age" : 10 })
```

Vous avez surement remarqué le temps que prend cette requête, aux environs de 10 secondes.

3.2. Cette requête filtre les documents de la collection puis comptabilise le nombre total de documents. Les fonctionnalités de MongoDB nous permettent d'évaluer la performance de cette requête avec la fonction **explain().**

Sans argument (ou avec argument **explain("queryPlanner**")), la fonction renvoie le plan de requête qui va être exécuté.

Avec **true** comme argument (ou sous la forme suivante : **explain("executionStats")**), la fonction renvoie également des informations sur l'exécution de la requête.

```
db.personnes.explain().find({age:10})
db.personnes.explain(true).find({age:10})
db.personnes.explain("queryPlanner").find({age:10})
db.personnes.explain("executionStats").find({age:10})
```

A noter que la fonction **explain()** se place juste après le nom de la collection et avant les fonctions de **find** et **count**.

3.3. Dans cette première partie (**`explain("queryPlanner")`**), **le winning** plan nous indique que la requête est composée d'une seule étape (stage). Cette étape fait un scan de la collection pour filtrer les documents satisfaisant le prédicat. Vous pouvez utiliser une requête multi-étape pour voir de quoi est composé le plan de requête, par exemple :

```
db.personnes.explain("queryPlanner").find({age:10}).count()
```

3.4. Dans les informations issues des statistiques d'exécution de cette requête, repérez les 2 informations importantes : **`"executionTimeMillis"`** et **`"totalDocsExamined"`** et reportez leurs valeurs :

3.5. Nous allons maintenant lancer une requête sur le champ indexé "**compteur**" puis lancer de nouveau un **explain**() pour voir les différences avec la requête sans indexe au niveau du temps d'exécution et du nombre de documents consultés durant cette requête :

```
db.personnes.explain("executionStats").find({compteur:10})
```

Cette fois-ci, la requête n'a plus besoin de scanner toute la collection : seulement un seul document a été examiné **("totalKeysExamined" : 1,**). La recherche a été fortement accélérée puisque la durée d'exécution de la requête est estimée à 0 ms **("executionTimeMillis" : 0).**

4. RECHERCHE DE TYPE TEXT

Il est possible de créer plusieurs index sur des **champs distincts** mais aussi des **index multiples** sur **plusieurs champs**. La force de MongoDB est aussi de pouvoir proposer des **index de type spécifique**.

Avec des index de **type text**, on peut ainsi faire de la ***recherche optimisée pour le full texte*** dans un ou plusieurs champs**.**

4.1. Commencez par créer un index de type text sur le champ « prenom » de votre collection. La création de l'index va prendre quelques dizaines de secondes car il y a pas mal de matière à indexer.

```
db.personnes.createIndex({"prenom" : "text"})
```

Vous allez surement remarquer le temps que prend cette requête, environ 30 secondes, puisqu'il s'agit d'une indexation basée sur un autre principe d'optimisation orienté recherche textuelle.