MongoDB

https://www.mongodb.com/

- Part du marché des SGBD: (https://db-engines.com/en/ranking)

Les concepts

Le modèle de données

Un serveur MongoDB peut gérer plusieurs bases de données. Chacune de de ces bd peut contenir 0 ou plusieurs collections. Une collection regroupe des documents. Si on voulait faire une analogie avec le modèle relationnel, on dirait que le document est l'équivalent d'un enregistrement et que la collection est l'équivalent d'une table.

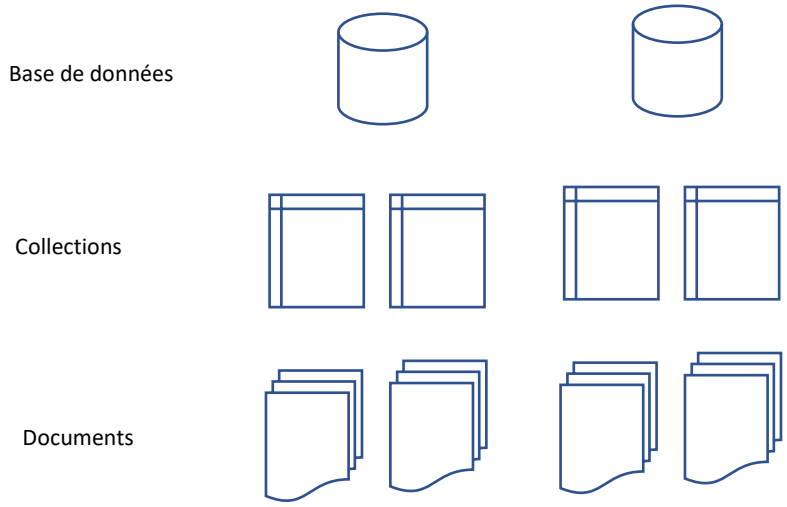

Figure 2: Un serveur MongoDB peut gérer plusieurs bases de données

Les documents

- Du point de vu de l'usager, un document est défini avec le langage JSON;
- Un document JSON est un ensemble d'associations { clé : valeur}. Le nom de clé **_id** est réservé au système;

```
{
    "num_ad": "20183383518",
     "nom": "Patoche",
     "prénom": "Alain"
}
```

- La clé est une chaine de caractères. La valeur associée à la clé peut être une chaine, un entier, un boolean, null, un tableau [], …;

Les collections

- Une collection est un regroupement de documents;
- Une collection n'a pas de structure prédéfinie comme une table dans une base de données relationnelle;
- Des documents de forme différente peuvent être ajouté dans une même collection;

  Ex. Les deux documents suivants pourraient être contenus dans la même collection;

```
{
    "nom": "Informatique",
    "code": "420"
}
```

```
{
    "num_ad": "20183383518",
    "nom": "Patoche",
    "prénom": "Alain"
}
```

- Typiquement on veut conserver des documents d'un même type dans une collection;

Les base de données

- Les collections sont regroupées dans base de données;
- Un serveur MongoDB peut gérer plusieurs bases de données;
- Les bases de données suivantes sont utilisées par le gestionnaire MongoDB;
  - **admin**
    Utilisé pour la gestion des usagers et des autorisations;
  - **local**
    Utilisé pour conserver les collections que l'on ne veut pas répliquer à d'autre serveur MongoDB;
  - **config**
    Utilisé pour la gestion du mode « shared » (partitionnement des donnés entre plusieurs serveur MongoDB);

Installer MongoDB

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

- Téléchargez la version la plus récente du logiciel (choisissez la version en format .zip);
- Décompressez le fichier sur le bureau;
- Démarrez un interpréteur de commandes (cmd.exe) et déplacez-vous dans le sous-répertoire 'bin' du logiciel MongoDB que vous avez décompressé;
- Vous devez créer le répertoire où seront conservés les fichiers des bases de données (par défaut MongoDB utilise le répertoire 'C:\data\db' pour conserver ses fichiers);
    - Créez le répertoire 'mongo_data' sur le bureau;

Démarrage du serveur MongoDB

- Dans l'interpréteur de commande que vous avez démarré, taper la commande suivante en spécifiant le chemin vers le répertoire que vous avez créé.

```
mongod.exe -dbpath C:\Users\beaul\OneDrive\Desktop\mongo_data
```

- Pour arrêter MongoDB simplement, tapez Ctrl-C dans la fenêtre de commandes;

Installer des pilotes dans Visual Studio

- Tapez la commande suivante dans la console du gestionnaire de paquets de Visual Studio.

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

- Pour se connecter au serveur MongoDB à partir d'un programme (.NET/Form) utilisez la chaine de connexion suivante;

```
string connectionString = "mongodb://localhost:27017";
MongoClient clientMDB = new MongoClient(connectionString);
```

L'interpréteur de commandes MongoDB

- Démarrez un interpréteur de commandes (cmd.exe) et déplacez-vous dans le sous-répertoire '**bin'** du logiciel MongoDB que vous avez décompressé;

- Pour lancer l'interpréteur de commandes MongoDB tapez la commande « mongo »;

```
C:\Users\beaul\mongodb-win-4.2.1\bin>mongo.exe
MongoDB shell version v4.2.1
connecting to:
mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceNam
e=mongodb
```

- La variable globale **db** fait référence à la base donnée en utilisation;

```
> db
test

> db.dropDatabase()
> db.collection.dropCollection()
```

- On peut changer la bd en utilisation avec la commande **use**. Notez qu'il n'est pas nécessaire de créer la base de données avant de pouvoir l'utiliser. La création de la bd et des collections se fera automatiquement au moment d'ajouter des documents ;
  La commande **use** définit la bd sur laquelle toutes les commandes s'appliqueront;

  La commande **show dbs** permet d'afficher toutes les bd sur le serveur. Notez que la bd kba_demo d'apparait pas puisqu'elle n'a pas encore été créée;

```
> use kba_demo
switched to db kba_demo
> db
kba_demo

C#

clientMDB = new MongoClient(connectionString);
IMongoDatabase mongoDBs = clientMDB.GetDatabase("kba_demo");

> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB

C#

var dbList = clientMDB.ListDatabases();

foreach (var item in dbList.ToList())
   Console.WriteLine(item);
```

- La commande **show collections** permet d'afficher toutes les collections dans la base de données couramment en utilisation; celle identifier avec la commande **use**. Notez qu'aucune collection n'existe encore dans la **kba_demo**;

```
> show collections
>

C#

var collList = mongoDBs.ListCollections().ToList();
foreach (var item in collList)
{
   Console.WriteLine(item);
}
```

Les commandes insert, find, update

- La commande **insert** permet d'ajouter un document dans une collection d'une base de données. Notez qu'après avoir ajouté d'un document, la bd kba_demo ainsi que la collection programmes ont été créées;

1. Ajouter un programme à la collection

```
> prog = {
...      "nom": "Bière",
...      "code": "555"
... }
{ "nom" : "Bière", "code" : "555" }

> db.prog_etudiants.insert(prog)
WriteResult({ "nInserted" : 1 })
>

C#

IMongoCollection<Programme> programmes =
           mongoDBs.GetCollection<Programme>("prog_etudiants");

Programme prog = new Programme("Bière", "999");

programmes.InsertOne(prog);
```

- La commande **find** permet de rechercher dans une collection;

1. Afficher tous les programmes

```
> db. prog_etudiants.find()

C#

var filter = Builders<Programme>.Filter.Empty;

var res = programmes.Find<Programme>(filter);
foreach (var prog in res.ToList())
   Console.WriteLine(prog);
```

Figure 3: Sans paramètres, la commande affiche tous les documents de la collection;

2. Afficher toute l'information du programme 300

```
> db. prog_etudiants.find({code: "300"})
{ "_id" : ObjectId("5de2c822abbd2d7941bcfe40"), "nom" :
"Éducation de l'enfance", "code" : "300" }
>

C#

var filter = Builders<Programme>.Filter.Eq("code", "300");
var resultDoc = programmes.Find<Programme>(filter).ToList();
foreach (var item in resultDoc)
{
   Console.WriteLine(item.ToString());
}
```

Figure 4: Requête avec un critère

3. Afficher uniquement le nom du programme 300

```
> db. prog_etudiants.find({code: "300"}, {"nom":1,"_id":0})
{ "nom" : "Éducation de l'enfance" }

C#

var filter = Builders<Programme>.Filter.Eq("code", "300");

var projection = Builders<Programme>.Projection.Include("nom")
                                                .Exclude("_id");

var resultDoc = programmes.Find<Programme>(filter)
                          .Project(projection).ToList();
```

Figure 5: requête avec critère et projection

4. Rechercher le programme d'un étudiant

```
db.prog_etudiants.find({"étudiants.nom":"Fafar"},
                       {"_id":0,"code":1, "nom":1})

C#

var filter = Builders<Programme>.Filter.Eq("étudiants.nom",
                                           "Fafar");

var resultDoc = programmes.Find<Programme>(filter).ToList();
```

Figure 6: Afficher le programme de l'étudiant 'Marie Fafar'

- La commande **update** permet de mettre à jour les données;

1. Modifier le code d'un programme

```
> db.prog_etudiants.update({"code": "666"},
                           { $set :{"code" : "991"}},
                           {multi:true})

WriteResult({ "nMatched" : 4,"nUpserted" : 0, "nModified" : 4})

C#

var filter = Builders<Programme>.Filter.Eq("code", "991");
var update = Builders<Programme>.Update.Set("code", "661");

programmes.UpdateMany(filter, update);
```

Figure 7: Mettre à jour la valeur d'une clé avec un critère

2. Ajouter un étudiant dans un programme

```
> db.prog_etudiants.update({"code": "320"},
                           { $push : { "étudiants":
                                       {"num_ad" : "2017545668",
                                        "nom" : "Lucia",
                                        "prénom" : "Luciano"
                                       }
                                     }
                            })

C#

var filter = Builders<Programme>.Filter.Eq("code", "320");
var update = Builders<Programme>.Update.AddToSet("étudiants",
                                  new Etudiant("Joe","Perlu"));

programmes.UpdateOne(filter, update);
```

Figure 8: Ajout d'un élément dans un tableau

3. Supprimer un étudiant d'un programme

```
> db.prog_etudiants.update({"code": "320"},
                           { $push : { "étudiants":
                                       {"num_ad" : "2017545668",
                                        "nom" : "Lucia",
                                        "prénom" : "Luciano"
                                       }
                                     }
                            })

C#

var filter = Builders<Programme>.Filter.Eq("code", "320");
var update = Builders<Programme>.Update.AddToSet("étudiants",
                                  new Etudiant("Joe","Perlu"));

programmes.UpdateOne(filter, update);
```

Figure 9: Supprimer un élément dans un tableau

Implémenter une relation de 1-N

Imbriquer les documents de type étudiants dans un champ tableau du document programme

```
    {
    "_id": "p001",
    "nom": "Informatique",
    "code": "420",
    "étudiants": [
      {
        "num_ad": "20183383518",
        "nom": "Patoche",
        "prénom": "Alain"
      },
      {
        "num_ad": "20163383518",
        "nom": "Leroy",
        "prénom": "Simba"
      },
      {
        "num_ad": "20173383518",
        "nom": "Fafar",
        "prénom": "Marie"
      },
      {
        "num_ad": "20153383518",
        "nom": "Couturier",
        "prénom": "Sylvain"
      }
    ]
    }
```

Figure 10: Relation implémenté en imbriquant les documents associés (kba_demo.prog_etudiants)

Ajouter des références vers des documents de type étudiant dans un champ tableau du document programme

```
      {
         "_id": "e001",
         "num_ad": "20183383518",
         "nom": "Patoche",
         "prénom": "Alain"
       }
       {
         "_id": "e002",
         "num_ad": "20163383518",
         "nom": "Leroy",
         "prénom": "Simba"
       }
       {
         "_id": "e003",
         "num_ad": "20173383518",
         "nom": "Fafar",
         "prénom": "Marie"
       }
       {
         "_id": "e004",
         "num_ad": "20153383518",
         "nom": "Couturier",
         "prénom": "Sylvain"
       }

++++++++++++++++++++++++++++++++++++++++

       {
          "nom": "Informatique",
          "code": "420",
          "étudiants": [ "e001", "e002", "e003", "e004"]
       }
```

Figure 11: La relation est implémentée en conservant des références sur les documents de type étudiant. ***(kba_demo. prog_etudiants_ref)***

Ajouter une référence vers des documents de type programme dans un document étudiant

```
  {
    "_id": "p001",
    "nom": "Informatique",
    "code": "420"
  }

  {
    "_id": "p002",
    "nom": "Soins infirmiers",
    "code": "320"
  }

  {
    "_id": "p003",
    "nom": "Génie électrique",
    "code": "430"
  }

  {
    "_id": "p004",
    "nom": "Arts visuels",
    "code": "412"
  }

  {
    "_id": "p005",
    "nom": "Éducation de l'enfance",
    "code": "300"
  }

++++++++++++++++++++++++++++++++++++

  {
    "_id": "e001",
    "num_ad": "20183383518",
    "nom": "Patoche",
    "prénom": "Alain",
    "prog": "p001"
  }
```

*Figure 12: : La relation est implémentée en conservant des références sur les documents de type programme. (**kba_demo. etudiants-prog_ref**)*

Classes C# pour le modèle programmes -- étudiants

```
    [BsonIgnoreExtraElements]
    class Programme
    {
        [BsonId]
        public ObjectId Id { get; set; }
        [BsonElement("nom")]
        public string Nom { get; set; }
        [BsonElement("code")]
        public string Code { get; set; }
        [BsonElement("étudiants")]
        public List<Etudiant> Etudiants { get; set; }

        public Programme(string nom, string code)
        {
            Nom = nom;
            Code = code;
            Etudiants = new List<Etudiant>();
        }
    }

    [BsonIgnoreExtraElements]
    class Etudiant
    {
        [BsonElement("nom")]
        public string Nom { get; set; }
        [BsonElement("prénom")]
        public string Prénom { get; set; }

        public Etudiant(string nom, string prénom)
        {
            Nom = nom;
            Prénom = prénom;
        }
    }
```

```
mongoimport --db kba_demo --collection prog_etudiants --file
"C:\Users\beaul\OneDrive - Collège Lionel-
Groulx\Cours\KBA\Privé\Exercices\KBA_MongoDB\prog-etudiants.txt"


mongoimport --db kba_demo --collection etudiants --file
"C:\Users\beaul\OneDrive - Collège Lionel-
Groulx\Cours\KBA\Privé\Exercices\KBA_MongoDB\etudiants.txt"


mongoimport --db kba_demo --collection programmes --file
"C:\Users\beaul\OneDrive - Collège Lionel-
Groulx\Cours\KBA\Privé\Exercices\KBA_MongoDB\programmes.txt"


mongoimport --db kba_demo --collection prog_etudiants_ref --file
"C:\Users\beaul\OneDrive - Collège Lionel-
Groulx\Cours\KBA\Privé\Exercices\KBA_MongoDB\prog-etudiants_ref.txt"


mongoimport --db kba_demo --collection etudiants_prog_ref --file
"C:\Users\beaul\OneDrive - Collège Lionel-
Groulx\Cours\KBA\Privé\Exercices\KBA_MongoDB\etudiants-prog_ref.txt"


mongoimport --db kba_demo --collection joueurs --file
"C:\Users\beaul\OneDrive - Collège Lionel-
Groulx\Cours\KBA\Privé\Exercices\KBA_MongoDB\joueurs.txt"
```