ETML Informatique

Module: 165

Utiliser des bases de données NoSQL

ETML Informatique

Base de données NoSQL

Objectifs

2. Implémenter une base de données NoSQL et y insérer les données.

ETML Informatique

MongoDB

Le champ _id et les ObjectId

MongoDB
Le champ _id

Dans MongoDB, chaque document stocké dans une collection nécessite un champ **_id** unique qui fait office de clé primaire .

Si un document inséré omet le champ **_id**, le pilote MongoDB génère automatiquement un spécifique de type **ObjectId** pour le champ **_id**.

ETML Informatique

MongoDB
Le champ _id

Le champ **_id** a le comportement et les contraintes suivants :

- Par défaut, MongoDB crée un index unique sur le champ **_id** lors de la création d'une collection.

- Le champ **_id** est toujours le premier champ des documents. Si le serveur reçoit en premier un document qui ne contient pas le champ **_id**, alors le serveur déplacera le champ au début.

- Le champ **_id** peut contenir des valeurs de n'importe quel type de données BSON , autre qu'un tableau.

ETML Informatique

MongoDB
Le champ _id

Vous pouvez mettre n'importe quelle valeur dans le champ **_id** du moment qu'il soit unique, excepté un type tableau [].

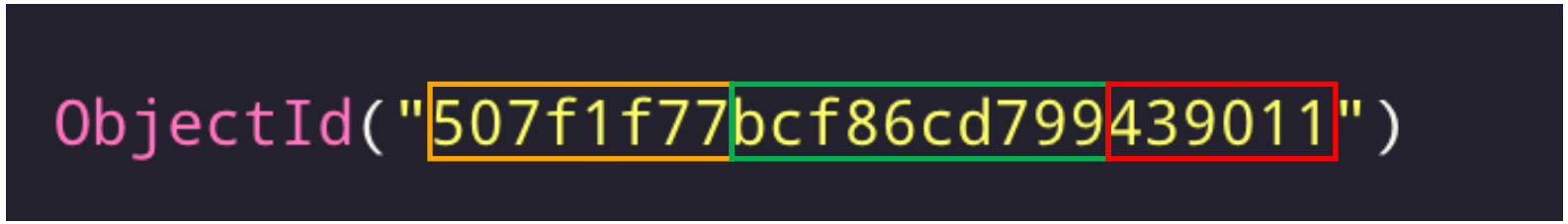
```
use('testdb');
db.users.insertOne({ firstname: 'Paul', lastname: 'Blinken', _id: new Date() });
```

ETML Informatique

MongoDB
Le ObjectId

Un **ObjectId** est un type **BSON**.
Les **ObjectId** font **12 octets**, ils sont rapides à générer et ordonnés.

- **4 octets** sont un timestamp de création.
- **5 octets** sont des valeurs aléatoires assurant l'unicité.
- **3 octets** sont un compteur qui s'incrémente et qui démarre à une valeur aléatoire.



- Vous pouvez mettre n'importe quelle valeur dans le champ **_id** du moment qu'elle soit unique.

ETML Informatique

MongoDB
Le ObjectId

Pour générer un nouvel **ObjectId**, utilisez le fonction **ObjectId()** sans argument :

```
newObjectId = ObjectId()
```

Dans cet exemple, la valeur de **newObjectId** :

```
ObjectId("507f1f77bcf86cd799439011")
```

ETML Informatique

ETML Informatique

MongoDB
Les opérations: CRUD

MongoDB
Prérequis

Veuillez ajouter la collection « **employes** » dans la base de données « **db_entreprise** » avec le fichier « **employes.zip** » fournit par votre enseignant.

Décompressez ce fichier dans votre répertoire « **C:\165_docker\backupdb** » et utilisez la commande « **mongorestore** » afin d'ajouter cette nouvelle collection au format BSON.

ETML Informatique

MongoDB
Les opérations: CRUD

Create

Read

Update

Delete

ETML Informatique

MongoDB
Opération Create

insertOne()

insertMany()

ETML Informatique

MongoDB
Opération Create

La méthode **insertOne()** permet d'insérer un document dans une collection.

Si vous ne précisez pas d'**ObjectId**, *mongod* en crée un automatiquement, car chaque document doit obligatoirement avoir un identifiant unique dans le champ **_id**.

ETML Informatique

MongoDB
Opération Create

Dans la base de données « **db_entreprise** » nous désirons ajouter un nouvel employé qui se nomme « **Müller Pascal** » avec « **12 ans** » d'ancienneté. Nous ne connaissons pas encore son adresse.

MongoDB
Opération Create

```
use('db_entreprise');

db.employes.insertOne({
  "nom": "Müller",
  "prenom": "Pascal",
  "anciennete": 12,
});
```

MongoDB
Opération Create

La méthode **insertMany()** permet d'insérer plusieurs documents dans une collection.

ETML Informatique

MongoDB
Opération Create

Maintenant, nous désirons ajouter 2 nouveaux employés « **Remond Félix** » et « **Uka Félix** », tous deux ont « **10 ans** » d'ancienneté. Nous ne connaissons pas encore leur adresse.

MongoDB
Opération Create

```
use('db_entreprise');

db.employes.insertMany([
  { "nom": "Remond", "prenom": "Félix", "anciennete": 10 },
  { "nom": "Uka", "prenom": "Félix", "anciennete": 10 }
]);
```

ETML Informatique

ETML Informatique

MongoDB
Opération Read

MongoDB
Opération Read

findOne()

find()

ETML Informatique

MongoDB
Opération Read

La méthode **findOne()** permet de retourner le premier document qui satisfait une requête donnée, passée en paramètre.

ETML Informatique

MongoDB
Opération Read

Nous désirons retrouver le premier employé avec comme prénom « **Félix** » et « **10 ans** » d'ancienneté.

```
use('db_entreprise');

db.employes.findOne({prenom: "Félix", anciennete: 10 });
```

ETML Informatique

MongoDB
Opération Read

Maintenant, nous désirons retrouver le premier employé qui habite à « Paris ».

```
use('db_entreprise');

db.employes.findOne({"adresse.ville": "Paris" });
```

ETML Informatique

Base de données NoSQL
Opération Read

La méthode **find()** permet de rechercher un ou plusieurs documents qui correspondent à la requête passée.

ETML Informatique

Base de données NoSQL
Opération Read

Trouvez tous les employés qui ont une ancienneté de « **7 ans** ».

```
use('db_entreprise');

db.employes.find({anciennete: 7 });
```

ETML Informatique

Base de données NoSQL
Opération Read

Combien d'employé habite à « **Toulouse** »?

```
use('db_entreprise');

db.employes.find({"adresse.ville": "Toulouse" }).count();
```

ETML Informatique

Base de données NoSQL
Opération Read

Attention ! La méthode **find()** retourne un curseur et non pas l'ensemble des documents.

```
db.users.find().count()

const curseur = db.users.find();
while (curseur.hasNext()) {
    const document = curseur.next()
    printjson(document.name);
}
```

```
use('test');
db.users.find().toArray()

const curseur = db.users.find();
curseur.forEach(d => printjson(d.firstname));
```

ETML Informatique

MongoDB
Opération Update

MongoDB
Opération Update

- updateOne()
- findOneAndUpdate()
- updateMany()
- replaceOne()
- findOneAndReplace()

ETML Informatique

MongoDB
Opération Update

La méthode **updateOne()** permet de mettre à jour un unique document qui répond à la requête passée en premier argument.

- L'opérateur de mise à jour **$set** permet de remplacer la valeur d'un ou plusieurs champs par les valeurs spécifiées.
- L'opérateur de mise à jour **$unset** permet de supprimer un ou plusieurs champs.

```
use('test');
db.users.updateOne({firstname: 'Julie'}, {
    $set: {
        name: 'Juliette',
        age: 22
    }
})
```

Ici, nous mettons à jour le premier document qui a la propriété ***name*: 'Julie'**.

La valeur du champ *name* est modifiée et le champ *age* est ajouté avec la valeur **22**.

MongoDB
Opération Update

Modifier le prénom de l'employé nommé « **Uka** » et ajouté lui son téléphone « **tel** » avec son numéro « **123456789** ».

```
use('db_entreprise');

db.employes.updateOne({ nom: "Uka"}, {
  $set: {
    prenom: "Julie",
    tel: 123456789
  }
});
```

ETML Informatique

MongoDB
Opération Update

La méthode **findOneAndUpdate()** permet de mettre à jour un document et de retourner le document avant ou après modification dans une même opération.

La méthode **findOneAndReplace()** permet de remplacer un document et de retourner le document avant ou après le remplacement dans une même opération.

Pour retourner le document après sa mise à jour, il faut utiliser une option.
Le nom de l'option n'étant pas harmonisé entre les *drivers* et *shells*, cela peut être **returnNewDocument**, **returnDocument** ou **new**.

MongoDB
Opération Update

Maintenant nous désirons ajouter l'adresse de l'employé « **Remond** » et que la requête retourne le document ainsi modifié.

L'adresse est « **1000 Lausanne** » au numéro « **22** ».

MongoDB
Opération Update

```
use('db_entreprise');

db.employes.findOneAndUpdate({ nom: "Remond"}, {
  $set: {
    adresse: {
      numero: 22,
      codepostal: 1000,
      ville: "Lausanne"
    }
  }
}, { returnDocument: "after" });
```

MongoDB
Opération Update

Nous voulons supprimer l'adresse de l'employé « **King David** » et retourner le document avant cette suppression.

```
use('db_entreprise');

db.employes.findOneAndUpdate({ nom: "King", prenom: "David"}, {
  $unset: {
    adresse: {}
  }
});
```

ETML Informatique

MongoDB
Opération Update

La méthode **updateMany()** permet de mettre à jour tous les documents qui répondent à la requête passée en premier argument.

```
use('test');
db.users.updateMany({firstname: 'Julie'}, {
    $inc: {
        age: 12
    },
    $set: {
        name: 'Juliette'
    }
})
```

Cette requête va mettre à jour tous les documents qui ont un champ ayant pour valeur ***name*: 'Julie'**.

La valeur du champ *name* sera mise à jour en 'Juliette' et la valeur du champ *age* sera incrémenté de **12** s'il existe. S'il n'existe pas, il sera créé avec la valeur **12**.

MongoDB
Opération Update

La méthode **replaceOne()** permet de remplacer un unique document qui répond à la requête passée en premier argument par un autre document passé en second argument.

```
use('test');
db.users.replaceOne({name: 'Alan'},
    {
        age: 25,
        name: 'Alain'
    },
    {upsert: true}
)
```

Si un document avec un champ *name* ayant pour valeur **Alan** existe alors il sera remplacé par un nouveau document **{ age: 25, name: 'Alain' }**.

Dans le cas contraire, un nouveau document **{ age: 25, name: 'Alain' }** sera inséré.

ETML Informatique

ETML Informatique

MongoDB
Opération Delete

MongoDB
Opération Delete

deleteOne()

findOneAndDelete()

deleteMany()

drop()

dropDatabase()

ETML Informatique

Base de données NoSQL
Opération Delete

La méthode **deleteOne()** va supprimer le premier document qui répond à la requête spécifiée.

```
use('test');
db.users.deleteOne({firstname: 'Alan'})

{
  "acknowledged": true,
  "deletedCount": 1
}
```

deleteCount contient le nombre de documents supprimés qui sera toujours de **0** (aucun document pour le filtre passé) ou **1** (le premier document de la collection pour le filtre spécifié)

ETML Informatique

Base de données NoSQL
Opération Delete

Supprimez l'employé nommé « **Briu** » et habitant à « **Foix** ».

```
use('db_entreprise');

db.employes.deleteOne({ nom: "Briu", "adresse.ville": "Foix"});
```

ETML Informatique

MongoDB
Opération Delete

La méthode **findOneAndDelete()** permet de supprimer le premier document correspondant au filtre et de le retourner.

ETML Informatique

MongoDB
Opération Delete

La méthode **deleteMany()** va supprimer tous les documents qui répondent à la requête spécifiée.

```
use('test');
db.users.deleteMany({firstname: 'Alan'})

{
  "acknowledged": true,
  "deletedCount": 2
}
```

deleteCount contient le nombre de documents supprimés.

Pour supprimer l'intégralité des documents d'une collection, il suffit de faire : **deleteMany({})**

MongoDB
Opération Delete

Supprimer tous les employés qui se prénomme « **Christophe** ».

```
use('db_entreprise');

db.employes.deleteMany({ prenom: "Christophe"});
```

ETML Informatique

MongoDB
Opération Delete

La méthode **drop()** va supprimer la collection sur laquelle elle est appelée.

Attention ! La collection et tous les documents qu'elle contient seront définitivement supprimés.

```
db.users.drop()
```

La méthode **dropDatabase()** va supprimer la base de données sur laquelle elle est appelée.

Attention ! La base de données, toutes les collections et tous les documents qu'elle contient seront définitivement supprimés.

```
db.dropDatabase()
```

MongoDB
Méthodes références

Retrouvez toutes les méthodes pour interagir avec MongoDB :

https://www.mongodb.com/docs/manual/reference/method/

ETML Informatique