MongoDB, index et agrégation

Avec MongoDB, comme avec les SGBD, il est possible d'aller plus loin que le simple CRUD, en créant des index ou en utilisant les agrégations.

Rappels

Avec MongoDB, il faut déclarer la base à utiliser et si elle n'existe pas, le moteur la créée :

```
Use ecole
```

L'ajout de collections est implicite lors de l'ajout d'un document :

```
db.eleves.insert({nom : "Granger", prenom : "Hermione"})
db.eleves.insert({nom : "Londubat", prenom : "Neville", classe : "Seconde"})
db.eleves.insert({nom : "Weasley", prenom : "Ronald", classe : "Première"})
db.eleves.insert({nom : "Malfoy", prenom : "Drago", age : 16 })
```

Vous pouvez constater que tous les documents n'ont pas la même structure.

Les index

Comme avec les SGBD, les index améliorent le temps d'exécution des requêtes, surtout si elles sont appelées fréquemment. MongoDB dispose d'index au niveau des collections. Le champ _id est indexé par défaut et il est impossible de le supprimer.

Pour poser un index sur le champ nom dans l'ordre ascendant (**1**, ce sera **-1** pour DESC), utilisez la commande **createIndex()** :

```
db.eleves.createIndex({nom:1});
```

Vous pouvez poser un index sur plusieurs champs, comme nom ET prenom (ASC tous les deux) :

```
db.eleves.createIndex({nom:1, prenom:1});
```

Comme en SQL, si vos requêtes se basent sur prenom, l'index ne sera pas utilisé !
Si vous requêtez sur le nom ou sur le nom ET le prénom, celui-ci sera sollicité.

Il est aussi possible de créer des index uniques, c'est à dire, un index sans doublon :

```
db.eleves.createIndex({nom:1}, {unique: true});
```

Notez que dans cet exemple, choisir nom comme index unique n'est pas très pertinent, car il ne fonctionnera plus dès qu'il y aura un doublon dans les noms.

Vous pouvez à tout moment vérifier les index d'une collection :

```
db.eleves.getIndexes();
```

Ou en supprimer un :

```
db.eleves.dropIndex('nom_1_prenom_1');
```

L'agrégation

Introduit dans la version 2.1 de MongoDB, le framework d'agrégation est une excellente alternative pour les opérations que l'on pouvait faire avec map-reduce. Plus simple que ce dernier, l'agrégation permet

d'agréger des données pour un ressortir des totaux, moyennes, min/max, etc.…

Avant de commencer, ajoutons les documents suivants à notre collection d'élèves :

```
db.eleves.insert({nom : "Chang", prenom : "Cho", classe: "Première", age : 17})
db.eleves.insert({nom : "Weasley", prenom : "Ginny", classe: "Seconde", age : 16})
db.eleves.insert({nom : "Diggory", prenom : "Cedric", classe: "Première", age : 18})
```

Grâce aux agrégats, il est possible de regrouper les documents sur un champ voulu. Par exemple, nous pouvons connaitre les élèves par classe grâce à la méthode **aggregate()**:

```
db.eleves.aggregate({$group:
	{_id:"$classe", qui:
		{$push:
			{nom:"$nom",prenom:"$prenom"}
		}
	}
})
```

Le pipeline **$group** permet d'afficher de nouveaux documents à partir des documents de la collection regroupé par **_id**. Cet attribut, obligatoire, fait référence, grâce au préfixe **$**, à un attribut des documents initiaux. D'ailleurs, à chaque fois qu'il faut faire référence à ces attributs, il faut utiliser le **$**. Enfin, lorsqu'on veut d'autres informations, il faut faire un **push** dans un tableau.

L'agrégation permet certaine opération, comme compter les documents ayant le même attribut :

```
db.eleves.aggregate({$group:
	{_id : "$classe", nb :
		{$sum:1}
	}
})
```

Cette dernière commande permet de connaitre le nombre d'élèves par classe. D'autres opérations sont possibles grâce aux pipelines suivants :

- **$avg** : moyenne de la valeur donnée des documents de la collection,
- **$min** : valeur minimale des documents de la collection,
- **$max** : valeur maximale des documents de la collection,
- **$addToSet** : insère la valeur dans un tableau du document résultant, sans doublon,
- **$first** : premier document de la collection sur la base du regroupement,
- **$last** : dernier document de la collection sur la base du regroupement.

```
db.eleves.aggregate({$group:{_id : "$classe", nb : {$avg:"$age"}}})
```

Il est possible de sélectionner qu'une partie des données à grouper avec le pipeline **$match** :

```
db.eleves.aggregate(
	{$match:
		{nom:"Weasley"}
	},
	{$group:
		{_id:"$nom", nb:{$sum:1}}
	}
)
```

Il est possible d'ajouter des tris, grâce au pipeline **$sort** :

```
db.eleves.aggregate({$group:{_id:"$classe"}},{$sort:{_id:-1}})
```

Pour aller plus loin

Vous pouvez consulter la doc officielle :

Sur les index : https://www.mongodb.com/docs/manual/indexes/

Sur les agrégats : https://www.mongodb.com/docs/manual/aggregation/

Un petit blog en français :

Sur les index : https://practicalprogramming.fr/mongodb-index

Sur les agrégats : https://practicalprogramming.fr/aggregation-mongo