

Bases de données NOSQL et Big data

TP4 : Bases de données NOSQL - mongo db

Diplôme National d'Ingénieur en Informatique

Spécialité :
Génie Logiciel

Réalisée par :
Oussama Ben Slama

Année Universitaire 2024/2025

Table des matières

1	Introduction à MongoDB	2
1.1	Bases de données NoSQL	2
1.2	MongoDB	2
1.3	Commandes MongoDB	2
1.3.1	1. Créer une base de données : shipsdb	2
1.3.2	2. Créer une collection : ships	3
1.3.3	3. Insérer des documents	3
1.3.4	4. Méthodes Find	4
1.3.5	5. Méthodes Update	5
1.3.6	6. Méthodes Remove	6
2	Correction Exercice 1	7
2.1	Afficher toutes les collections de la base	8
2.2	Afficher toutes les collections et tous les documents de la base	9
2.3	Compter le nombre de documents de la collection *employés*	10
2.4	Insérer deux employés	10
2.5	Afficher la liste des employés dont le prénom est David	11
2.6	Afficher la liste des employés dont le prénom commence ou se termine par D	12
2.7	Afficher la liste des personnes dont le prénom commence par D et contient exactement 5 lettres	13
2.8	Afficher les noms et prénoms de chaque employé ayant une ancienneté > 10 ans	14

Chapitre 1

Introduction à MongoDB

1.1 Bases de données NoSQL

Les bases de données NoSQL sont des systèmes de gestion de bases de données qui ne suivent pas le modèle relationnel classique. Elles sont conçues pour gérer de grandes quantités de données non structurées ou semi-structurées.

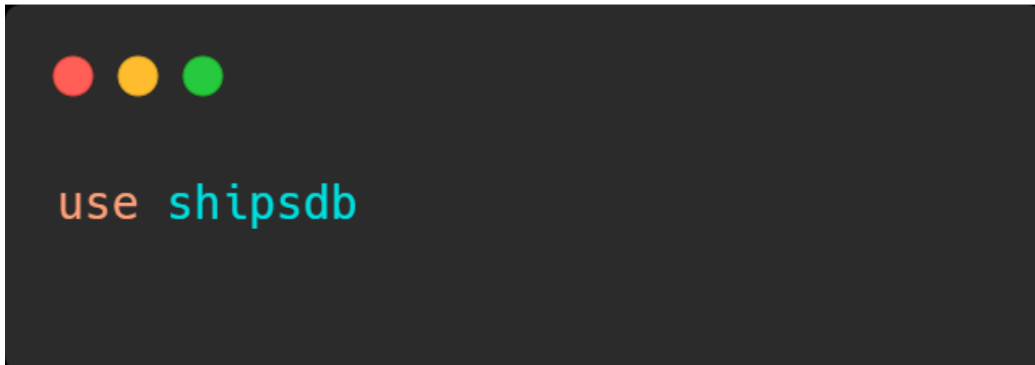
1.2 MongoDB

MongoDB est une base de données NoSQL orientée documents, qui stocke les données sous forme de documents JSON. Elle est populaire pour sa flexibilité, sa capacité à gérer de grands volumes de données, et son support pour les requêtes complexes.

1.3 Commandes MongoDB

1.3.1 1. Créer une base de données : `shipsdb`

Dans MongoDB, nous utilisons la commande `use` pour créer ou basculer vers une base de données.



```
use shipsdb
```

FIGURE 1.1 –

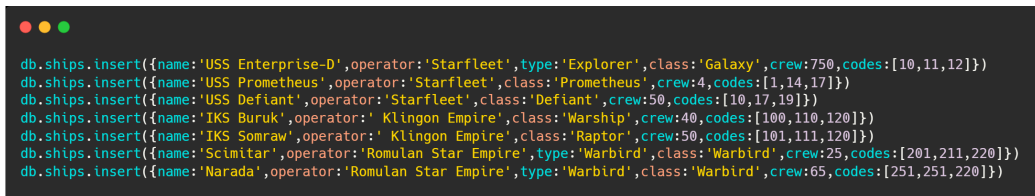
1.3.2 2. Créer une collection : ships



```
db.createCollection("ships")
```

FIGURE 1.2 –

1.3.3 3. Insérer des documents




```
db.ships.insert({name:'USS Enterprise-D',operator:'Starfleet',type:'Explorer',class:'Galaxy',crew:750,code:[10,11,12]})
db.ships.insert({name:'USS Prometheus',operator:'Starfleet',class:'Prometheus',crew:4,code:[1,14,17]})
db.ships.insert({name:'USS Defiant',operator:'Starfleet',class:'Defiant',crew:50,code:[10,17,19]})
db.ships.insert({name:'IKS Buruk',operator:'Klingon Empire',class:'Warship',crew:40,code:[100,110,120]})
db.ships.insert({name:'IKS Somraw',operator:'Klingon Empire',class:'Raptor',crew:50,code:[101,111,120]})
db.ships.insert({name:'Scimitar',operator:'Romulan Star Empire',type:'Warbird',class:'Warbird',crew:25,code:[201,211,220]})
db.ships.insert({name:'Narada',operator:'Romulan Star Empire',type:'Warbird',class:'Warbird',crew:65,code:[251,251,220]})
```

FIGURE 1.3 –

1.3.4 4. Méthodes Find


Trouver tous les documents



```
db.ships.find().pretty()
```

FIGURE 1.4 –


Trouver avec un filtre



```
db.ships.findOne({ name: "Oussama" })
```

FIGURE 1.5 –

Trouver avec des projections (uniquement des champs spécifiques)




```
db.ships.find({}, {name:true, _id:false})
```

FIGURE 1.6 –

1.3.5 5. Méthodes Update


Mettre à jour un document



```
db.ships.updateOne(  
  { name: "Oussama" },  
  { $set: { crew: 20 } }  
)
```

FIGURE 1.7 –

Mettre à jour plusieurs documents




```
db.ships.updateMany(  
  { crew: 20 },  
  { $set: { crew: 10 } }  
)
```

FIGURE 1.8 –

1.3.6 6. Méthodes Remove


Supprimer un document



```
db.ships.deleteOne({ name: "Oussama" })
```

FIGURE 1.9 –


Supprimer plusieurs documents



```
db.ships.deleteMany({ crew: 20 })
```

FIGURE 1.10 –

Supprimer une collection



```
db.ships.drop( )
```

FIGURE 1.11 –

Chapitre 2

Correction Exercice 1

Dans cet exercice, nous allons exécuter quelques requêtes telles que :

2.1 Afficher toutes les collections de la base

```
> use ex1DB
< switched to db ex1DB
> show collections
< employes
> db.employes.find()
< {
  _id: ObjectId('511d0a9f181b16509ae5f7d8'),
  nom: 'Dupond',
  prenom: 'Jean',
  anciennete: 10,
  adresse: {
    numero: 77,
    codepostal: 31000,
    ville: 'Toulouse'
  }
}
{
  _id: ObjectId('511d0a9f181b16509ae5f7d9'),
  nom: 'King',
  prenom: 'David',
  anciennete: 27,
  adresse: {
    numero: 78,
    codepostal: 33000,
    ville: 'Bordeaux'
  }
}
{
```

FIGURE 2.1 – Affichage des collections disponibles dans la base de données

2.2 Afficher toutes les collections et tous les documents de la base

```
> use ex1DB
< switched to db ex1DB
> show collections
< employes
> db.employes.find()
< {
  _id: ObjectId('511d0a9f181b16509ae5f7d8'),
  nom: 'Dupond',
  prenom: 'Jean',
  anciennete: 10,
  adresse: {
    numero: 77,
    codepostal: 31000,
    ville: 'Toulouse'
  }
}
{
  _id: ObjectId('511d0a9f181b16509ae5f7d9'),
  nom: 'King',
  prenom: 'David',
  anciennete: 27,
  adresse: {
    numero: 78,
    codepostal: 33000,
    ville: 'Bordeaux'
  }
}
{
```

FIGURE 2.2 – Affichage des collections et des documents de la base de données

2.3 Compter le nombre de documents de la collection *employés*

```
> db.employes.find({}).count()
< 48
ex1DB>
```

FIGURE 2.3 – Comptage du nombre de documents dans la collection *employés*

2.4 Insérer deux employés

```
> db.employes.insertOne({nom:'oussama',prenom:'ben slama',anciennete:10})
< {
  acknowledged: true,
  insertedId: ObjectId('6749ddf05410895b9b6380c')
}
> db.employes.insertOne({nom:'oussama',prenom:'ben slama',prime:10})
< {
  acknowledged: true,
  insertedId: ObjectId('6749de5505410895b9b6380d')
}
ex1DB>
```

FIGURE 2.4 – Insertion de deux nouveaux employés dans la base de données

2.5 Afficher la liste des employés dont le prénom est David

```
> db.employees.find({prenom:'David'})
< {
  _id: ObjectId('511d0a9f181b16509ae5f7d9'),
  nom: 'King',
  prenom: 'David',
  anciennete: 27,
  adresse: {
    numero: 78,
    codepostal: 33000,
    ville: 'Bordeaux'
  }
}
{
  _id: ObjectId('511d0a9f181b16509ae5f7df'),
  nom: 'Ving',
  prenom: 'David',
  anciennete: 17,
  adresse: {
    numero: 28,
    codepostal: 33000,
    ville: 'Bordeaux'
  }
}
{
  _id: ObjectId('511d0aa0181b16509ae5f7e5'),
  nom: 'Kingaba',
  prenom: 'David',
  anciennete: 23,
```

2.6 Afficher la liste des employés dont le prénom commence ou se termine par D

```
>_MONGOSH
> db.employees.find({ "prenom": { "$regex": "^D|D$", "$options": "i" } }
)
< {
  _id: ObjectId('511d0a9f181b16509ae5f7d9'),
  nom: 'King',
  prenom: 'David',
  anciennete: 27,
  adresse: {
    numero: 78,
    codepostal: 33000,
    ville: 'Bordeaux'
  }
}
{
  _id: ObjectId('511d0a9f181b16509ae5f7df'),
  nom: 'Ving',
  prenom: 'David',
  anciennete: 17,
  adresse: {
    numero: 28,
    codepostal: 33000,
    ville: 'Bordeaux'
  }
}
```

FIGURE 2.6 – Liste des employés dont le prénom commence ou se termine par *D*

2.7 Afficher la liste des personnes dont le prénom commence par D et contient exactement 5 lettres

```
<
> db.employees.find({ "prenom": { "$regex": "^D.{4}$", "$options": "i" } }
)
< {
  _id: ObjectId('511d0a9f181b16509ae5f7d9'),
  nom: 'King',
  prenom: 'David',
  anciennete: 27,
  adresse: {
    numero: 78,
    codepostal: 33000,
    ville: 'Bordeaux'
  }
}
{
  _id: ObjectId('511d0a9f181b16509ae5f7df'),
  nom: 'Ving',
  prenom: 'David',
  anciennete: 17,
  adresse: {
    numero: 28,
    codepostal: 33000,
    ville: 'Bordeaux'
  }
}
{
  _id: ObjectId('511d0aa0181b16509ae5f7e5'),
```

FIGURE 2.7 – Liste des personnes dont le prénom commence par *D* et contient 5 lettres

2.8 Afficher les noms et prénoms de chaque employé ayant une ancienneté > 10 ans

```
> db.employes.find(
  { "anciennete": { $gt: 10 } }, // Filtre : ancienneté > 10
  { "_id": 0, "nom": 1, "prenom": 1 } // Projection : uniquement les champs nom et prénom
)
< {
  nom: 'King',
  prenom: 'David'
}
{
  nom: 'Rupont',
  prenom: 'Jean'
}
{
  nom: 'Ving',
  prenom: 'David'
}
{
  nom: 'Bass',
  prenom: 'Vincent'
}
```

FIGURE 2.8 – Liste des employés ayant une ancienneté supérieure à 10 ans