

# Cours de Bases de données NOSQL (MongoDB)

[madaniabdellah@gmail.com](mailto:madaniabdellah@gmail.com)

# Plan du cours

- Rappels :
  - JSON
  - Bases de Données Relationnelles
- Introduction : Mouvement NoSQL
- MongoDB
  - Caractéristiques de MongoDB
  - **Requêtes avec MongoDB**
  - MongoDB et Java
  - MongoDB et PHP
  - Réplication et reprise sur panne dans MongoDB
  - Partitionnement dans MongoDB
  - TextSearch

## HOW TO WRITE A CV



# MongoDB

- `use base_donnee` : crée une base de données si elle n'existe pas, sinon y accède
- `db` : affiche la base de données courante
- `show dbs` : affiche la liste des bases de données
- `db.dropDatabase()` : supprime la base de données courante

`use mabase`

`db`

`Show dbs`

# MongoDB

- `db.createCollection(col)` : crée la collection « col »
- `show collections` : liste des collections
- `db.col.drop()` : supprime la collection « col »

`use mabase`

`db.createCollection(«livres »)`

`show collections`

`db.livres.drop()`

`show collections`

# MongoDB

- `db.collection.insert( document )` : insère un document dans une collection

```
db.etudiants.insert(  
  {  
    'prenom': 'Camille',  
    'nom': 'Simon'  
  }  
)
```

# MongoDB

- `db.collection.find()` : tous les documents sont renvoyés
- `db.collection.find(requête)` : requête est un tableau clefs / valeurs spécifiant des opérateurs sur les champs des documents recherchés
- `db.collection.find(requête, projection)` : projection est un tableau permettant de limiter les champs que l'on souhaite consulter dans les documents recherchés

# MongoDB

- `db.collection.findOne()` : fait la même chose que `db.collection.find()` mais sans s'embarrasser d'un curseur, lorsqu'on souhaite récupérer un document unique (par son identifiant, par exemple).

- Remarque :

La méthode `find()` retourne un curseur même si elle retourne une seule valeur

# MongoDB

```
db.etudiants.find()
```

```
db.etudiants.find( { 'prenom': 'Camille' } )
```

```
db.etudiants.find( { 'nom': /^i/ } )
```

```
db.etudiants.find( { 'nom': /i$/ } ) .count()
```

```
db.etudiants.findOne()
```

```
db.etudiants.find( { ville: 'casa' }, { _id: 0 } ).sort( { moyenne: -1 } )
```

```
db.etudiants.find( { }, { _id: 0 } ).sort( { moyenne : -1 } ).limit(5)
```

```
db.etudiants.find( { }, { _id: 0 } ).sort( { moyenne: 1 } ).skip(10).limit(5)
```



# MongoDB

Opération	Syntax	Example	RDBMS Equivalent
=	{<key>:<value>}	db.mycol.find({"by":"tutorials point"}).pretty()	where by =
<	{<key>:{\$lt:<value>}}	db.mycol.find({"likes":{\$lt:50}}).pretty()	where likes < 50
<=	{<key>:{\$lte:<value>}}	db.mycol.find({"likes":{\$lte:50}}).pretty()	where likes <= 50
>	{<key>:{\$gt:<value>}}	db.mycol.find({"likes":{\$gt:50}}).pretty()	where likes > 50
>=	{<key>:{\$gte:<value>}}	db.mycol.find({"likes":{\$gte:50}}).pretty()	where likes >= 50
!=	{<key>:{\$ne:<value>}}	db.mycol.find({"likes":{\$ne:50}}).pretty()	where likes != 50

# MongoDB

- `db.mycol.find( {key1:value1, key2:value2} ).pretty()` : la virgule représente le et logique
- `db.mycol.find( { $and:[ {key1: value1}, {key2:value2} ] } ).pretty()` : `$and` représente le et logique
- `db.mycol.find( { $or:[ {key1: value1}, {key2:value2} ] } ).pretty()` : `$or` représente le ou logique

```
db.mycol.find( { "moyenne": { $gt: 10 }, $or: [ { "prenom": "Mohamed" }, { "ville": "Casa" } ] } ).pretty()
```

# MongoDB

- `Db.mycol.find( {key : { $exists : true | false } })` : vérifie l'existence d'un champ
- `Db.mycol.find( {key : { $in : [valeur1, valeur2, ...] } })` : vérifie si une valeur existe dans un tableau d'élément

1. `Db.employees.find( {email : { $exists : true } })` : les employés ayant un email
2. `Db.employees.find( {ville : { $in : ['Casa', 'Rabat', 'Fes'] } })` : les employés de Casa, Rabat ou Fes.

# MongoDB

- `db.mycol.update(critere, donnée)` : est utilisée pour modifier un document.

```
db.mycol.update( {'title':'MongoDB Overview'}, {$set: {'title':'New MongoDB Tutorial'}} )
```

```
db.mycol.update( {'title':'MongoDB Overview'}, {$set: {'title':'New MongoDB Tutorial'}} , {multi:true} )
```

```
db.mycol.update( {}, {$inc : {salaire : 400}} , {multi:1} )
```

```
db.mycol.update( {}, {$push : {loisirs : "sport"}} , {multi:1} )
```

```
db.mycol.update( {}, {$rename: {loisirs : "hobbies"}} , {multi:1} )
```

```
db.mycol.update( {}, {$unset : {loisirs : 1}} , {multi:1} )
```

# MongoDB

- `db.mycol.remove(critère)` : supprime des documents d'une collections

`db.mycol.remove( {'title':'MongoDB Overview'} )` : supprime plusieurs documents

`db.mycol.remove( {'title':'MongoDB Overview'}, 1 )` :  
supprime seulement le premier

`db.mycol.remove()` : supprime tous les documents

# MongoDB

- Une agrégation permet de transformer et de combiner des documents dans les collections

```
db.mycol.aggregate([{$group: {_id:'$sexe', total: {$sum:1}}}] )
```

```
db.mycol.aggregate([{$group: {_id:'$sexe', som: {$sum:'$salaire'}}}] )
```

```
db.mycol.aggregate([{$group: {_id:'$sexe', max: {$max:'$salaire'}}}] )
```

```
db.mycol.aggregate([{$group: {_id:'$sexe', min: {$min:'$salaire'}}}] )
```

```
db.mycol.aggregate([{$group: {_id:'$sexe', moy: {$avg:'$salaire'}}}] )
```