
mongoDB

Plan

- Introduction
- Rappel sur JSON
- Présentation
- Comparaison avec SQL
- Requêtes
- Se connecter depuis Node.js

Introduction:

- MongoDb est un SGBD:
  - Open source
  - NoSQL
  - Orienté document
  - Scalable

Humongous : énorme ou immense

Rappel sur le JSON

- Le JSON (JavaScript Object Notation) est un format de représentation textuelle des données dérivé de la notation des objets du langage JavaScript. Toutefois il est indépendant du JavaScript et de tout autre langage de programmation.

- Le JSON permet de représenter de l'information structurée comme le permet XML par exemple.

Objet JSON

- Un objet JSON a pour but de représenter des informations (valeurs) accompagnées d'étiquettes (champs) permettant de les identifier.
- Un objet JSON ne comprend que deux types d'éléments structurels :
  - des ensembles de paires nom/valeur
  - des listes ordonnées de valeurs
- Les valeurs représentent trois types de données :
  - des objets
  - des tableaux
  - des valeurs génériques de type tableau, objet, booléen, nombre, chaîne ou null
- Les champs par contre ne peuvent être que des chaînes de caractères.

Exemple simple d'objet JSON

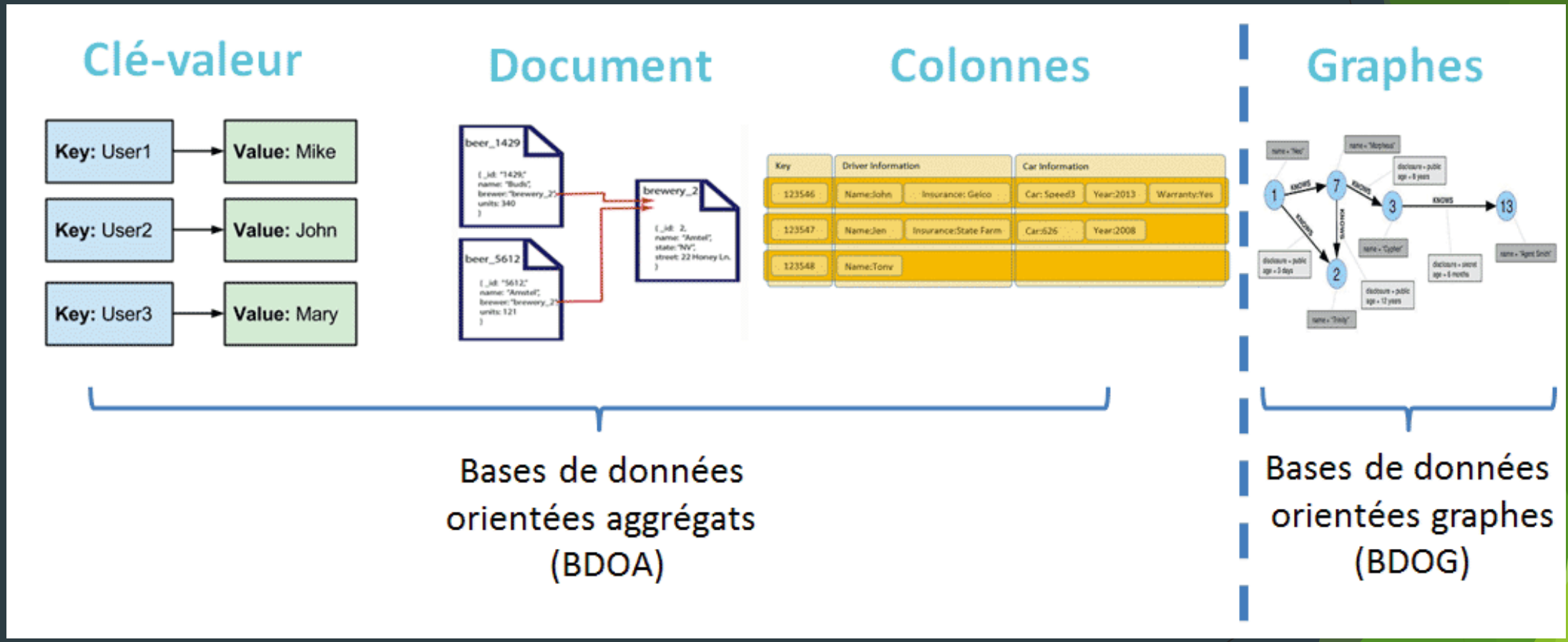
```
{
   name: "sue",                          ← field: value
   age: 26,                              ← field: value
   status: "A",                          ← field: value
   groups: [ "news", "sports" ]          ← field: value
}
```

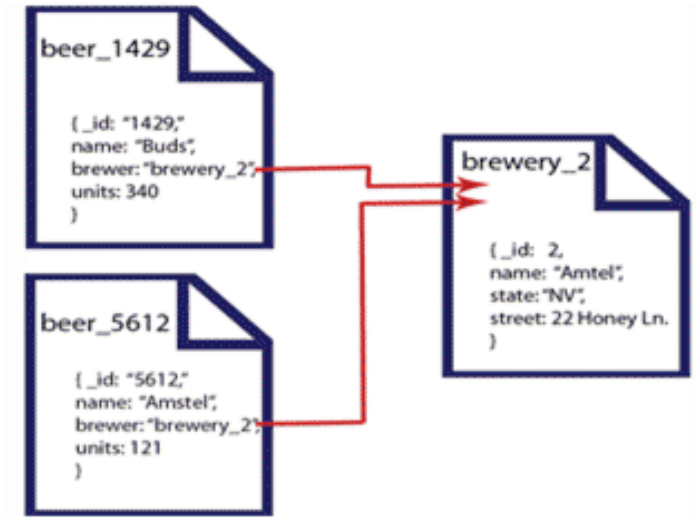
Exemple 2:

```
{
"actor": { "name": "De niro", "nickname" : "Bobby" },
"parts" : [
{"character" : "Neil McCauley", "film" : "Heat"},
{"character" : "Don Corleone","film" : "The godfather pt 2"}
]
}
```

Présentation de MongDB

Les 4 types de SGBD NoSql





Requétage plus complet

Flexibilité

Evolutif au cours du temps

Duplication des données

Cohérence pas forcément assurée

Exemples d'implémentation

CouchDB relax

mongoDB

terrastore

Orienté documents, Qu'est ce cela signifie ?

- Dans un système de base de données relationnelles les données sont stockées par ligne dans des tables. Et il est souvent nécessaire de faire des jointures sur plusieurs tables afin de tirer des informations assez pertinentes de la base.

- Dans MongoDB, les données sont modélisées sous forme de document sous un style JSON.

- On ne parle plus de tables, ni d'enregistrements mais de collections et de documents. Ce système de gestion de données nous évite ainsi de faire des jointures de tables car toutes les informations propres à un certain donnée sont stockées dans un même document.

SGBDR

Table **Acteur**:

id	nom	prenom
1	Johansson	Scarlett
2	Phoenix	Joaquim

Table **Acteur_Film:**

film_id	acteur_id
1	1
1	2
2	1

Table **Film**:

id	titre
1	Her
2	Avengers

NoSQL

```
{_id: "Her", acteurs : [{nom:"Johansson", prenom:"Scarlett"}, {nom:"Phoenix", prenom:"Joaquim"}]}
{_id: "Avengers", acteurs : [{nom:"Johansson", prenom:"Scarlett"}]}
```

#	title	stuff	moar
1	Bla bla	Mdr	xD
3	TEST	Lmfao	XML
4	Azerty	GUI	Lol

```
{ title: "Bla bla", stuff: "Mdr", moar: "xD" }
{ title: "TEST", stuff: "Lmfao", moar: "XML" }
{ title: "Azerty", stuff: "GUI", moar: "Lol" }
```

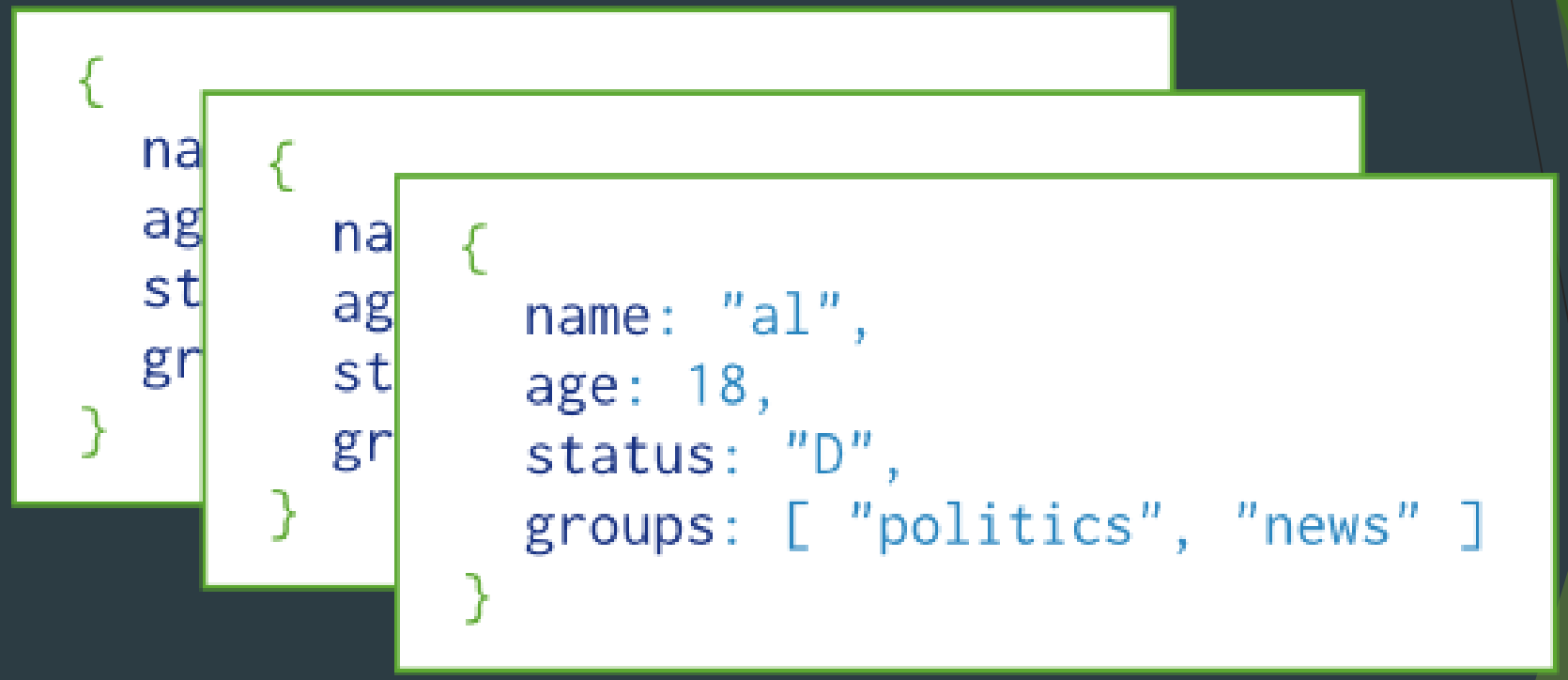
Des collections et des documents

Document

- Les documents sont les unités de base dans une base MongoDB. Ils sont équivalents aux objets JSON et sont comparables aux enregistrements d'une table dans une base de données relationnelle.
- Tout document appartient à une collection et a un champ appelé _id qui identifie le document dans la base de données.
- MongoDB enregistre les documents sur le disque sous un format BSON (JSON binaire).

Collection

- Une collection est un ensemble de documents, l'équivalent d'une table en relationnel. Contrairement aux bases de données relationnelles, les champs des documents d'une collection sont libres et peuvent être différents d'un document à un autre. Le seul champ commun est obligatoire est le champ "_id".


{
na
ag
st
gr
}
{
na
ag
st
gr
}
{
name: "al",
age: 18,
status: "D",
groups: ["politics", "news"]
}

Collection

Comparaison MongoDB, SQL

SQL	MongoDB
database	database
table	collection
enregistrement	document
colonne	champ
index	index
jointure	objet, dénormalisation
clef primaire	clef primaire (Dans MongoDB la clef primaire est automatiquement attribué au champ _id)
clef étrangère	reférence

Requêtes

- Une requête porte sur une collection
- On pourra spécifier dans la requête des conditions et des critères qui permettent d'identifier le document qui sera retourné au client.

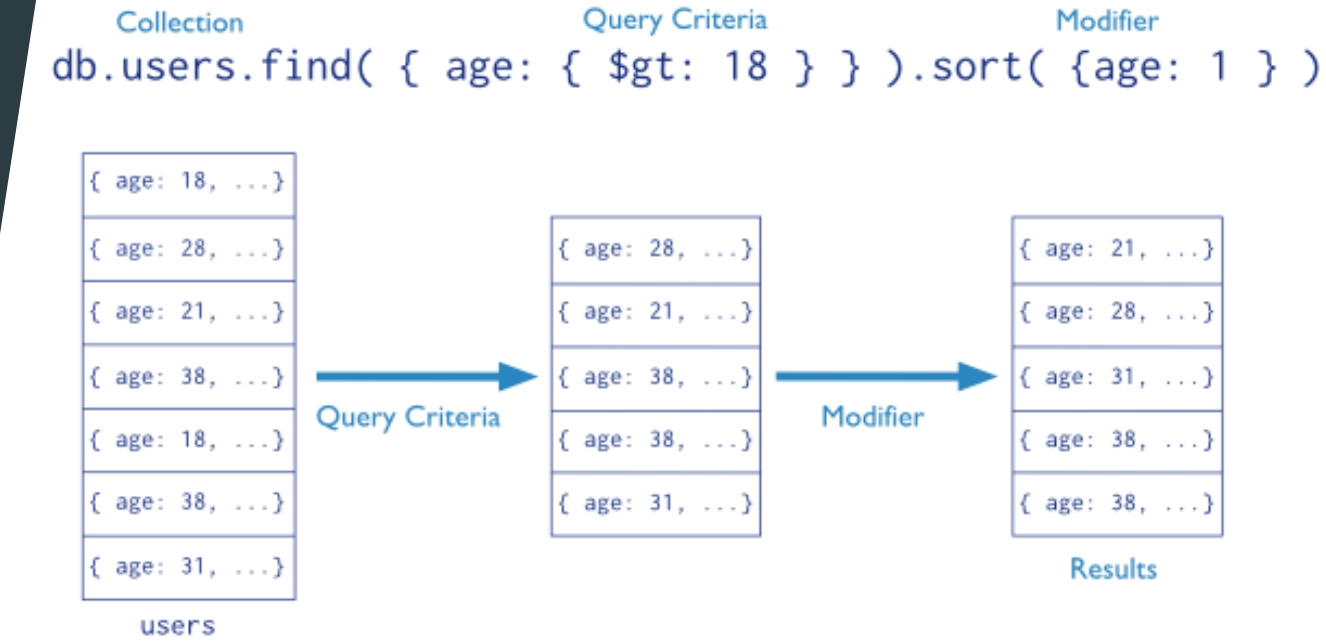
Projection

db.mycollection.find(arg1[, arg2])

- pour faire une projection dans une collection (select)

•arg1 est un objet JSON qui permet de faire une restriction (where)

•arg2 : objet JSON contenant les champs à projeter



Insertion

db.mycollection.insert(arg)

- pour insérer des données dans une collection
- arg : Objet JSON ou tableaux d'objet JSON correspondant au(x) document(s) à insérer dans la collection
- On peut insérer un seul document, comme on peu insérer un ensemble de documents.
- Le champ _id n'est pas obligatoire lors de l'insertion du document. Il sera automatiquement généré par le moteur de base de données s'il n'est pas donné.

```
db.Movies.insert(
{
    _id:"1",
    "nom" : "Training Day",
    "acteurs" : [
        {
            "nom" : "Washinton",
            "prenom" : "Denzel"
        },
        {
            "nom" : "Hawke",
            "prenom" : "Ethan"
        }
    ]
}
)
```

Mise à jour

- db.mycollection.update(arg1[, arg2])
- mettre à jour une collection ou un champ d'un document
- arg1 : permet de faire la restriction sur le(s) document(s) à mettre à jour
- arg2 : attribue une nouvelle valeur à un champ : utilisation de la fonction $set

db.Movies.update({_id:"1"}, {$set:{nom:"Memento"}})
pour renommer le film qui a pour _id 1 par "Memento".

Suppression

db.mycollection.remove(arg)

- arg : permet de faire la restriction sur le(s) document(s) à mettre à supprimer

db.Movies.remove({_id:"1"})
enlève de la collection Movies le document avec _id 1.

Autres fonctions

MongoDB propose également les fonctions:

- sort(),
- count(),
- $push (ajouter un élément à un tableau),
- $pop(enlever un élément d'un tableau),
- $gt(plus grand que),
- $lt(plus petit que),
- $gte(plus grans ou égal à),
- $or,
- $and,
- $in,
- $all,
- $exist,
- $type,
- $regex...

Se connecter depuis Node.js

```
// Récupération du client mongodb
var mongoClient = require('mongodb').MongoClient;

// Paramètres de connexion
var url = 'mongodb://localhost/db_name';

// Connexion au serveur avec la méthode connect
mongoClient.connect(url, function (err, db) {
    if (err) {
        return console.error('Connection failed', err);
    }
    console.log('Connection successful on ', url);

    // Nous allons travailler ici ...

    // Fermeture de la connexion
    db.close()
});
```

Insertion d'un document

```
// Récupération de la collection users
var collection = db.collection('users');

// Création de deux objets users
var user1 = {firstName: 'Foo', lastName: 'Fighters'};
var user2 = {firstName: 'Bob', lastName: 'Dylan'};

// Enregistrement de plusieurs objets en db avec insertMany
collection.insertMany([user1, user2], function (err, result) {
    if (err) {
        console.error('Insert failed', err);
    } else {
        console.log('Insert successful', result);
    }

    db.close()
});

// Enregistrement d'un objet en db avec insertOne
//collection.insertOne(user1);
```

Mise à jour d'un document

```
// Mise à jour d'un document
collection.updateOne({firstName: 'Bob'}, {$set: {lastName: 'Marley'}}, function (err, result)
    if (err) {
        console.error('Update failed', err);
    } else {
        console.log('Update successful', result);
    }

    db.close()
});

// Mise à jour de plusieurs documents
//collection.updateMany();
```

Suppression d'un document

```
// Suppression d'un document
collection.deleteOne({firstName: 'Bod'}, function (err, result) {
    if (err) {
        console.error('Remove failed', err);
    } else {
        console.log('Remove successful', result);
    }

    db.close()
});

// Suppression plusieurs documents
//collection.deleteMany();
```

Rechercher un document

```
// Récupération de tous les documents de la collection
collection.find().toArray(function (err, result) {
    if (err) {
        console.error('Find failed', err);
    } else {
        console.log('Find successful', result);
    }

    db.close()
});

// Récupération à partir d'une requête (firstName like b)
//var query = {firstName: /b/ };
//collection.find(query);

// Récupération du 1er document avec l'aggregationCursor limit
//collection.find().limit(1);
```