cours 5A big data REMY Guillaume

mardi 17 novembre 2020
10:52

couchDB : http://127.0.0.1:5984/_utils
cmd :
curl -X GET http://localhost:5984
add bdd : curl -u admin:admin -X PUT http://localhost:5984/etudiants
add doc : curl -u admin:admin -X PUT http://localhost:5984/etudiants/doc1 -d {\"prenom\":\"Jean\"}
add fichier json : curl -u admin:admin -X POST http://localhost:5984/etudiants/_bulk_docs -d @filmscouchdb.json -H "Content-Type:application/json"

MongoDB :
lancer mongod : mongod.exe --dbpath 5a2i
lancer mongo : mongo.exe

```
>
> use demo
switched to db demo
> db.createCollection("cours")
{ "ok" : 1 }
> show collections
cours
> db.cours.insert({"intitulé":"Services Web"})
WriteResult({ "nInserted" : 1 })
```

```
> db.cours.find()
{ "_id" : ObjectId("5fc4be87697e17fc3628c4cf"), "intitulé" : "Services Web" }
> db.cours.insert({"intitulé":"Micro-services"})
WriteResult({ "nInserted" : 1 })
> db.cours.find()
{ "_id" : ObjectId("5fc4be87697e17fc3628c4cf"), "intitulé" : "Services Web" }
{ "_id" : ObjectId("5fc4beb6697e17fc3628c4d0"), "intitulé" : "Micro-services" }
```

Import doc json : mongoimport.exe -d demo30nov -c restaurants --file restaurants.json

Exemple commande robo 3T : db.getCollection('restaurants').find({"borough":"Brooklyn", "cuisine":"Italian"}) →permet de trouver les documents correspondant au pattern entre {…}

db.getCollection('restaurants').find({"borough":"Brooklyn", "cuisine":"Italian", "address.zipcode":"10019"}).count() → renvoi le nombre de document validant le pattern

Recherche par mot clé dans le nom : db.getCollection('restaurants').find({"name": /pub/i})

Projection (pour renvoyer que le nom par exemple) :
db.getCollection('restaurants').find({"borough":"Brooklyn", "cuisine":"Italian"}, {"name":1, "_id":0})
on ajoute un 2ème paramètre à find()

.find() renvoi les documents d'une collection
.findOne() renvoi le premier document

Exemple avec opérateurs : Query and Projection Operators — MongoDB Manual

- Restaurant avec un score supérieur ou égal à 15 :
  db.getCollection('restaurants').find({"grades.score":{$gte: 15, $not:{$lt:14}}})
- Restaurant avec un note inf à 5 et grade = A :

```
db.getCollection("restaurants").find( {
   "grades": {
      $elemMatch: {
         "score": {
            $lt: 5 },
         "grade":"A" }
   }
},
{
   "grades.grade":1, "grades.score":1, "_id":0
}
)
```

- Restaurant grade le plus récent = B :

```
db.getCollection("restaurants").find( {
   "grades.0.grade":"B"
},
{
   "name":1, "_id":0
}
)
```

- voir tous les types de cuisine :
  db.getCollection("restaurants").distinct("cuisine")

Avec des variables :

- resto grade B :
  match = {$match: {"grades.0.grade": "B"}}; →équivalent de find()
  project = {$project: {"name":1, "_id":0}}; →projection
  db.restaurants.aggregate([match, project]); →aggrégation

- resto grade B classés par zipcode :
  match = {$match: {"grades.0.grade": "B"}};
  project = {$project: {"name":1, "_id":0, "address.zipcode":1}};
  db.restaurants.aggregate([match, project]);
  sort = {$sort: {"address.zipcode":1}}; →trier les données par zipcode
  db.restaurants.aggregate([match, project, sort]); →relance commande avec triage

- resto avec un B en première évaluation :
  db.restaurants.count({"grades.0.grade":"B"})
  OU avec group
  match = {$match: {"grades.0.grade": "B"}};
  group = {$group: {"_id":null, "somme":{$sum:1}}};
  db.restaurants.aggregate([match, group]);

- nombre de resto dans un quartier :
  db.restaurants.distinct("borough") →pour voir tous les quartiers

match = {$match: {"borough": "Bronx"}}; →quartier du Bronx
group = {$group: {"_id":"$borough", "somme":{$sum:1}}}; →somme des resto dans ce quartier
db.restaurants.aggregate([match, group]);

Mettre à jour un document :

- changer la valeur du champ cuisine :

```
db.restaurants.update(
{
   "_id": ObjectId("5fc4cc846a4f8c5608cf5655") →ID récup avec un findOne() par exemple
},
{
   $set: {"cuisine":"Lorraine"} →modif de la valeur
}
)
```

- ajouter une nouvelle clé :

```
db.restaurants.update(
{
   "_id": ObjectId("5fc4cc846a4f8c5608cf5655")
},
{
   $set: {"test":"ajout d'une clé"} →ajout
}
)
```

- supprimer une clé :

```
db.restaurants.update(
{
   "_id": ObjectId("5fc4cc846a4f8c5608cf5655")
},
{
   $unset: {"test":1} →suppression
}
)
```