Les requêtes de recherche en MongoDB

Affichage de documents

- MongoDB propose deux types de requêtes simples **find**() et **findOne()**:

  - **findOne()**: Permet de sélectionner **le premier document d'une collection**

```
db. le_nom_de_la_collection.findOne()
```

```
db.Stagiaires.findOne()
```

  - **find() :** Sélectionner **tous les documents d'une collection**

```
db. le_nom_de_la_collection.find()
```

```
db.Stagiaires.find()
```

Les paramètres de find()

- **find()** : possède deux paramètres comme suit:

```
db.MaCollection.find(restriction, projection)
```

- Où :

1. **Restriction**: est une (des) **conditions de filtrage** des documents
2. **Projection** : spécifier les champs qui vont être affichés ou ceux qui ne vont pas être affichés

Restrictions dans la sélection

- La restriction permet de sélectionner les documents qui vérifient **une condition**.
- La condition est un **objet JSON** avec les critères de sélection des documents.

```
db.le_nom_de_la_collection.find (    { "champ1":"val1","champ2":"val2" }  )
```

- **find()** : Sélectionne **les documents** dont les valeurs des champs (champ1, champ2) sont respectivement **val1** et **val2**

- **findOne() :** Sélectionne **le premier document** dont les valeurs des champs (champ1, champ2) sont respectivement **val1** et **val2**

```
db.le_nom_de_la_collection.findOne( {"champ1":"val1","champ2":"val2" } )
```

Exemple1

```
db.Stagiaires.findOne({"nom":"Alami"})
```

```
{
    "_id" : ObjectId("62e0249adc15f0d3b0bc2838"),
    "nom" : "Alami",
    "prenom" : "Amina",
    "filiere" : {
        "_id" : "DD",
        "intitule" : "Développement digital"
    },
    "moy1A":14.50",
    "niveau" : "2A",
    "option" : "Mobile"
}
```

Exemple2

- Sélectionner les stagiaires qui ont le nom égal à Alami

```
db.Stagiaires.find({"nom":"Alami"})
```

Projection

- La projection permet de limiter les informations retournées en précisant les champs souhaités dans un document JSON passé en paramètre à la requête **find** ou **findOne**:

- **find()** :Sélectionne les champs (champ1, champ2) de tous les documents d'une collection

```
db.nom_de_la_collection. find ( {} , { "champ1": 1,"champ2": 1 } )
```

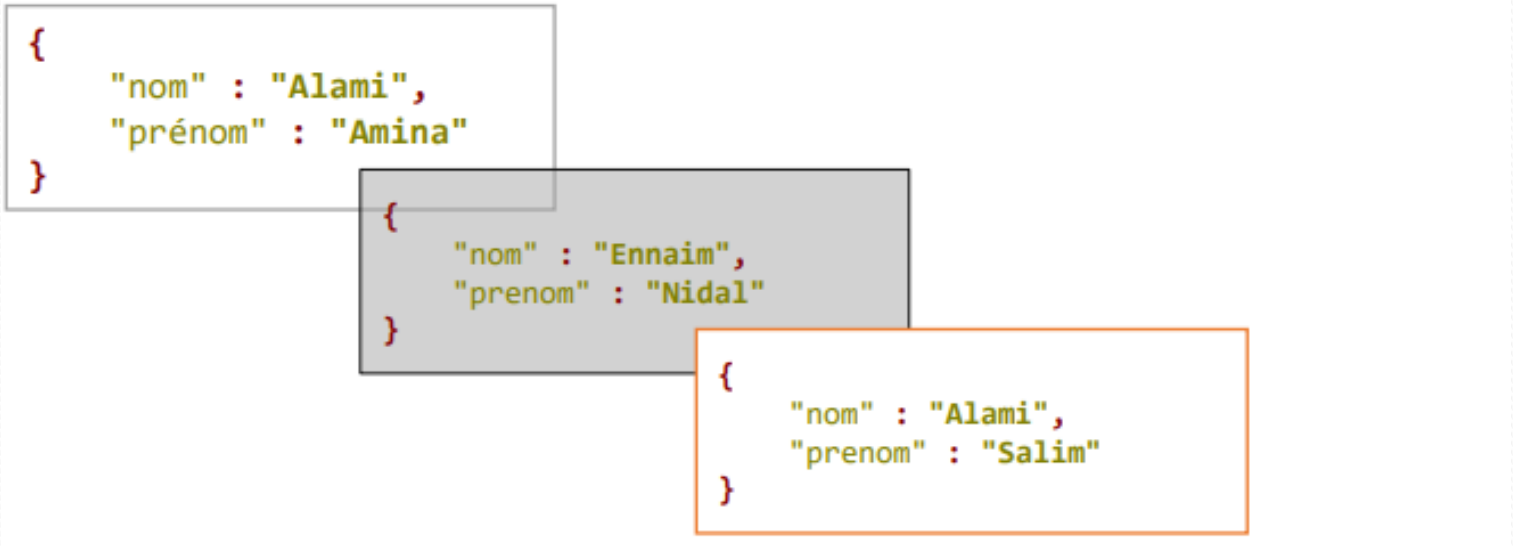
Signifie qu'il n'y a pas de condition

- 1 : signifie que **le champ doit figurer** dans la sélection
- 0: signifie que **le champ ne doit pas figurer** dans la sélection, c'est la valeur par défaut sauf pour l'_id
- Remarque: ce qui est dit pour find() et aussi valable pour findOne()

Exemple

- Sélectionner les nom et prénom des stagiaires:

```
db.Stagiaires.find({},{"nom":1,"prenom":1,"_id":0})
```



Exemple

- Sélectionner les prénoms et les options des stagiaires dont le nom est "Alami" :

```
db.Stagiaires.find({"nom": "Alami"},{"prenom":1,"option":1,"_id":0})
```

```
{
    "prenom" : "Amina",
    "option" : "Mobile"
}
```

```
{
    "prenom" : "Salim",
    "option" : "FullStack"
}
```

Exemple

- On peut créer les deux fichiers JSON de la sélection et la projection en dehors de la requête :

```
projection = {"prenom":1,"option":1,"_id":0}
selection = {"nom":"Alami"}
db.Stagiaires.find(selection,projection)
```

```
{
    "prenom" : "Amina",
    "option" : "Mobile"
}
```

```
{
    "prenom" : "Salim",
    "option" : "FullStack"
}
```

Exemple

- Sélectionner les noms et prénoms des stagiaires de l'option "Mobile ":

```
projection = {"nom":1,"prenom":1,"_id":0}
sélection = {"option": "Mobile"}
db.Stagiaires.find(sélection , projection)
```

```
{
    "nom" : "Alami",
    "prénom" : "Amina"
}
```

Sélectionner un sous champ

- Pour spécifier un sous champ d'un champ, il est nécessaire d'utiliser le formalisme : champ.souschamp

```
projection = {"nom":1,"prenom":1,"_id":0}
sélection = {"filiere.intitule": "Developpement digital"}
db.Stagiaires.find(sélection , projection)
```

```
{
    "nom" : "Alami",
    "prénom" : "Amina"
}
```

```
{
    "nom" : "Alami",
    "prénom" : "Salim"
}
```

Fonction COUNT()

- Une fonction très utile pour faire des **dénombrements** suite à des **sélections et connaitre la taille du résultat** ,
- **Count()** s'ajoute à la suite d'une fonction **find** et **retourne le nombre de documents renvoyés :**

```
db.Stagiaires.find().count()
```

```
4  // nombre total de Stagiaires dans la collection
```

```
db.Stagiaires.find({"niveau":"2A"}).count()
```

```
3  // nombre de stagiaires en deuxième années
```

Fonction LIMIT()

- Cette fonction permet de **limiter le nombre de documents** renvoyés par la fonction **find**,
  - limit(n) s'ajoute à la suite d'une fonction find et retourne les **n** premiers documents résultats,
  - limit(), sans spécification du nombre n, retourne tous les documents renvoyés par find

```
db.Stagiaires.find({},{"nom":1,"prenom":1,"_id":0}).limit(2)
```

```
{
    "nom" : "Alami",
    "prenom" : "Amina"
}
{
    "nom" : "Ennaim",
    "prenom" : "Nidal"
}
```

La fonction DISTINCT

- Une fonction très utile pour **lister les valeurs** prises par le champ de la collection passé en paramètre,
- **distinct()** retourne les valeurs prises par le champ sous forme de tableau

```
db.Stagiaires.distinct("option")
```

```
[
    "Cyber Security",
    "FullStack",
    "Mobile"
]
```

La fonction DISTINCT

- On utilise toujours le même formalisme **champ.souschamp** pour lister les valeurs d'un sous champ:

```
db.Stagiaires.distinct("filiere.intitule")
```

```
[
    "Développement digital",
    "Infrastructure digitale"
]
```

Les opérateurs de comparaison

- On a souvent besoin de faire des condition à l'aide d'opérateurs de comparaison:

Operateur	Description
$eq	***Equal*** : =
$gt	***Greater Than*** : >
$gte	***Greater Than or Equal*** : >=
$lt	***Less Than*** : <
$lte	***Less Than or Equal*** : <=
$ne	***Not Equal*** : ≠

Exemple d'utilisation d'opérateur de comparaison

- Sélectionner les stagiaires dont la moyenne de la première année est supérieure ou égale à 14.00:

```
selection = {"moy1A":{"$gte":14.00}}

db.Stagiaires.find(selection,{"nom":1,"prenom":1,"_id":0})
```

```
{
    "nom" : "Alami",
    "prenom" : "Amina"
}
{
    "nom" : "Ennaim",
    "prenom" : "Nidal"
}
```

Exemple d'utilisation d'opérateur de comparaison avec une date

- On peut aussi vérifier une date en mongodb à l'aide d'opérateurs de comparaison,

```
db.stagiaires.find( {"datenaissance":{"$gt": new Date(2003,05,14)} } )
```

Opérateurs de comparaison de liste

- Équivaux successivement à : IN et NOT IN de SQL

Operateur	Description
$in	Appartient à la liste
$nin	N'appartient pas à la liste

Exemple

- Sélectionner les stagiaires dont l'option est FullStack ou Cyber Security:

```
selection = {"option":{"$in":["FullStack","Cyber Security"]}}

db.Stagiaires.find(selection,{"nom":1,"prenom":1,"_id":0})
```

```
{
    "nom" : "Ennaim",
    "prenom" : "Nidal"
}
{
    "nom" : "Alami",
    "prenom" : "Salim"
}
```

Les opérateurs logiques

- Les opérateurs logiques qu'on a besoin dans les conditions de sélection :

Operateur	Description
$and	ET Logique
$or	OU Logique
$nor	Not (OU Logique)

Exemple

- Sélectionner les stagiaires qui ont une moyenne de la première année supérieure ou égale à 15.00 ou bien qui ont choisi l'option est Mobile:

```
selection = { "$or": [ { "moy1A" : { "$gte" : 15 } }, { "option": "Mobile" } ] }

db.Stagiaires.find(selection,{"nom":1,"prenom":1,"_id":0})
```

```
{
    "nom" : "Alami",
    "prenom" : "Amina"
}
```

```
{
    "nom" : "Ennaim",
    "prenom" : "Nidal"
}
```

L'opérateur $not

- Il sélectionne les documents qui **ne correspondent pas aux expressions de la requête**. Cela inclut les documents qui ne contiennent pas le champ.

- ***Exemple:***

```
db.stagiaires.find( { "Moy1A": { $not: { "$lt": 10 } } } )
```

L'opérateur exists

- Permet de vérifier l'existence ou non d'un champ, on lui donne comme valeur, un booléen **true** ou **false**:

Operateur	Description
$exists	Teste la présence ou non d'un champ

Exemple

- Sélectionner les stagiaires qui n'ont pas d'option :

```
selection = { "option": {$exists : false }}

db.Stagiaires.find(selection,{"nom":1,"prenom":1,"_id":0})
```

```
{
    "nom" : "Dalil",
    "prenom" : "Karima"
}
```

Les expressions régulières

- Les expressions régulière nous permettent de séléctionner des documents dont certains champs correspondant à un certain format.

```
db.Stagiaires.find( { "champ1": {"$regex" : "LeModele" } } )
```

- **Lemodèle**: est le format qu'on souhaite obtenir pour le champ spécifié.
- Les symboles des expressions régulières en MongoDB sont les mêmes que celles utilisées dans les autres langages.

Exemples

- Sélectionner les stagiaires qui ont un nom commençant par E:

```
db.Stagiaires.find( { "nom": {"$regex" : "^E" } } )
```

- Sélectionner les stagiaires qui ont un nom **commençant par E et se termine par r**

```
db.Stagiaires.find( { "nom": {"$regex" : "^E.*r$" } } )
```

Trier une sélection selon un champ

- On utilise pour cela la fonction sort() :

```
db.Stagiaires.find().sort( { "nom":  1 })
```

- 1 : signifie que le trie est ascendant
- -1 : signifie que le trie est descendant