RAPPORT TRAVAIL VTE

21/04/2020

Mohammed Anouar Chairi Bounekoub

1. *Manière choisie pour ajuster la collection « cours » :*

Définition :

```
{  $ set :  {  < newField >:  < expression > ,  ...  }  }
```

Le $set permet de rajouter un nouveau champ aux documents.

Remarque : Si le nom du nouveau champ que l'on veut rajouter existe déjà, le $set remplace les valeurs existantes de ce champ par des valeurs nouvelles.

```
{ $rename :  {  < field1 >:  < newName1 > }
```

Le $rename met à jour le nom du champ existant.

L'ancien champ doit être différent du nouveau champ.

Voici la requête que j'ai faite pour ajuster ma collection :

db.cours.update ({},{ $set:{prerequis:[]}},{multi:true})

1. *Tout d'abord, j'ai fait un update de ma collection pour rajouter (set) un champ prérequis de vecteur vide pour chaque ligne de ma collection.*

2. *Ensuite, j'ai mis à jour le champ prérequis pour les cours qui ont des prérequis directs.*

   Grâce aux requêtes suivante :

   db.getCollection("cours").update({_id:"4IAPF"},{$set:{prerequis:['4IPID','4ISIP']}});

   db.getCollection("cours").update({_id:"4IASR"},{$set:{prerequis:['4IBRE']}});

   db.getCollection("cours").update({_id:"4IEI2"},{$set:{prerequis:['4IAPF','4IASR','4INEB','4IPLG','4XORG','4XSTA','4IMA2','4ISE2','4ISO2','4XICP','4IVTE']}});

   db.getCollection("cours").update({_id:"4IGEB"},{$set:{prerequis:['4IIBD']}});

   db.getCollection("cours").update({_id:"4IPAC"},{$set:{prerequis:['4ITGP','4IPAI']}});

   db.getCollection("cours").update({_id:"4IPDB"},{$set:{prerequis:['4IPO2','4IGEB']}});

   db.getCollection("cours").update({_id:"4IPDW"},{$set:{prerequis:['4IWPB']}});

   db.getCollection("cours").update({_id:"4IPID"},{$set:{prerequis:['4IPDB']}});

   db.getCollection("cours").update({_id:"4IPO2"},{$set:{prerequis:['4IPAP']}});

   db.getCollection("cours").update({_id:"4ISIP"},{$set:{prerequis:['4IPAC']}});

3. *Pour modifier le champs etcs en ects j'ai fait un updateMany en utilisant l'opérateur $rename :*

   db.getCollection("cours").updateMany({},{$rename:{etcs:'ects'}})

2. *Script des 10 Requêtes avec une petite explication*

1. *Pour chaque cours, changer le champ « etcs » en « ects » et ceci en une seule requête.*

   db.getCollection("cours").updateMany({},{$rename:{etcs:'ects'}})

   Explication mise au point 1. Manière….

2. *À partir d'une seule requête afficher le nombre de cours, le nombre de périodes et le nombre d'ects.*

   Définition : La pipeline d'agrégation est un cadre d'affiliation basée sur le concept de pipelines de traitement de donnée.

   Pipeline : En MongoDB la pipeline est composé d'étape. Les documents sont transformés lors de leur passage dans les étapes. Par exemple, une étape peut générer ou filtrer un nouveau document.

   $group : Le $group permet de saisir les documents par l'expression spécifiée(_id). Le champ (_id) de chaque document contient le groupe unique par valeur. Le document peut contenir des champs qui contiennent des valeurs calculées. Les valeurs calculées se créent grâce aux expressions d'accumulateur.

   Accumulateur $sum : Permet de calculer et de renvoyer la somme des valeurs numériques. Il ignore les valeurs non numériques.

   $project : permet de prendre un document qui peut spécifier l'inclusion des champs, suppression de l'_id, l'ajout d'un nouveau champ et l'initialisation des valeurs des champs existants dernièrement on peut aussi spécifier l'exclusion d'un champ.

   Par exemple : pour l'exclusion d'un champ, on met le nom du champ à 0 pour le cacher.

```
{  $ project :  {  "< _id >" :  0 ,  "<field2>" :  0 ,  ...  }  }
```

   Réponse exercice :

   db.cours.aggregate ([

   {$group: {_id:0, nbCours:{$sum:1},nbECTS:{$sum:'$ects'}, nbPeriodes:{$sum:'$nbPeriodes'}}},

   {$project:{_id:0}} .

   ])

3. *Affichez tous les codes des cours multi-sections, ils ont un code avec le format « 4X… »*

   Définition : db.collection.find({requete} ,{projection})

   *Le find() fonction qui peut être vue comme une requête en SQL « « Select * from table ».*

Le deuxième paramètre représente le select et le premier paramètre représente le where.

Le paramètre de projection : permet de renvoyer le champ dans le document.

Si la valeur du champ est à 1 ou true le champ sera affiché.

Si la valeur du champ est à 0 ou false le champ ne sera pas affiché.

Le paramètre requête : permet d'effectuer une recherche avec des valeurs recherchées.

Par exemple :db.collection.find({_id :5},{}) //renvoie les documents dont l'_id égal 5.

(Pour avoir accès aux champs d'un document incorporé, il faut utiliser la notation par point « Db.collection.({"cours.nbheure" : 135},{}) »).

Opérateur $regex : renvoie des documents dont la valeur du champ correspond à l'expression régulière.

Réponse :

```
db.getCollection("cours").find({_id:{$regex :/^4X/}},{_id:1,intitule:1})
```

Deuxième manière de procéder qui donne le même résultat :

```
db.getCollection("cours").find({_id:/4X/},{_id:1,intitule:1}
```

4. *Liste des cours que l'on doit avoir réussi pour suivre un cours donné. Le cours peut être stocké dans une variable JS : LeCours = '4IPDB'*

   Essayez de répondre à cette question avec juste un champ «prérequis » direct et l'opérateur « $graphLookup ».

Définition :$graphLookup effectue une recherche récursive sur une collection.

Dans le $GraphLookup on retrouve les champs suivant :

From : cible la collection que l'on va utilisée .

startWith : on va retrouver la valeur avec laquelle on va commencer la recherche récursives. Le startWith peut agir comme un tableau de valeur.

connectFromField : nom du champ dont la valeur utilisé permet de correspondre récursivement à celle des autres documents de la collection.

connectToField : nom du champ dans un autre document qui vont être comparer avec les valeurs dans le paramètre connectFromField .

As : Nom du champ qui va nous permettre de consulter/utiliser les informations des documents parcouru dans le graphLookup.

Réponse :

```
cours = db.cours.findOne({_id : '4IPDB'}.id)

db.cours.aggregate ([{

    $graphLookup: {

    from: "cours",

    startWith: '$prerequis',
```

connectFromField: 'prerequis',

connectToField:'_id',

as :'prerequies',

}}, { $match :{_id:cours}},{ $project:{_id:0,'cours':"$prerequies._id"}}])

5. *Liste des déterminants, résultat trié par ordre alphabétique sur l' «_id» :*

L'affichage en Studio3T se fait sur plusieurs lignes, ici j'ai tout condensé en une seule ligne :

1. *Affichage avec un find et un foreach.*

Définition : Pour cet exercice j'ai utilisé la méthode sort () pour trier les id en ordre croissant on précise le nom du champ et on met comme valeur 1 pour ordre croissant et -1 pour trier en ordre décroissant.

La méthode foreach() combinée avec une function va nous permettre d'avoir l'affichage demandé.

Le foreach va nous permettre d'appliquer une fonction javaScript pour chaque document.

On peut utiliser le foreach() pour : Mise à jour de document.

Suppression de document.

Manipulation de document.

Dans la function on fait appelle au print() de javascript pour avoir le bon affichage .

Exemple : db.collection.find().sort({_id :{1}) Trie croissant

Db.collection.find().sort(_id :{-1})trie décroissant

```
db.cours.find({determinant:true}).sort({_id:1}).forEach(
   function(p){
      print(p._id)
})
```

2. *Affiche avec un aggregate.*

Définition : $match filtre les documents pour transmettre les documents qui correspondent à la prochaine étape du pipeline.

Procédé :

Pour cette manière, nous avons utilisé un match pour récupérer que les cours à true

Ensuite grâce au $sort j'ai trié les _id par en ordre croissant.

Et en dernier lieux j'ai fait un $project pour n'afficher que les _id.

```
db.cours.aggregate ([
    {$match:{determinant:true}},
```

```
{$sort:{_id :1}},
 {$project:{_id:1}}
   ])
```

6. *Liste des cours n'ayant aucun prérequis.*

   Réponse :

   db.cours.find({prerequis:[]},{id:1})

   Procédé : Pour cet exercice j'ai utilisé un simple find() dans le premier paramètre

   J'ai demandé à récupérer tous les cours qui n'ont aucun prérequis et dans le deuxième paramètre d'afficher seulement l'id de ligne dont les prérequis sont vides.

7. *Liste des cours ayant le plus de prérequis directs avec le nombre de prérequis.*

   Définition : $cond permet d'évaluer une expression boolean pour renvoyer une des deux valeurs. $cond requiert les 3 arguments (if-then-else).

   $isArray détermine si l'opérande est un tableau. Si c'est le cas elle renvoie true.

   Procédé : Tout d'abord j'ai créé une variable max que j'utiliserai dans la méthode foreach. Dans le $project je crée un nouveau champ (numberPrerequis)qui va prendre dans chaque cours le nombre max de prérequis direct. Ensuite dans le foreach je compare numberPrerequis à la variable max que j'ai créé et l'ai mise à 0.

   Si le numberPrerequis est plus grand que le max à ce moment-là, la valeur max change et devient la plus grande valeur. Ensuite, j'affiche l'id et le nombre de prérequis du cours concerné.

   Réponse :

```
var max = 0;
db.cours.aggregate([
  {
 $project: {
numberPrerequis: { $cond: { if: { $isArray: "$prerequis" }, then: { $size: "$prerequis" }, else: "NA"} },
    }
  }
]).forEach(function(p){
   if (p.numberPrerequis > max) {
     max = p.numberPrerequis;
```

```
        print(p._id + " avec " +max+" prerequis");
    }
    })
```

8. *Pour l'examen « EIVTE1920_1S» calculer la moyenne des résultats*

   Définition :

   $first : Retourne la valeur d'une expression au premier document d'un groupe de document qui partage le même groupe par clé.

*Procédé : dans le "aggregate", j'ai utilisé le $group pour récupérer dans le champ id les résultats des notes de chaque élève. Ensuite, grâce à l'expression $first qui m'a permis de récupérer la note totale de l'examen, cela va me permettre de le diviser dans la partie du $Project. Dans le "project", je crée une noteMoyenne qui va contenir le résultat de l'opération suivante : noteMoyenne=(moyenneResultatObtenueDeChaqueEleve*100)/NoteMaximumExamen*

Réponse :

```
db.examens.aggregate([
  {$group : {_id : '$resultats.note',totalss:{$first:'$total'}} },
  {$project : {_id:0,noteMoyenne : {$divide :[ {$multiply : [{$avg : '$_id'},100]},'$totalss']}} }
])
```

9. *Pour l'examen « EIVTE1920_1S » afficher toutes les réussites*

   Le résultat devra être enregistré dans une nouvelle collection « result » au moyen du dernier étage du pipeline.

   Définition :

   $unwind : permet d'éclater le vecteur ce qui signifie que chaque document du vecteur va être séparé.

   Par exemple($unwid) : db.cours.aggregate(

```
[
{$unwind : ''$prerequis''},
{$match :{_id : ''4IPAC''}}
]
)
```

   Résultat :

{ "_id" : "4IPAC", "determinant" : true, "intitule" : "Projet d'analyse et de conception", "nbPeriodes" : 100.0, "ects" : 10.0, "prerequis" : "4ITGP" }

{ "_id" : "4IPAC", "determinant" : true, "intitule" : "Projet d'analyse et de conception", "nbPeriodes" : 100.0, "ects" : 10.0, "prerequis" : "4IPAI" }

Exemple sans $unwind :

```
db.cours.aggregate(
[
{$match :{_id : ''4IPAC''}}
]
)
```

Résultat :

{ "_id" : "4IPAC", "determinant" : true, "intitule" : "Projet d'analyse et de conception", "nbPeriodes" : 100.0, "ects" : 10.0, "prerequis" : ["4ITGP", "4IPAI"] }

$out : prend le document retourner dans le pipeline de l'agrégation et met les informations dans une nouvelle collection que nous avons nommer.

$out doit absolument se retrouver au dernier étage du pipeline.

Réponse :

```
db.examens.aggregate([
   {$unwind: "$resultats"},
{$match : {"resultats.note":{$gt:20, $not:{$lt:20}}}},
   {$group : {_id : '$resultats',total:{$first:'$total'}} },

   {$project : { cours:'4VTE',_id: '$_id.etudiant', note : {$divide :[ {$multiply : [{$avg : '$_id.note'},100]},'$total']}} },
     {$sort:{_id:1}},
     { $out : "result" }
])
```

10. Pour un étudiant donné, affichez la liste des cours qu'il a réussis avec plus de 70 %.

Définition : $push permet de regrouper des documents dans un seul et même groupe.

Par exemple : Avoir les réussites dans le groupe resPlus70.

$gt et $lt accumulateur qui permet de dire que l'on recherche des valeurs qui sont soit plus grande ($gt) ou plus petite ($lt).

```
db.etudiants.aggregate([
```

```
{$unwind: '$reussites'},
{$match : {"_id":"ET3","reussites.note":{$gt:70, $not:{$lt:70}}}},
{$group: {_id:'ET3', resPlus70: {$push: '$reussites'}}}, ])
```

3. Référence

Site : MongoDb : https://docs.mongodb.com/manual/

Syllabus VTE