

420-KBA-LG, programmation de bases de données

Saliha Yacoub

Les bases de données NoSQL

- Introduction aux bases de données NoSQL
- MongoDB, introduction
 - Créer une base de données
 - Insertion des données
 - Rechercher des données
 - Mise à jour des données
- Intégration avec C#

Les bases de données NoSQL

- Not Only SQL propose de laisser de côté certaines contraintes des bases de données relationnelles. (dénormalisation, pas de FK)
- Dans ce contexte, il est plus intéressant d'avoir un langage de haut niveau pour exploiter les bases de données.
- Contrairement aux BD SQL, qui fonctionnent toutes sous le même principe, il existe plusieurs types de BD No SQL
 - Clé/Valeurs: Redis(VmWare) , SimpleDB (Amazon)
 - Des lignes vers les colonnes: le stockage des données est sous forme de colonne plutôt que de lignes. BigTable(Google), HBase
 - Gestion de documents: MongoDB, Cassandra.
 - Orienté Graph:Neo4J

Les bases de données NoSQL

Part du marché des SGBD: Les SGBD relationnels dominant clairement le marché (<https://db-engines.com/en/ranking>)

Quand utilise-t-on le SQL ?

- Les données doivent être structurées. L'organisation est connue (ou pourrait être connue) d'avance.
- L'intégrité des données doit-être respectée
- Les transactions sont importantes. Le principe ACID est important

Quand utilise t-on le NoSQL

- La structure de données n'est pas importante. Évolutive et pas connue d'avance
- Gestion de beaucoup de données structurées, et non structurée.
- BASE: (contrairement à ACID)
 - **Basically Available** : quelle que soit la charge de la base de données, le système garantie la disponibilité des données.
 - **Soft-state** : La base peut changer lors des mises à jour ou lors d'ajout/suppression de serveurs. La base NoSQL n'a pas à être cohérente à tout instant
 - **Eventually consistent** : À terme, la base atteindra un état cohérent

Les bases de données NoSQL(MongoDB)

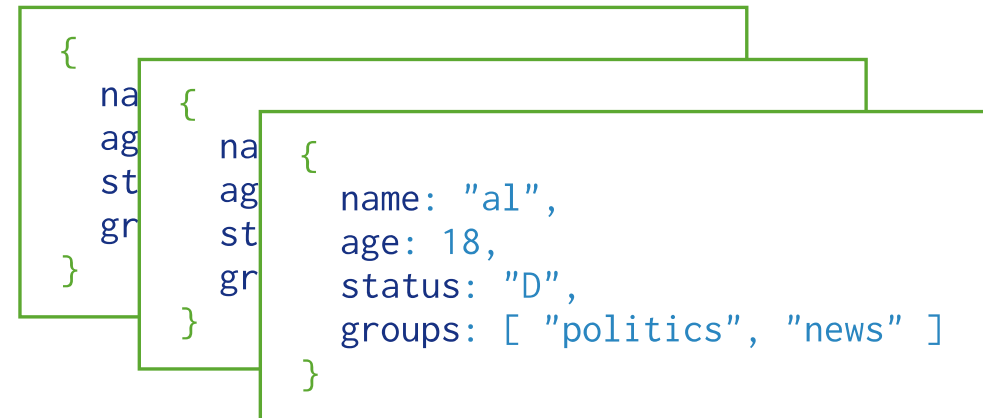
```
{  
  name: "sue",  
  age: 26,  
  status: "A",  
  groups: [ "news", "sports" ]  
}
```

← field: value
← field: value
← field: value
← field: value

- Un enregistrement dans MongoDB est un document, qui est une structure de données composée de paires de champs et de valeurs.
- Un document est encapsulé dans des accolades {...}, pouvant contenir des listes de clés/valeurs
- Les documents MongoDB sont similaires aux objets JSON.
- Une valeur peut être un type scalaire (entier, nombre, texte, booléen, null), des listes de valeurs [...], ou des documents imbriqués

Les bases de données NoSQL (MongoDB)

- Une collection est un ensemble de documents.
- C'est comme une table dans une base de données relationnelle.
- Les collections se trouvent dans une base de données



Collection

Les bases de données NoSQL (MongoDB)

Installation : télécharger le serveur MongoDB à l'adresse

<https://www.mongodb.com/download-center/community>

Vous devez télécharger le MSI (Microsoft System Installer). Les instructions d'installation sont ici:

<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/>

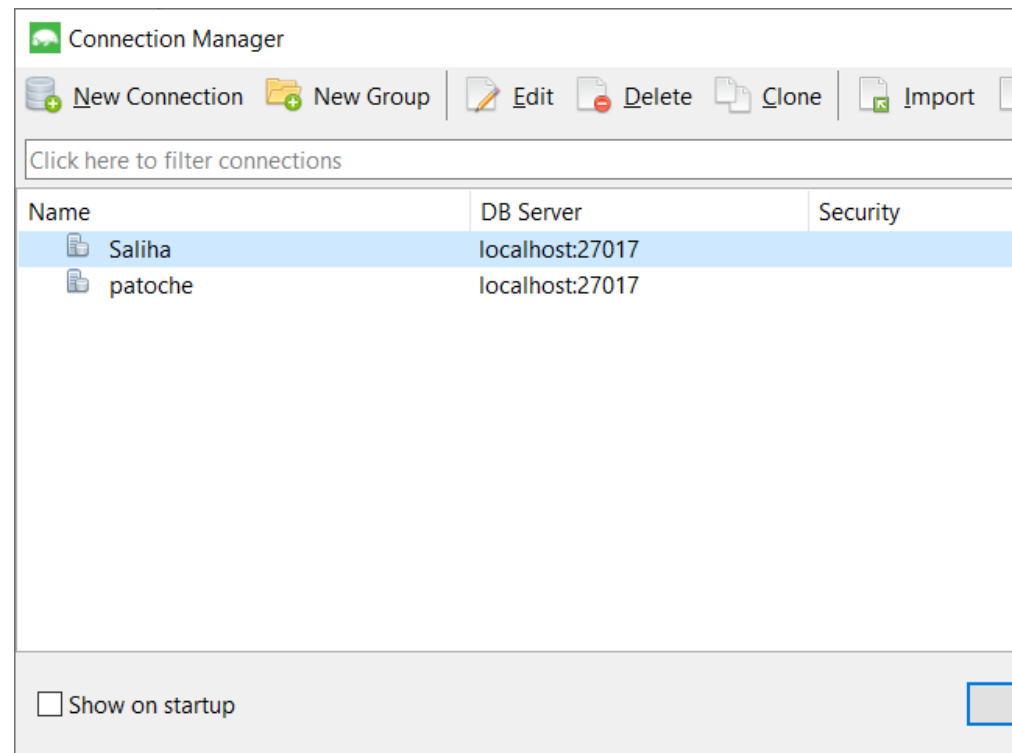
Il suffit de double cliquer sur l'exécutable et votre serveur s'installe. Cette installation inclue Mongo Compass qui est une interface graphique par laquelle vous pouvez exploiter votre serveur.

Pour exploiter votre serveur, on vous recommande de télécharger et installer Studio 3T qui offre une belle interface pour vos requêtes. (c'est juste une interface... pas de serveur)

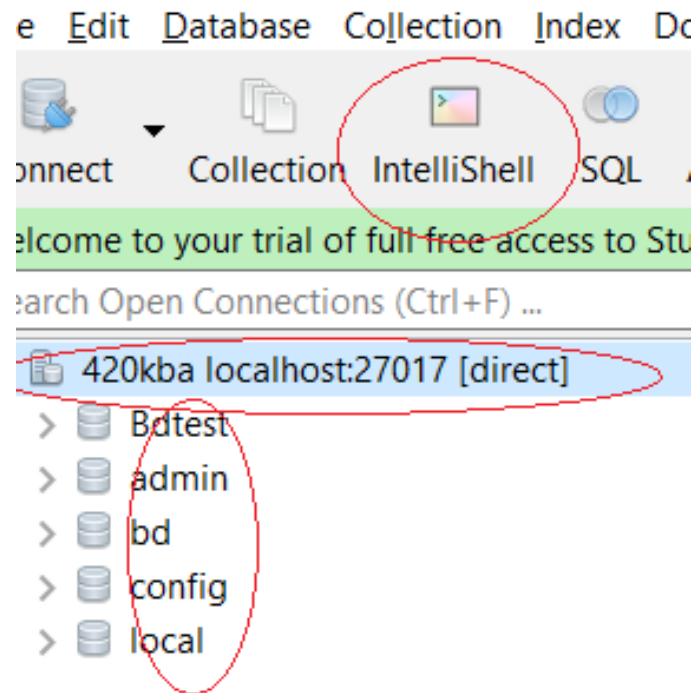
Sinon vous pouvez passer par l'interface de commandes.

Les bases de données NoSQL (MongoDB)

Par Studio 3T, vous pouvez utiliser une connexion existante ou créer une nouvelle connexion



Les bases de données NoSQL (MongoDB)



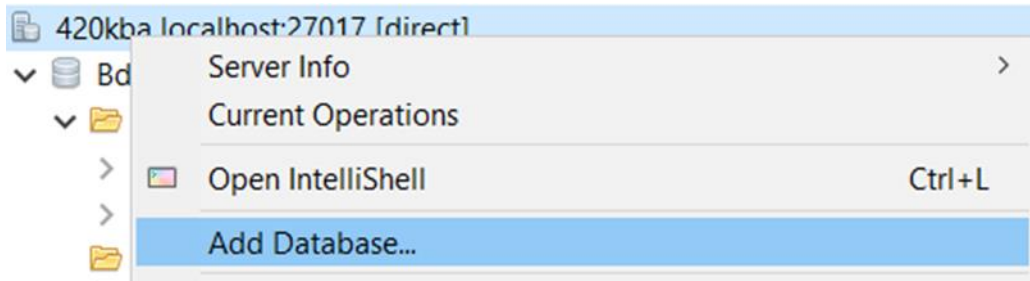
Une fois connecté, vous pouvez accéder à IntelliShell

IntelliShell: vous permet d'avoir la feuille pour écrire les requêtes

On s'est connecté avec la connexion 420kba

Il y a 5 bases de données dont deux créées par l'utilisateur

Les bases de données NoSQL (MongoDB)



Add Database vous permet d'ajouter une base de données.

Une fois la BD ajoutée, faites USE (nomBD) use bdSimba;

la commande **use nomBD** permet de créer la base de données si celle-ci n'existe pas..

Une fois que la BD est créé (avec use), vous devez créer vos collections.

La commande **db.createCollection("nomCollection");** permet de créer une collection

Les bases de données NoSQL (MongoDB)

Exemple 1:

```
use bdSimba; // va créer la bd bdSimba
db.createCollection("contacts"); // va créer la collection contacts dans bdSimba
db.contacts.insertOne
(
    {
        "nom":"Saliha",
        "dep":"info"
    }
);
```

La commande `db.nomCollection.insertOne({ liste des champs du documents})` permet d'insérer un document à la fois dans la collection

Les bases de données NoSQL (MongoDB)

Exemple 2:













```
db.contacts.insertOne(  
{  
  "nom": "Poitras",  
  "dep":  
    {"code": 420, nom: "info"},  
  "cours": "kba"  
}  
);
```

Remarquez que le champ dep a lui-même deux champs.

Les bases de données NoSQL (MongoDB)

La commande `insertOne(document)` permet de faire une insertion dans une collection un document à la fois. Si Aucun id n'est fourni, le système va attribuer un identificateur par défaut

La figure suivante, vous permet de voir les documents avec un id fournit par le système lorsqu'il y en a pas.

 5ddf27722b73c1f433a60a	 20.0	 "Gable	 "Alain	 { 2 fields }
 5ddf27722b73c1f433a60b	 21.0	 "Primogene	 "Alain	 { 2 fields }
 5ddf27722b73c1f433a60c	 22.0	 "Lechat	 "Alain	
 1.0	 11.0	 "Patoche	 "Alain	
 2.0	 12.0	 "Patoche	 "Alain	
 3.0	 13.0	 "Patoche	 "Alain	
 4.0	 20.0	 "Gable	 "Alain	 { 2 fields }
 5.0	 21.0	 "Primogene	 "Alain	 { 2 fields }
 6.0	 22.0	 "Lechat	 "Alain	

Les bases de données NoSQL (MongoDB)

```
db.Programme.insertOne(  
{"_id":21, "numad": 11, "nom":"Ruby" , "prenom":"Robin"});
```

Va faire une insertion avec 21 comme identifiant pour Ruby.

insertMany([document1, document2, ..] permet de faire plusieurs insertion à la fois.

```
db.Programmes.insertMany(  
{ "_id":1, "numad": "11000", "nom" : "Patoche" , "prenom": "Alain"},  
{ "_id":2, "numad": "1200", "nom": "Patoche" , "prenom": "Voila"},  
{ "_id":3, "numad": "1300", "nom": "Lechat" , "prenom": "Simba"}  
)
```

Les bases de données NoSQL (MongoDB)

```
db.Programmes.insertMany([
  { "_id":4 ,"numad": "2000", "nom":"Gable" ,"prenom":"Alain",
    "programme": {"code":420,"nomprog":"info"}
  },
  { "_id":5 ,"numad": "2000", "nom":"Leroy" ,"prenom": "Yanick",
    "programme": {"code":410,"nomprog":« soins"}
  },
  ])
```

Les deux documents ont des documents imbriqués

Les bases de données NoSQL (MongoDB)

La commande find() , sélectionne un ou des documents dans la collection et retourne le résultat dans un curseur. Si aucun argument n'est fourni, la méthode retourne TOUS les documents.

Exemple1

`db.Programmes.find({"nom":"Patoche"});` va retourner tous les étudiants dont le nom est Patoche.

`db.Programmes.find({"nom":"Patoche", "prenom":"Alain"});` va retourner les noms des étudiants dont le nom est Patoche et le prenom Alain.

`db.Programme.find({"nom":"Patoche"},{nom:1,prenom:1});` seuls les noms et les prénoms seront affichés

`db.Programmes.find({ "etudiant.nom": "Yanick" });` **on cherche dans un document imbriqué**

Le document Programmes, contient le document Etudiant

Les bases de données NoSQL (MongoDB)

Opérateurs de comparaison:

\$gt: retourne les document dont la valeur est plus grande que la valeur passée. Il y a aussi **\$gte** pour supérieur ou égale

Syntaxe: {field: {\$gt: value} }

Exemple: db.employes.find({"Salaire": {\$gt:45000}});

\$lt, pour plus petit. Il y a aussi **:\$lte**

\$eq: pour l'égalité. Il y a aussi le **\$ne**

\$in (semblable au IN du SELECT, sauf que els valeurs sont fournies entre [])

Syntaxe: { field: { \$in: [<value1>, <value2>, ... <valueN>] } }

Exemple: db.employes.find({"Salaire": {\$in:[45000,50000,35000]}}).

Il y a aussi **\$nin** pour not in

Les bases de données NoSQL (MongoDB)

La commande **update()**. Elle permet de mettre à jour des informations contenues dans un document.

Syntaxe: `db.collection.update(query, update, options)`

- `query` , indique le document à mettre à jour.
- `update`, le document de mise à jour
- `option` indique les options de mise à jour (si le document à mettre à jour n'existe pas, faut-il l'insérer ?).

`$inc`: permet de faire une incrémentation d'un champ par une valeur: Utile pour les UPDATE.

Syntaxe: `$inc: { <field1>: <amount1>, <field2>: <amount2>, ... }` }

Les bases de données NoSQL (MongoDB)

```
Exemple: db.employes.update (  
  {"_id":1},  
  {  
    $inc: {"Salaire": 20}  
  }  
);
```

```
db.Programmes.update(  
  {"_id":11},  
  {  
    $inc: { "salaire": 20 },  
  })
```

nd	Document	Find	Find	Find	Find	Find	Text	Find	Find
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })									

Comme id=11, a été trouvé alors le nombre d'insertions est égale à 0 alors que le nombre de mise à jour est 1

Les bases de données NoSQL (MongoDB)

Comme id=99, n'a pas été trouvé et l'option upsert est à true alors le nombre d'insertions est égale à 1 alors que le nombre de mise à jour est 0

```
50
51 db.Programmes.update(
52     { "_id":99} ,
53
54     { "numad": "20",
55       "nom"  :"Simpson",
56       "prenom":"Bart"
57     },
58
59     { upsert: true })
70
```

Document	Find	Find	Find	Find	Find	Text	Find	Find	Document	Document
1	WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 99 })									
2										

Les bases de données NoSQL (MongoDB)

La commande **remove()** permet de supprimer un ou plusieurs documents selon le critère fournis

Exemple :Suppression

```
db.Programmes.remove({"_id":99});  
db.Programmes.remove({"nom":"Ruba"});
```

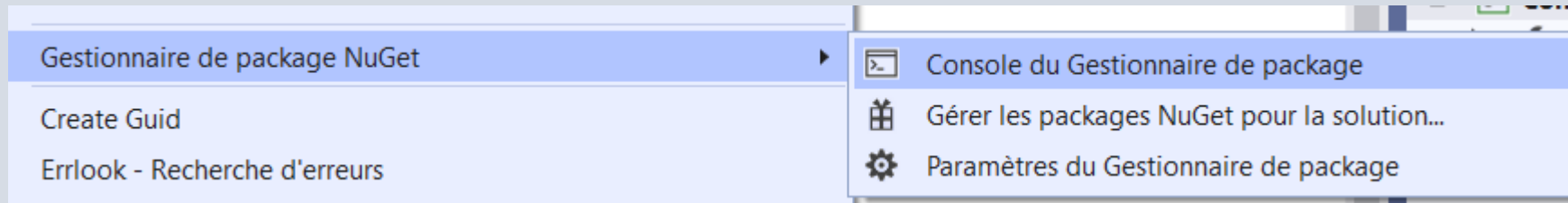
La commande **count()** permet de compter le nombre de documents à l'intérieur d'une collection.

```
db.Programmes.count();  
db.Programmes.count({"nom":"Patoche"});
```

Les bases de données NoSQL (MongoDB)

C# et MongoDB

Dans la console du gestionnaire des Packages, tapez la commande suivante:



```
Install-Package MongoDB.Driver -Version 2.9.3
```

Puis.....

```
using MongoDB.Driver;
```

```
using MongoDB.Bson;
```

La chaîne de connexion est de la forme: "mongodb://localhost:27017":

```
private const string connectionString = "mongodb://localhost:27017";
```

Les bases de données NoSQL (MongoDB)

Une fois connecté au serveur, il faudra indiquer quelle base de données utilisée. Puis quelle collection de la base de données. Voici les étapes:

```
MongoClient mong = new MongoClient(connectionString);  
IMongoDatabase db = mong.GetDatabase("bdSimba");  
IMongoCollection<BsonDocument> collectionDoc = db.GetCollection<BsonDocument>("Programme");
```

À partir de maintenant, on peut insérer, rechercher, modifier et supprimer:

Pour rechercher, il faudra définir un critère de recherche. Ce critère est vide lorsqu'aucun critère n'est défini.

Les bases de données NoSQL (MongoDB)

Exemple 1: Afficher tout (aucun critère)

```
var critere = Builders<BsonDocument>.Filter.Empty;
var resultat = colectionDoc.Find(critere).ToList();
foreach (var doc in resultat)
{
    Console.WriteLine(doc);
    Console.Read();
}
```


Les bases de données NoSQL (MongoDB)

Exemple 2:

```
var filter2 = Builders<BsonDocument>.Filter.Eq("nom", "Lechat");
var resultat2 = colectionDoc.Find(filter2).ToList();

foreach (var doc2 in resultat2)
{
    Console.WriteLine(doc2);
    Console.Read();
}
```

Les bases de données NoSQL (MongoDB)

Insertion:

```
var documnt = new BsonDocument
  {
    { "_id","111"},
    { "numad","123"},
    { "nom","Poitras"},
    { "prenom","Alain"}
  };
collectionDoc.InsertOne(documnt);
```

Les bases de données NoSQL (MongoDB)

Mise à jour: Update();

Permet de mettre à jour un document (ou plusieurs) selon le critère fourni.

```
{  
    var filter3 = Builders<BsonDocument>.Filter.Eq("_id", 6);  
    var update = Builders<BsonDocument>.Update.Set("nom","Patoche");  
    collectionDoc.UpdateOne(filter3, update);  
}
```

Les bases de données NoSQL (MongoDB)

```
Mise à jour: Update();  
{  
var filter3 = Builders<BsonDocument>.Filter.Eq("nom", "Poitras");  
    var update = Builders<BsonDocument>.Update.Set("nom","Poupon");  
    //colectionDoc.UpdateOne(filter3, update);  
    colectionDoc.UpdateMany(filter3, update);  
}
```

UpdateOne(): même s'il y a plusieurs documents retournés par le résultat de la recherche, seul le premier sera mis à jour.

UpdateMany(), tous les documents correspondant à la recherche seront mis à jour.

Les bases de données NoSQL (MongoDB)

Suppression d'un document: DeleteOne() ou DeleteMany();

```
{
```

```
var filter4 = Builders<BsonDocument>.Filter.Eq("_id", "103");
```

```
collectionDoc.DeleteOne(filter4);
```

```
}
```