



Bases de Données Avancées

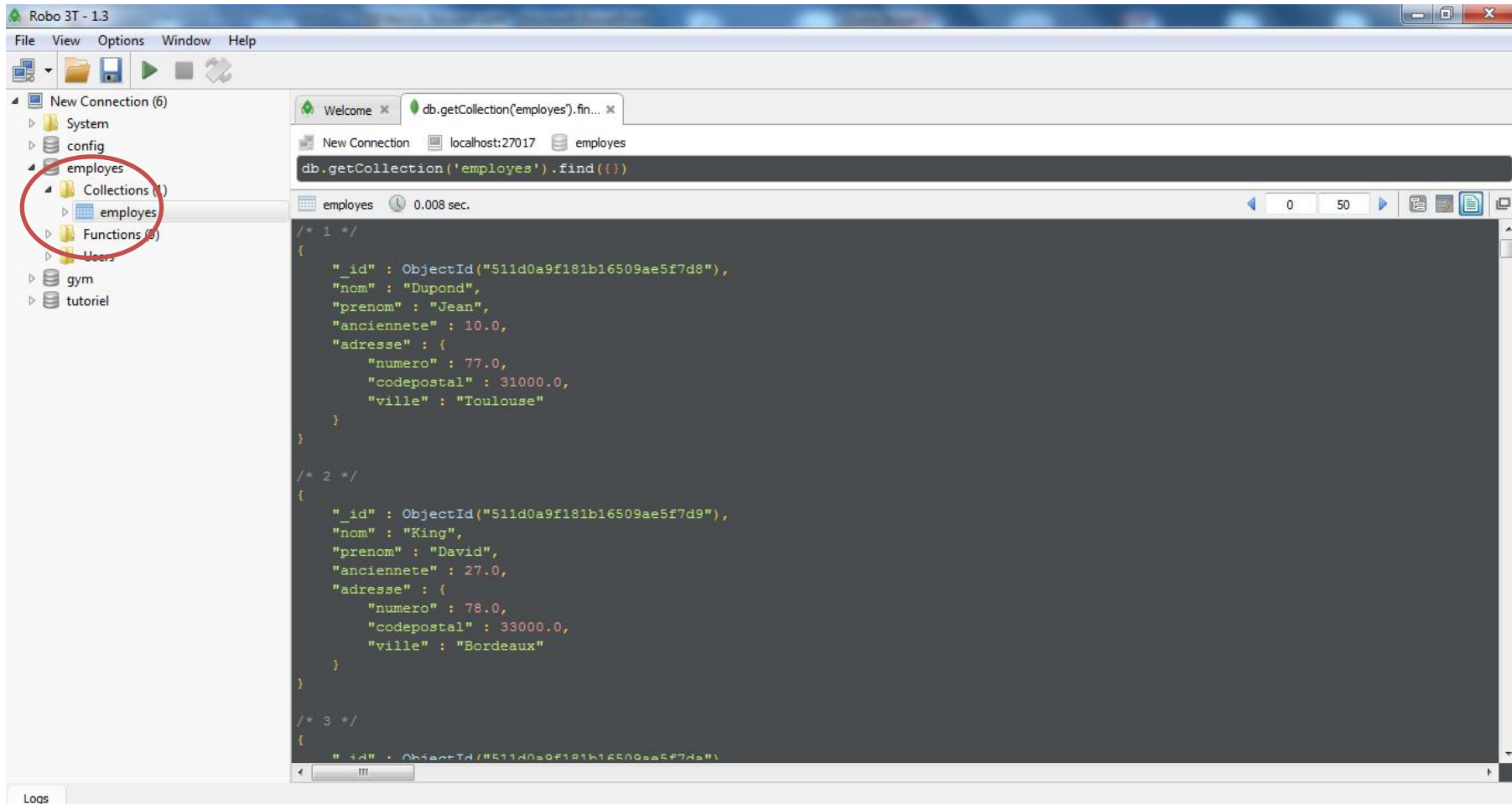
TP Révision : MongoDB

- USTHB Master 01 IL-
M. AZZOUZ

Dernière mis à jour :Juillet 2020

TP Révision

Vue d'ensemble sur la BD employe:



TP Révision

Collection employees:

Robo 3T - 1.3

File View Options Window Help

New Connection (6)

- System
- config
- employees
 - Collections (1)
 - employees
 - Functions (0)
 - Users
- gym
- tutorial

Welcome x db.getCollection('employees').fin... x

New Connection localhost:27017 employees

```
db.getCollection('employees').find({})
```

employees 0.008 sec.

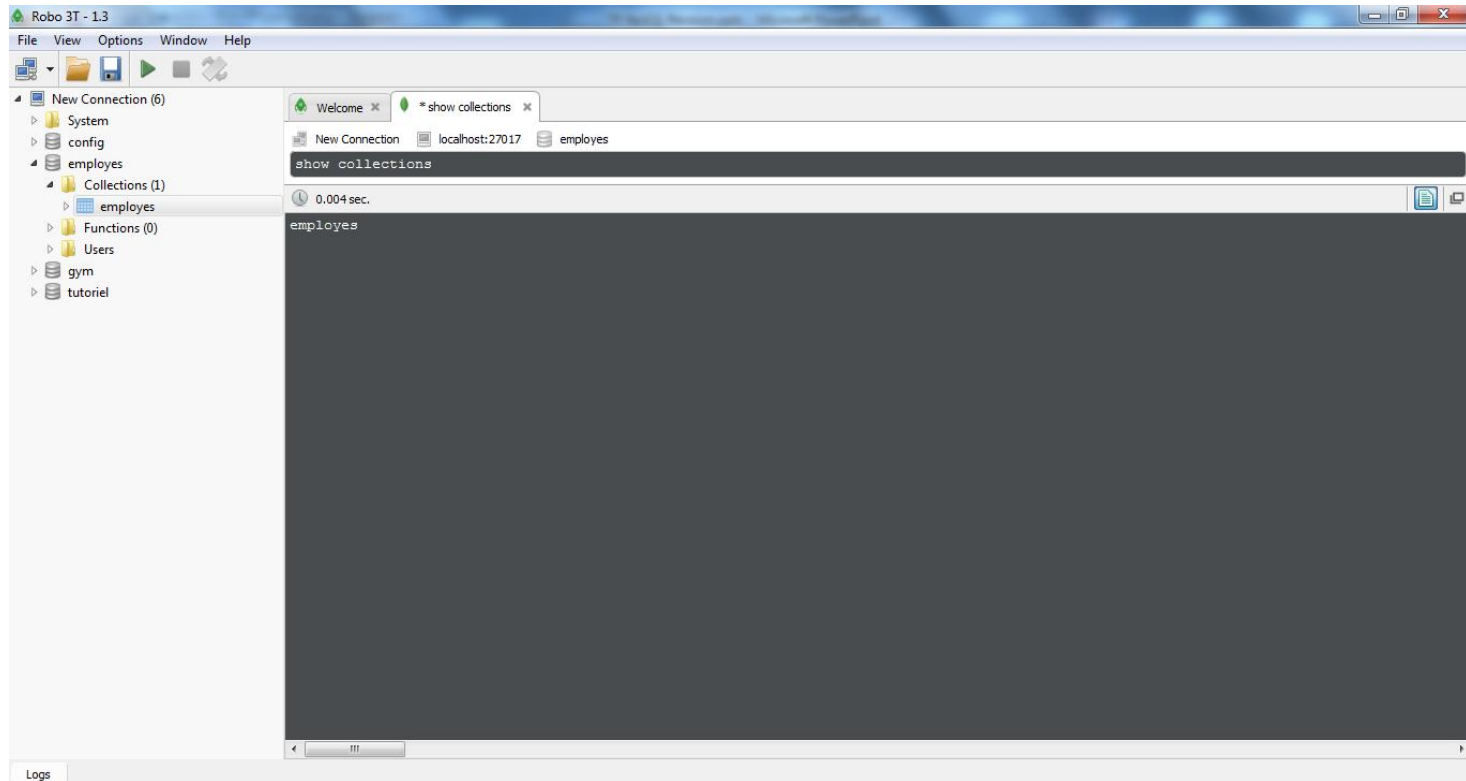
Key	Value	Type
(1) ObjectId("511d0a9f181b16509ae5f7d8")	{ 5 fields }	Object
(2) ObjectId("511d0a9f181b16509ae5f7d9")	{ 5 fields }	Object
(3) ObjectId("511d0a9f181b16509ae5f7da")	{ 5 fields }	Object
(4) ObjectId("511d0a9f181b16509ae5f7db")	{ 5 fields }	Object
(5) ObjectId("511d0a9f181b16509ae5f7dc")	{ 6 fields }	Object
(6) ObjectId("511d0a9f181b16509ae5f7dd")	{ 7 fields }	Object
(7) ObjectId("511d0a9f181b16509ae5f7de")	{ 5 fields }	Object
(8) ObjectId("511d0a9f181b16509ae5f7df")	{ 5 fields }	Object
(9) ObjectId("511d0a9f181b16509ae5f7e0")	{ 5 fields }	Object
(10) ObjectId("511d0aa0181b16509ae5f7e1")	{ 5 fields }	Object
_id	ObjectId("511d0aa0181b16509ae5f7e1")	ObjectId
nom	Motin	String
prenom	Roger	String
anciennete	2.0	Double
adresse	{ 4 fields }	Object
numero	67.0	Double
rue	Jean Moulin	String
codepostal	31000.0	Double
ville	Toulouse	String
(11) ObjectId("511d0aa0181b16509ae5f7e2")	{ 6 fields }	Object
(12) ObjectId("511d0aa0181b16509ae5f7e3")	{ 7 fields }	Object
(13) ObjectId("511d0aa0181b16509ae5f7e4")	{ 5 fields }	Object
(14) ObjectId("511d0aa0181b16509ae5f7e5")	{ 5 fields }	Object
(15) ObjectId("511d0aa0181b16509ae5f7e6")	{ 5 fields }	Object

Logs

TP Révision

a. afficher toutes les collections de la base

➤ **Réponse :** show collections



TP Révision

b. afficher tous les documents de la base

➤ **Réponse :**

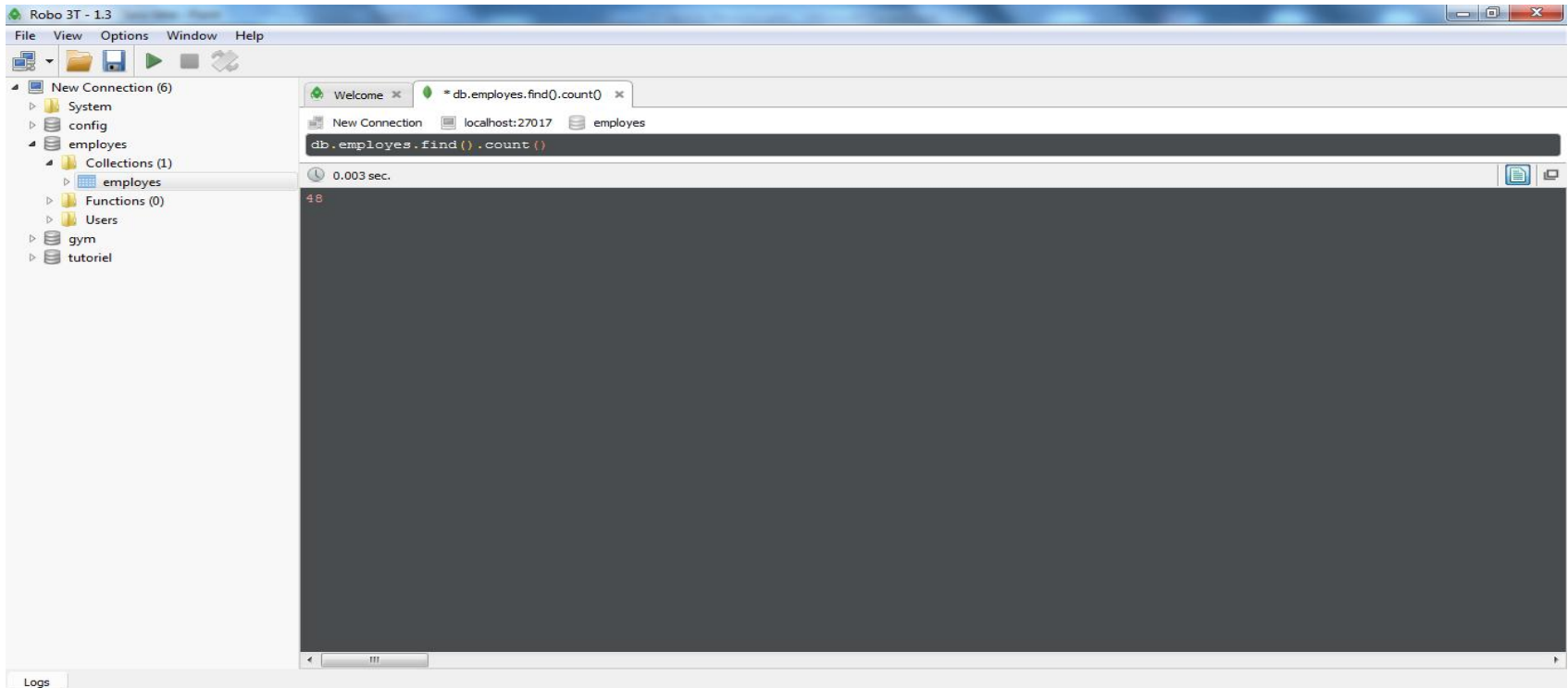
```
var coll = db.getCollectionNames()
for ( var i = 0; i < coll.length ; i ++ )
{
  db.getCollection(coll[i]).find()
}
```

TP Révision

c. compter le nombre de documents de la collection employees

➤ Réponse :

```
db.employees.find().count()
```



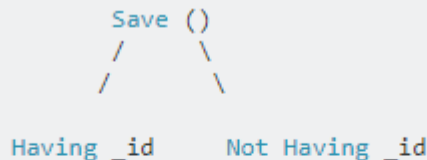
TP Révision

d. insérer de deux manières différentes deux employés avec les champs nom, prénom et soit prime soit ancienneté

➤ Réponse :

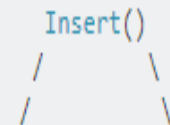
```
db.employees.insert ({ "nom" : "Alan", "prenom" : "Joe",  
"anciennete" : 10 });
```

```
db.employees.save({ "nom" : "Wick", "prenom" : "John",  
"prime" : 150 });
```



-> In this case save will do upsert to insert. Now what that means, it means take the document and replace the complete document having same _id.

-> It will do normal insertion in this case as insert() do.



-> E11000 duplicate key error index:

-> Insert a new doc inside the collection.

TP Révision

e. afficher la liste des employés dont le prénom est David

➤ Réponse :

```
db.employees.find (  
  { "prenom" : "David"},  
  { "_id": 0 });
```


TP Révision

f. afficher la liste des employés dont le prénom commence ou se termine par D

➤ Réponse :

```
db.employees.find({$or: [{ "prenom" :/^D/},  
{"prenom" :/d$/}]})
```

Ou bien

```
db.employees.find({"prenom" :/^D.* | .*d$/})
```

TP Révision

g. afficher la liste des personnes dont le prénom commence par D et contient exactement 5 lettres

➤ Réponse :

```
db.employees.find({"prenom":/^D/}). forEach
(  
function (p){  
if(p.prenom.length == 5){  
print (p)  
}})
```

Ou bien

```
db.employees.find({"prenom":/^D[a-z]{4}$/});
```

TP Révision

h. afficher la liste des personnes dont le prénom commence et se termine par une voyelle

➤ **Réponse :**

```
db.employees.find(  
{  
  "prenom": /^[AEIOUY].*[aeiouy]$/  
}) ;
```

TP Révision

- i. afficher la liste des personnes dont le prénom commence et se termine par une même lettre

➤ Réponse :

```
db. employees.find().forEach(function (p){  
  pre = p.prenom.toLowerCase ();  
  if( pre[0] == pre

```
pre.length-1
```

]){  
    print ( p )  
  }  
})
```

TP Révision

j. afficher les nom et prénom de chaque employé ayant une ancienneté > 10

➤ Réponse :

```
db.employees.find (  
  { "anciennete" :{ $gt :10}} ,  
  
  { "_id" :0, "nom" :1, "prenom" :1}  
);
```

TP Révision

k. afficher les nom et adresse complète des employés ayant un attribut rue dans l'objet adresse

➤ Réponse :

```
db. employes.find (
```

```
  {"adresse.rue": { $exists : true } },
```

```
  { "nom" :1, "adresse" :1}
```

```
);
```

TP Révision

1. incrémenter de 200 la prime des employés ayant déjà le champ prime

➤ Réponse :

```
db.employees.updateMany (  
  { "prime" :{ $exists : true }},  
  { $inc :{ "prime" :200}});
```

Ou bien

```
db.employees.update (  
  { "prime" :{ $exists : true }},  
  { $inc :{ "prime" :200}} ,  
  { multi : true })
```

TP Révision

m. afficher les trois premières personnes ayant la plus grande valeur d'ancienneté

➤ **Réponse :**

```
db.employees.find({"anciennete":  
db.employees.aggregate([  
{$group: {_id: null,  
"anciennetemax": {$max:"$anciennete"}}}  
]).next().anciennetemax  
},  
{"_id":0}).limit (3) ;
```


TP Révision

n. regrouper les personnes dont la ville de résidence est Toulouse (afficher nom, prénom et ancienneté)

➤ **Réponse :**

```
db.employees.find (
```

```
  {"adresse.ville": "Toulouse"},
```

```
  { "_id" :0, "nom" :1, "prenom" :1, "anciennete" :1}  
);
```

TP Révision

o. afficher les personnes dont le prénom commence par M et la ville de résidence est soit Foix soit Bordeaux

➤ **Réponse :**

```
db. employes . find (  
  {"prenom" : /^M/,  
  "adresse.ville": {$in : ["Foix", "Bordeaux"]}  
});
```

TP Révision

p. mettre à jour l'adresse de Dominique Mani : nouvelle adresse (numero : 20, ville : 'Marseille', codepostal : '13015'). Attention, il n'y aura plus d'attribut rue dans adresse

➤ **Réponse :**

```
db.employees.update (  
  { "prenom" : "Dominique", "nom" : "Mani"},  
  { $set : { "adresse.numero" : 20,  
    "adresse.ville" : "Marseille",  
    "adresse.codepostal" : "13015"},  
    $unset : { "adresse.rue" : 1 }  
  });
```

TP Révision

q. attribuer une prime de 1 500 à tous les employés n'ayant pas de prime et dont la ville de résidence est différente de Toulouse, Bordeaux et Paris.

➤ Réponse :

```
db.employees.updateMany (
```

```
  {"adresse.ville":{"$nin":["Paris","Toulouse","Bordeaux"]},  
  "prime" :{ $exists : false }  
},
```

```
  { $set :{ "prime" :1500}}  
);
```

TP Révision

- r. remplacer le champ téléphone, pour les documents ayant un champ tel), par un tableau nommé téléphones contenant la valeur du champ tel(le champ tel est à supprimer)

➤ **Réponse :**

```
db.employees.find ({ "tel" : { $exists : true } }).forEach (
function (p) {
db.employees.update(
{ "_id" : p._id },
{$push : { "telephones" : p.tel },
$unset : { "tel" : 1 }}
)
})
```

TP Révision

s. créer un champ prime pour les documents qui n'en disposent pas et de l'affecter à 100 * nombre de caractère du nom de la ville

➤ Réponse :

```
db.employees.find ({ "prime" : { $exists :false}}) . forEach  
(function ( p ){  
  db.employees.update (  
    { "_id" : p._id },  
    { $inc :{ "prime" : 100*p.adresse.ville.length }}  
  ));
```

TP Révision

- t. créer un champ mail dont la valeur est égale soit à `nom.prénom@formation.fr` pour les employés ne disposant pas d'un champ téléphone, soit à `prénom.nom@formation.fr` (nom et prénom sont à remplacer par les vraies valeurs de chaque employé)

➤ **Réponse :**

```
db.employees.find().forEach (
function (p) {
email = p.nom + '.' + p.prenom + '@formation.fr ';
if (p.telephones)
email = p.prenom + '.' + p.nom + '@formation.fr ';
db.employees.update(
{ "_id" :p._id },
{ $set :{ "mail" : email }}}
));
```

TP Révision

u. calculer et afficher la somme de l'ancienneté pour les employés disposant du même prénom

➤ Réponse :

```
db.employees.aggregate(  
  { $group : { "_id" : "$prenom",  
    "ancienneteCum" : { $sum : "$anciennete" } } },  
  { $sort : { "_id" : 1 } }  
);
```