Les Documents

Dans chacune de nos collections, nous pourrons trouver plusieurs groupe de données, chacune de ces entrées sont nommées "Document". Les documents sont l'équivalent des "ROW" dans les BDD classiques.

Ajouter un Document

Une fois que l'on a indiqué quelle BDD nous souhaitons utiliser, on pourra utiliser cette commande :

```
db.nomCollection.insertOne(document)
```

Les documents sont au format JSON. Une fois la commande entrée, le gestionnaire de BDD vous retournera un objet. Celui-ci contiendra un Boolean indiquant si tous s'est bien passé et l'id automatiquement donné au document.

Nous avons aussi la possibilité d'ajouter plusieurs documents d'un seul coup :

```
db.nomCollection.insertMany([document1,document2...])
```

À la différence d'une BDD classique, rien n'oblige à ce que tous les documents ai la même structure.

Si un document possède certaines propriétés et qu'un autre de la même collection en a des différentes, cela ne posera aucun problème. Même si pour une question de logique, nous essayerons de garder la même structure.

Trouver un Document

Une fois que l'on a quelques documents dans notre collection, nous voudrions récupérer ces informations stockés.

Fonctions de recherche

Pour cela nous avons accès à deux fonctions :

```
# Récupérer tous les résultats :
db.collectionName.find();
# Récupérer le premier résultat :
db.collectionName.findOne();
```

Mais cela est peu précis. Si nous souhaitons faire une recherche plus précise, nous pouvons ajouter à ces fonctions un paramètre sous le forme d'un objet :

```

```

```
db.collectionName.find({propertyToSearch:ValueToSearch});
# par exemple :
db.users.find({email:"maurice@gmail.com"});
```

Comparateurs

Ceci est bon pour trouver une information précise. Mais parfois nous souhaitons trouver des informations plus grande ou plus petite qu'une valeur, pour cela nous auront besoin d'un de ces mots clefs :

- **$eq** égale à.
- **$lt** plus petit que (less than).
- **$lte** plus petit ou égale à.
- **$gt** plus grand que (greater than).
- **$gte** plus grand ou égale à.
- **$ne** non égale.
- **$in** dans le tableau.
- **$nin** Pas dans le tableau.

Ils s'utiliseront de la même façon :

```
db.collectionName.find({propertyName:{$comparateur:valeur}});
# Exemple :
db.users.find({age:{$gt:18}});
```

Les Opérateurs Logiques

On peut aussi combiner les comparaisons avec des opérateurs "AND", "OR"...

- **$and** Le document doit comporter toute les valeurs suivantes.
- **$or** Le document doit comporter l'une des valeurs suivantes.
- **$not** Le document ne doit pas comporter toute les valeurs suivantes.
- **$nor** Le document ne doit pas comporter l'une des valeurs suivantes.

Pour les utilisers nous utiliseront la syntaxe suivante :

```
db.collectionName.find({$operateur:[{property1:value},{property2:value}]});
# Exemple :
db.users.find({$and:[{age:{$gt:18}},{email:/gmail.com$/}]})
```

Comme vous pouvez le voir dans ce dernier exemple, si on souhaite rechercher une information incomplète, on peut aussi utiliser des REGEX.

Mettre à jour un document

Pour mettre à jour un document nous allons devoir utiliser la syntaxe suivante :

```
# Modifer un document.
db.nomCollection.updateOne(filter,update);
# Modifier plusieurs documents.
db.nomCollection.updateMany(filter,update);
# Remplacer un document.
db.nomCollection.replaceOne(filter,document);
```

- Les filtres sont les mêmes que ce que l'on utilise pour la méthode "*find*".
- update est un objet contenant un "**update operator**" qui contiendra lui même les propriétés à valeur à mettre à jour.

Voici un exemple :

```
db.users.updateOne({prenom:"Maurice"},{$set:{age:42}})
```

Dans cet exemple on cherche un utilisateur avec comme prénom "Maurice" et on va indiquer qu'il a la propriété "age" qui vaut 42. Si cette propriété existe, elle sera mis à jour. Sinon elle sera créé.

On notera que l'**update operator** utilisé est "**$set**". C'est le plus important car il permet de paramétrer une valeur à un champ donné. Mais vous pourrez en trouver d'autres dans la documentation officielle.

Supprimer un document

Nous pouvons ajouter, lire et mettre à jour, il ne nous reste plus qu'à supprimer. Ce n'est pas plus difficile et voici les deux méthodes principales :

```
# Supprimer un élément
db.nomCollection.deleteOne(filter);
# Supprimer plusieurs éléments
db.nomCollection.deleteMany(filter);
```

Avec encore une fois les même filtres que pour "*find*". Voici un petit exemple :

```
db.users.deleteMany({age:{$lt:18}});
```

Ici on supprime tous les utilisateurs mineurs.