

Couch DB

- Installer Couch DB et curl (<https://o7planning.org/fr/11617/installation-de-curl-sous-windows>)
- Pour démarrer couch DB aller dans gestionnaire des tâches → services → chercher le service « Apache couchDB » → puis ouvrir les services et démarrer le service « Apache couchDB »
- Pour se connecter sur l'interface web de couch DB : <http://127.0.0.1:5984/> [utils](#)
- Créer une base de données « étudiants » :

```
C:\Users\thiba\Downloads\curl-7.73.0-win64-mingw\bin>curl -u admin:admin -X PUT http://127.0.0.1:5984/etudiants {"ok":true}
```

- Renvoie le nbre de documents de la base de données « étudiant » avec le champs « doc_count »:

```
C:\Users\thiba\Downloads\curl-7.73.0-win64-mingw\bin>curl -u admin:admin -X GET http://127.0.0.1:5984/etudiants {"db_name":"etudiants","purge_seq":"0-g1AAAABXejZLYWBgYMpgTmEQTM4vTc5ISXLIyU9OzMnILy7JAUnlsQBjhgYg9R8IshIZ8KhNZEiqhyjKAgBm5Rxs","update_seq":"0-g1AAAABXejZLYWBgYMpgTmEQTM4vTc5ISXLIyU9OzMnILy7JAUnlsQBjhgYg9R8IshIZ8KhNZEiqhyjKAgBm5Rxs","sizes":{"file":16700,"external":0,"active":0},"props":{"doc_del_count":0,"doc_count":0,"disk_format_version":8,"compact_running":false,"cluster":{"q":2,"n":1,"w":1,"r":1},"instance_start_time":"0"}}
```

Ici aucun document pour le moment

- Insertion d'un document

```
C:\Users\thiba\Downloads\curl-7.73.0-win64-mingw\bin>curl -u admin:admin -X PUT http://127.0.0.1:5984/etudiants/doc1 -d {"prenom":"thibaud"} {"ok":true,"id":"doc1","rev":"1-2623a5cee59592d3c845b573e47fad23"}
```

- Récupérer document

```
C:\Users\thiba\Downloads\curl-7.73.0-win64-mingw\bin>curl -u admin:admin -X GET http://127.0.0.1:5984/etudiants/doc1 {"_id":"doc1","_rev":"1-2623a5cee59592d3c845b573e47fad23","prenom":"thibaud"}
```

- Importer un seul film

```
C:\Users\thiba\Downloads\curl-7.73.0-win64-mingw\bin>curl -u admin:admin -X PUT http://127.0.0.1:5984/etudiants/film1 -d @film1.json -H "Content-Type:application/json" {"ok":true,"id":"film1","rev":"1-cfa6566fa0deb48992c3a8b313223f68"}
```

- Importer une liste de film sous la forme d'un doc il va séparer la liste en plusieurs fichier de films différents

```
C:\Users\thiba\Downloads\curl-7.73.0-win64-mingw\bin>curl -u admin:admin -X POST http://127.0.0.1:5984/etudiants/_bulk_docs -d @film2.json -H "Content-Type:application/json"
```

<input type="checkbox"/>	movie:10098	movie:10098
<input type="checkbox"/>	movie:1018	movie:1018
<input type="checkbox"/>	movie:103	movie:103
<input type="checkbox"/>	movie:103731	movie:103731
<input type="checkbox"/>	movie:10669	movie:10669
<input type="checkbox"/>	movie:10675	movie:10675
<input type="checkbox"/>	movie:10835	movie:10835
<input type="checkbox"/>	movie:10889	movie:10889
<input type="checkbox"/>	movie:1091	movie:1091
<input type="checkbox"/>	movie:10935	movie:10935
<input type="checkbox"/>	movie:11	movie:11
<input type="checkbox"/>	movie:110415	movie:110415

Mongo DB et Robot 3T

Commandes utiles :

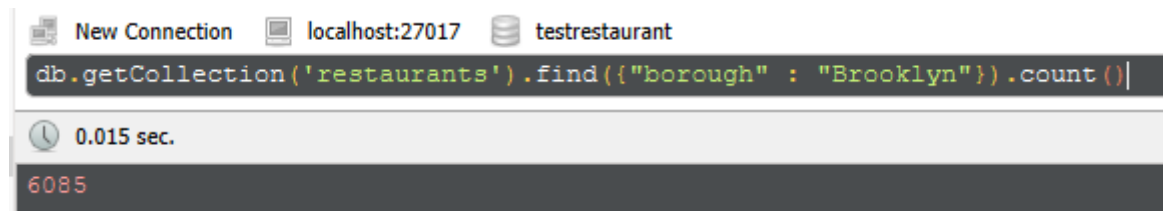
- Mongod.exe pour lancer le server
- Mongo.exe pour lancer le client
- Mongo import pour insérer des données dans le serveur mysql

```
C:\Program Files\MongoDB\Server\4.0\bin>mongoimport -d testrestaurant -c restaurants --file restaurants.json
2020-11-30T11:39:38.494+0100    connected to: localhost
2020-11-30T11:39:39.137+0100    imported 25357 documents
```

- Use demo //changer de BD
- Db.createCollection("cours") // Créer une nouvelle collection
- Show collections // Afficher les collections
- Db.cours //Rentrer dans la base de données cours pour effectuer des actions
- Db.insert // Insérer des données dans la base
- Db.find // renvoie tous les documents
- Db.findOne // renvoie un document

On va utiliser robot 3T afin d'effectuer des requêtes sur notre base de données.

Exemple de la requête utilisée pour trouver le nombre de restaurant à Brooklyn :



Filtre et projection

Satisfaire un pattern dans un document JSON. Nous allons dans ce TP utiliser le logiciel Robot 3T qui est un équivalent de PhpMyAdmin c'est-à-dire un client d'une base de données MySQL.

La commande suivante renvoie tous les restaurants du quartier Manhattan qui propose comme type de cuisine « Irish » situé à l'adresse 100019 et qui contient le mot pub (cette notation nous permet de le rendre insensible à la casse)

```
db.getCollection('restaurants').find({
  "borough" : "Manhattan",
  "cuisine" : "Irish",
  "address.zipcode" : "10019",
  "name" : /pub/i})|
```

La commande suivante nous permet de compter le nombre de restaurant avec les paramètres défini précédemment. On s'aperçoit qu'il y en a 8.

```
db.getCollection('restaurants').find({
  "borough" : "Manhattan",
  "cuisine" : "Irish",
  "address.zipcode" : "10019",
  "name" : /pub/i}).count()|
```

0.021 sec.

8

Pour renvoyer la liste des restaurant avec les filtres suivant mais en affichant seulement la liste des noms des restaurants.

```
db.getCollection('restaurants').find({
  "borough" : "Manhattan",
  "cuisine" : "Irish",
  "address.zipcode" : "10019",
  "name" : /pub/i},|
{
  "name":1,
  "_id":0
})
```

Renvoyer en plus le type de cuisine par rapport à la requête précédente.

```
db.getCollection('restaurants').find({
  "borough" : "Manhattan",
  "cuisine" : "Irish",
  "address.zipcode" : "10019",
  "name" : /pub/i},
{
  "name":1,
  "_id":0,
  "cuisine":1
})
```

Filtrage avec des comparateurs

Nous allons maintenant renvoyer tous les restaurants qui se trouvent dans un quartier ayant un score inférieur à une certaine valeur. Nous allons renvoyer les noms et les scores des restaurants de Manhattan dont le score est inférieur à 5. Sans `$not:{$gte:5}` dès qu'un restaurant a un score inférieur à 5, la requête nous envoie tous les autres scores.

```
db.restaurants.find({
  "borough" : "Manhattan",
  "grades.score": {
    $lt: 5,
    $not:{
      $gte: 5
    }
  },
  {
    "name":1, "_id":0, "grades.score":1
  }
})
```

restaurants 0.032 sec.

```
/* 1 */
{
  "grades" : [
    {
      "score" : 2
    },
    {
      "score" : 2
    },
    {
      "score" : 2
    },
    {
      "score" : 2
    },
    {
      "score" : 2
    }
  ],
  "name" : "Cafe Madison"
}
```

Renvoie le grade et le score des restaurants dont le grade est égal à A et inférieur à 5

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

Renvoie le nom et le grade du restaurant dont le premier élément de la liste grade est un score de B.

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

Recherche pour lister tous les types de cuisine

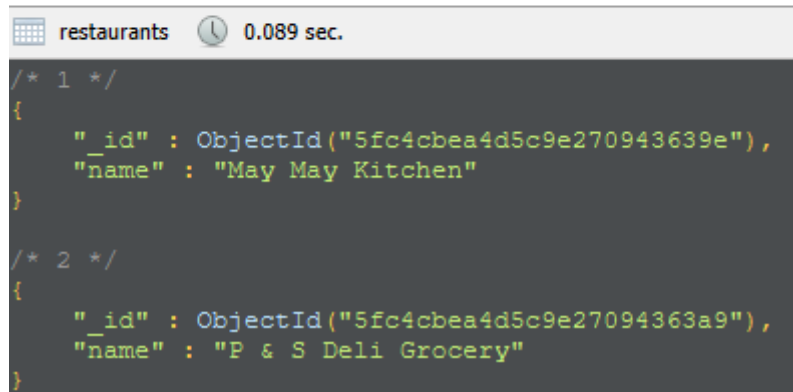
```
db.restaurants.distinct("cuisine")
```

Agrégation

On va définir une entité comme étant liée à plusieurs entités de classes différentes.

Fait un agrégat et nous renvoie le nom du restaurant qui à comme premier grade la lettre C.

```
db.restaurants.aggregate(
[
  {
    $match:{
      "grades.0.grade":"B"
    }
  },
  {
    $project:{
      "name":1
    }
  }
])
```



```
restaurants 0.089 sec.
/* 1 */
{
  "_id" : ObjectId("5fc4cbea4d5c9e270943639e"),
  "name" : "May May Kitchen"
}
/* 2 */
{
  "_id" : ObjectId("5fc4cbea4d5c9e27094363a9"),
  "name" : "P & S Deli Grocery"
}
```

On peut voir ici que le restaurant May May kitchen en fait partie.

```
match = {$match : {"grades.0.grade":"B"}}; //db.restaurants.find()
project = {$project : {"_id":0}}; //db.restaurants.find({}, {}), deux paramètres avec projection...
db.restaurants.aggregate([match, project]); // aggrégation...
sort = {$sort : {"address.zipcode":1}};
db.restaurants.aggregate([match, project, sort]);
```

L'agrégation peut se définir comme telle : la classe globale contient une référence à une autre classe et est dit d'avoir la propriété de cette classe. Chaque classe référencée est considérée comme partie de la classe globale.

Nous allons écrire une requête qui permet de renvoyer le nombre de restaurants ayant dans la dernière évaluation un B.

On remarque ici que nous obtenons un résultat de 25357 quand nous faisons un aggregate de match_c et groupe.

```
match = {$match : {"grades.0.grade":"B"}}; //db.restaurants.find()
project = {$project : {"_id":0}}; //db.restaurants.find({}, {}), deux paramètres avec projection...
db.restaurants.aggregate([match, project]); // aggrégation...
sort = {$sort : {"address.zipcode":1}};
db.restaurants.aggregate([match, project, sort]);
db.restaurants.count({"grades.0.grade":"B"}) //Appliquer un filtre
db.restaurants.find({"grades.0.grade":"B"}).count()
match_c = {$match: {}};
groupe = {$group : {"_id":null, "somme":{$sum:1}}}; //Fait la somme
db.restaurants.aggregate([match_c, groupe]);
```

match = {\$.. ✕ project = .. ✕ db.restaur.. ✕ sort = {\$s.. ✕ db.restaur.. ✕ db.restaur.. ✕ db.restaur.. ✕ match_c = .. ✕ groupe = .. ✕

restaurants 0.009 sec.

```
/* 1 */
{
  "_id" : null,
  "somme" : 25357.0
}
```

On remarque ici que nous obtenons un résultat de 25357 quand nous faisons un aggregate de groupe.

```
match = {$match : {"grades.0.grade":"B"}}; //db.restaurants.find()
project = {$project : {"_id":0}}; //db.restaurants.find({}, {}), deux paramètres avec projection...
db.restaurants.aggregate([match, project]); // aggrégation...
sort = {$sort : {"address.zipcode":1}};
db.restaurants.aggregate([match, project, sort]);
db.restaurants.count({"grades.0.grade":"B"}) //Appliquer un filtre
db.restaurants.find({"grades.0.grade":"B"}).count()
match_c = {$match: {}};
groupe = {$group : {"_id":null, "somme":{$sum:1}}}; //Fait la somme
db.restaurants.aggregate([groupe]);
```

match = {\$.. ✕ project = .. ✕ db.restaur.. ✕ sort = {\$s.. ✕ db.restaur.. ✕ db.restaur.. ✕ db.restaur.. ✕ match_c = .. ✕ groupe = .. ✕

restaurants 0.009 sec.

```
/* 1 */
{
  "_id" : null,
  "somme" : 25357.0
}
```

On peut en conclure que les deux aggregate sont similaire et effectue la même chose et compte le même nombre de restaurants.

Pour changer la valeur d'un champ dans la base, on utilise la commande update comme ci-dessous

```
db.restaurants.update(  
  {  
    "_id": ObjectId("5fc4cbea4d5c9e2709436396")  
  },  
  {  
    $set : {"cuisine":"lorraine"}  
  }  
);  
db.restaurants.findOne();
```

```
"borough" : "Queens",  
"cuisine" : "lorraine",  
"grades" : [
```

On observe bien que le champ cuisine à bien été modifié avec la valeur « lorraine »