MongoDB

Connexion

```
mongo --host mongodb.ensimag.fr -u <login> <db name> -p
```

Par défaut :

- `<db name>` identique au `<login>`
- Mot de passe identique au `<login>`

C'est une bonne idée d'ajouter dans votre `.bashrc` la ligne suivante :

```
alias mongo='mongo --host mongodb.ensimag.fr -u $USER $USER -p'
```

Changement de mot de passe

```
db.updateUser("<login>", { pwd : "<new passwd>" })
```

Modèle de données

Une *base de données* MongoDB contient des *collections* de *documents*. L'objet `db` représente la base de données.

Un *document* est un ensemble de couples (*attribut*, *valeur*), il est noté `{ "att1" : val1, "att2" : val2, … }`. Une valeur peut être une constante (exemple : `"toto"`, `223.6`), un tableau de valeurs (noté `[ val1, val2, …]`) ou même un document (entre `{}`). Toutes les combinaisons sont possibles.

Tous les documents ont un identifiant unique dans la base de données : l'attribut `_id` de type `ObjectId`. Soit l'identifiant est donné explicitement à la création du document, soit il est généré automatiquement.

Commandes

Les commandes Mongo fonctionnent sur le principe du pipeline : les commandes peuvent être enchainées (on commence toujours par `db`), chacune d'exécutant sur le résultat de la précédente.

Exemple :

```
db.test.find(…).sort(…)
```

- `test` est une *collection* de la base de données (`db`)
- `find(…)` s'applique sur la *collection* `test`
- `sort(…)` s'applique sur le résultat du `find`

La suite présente les commandes les plus fréquemment utilisées (pour le reste, allez voir la documentation en ligne : https://docs.mongodb.com/v3.6/).

pretty

Purement cosmétique: permet d'indenter automatiquement le résultat de la requête affiché dans le terminal afin de le rendre plus lisible.

Exemple :

```
db.eleves.find().pretty()
```

insert(documents)

Insère un document (`{…}`) ou un tableau de documents (`[{…}, {…}, …]`) dans une collection.

Exemple :

```
db.eleves.insert({
  "_id" : ObjectId("5d5bc26f0f38bea8573f29b4"),
  "prenom" : "Maître",
  "nom" : "Yoda",
  "email" : "master.yoda@ensimag.fr",
  "filière" : "ISI"
})
```

remove(filter)

Supprime des documents d'une collection. Le paramètre `filter` (facultatif) est un document indiquant le filtre sélectionnant les documents à supprimer (pas de filtre => supprime tous les documents !)

Exemple :

```
db.eleves.remove({"nom" : "Yoda"})
```

Un document dans le paramètre `filter` peut comprendre des opérateurs. Les principaux opérateurs sont décrits dans la suite.

Opérateurs de comparaison

`$gt`, `$gte`, `$lt`, `$lte`, `$ne`

Exemple :

```
db.notes.find({"note": {$lt : 10}})
```

Tests sur les attributs

`"attribut" : { $exists : true }` : teste si l'attribut existe dans le document.

`"attribut" : { $type : <BSON type> }` : teste le type de l'attribut

Attributs booléens

`$and`, `$or`, `$not`, `$nor`

Leur valeur doit être un tableau de documents de filtrage.

Exemple :

```
db.notes.find({ $and : [{"note": {$gte: 7}} , {"note": {$lte: 10}}]})
```

update(filtre, updates)

Modifie un seul document. Le paramètre `filter` sert à spécifier le document à modifier (le premier sélectionné). Le paramètre `update` peut être :

- un document simple : dans ce cas le document d'origine est remplacé par le nouveau
  ```
  db.eleves.update({"nom" : "Yoda"},
                   {"email" : "maitre@yoda.fr"})
  ```
- Un document utilisant les opérateurs `$set` ou `$unset`
  - `$set` ajoute ou modifie un attribut
  - `$unset` supprime un attribut

  ```
  db.eleves.update({"nom" : "Yoda"},
                   {$set: {"email" : "maitre@yoda.fr"}})
  ```

Il existe une commande `updateMany()` pour pouvoir modifier plusieurs documents en une seule fois.

drop()

Supprime une collection.

find(filter, projection)

Recherche des documents dans une collection. Le paramètre `filter` (facultatif) est un document de filtrage. Le paramètre `projection` (facultatif, mais obligatoirement après `filter`) est un document indiquant les attributs attendus. Il contient des couples (*attribut* : *constante*). Si la constante vaut `1`, l'attribut sera affiché. S'il vaut `0`, l'attribut ne sera pas affiché. L'identifiant du document (attribut `_id`) est affiché par défaut.

Exemple :

```
db.eleves.find({"nom" : "Solo"})
```

aggregate(param)

La commande aggregate permet de calculer des expressions de calcul complexes sur des groupes. Le paramètre `param` peut être un simple document contenant une expression de calcul utilisant l'opérateur $group, ou un tableau de documents expressions de calcul dont au moins un utilise l'opérateur $group. Dans ce dernier cas, les expressions de calcul seront évaluées en pipeline (chacune s'appliquant sur le résultat de la précédente).

Les opérateurs des expressions peuvent être, entre autres :

- `$group : { "_id" : <attribut(s) de groupement>, <attributs> : <expression>, …}`
  l'attribut `_id` définit les attributs de groupement.
  les attributs suivants sont des expressions calculées sur les groupes (ex : `$avg`, `$sum`, `$min`, `$max`, ...)
- `$match : <document de filtrage>`
  Filtre les documents
- `$sort : <ordre>`
  Trie les documents. Le paramètre ordre est un document contenant des couples (attribut : constante) indiquant les attributs de tri ainsi que l'ordre de tri. Si la constante vaut 1, le tri se fera par ordre croissant sur l'attribut. Si la constante vaut -1, le tri se fera par ordre décroissant sur l'attribut.
- `$project : <projection>`
  Projette le résultat précédent sur une partie de ses attributs.
- `$lookup : { "from" : "<collection>",`
  `"localField" : <local attributes>,`
  `"foreignField" : <foreign attributes>,`
  `"as" : "<result local name>"}`
  L'opérateur `$lookup` effectue l'équivalent d'une jointure (pas très efficace). Dans le résultat, nommé dans la suite de l'`aggregate` par l'attribut `as`, les attributs locaux listés dans `localField` correspondront à ceux listés dans `foreignField` de la collection `from`.

Exemples :

```
db.notes.aggregate(
  [{$match: {"note": {$gte : 10}}} ,
   {$group: {"_id": "$cours", "moyenne": {$avg: "$note"}}}
  ])

db.notes.aggregate(
  [{$group: {"_id": "$cours", "moyenne": {$avg: "$note"}}},
```

```
    {$sort: {"moyenne": -1}}
  ])

db.notes.aggregate(
  [{$group: {"_id": "$cours", "moyenne": { $avg: "$note"}}},
   { $group: {"_id": null , "moyenne": { $min: "$moyenne"}}}
  ])
db.eleves.aggregate(
  [{ $lookup: {"from": "notes",
               "localField": "_id",
               "foreignField": "numEleve",
               "as": "notes"}},
   { $project: {"nom": 1, "prenom": 1, "notes.note": 1}}
  ])
```

sort(order)

La commande sort permet de trier des documents résultant d'une autre commande.

Exemple :

```
db.eleves.find().sort({"nom" : 1, "prenom" : 1 })
```

distinct(attributes, filter)

La commande `distinct` propose d'éliminer les doublons dans une requête simple. Le paramètre `attributes` est le tableau des attributs à projeter et le paramètre `filter` (facultatif) est un document condition de filtrage.

Copier la base de données du TP

Connectez-vous à MongoDB et exécutez les commandes suivantes :

```
use TP
var resto = db.restaurants.find()                          // copie de la BD

use <login>
resto.forEach(function(r) { db.restaurants.insert(r); })  //insertion de la copie
```

(en remplaçant `<login>` par votre login Agalan)