

Introduction à MongoDB : corrigé

Stéphane Malandain – Printemps 2024

1. A faire

Répondre aux questions suivantes

1. Quels sont les sportifs (identifiant, nom et prénom) qui ont entre 20 et 30 ans ?

```
db.Sportifs.find(
{
  "Age": { "$gte": 20, "$lte": 30 }
},
{
  "_id": 0,
  "IdSportif": 1,
  "Nom": 1,
  "Prenom" :1
})
```

2. Quels sont les gymnases de "Villetaneuse" ou de "Sarcelles" qui ont une surface de plus de 400 m2 ?

```
db.Gymnases.find(
{
  "Ville":{"$in":["VILLETANEUSE", "SARCELLES"]},
  "Surface": { "$gt": 400 }
},
{
  "_id": 0,
  "NomGymnase": 1,
  "Ville": 1,
  "Surface" :1
})
```

3. Quels sont les sportifs (identifiant et nom) qui pratiquent du hand ball ?

```
// pour spécifier un sous-item d'un item, utilisez le formalisme item.sousitem
db.Sportifs.find(
  {
    "Sports.Jouer": "Hand ball"
  },
  {
    "_id": 0,
    "IdSportif": 1,
    "Nom": 1
  }
)
```

4. Dans quels gymnases et quels jours y a t-il des séances de hand ball ?

Depuis la version 2.2, MongoDB propose un Framework d'agrégation. Le framework d'agrégation comporte un ensemble large d'opérateurs :

- \$project : redéfinition des documents (si nécessaire).
- \$match : restriction sur les documents à utiliser.
- \$group : regroupements et calculs à effectuer.
- \$sort : tri sur les agrégats.
- \$unwind : découpage de tableaux.
-

Une requête d'agrégation est de la forme `db.collection.aggregate([{...}, {...}, ...])`.

```
db.Gymnases.aggregate( [
  { $unwind : "$Seances" },
  { $match: { "Seances.Libelle" : "Hand ball" }},
  { $group: {
    "_id": {
      "Nom": "$NomGymnase ",
      "Ville": "$Ville",
      "Jour": { $toLower: "$Seances.Jour" }
    },
    "nb": { $sum: 1 }
  }},
  { $sort: {
    "_id.Ville": 1,
    "_id.Nom": 1,
    "nb": -1
  }}
])
```

5. Quels sportifs (identifiant et nom) ne pratiquent aucun sport ?

```
db.Sportifs.find(
  {
    "Sports" : { "$exists" : false }
  },
  {
    "_id": 0,
    "Nom": 1
  })
```

6. Quels gymnases n'ont pas de séances le dimanche ?

```
db.Gymnases.find(
  {"Seances.Jour": { "$nin" : [ "dimanche", "Dimanche" ] }
},
  { "_id": 0,
    "NomGymnase": 1,
    "Ville": 1,
    "Seances.Jour": 1
  }
)
```

7. Quels gymnases ne proposent que des séances de basket ball ou de volley ball ?

```
db.Gymnases.find(
  {
    "$nor": [
      { "Seances.Libelle": { "$ne": "Basket ball" } },
      { "Seances.Libelle": { "$ne": "Volley ball" } } ]
  },
  {
    "_id": 0,
    "NomGymnase": 1,
    "Ville": 1,
    "Seances.Libelle": 1
  })
```

8. Quels sont les entraîneurs qui sont aussi joueurs ?

```
db.Sportifs.find(
  {
    "Sports.Jouer" : { "$exists" : true },
    "Sports.Entraîner" : { "$exists" : true }
  },
  {
    "_id": 0,
    "Nom": 1
  }
)
```

9. Quels sont les sportifs qui sont des conseillers ?

```
db.Sportifs.find(
  {"IdSportif":{"$in":
db.Sportifs.distinct("IdSportifConseiller")
},
{
  "_id": 0,
  "Nom": 1
}
)
```

10. Pour le sportif "Kervadec" quel est le nom de son conseiller ?

```
db.Sportifs.find(
{ "IdSportif":db.Sportifs.findOne(
  {"Nom": "KERVADEC" }).IdSportifConseiller
},
{
  "_id": 0,
  "Sports": 0
}
)
```

11. Quels entraîneurs entraînent du hand ball et du basket ball ?

```
db.Sportifs.find(
  { "Sports.Entraîner": "Hand ball",
    "Sports.Entraîner": "Basket ball" },
  { "_id": 0,
    "Nom": 1,
    "Sports.Entraîner": 1 }
)

// ou

db.Sportifs.find(
  {$and: [
    { "Sports.Entraîner": "Hand ball" },
    { "Sports.Entraîner": "Basket ball" }
  ]
},
  { "_id": 0,
    "Nom": 1,
    "Sports.Entraîner": 1
  }
)
```

12. Quels sont les couples de sportifs (identifiant et nom et prénom de chaque) de même âge ?

```
Var sportifs1=db.Sportifs.find({},
{   "_id":0,
    "IdSportif":1,
    "Nom":1,
    "Prenom":1,
    "Age":1
})
).toArray();

var sportifs2 =sportifs1;

for(var i = 0; i < sportifs1.length; i++){
    for(var j = 0; j < sportifs2.length; j++){
        if(sportifs1[i].IdSportif<sportifs2[j].IdSportif) {
            if(sportifs1[i].Age==sportifs2[j].Age) {
                printjson('(' +sportifs1[i].IdSportif + ' ' + sportifs1[i].Nom+
                    '+sportifs1[i].Prenom+ ', '+ sportifs2[j].IdSportif + '
                    '+sportifs2[j].Nom+ ' '+sportifs2[j].Prenom+ ')');
            }
        }
    }
}
```

13. Quelle est la moyenne d'âge des sportives qui pratiquent du basket ball ?

```
db.Sportifs.aggregate([
    { $match: { "Sports.Jouer": "Basket ball", "Sexe": { $in: ["f", "F"] } }},
    { $group: { "_id": null, "AgeMoyen": { $avg: "$Age" } } }
])
```

14. Quels sont les sportifs les plus jeunes ?

```
// On calcule l'âge min et on le stocke dans une variable agemin. Le calcul de
// l'âge min se fait en utilisant la fonction « aggregate ».
```

```
var agemin=db.Sportifs.aggregate(
    [{ $group: { _id: null,
        "agemin": { $min: "$Age" } } }]).next();

// On cherche les sportifs avec Age = agemin.agemin.
db.Sportifs.find({"Age": agemin.agemin}, {"_id":0,"Sports":0})
```

On peut faire la requête en une seule ligne comme suit :

```
db.Sportifs.find({"Age":
db.Sportifs.aggregate([
{
  $group: {
    "_id": null,
    "agemin": {$min:"$Age"}}}).next().agemin
  },
{
  "_id":0,
  "Sports":0
}
])
```

15. Quels entraîneurs n'entraînent que du hand ball ou du basket ball ?

```
// on cherche ceux qui entraînent l'un ou l'autre, mais pas d'autres sports

db.Sportifs.find(
{$and: [{ $or: [{ "Sports.Entraîner": "Hand ball" },
                { "Sports.Entraîner": "Basket ball" }
            ]},
        { "Sports.Entraîner" : { $nin : autres }}]
},
{
  "_id": 0,
  "Nom": 1,
  "Sports.Entraîner" :1
})
```

16. Pour chaque sportif donner le nombre de sports qu'il arbitre

```
db.Sportifs.aggregate([
{ $match: { "Sports.Arbitrer": { $exists: true } }},
{ $unwind: "$Sports.Arbitrer"},
{ $group: { _id: "$Nom", "nbArbitrer": { $sum: 1 } } }
])
```

17. Pour chaque gymnase de Stains donner par jour d'ouverture les horaires des premières et dernières séances

```
db.Gymnases.aggregate([
{ $match: { "Ville": "STAINS" } },
{ $unwind: "$Seances" },
{ $project: { "nom": "$NomGymnase", "jour":
{$toLower: "$Seances.Jour" }, "h": "$Seances.Horaire"}},
{ $group: { _id: { "nom": "$nom", "jour": "$jour" },
"debut": { $min: "$h"}, "fin": { $max: "$h" } } }
])
```

18. Pour chaque entraîneurs de hand ball quel est le nombre de séances journalières qu'il assure ?

```
// On récupère la liste des identifiants des entraîneurs de Hand.
var entraîneursHand = db.Sportifs.find({ "Sports.Entraîneur" :
"Hand ball" }, { _id: 0, IdSportif: 1 }).toArray().map(function(e) {
return e.IdSportif });
// On les utilise pour faire la restriction (avec le $match) dans l'agrégation.
db.Gymnases.aggregate([
  { $unwind: "$Seances" },
  { $match: { "Seances.IdSportifEntraîneur": { $in: entraîneursHand
}}}},
  { $project : { "ent": "$Seances.IdSportifEntraîneur",
"jour": { $toLower: "$Seances.Jour" } }},
  { $group: { _id: { "entraîneur": "$ent", "jour": "$jour"},
nbSeances: { $sum: 1 } } }
])
```

19. Calculer le nombre de gymnases pour chaque ville

Le paradigme MapReduce en MongoDB

- MapReduce est un des mécanismes les plus complexes de requêtes de mongoDB. Il se base sur la spécification des deux fonctions Map et Reduce (écrites en javascript). Ces deux fonctions sont des fonctions définies par l'utilisateur.
- En mongoDB, la fonction Map, permet de générer des documents clés-valeurs pour les transmettre en entrée à Reduce. La fonction ne prend aucun paramètre, on accède à l'objet analysé via l'opérateur this. La fonction peut émettre des couples via la fonction emit(key, value) autant de fois que nécessaire dans la fonction.
- La fonction Reduce retourne le résultat agrégé à partir de ces documents en entrée. La fonction prend deux paramètres key et values (tableau des valeurs de la clé). Elle peut être appelée plusieurs fois pour la même clé, elle doit donc renvoyer une valeur de même type que celles dans le tableau.
- On doit passer un troisième paramètre, un littéral JSON, qui représente les options de la fonction. La principale option (out) est la collection dans laquelle le résultat sera placé. Si l'on veut voir le résultat, sans le stocker, il est possible d'indiquer out: { inline: 1 }

```
// La fonction Map, crée un document avec comme clé la ville
// et 1 comme valeur. Ce document est transmis à la fonction Reduce
var Map=function() {
  emit(this.Ville, 1); }

// La fonction Reduce, crée un document clé/valeur avec comme
// clé ville et nbgym qui indique le nombre de gymnases pour cette ville
var Reduce=function(cle, nbgym)
  {return Array.sum(nbgym); }

// Exécution des fonctions Map et Reduce, sur la collection des
// gymnases et affichage des résultats
db.Gymnases.mapReduce( Map, Reduce,
  { out: { inline: 1 } })
```

20. Calculer le nombre de séances pour chaque jour de la semaine.

```
var Map=function() {
  if (this.Seances) { //il y a des gymnases qui ne proposent pas de séances
    for (s of this.Seances) // pour chaque item de Seances, émettre ses jours
    { emit(s.Jour.toLowerCase(), 1);
    } } };
var Reduce =function(jours, nbseances) {
  return Array.sum(nbseances); };
db.Gymnases.mapReduce(Map, Reduce,
  { out: { inline: 1 }} )
```

21. Calculer la superficie moyenne des gymnases pour chaque ville.

Pour cela, vous devez calculer la somme des superficies ET le nombre de gymnase (à émettre dans un même objet et à réduire en tenant compte que ce double aspect)

```
var Map=function() {
  emit(this.Ville,
  { "nb": 1,
    "surface": this.Surface
  });
};

var Reduce=function(ville, valeurs) {
  var nb = 0, surface = 0;
  for (val of valeurs) {
    nb += val.nb;
    surface += val.surface;
  }
  return { "nb": nb,
    "surface": surface,
    "surfMoy": Math.round( surface / nb)
  }
};

db.Gymnases.mapReduce(Map, Reduce, { out: { inline: 1 }})
```