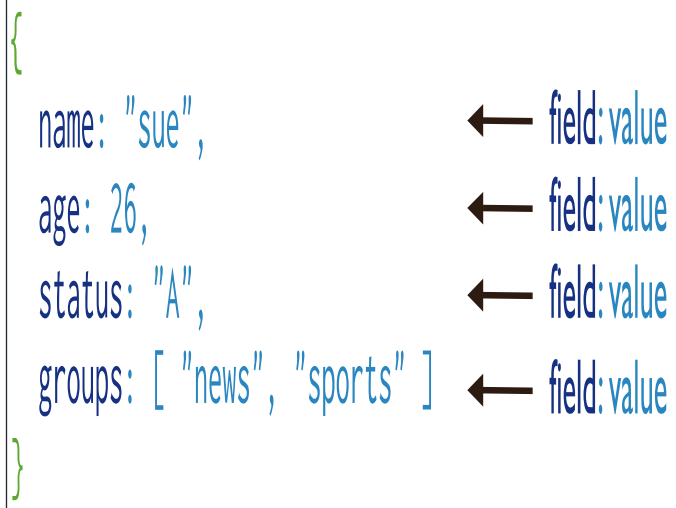
420-A – Programmation de bases de données

MongoDB, rappels

1. Introduction aux bases de données NoSQL
2. Quelques types de stockages de données
    - Gestion de documents
    - Clé/valeur
    - Lignes vers les colonnes
    - Orienté graphe
3. ACID vs BASE
4. Théorème de CAP
5. Présentation du travail

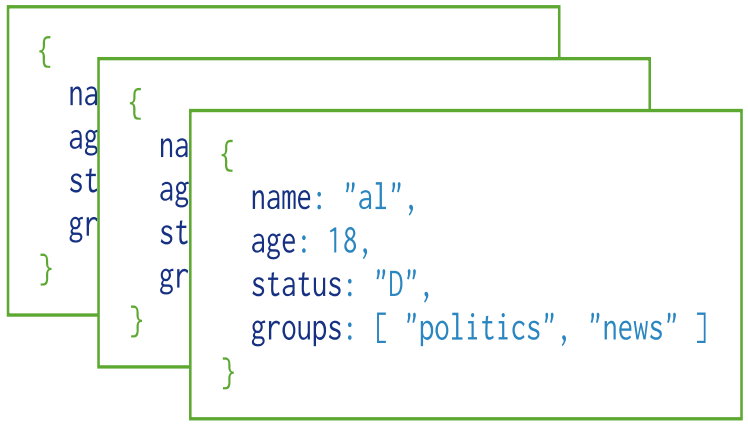
NoSQL: Documents

- Dans une BD orientée document, « un enregistrement » 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 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
- Le but de ce stockage est de manipuler des documents contenant des informations avec une structure complexe (types, listes, imbrications)



NoSQL: Documents

- 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



Application: MongoDB

MongoDB est une BD NoSQL orientée document

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

Application: 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.

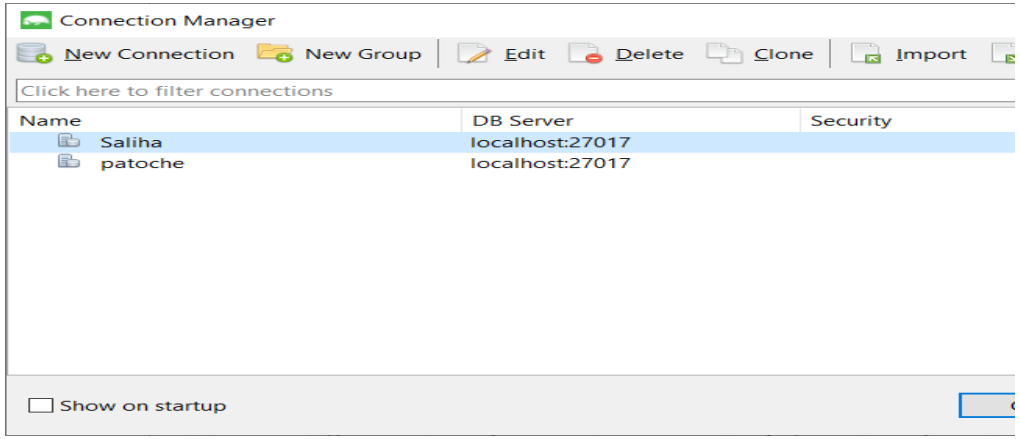
Pour exploiter votre serveur, on vous recommande de télécharger et installer **Studio 3T** à l'adresse: https://studio3t.com/download qui offre une belle interface pour vos requêtes. (c'est juste une interface… pas de serveur)

Vous pouvez avoir également **Robo 3T** complétement gratuit: https://robomongo.org/

Sinon vous pouvez passer par l'interface de commandes.

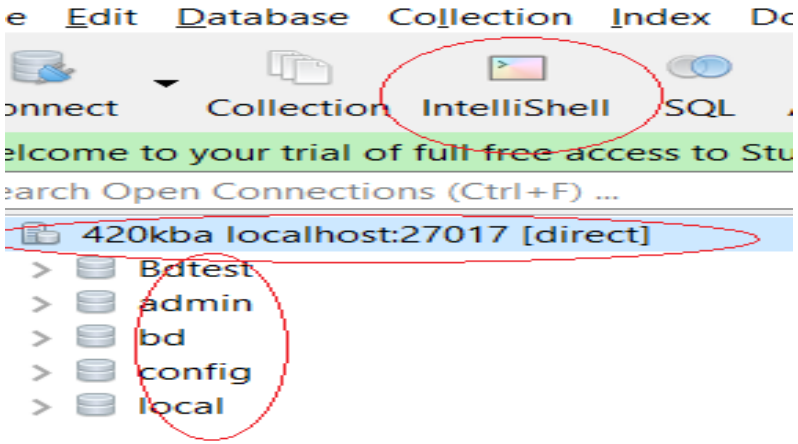
Application: MongoDB

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



Une fois connecté vous pouvez accédez à IntelliShel

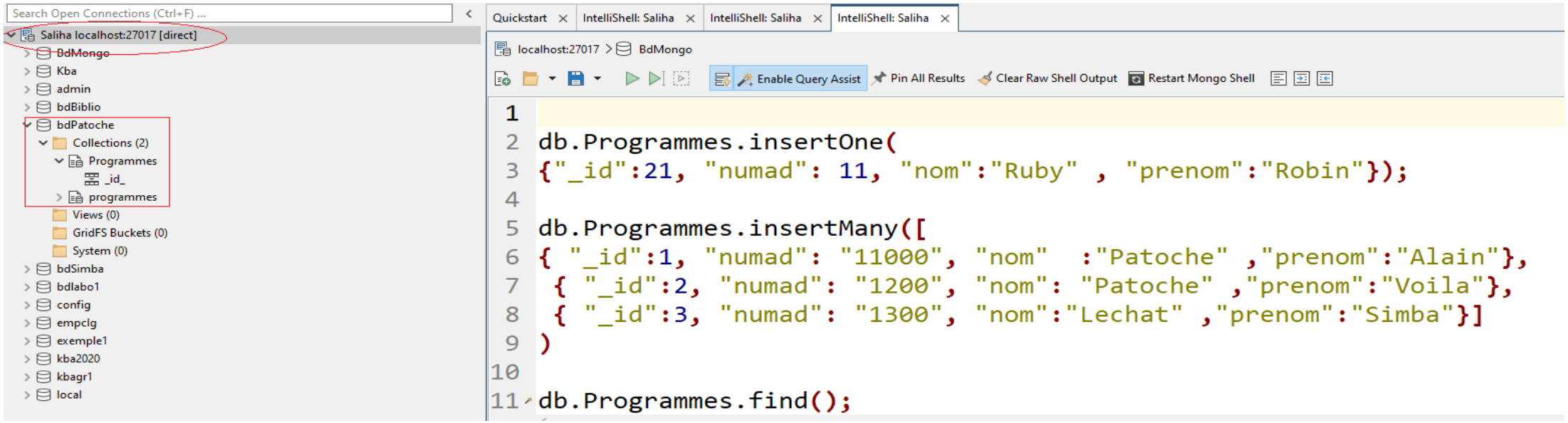
- IntelliShel: vous permet d'avoir la feuille pour écrire les requêtes
- On s'est connecté avec la connexion 420kba
- Il y a 5 bases de données dont deux crées par l'utilisateur



Application: MongoDB

Une fois connecté vous pouvez accédez à IntelliShel

- IntelliShel: vous permet d'avoir la feuille pour écrire les requêtes
- On s'est connecté avec la connexion Saliha
- Il y a plusieurs bases de données dont celles créés par l'utilisateur Saliha



Application: MongoDB

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 créée (avec use), vous devez créer vos collections.

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

Add Database vous permet d'ajouter une base de données

Application: 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"
        }
  );
```

Application: MongoDB

La commande : **`insertOne()`**

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

La commande **`insertMany()`**

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

Application: 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.

Application: 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 n'y en a pas.

5ddfeb27722b73c1f433a60a	20.0	Gable	Alain	{ 2 fields }
5ddfeb27722b73c1f433a60b	21.0	Primogene	Alain	{ 2 fields }
5ddfeb27722b73c1f433a60c	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	

Application: MongoDB

db.Programme.insertOne(

{"_id":21, "numad": 11, "nom":"Ruby" , "prenom":"Robin"});

Va faire une insertion avec 21 comme identifiant pour Ruby.

-----insertMany()

```
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"}]
);
```

Application: 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

Application: 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 Patcohe.

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

Application: 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

Application: MongoDB

Opérateurs logiques, exemple

```
db.contacts.find(
{
    "$and" : [
        { "numad" : {"$gt" : 202200002} },
        { "nom" : "Simpson" }
    ]}
);

db.Joueurs.find(
{           $or: [
                    { "solde": { $eq: 3000 } },
                    {"alias":"Saturne"}
           ]
}
);
```

Application: 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>, ... } }

Application: MongoDB

Exemple: db.employes.update

```
(
  {"_id":11},
  {
  $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 })
2
```

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

Application: 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

```
1
2  db.joueurs.update(
3          {"_id":101},
4  {$set:
5      {"alias": "gourou"}
6      },
7      {upsert:true}
8   );
```

Raw Shell Output | Shell Output (Documents) ×

50 | Documents 1 to 1

Result > acknowledged

_id	acknowledged	insertedId	matchedCount	modifiedCount	upsertedCount
	true	101.0	0.0	0.0	1.0

Application: 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.

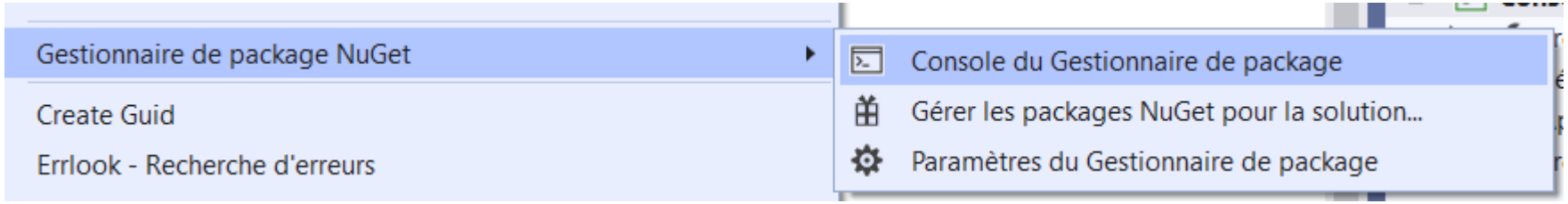
db.Programmes.count**();**

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

Application: 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 chaine de conenxion est de la forme: "mongodb://localhost:27017“:

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

Application: 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> colectionDoc = 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

Application: 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();
        }
```

Application: 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();
            }
```

Application: MongoDB

Insertion:

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

Application: 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");
        colectionDoc.UpdateOne(filter3, update);
}
```

Application: 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.

Application: MongoDB

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

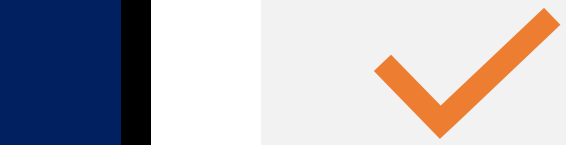
```
{
var filter4 = Builders<BsonDocument>.Filter.Eq("_id", "103");
   colectionDoc.DeleteOne(filter4);
}
```

Sources:

Sources:

- https://www.mongodb.com/docs/
- https://www.ibm.com/fr-fr/cloud/learn/cap-theorem
- https://www.oracle.com/fr/database/base-donnees-relationnelle-difference-non-relationnelle.html
- https://actualiteinformatique.fr/data/definition-nosql-not-only-sql-databas
- https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4462433-choisissez-votre-famille-nosql
- https://aws.amazon.com/fr/nosql/
- https://www.mongodb.com/docs/
- https://www.mongodb.com/blog/post/quick-start-c-sharp-and-mongodb-starting-and-setup
- https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mongo-app?view=aspnetcore-6.0&tabs=visual-studio
- https://www.mongodb.com/docs/drivers/csharp/

BD NoSQL



Conclusion

Questions