## Créer une nouvelle DB ou basculer vers un DB existant

Cette simple commande permet de créer une nouvelle base de données si elle n'existe pas ou de basculer vers la base de données existante. Dans MongoDB, "test" est la base de données par défaut et les utilisateurs utilisent "**tester**”DB une fois que Mongo Shell est connecté.

use DB\_Name

mongos> use DBTest

switched to db DBTest

La commande mentionnée est utilisée pour lister toutes les bases de données.

show dbs

mongo> show dbs

admin 0.000GB

config 0.002GB

DB\_Test 0.000GB

test 0.000GB

## Supprimer la base de données

La commande donnée aide l'utilisateur à supprimer la base de données requise. Exécutez la commande sur le client MongoDB. Veuillez vous assurer de sélectionner la base de données avant d'exécuter la commande drop. Sinon, la valeur par défaut "**tester**" Base de données.

**db.dropDatabase()**

## Créer une collection

Les collections sont similaires aux tables du SGBDR.

La commande Créer une collection se compose de deux paramètres. La collection se compose de zéro ou plusieurs documents. Par conséquent, pour créer une collection, le paramètre obligatoire à utiliser dans la commande est son nom et [paramètre facultatif](https://docs.mongodb.com/manual/reference/method/db.createCollection/) peut inclure le nom des documents, sa taille et son index.

* Créer une collection simple.

**Syntaxe**: db.createCollection(Name)

**Exemple**:

> use stock

switched to db stock

>

> db.createCollection("Articles")

{ "ok" : 1 }

>

> show collections   
Articles

* Créer une collection limitée

Dans ce cas, limitez la taille et le nombre de documents à insérer dans la collection. La collection plafonnée a une propriété pour supprimer les documents les plus anciens pour faire de la place pour les nouveaux documents.

**Syntaxe**:

db.createCollection(Name,{capped : true, size : sizeLimit , max : documentLimit })

**Mise en situation :** Créons une collection plafonnée, insérons un enregistrement et récupérons-le

> db.createCollection("clients",{capped:true,max:1,size:200})

## Supprimer une collection

La commande Drop Collection est similaire à DDL dans le SGBDR. Il acquiert des verrous sur la collection requise jusqu'à l'exécution de la commande. Drop collection supprime la collection de la base de données avec tous les index associés à cette collection. Pour supprimer la collection, la méthode drop () est requise.

Il renvoie true pour une suppression réussie et false en cas d'erreur ou si la base de données n'existe pas.

**Syntaxe**: collectionName.drop()

**Exemple**:

> use stock

switched to test

>

show collections

Articles

>

> db.Articles.drop()

true

>

> db.Articles.drop()

false

## Renommer la collection

MongoDB fournit renameCollection () méthode pour renommer la collection.

**Syntaxe**: db.collectionName.renameCollection(newCollectionName)

## Ajouter un document à la collection

Db.NomColl.InsertOne()

## Modifier un champ à la collection

$set ; permet d’ajouter un nouveau champ à la collection

db.your\_collection.update(

{},

{ $set: {"new\_field": 1} }

)

db.classe.updateMany({"\_id": {$in :[20,30,70]}},{$set : {"frais":920}})

## Supprimer un champ de la collection

db.classe.updateMany({}, {$unset: {"frais":""}});

## Renommer un champ :

db.collection.updateMany({}, {$rename:{"oldField":"newField"}}, false, true)

|  |  |
| --- | --- |
| Mysql | NOSQL |
| Base de données | Base de données |
| Table | collection |
| enregistrements | documents |

Requêtes

|  |  |
| --- | --- |
| Mysql | NOSQL |
| Select \* from classe | db.classe.find()  db.classe.find({},{}) |
| Select \* from classe  Where Effectif=22 | db.classe.find({"Effectif":22},{}) |
| Select Nom\_Classe, Effectif from classe  Where Effectif=22 | db.classe.find({"Effectif":22},{"Nom\_classe":1, "Effectif":1}) |
|  | filtre={"Effectif":22}  Project={"Nom\_classe":1,"Effectif":1}  db.classe.find(filtre , Project) |
| Select count(\*) from classe | db.classe.find().count() |
| Select count(\*) from classe  Where Effectif=22 | db.classe.find({"Effectif":22},{}).count() |
| Select \* from classe  Where Effectif>22 | db.classe.find({"Effectif": {$gt : 22} },{}) |
|  |  |
| Select \* from classe  Where Effectif>=22 | db.classe.find({"Effectif": {$gte : 22} },{}) |
| Select count(\*) from classe  Where Effectif>22 | db.classe.find({"Effectif": {$gt : 22} } , {}).count() |
| Select count(\*) from classe  Where Effectif > = 22 | db.classe.find({"Effectif": {$gte : 22} } , {}).count() |
|  |  |

Mise à jour

|  |  |
| --- | --- |
| Ajouter un champs (frais=850) à la collection, si il n’existe pas, sinon il sera créé | db.classe.updateOne({"\_id":20} , { $set :{"Frias":850}}) |
| Ajouter un champs (frais=900) aux documents dont le code 30,50 ,70 à la collection, si il n’existe pas, sinon il sera créé | db.classe.updateMany({"\_id":{ $in : [30,50,70]}} , { $set :{"Frias":900}}) |
| Lister la classe ayant l’effectif le plus grand | db.classe.find().sort({"Effectif":-1}).limit(1) |
| Lister les classes triées par ordre décroissant suivant l’effectif | db.classe.find({} , {}).sort({"Effectif":-1}) |
| Lister les classe ayant  Effectif=22  Frais >800 |  |
|  |  |
|  |  |
|  |  |