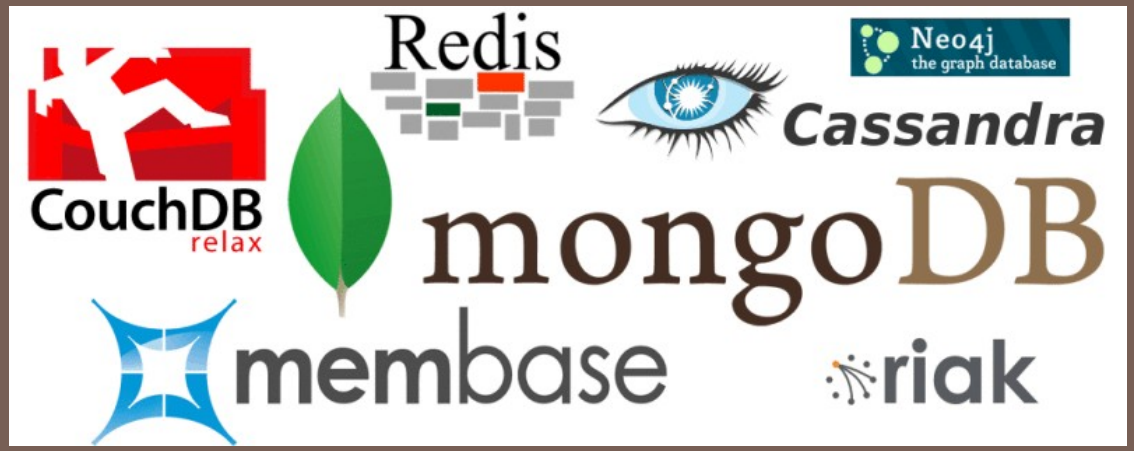


NoSQL : mongoDB

BD et SI

Frédéric Moal
Master 1 MIAGE

année 2018/2019

Plan de la présentation

- JSON
- MongoDB
- Requêtes en client Mongo
-

MongoDB

Pourquoi MongoDB ?

- Une BDD mature
  - Doodle, github, Ebay, McAfee, Adobe, Craiglist, ...
  - Une doc très fournie et une grande communauté
- Adaptation facile
  - Orientée document - format JSON
  - SQL > MongoDB : Easyyy ! mouaip...
  - Utilise Javascript
- Scalable
  - MapReduce en natif ou avec le connecteur ;
  - Réplication, sharding automatique
- Et plein d'autres trucs cools ...

JSON

Rapide rappel

JSON

JSON « **J**ava**S**cript **O**bject **N**otation » est un format d'échange de données, facile à lire par un humain et interpréter par une machine.

Basé sur JavaScript, il est complètement indépendant des langages de programmation mais utilise des conventions qui sont communes à toutes les langages de programmation (C, C++, Perl, Python, Java, C#, VB, JavaScript,….)

Deux structures :

- Une collection de clefs/valeurs Object
- Une collection ordonnée d'objets Array

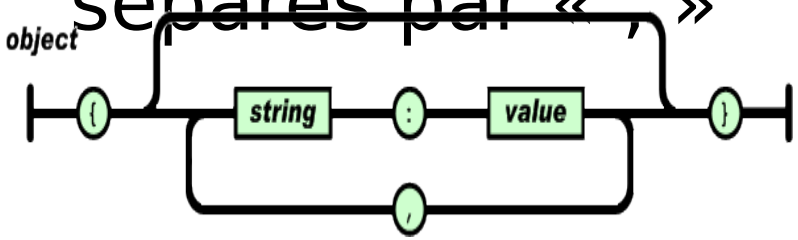
Exemple de JSON

```
{"menu": {
    "id": "file",
    "value": "File",
    "popup": {
      "menuitem": [
        {"value": "New", "onclick": "CreateNewDoc()"},
        {"value": "Open", "onclick": "OpenDoc()"},
        {"value": "Close", "onclick": "CloseDoc()"}
      ]
    }
 }}
```

JSON

Objet

Commence par un « { » et se termine par « } » et composé d'une liste non ordonnée de paire clefs/valeurs. Une clef est suivie de « : » et les paires clef/valeur sont séparés par « , »

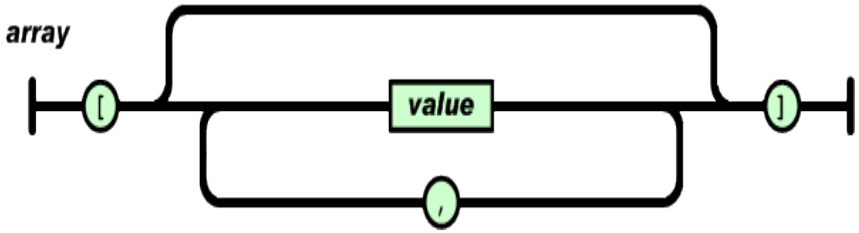



```
{ "id": 51,
"nom": "Mathematiques 1", "resume":
"Resume of math ", "isbn": "123654",
"categorie":
     {
     "id": 2, "nom": "Mathematiques",
     "description": "Description of
     mathematiques "
     },
"quantite": 42,
"photo": ""
}
```

JSON

ARRAY

Liste ordonnée d’objets commençant par « [« et se terminant par «] », les objets sont séparés l’un de l’au

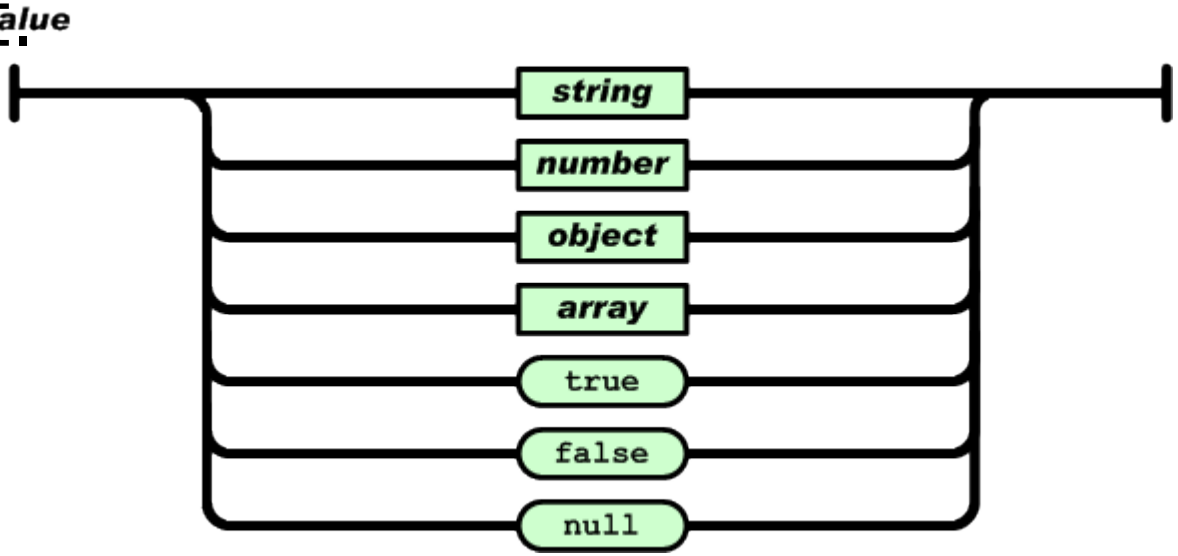



```
[
     { "id": 51,
     "nom": "Mathematiques 1",
     "resume": "Resume of math ",
     "isbn": "123654",
     "quantite": 42,
     "photo": ""
     },
     { "id": 102,
     "nom": "Mathematiques 1",
     "resume": "Resume of math ",
     "isbn": "12365444455",
     "quantite": 42,
     "photo": ""
     }
]
```

JSON

Value

Un objet peut être soit un string entre « ""» ou un nombre (entier, décimal) ou un boolean (true, false) ou null ou un objet.



MongoDB

retour

MongoDB

- MongoDB est fourni avec un client en ligne de commande
- 2 parties :
  - le serveur, écoute [en général] sur le port 20017
  - Le client qui s’y connecte

MongoDB avec docker

- On lance le serveur :

  docker run --name mongoserver -d mongo:tag

- On lance un client (mongoDB shell) :

  docker run -it --link mongoserver:mongoserv --rm mongo mongo --host mongoserv test

MongoDB avec docker

- Utilitaire externe : mongoimport (export) pour importer un jeu de données complet en JSON, csv, ...

MongoDB - les bases

- Le vocabulaire
  - database => database
  - table => collection
  - entrée, ligne, tuple => document
- Création, insertion
  - use db1
  - db.createCollection(”collection_name”)
  - db.collection_name.insert({field1 : ”value_string”, field2 : value_int, ..., fieldN: [arrayV1, ..., arrayVn]})

MongoDB - les bases

- Exemple :

  use *maBD;*

  db.createCollection('*users*');

  db.users.save ( ); //pas de quotes

- Requêtes orientées documents

  > db.users.find({ login : "james" });

  > db.users.find({ address.city : "London" }, {name : 1});

  > db.users.find({ age : { $gt : 40 } });

  //$gt, $gte, $lt, $lte, $ne, $in, $nin, $or, $and, $exists, $type…

  > db.users.find({ name : { $regex : "james", $options : "i" } });

  > db.users.count();

```
{
"_id": 100,
"name": "James Bond",
"login": "james",
"age": 50,
"address": {
"street" : "42 Class Street",
"city": "London"
}
}
```

MongoDB - MAJ

- Pas de transactions
- Format général : update({filtre},{operations}
- Opérations : modifications atomiques de documents
  - $set – Modifie une valeur
  - $unset – Supprime un attribut
  - $inc – Incrément
  - $push – Ajout dans un tableau
  - $pushAll – Plusieurs valeurs dans un tableau
  - $pull – Supprimer une valeur de tableau
  - $pullAll – Supprimer plusieurs valeurs

MongoDB - MAJ

```
> db.users.update( { “_id” :
ObjectId("4efa8d2b7d284dad101e4bc7") } ,
   { “$inc” : { “age” : 1 } }
> db.collection_name.update( { _id: 1 },
{
$inc: { field1: 5 },
$set: { field2: ”ABC123” }
} )
```

MongoDB - suppression

- Suppression : remove({filtre})

```
db.collection_name.remove({ })
//Delete the collection
db.collection_name.remove({field1 :
   value}) //Delete doc that match
```

MongoDB – les requêtes

- Exemple - Select, Projection

  db.collection_name.find({ }, {field1: 1})

  db.collection_name.find({ }, {field2: 0})

- Exemple – Where

```
db.collection_name.find({field: {$gt:v1,
   $lt:v2} });
// where v1 < field < v2
db.collection_name.find({field: value});
// where field == value
```

MongoDB – les requêtes

- Exemple - Select, Projection

  db.collection_name.find({ }, {field1: 1})

  db.collection_name.find({ }, {field2: 0})

- Exemple – Where

```
db.collection_name.find({field: {$gt:v1,
   $lt:v2} });
// where v1 < field < v2
db.collection_name.find({field: value});
// where field == value
```

MongoDB – les requêtes

- Order By équivalent

  db.collection_name.find().sort({field1: 1, field2: -1}) // Order by field1 ASC, field2 DESC

- Limit

  db.collection_name.find().limit(1) // Un doc

MongoDB – les requêtes

- Exemple - Group By

db.collection_name.aggregate([{$group : {_id : ”$field”} }]) //

- Group by field

db.collection_name.aggregate([{$group : {_id : ”$user”, count: {$sum: 1} } , {$match : {count : {$gte : 10} } } }]) // Group by field having count > 10

db.collection name.aggregate([{$group : {_id : ”$user”, count: { $sum: 1} }, {$match : {count : {$gte : 10 } } }, {$sort : {count : -1} }])

MongoDB – les requêtes

- Exemple :
- Valeurs distinctes

  db.users.distinct(“address.city”)
- Aggrégats

  db.users.aggregate(**[{$sort : {age : 1}}]);**

  db.users.aggregate(**[{$project : {login : 1, name:1}}]);**

  db.users.aggregate([

  {$match : {address.city : “London”}},

  {$sort : {age : 1}]);

  db.users.aggregate([

  {$group : {_id : “$address.city”, number : {$sum : 1}},

  {$sort : {number : 1}]);
- Séparer les instances d'un tableau

  db.users.aggregate([{$unwind : “$array”}]);

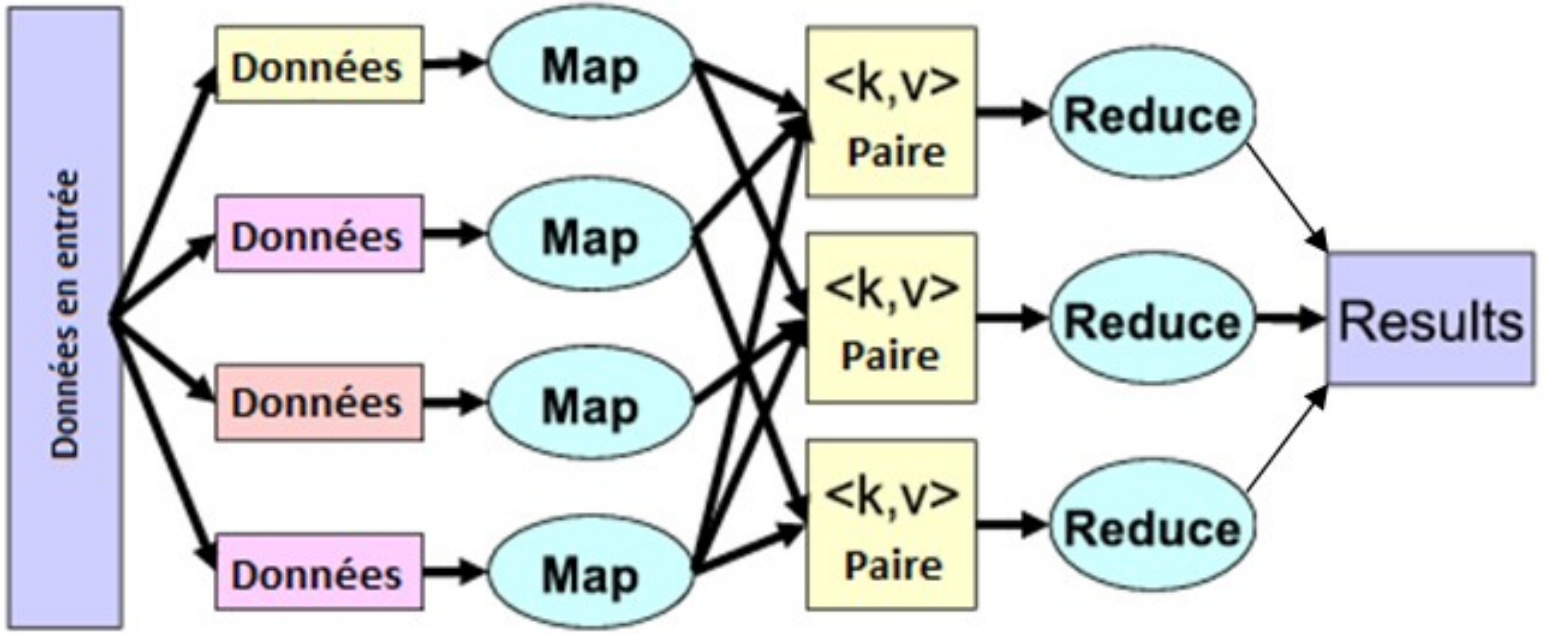
MongoDB – les requêtes

- Exemple – Cursor

```
var cursor = db.collection_name.find()
while (cursor.hasNext()) {
var doc = cursor.next();
db.collection_name.update({_id : doc._id},
  {$set : {field : value} })
}
```

MongoDB - MapReduce

- En natif



MongoDB - MapReduce

// Map : fonction appliquée à chaque document. Filtre et produit les éléments (clé/valeur) en sortie

var mapFunction = function () {**emit(this.address.city, this.login);}**

// Reduce : regroupe tous les documents sur la clé. Applique une fonction d'agrégat sur chaque ensemble.

var reduceFunction = function (**key, values) {return Array.count(values);}**

//queryParam : Paramètres d'exécution

var queryParam = {query : {}, out : "**result_set"}**

//query : Permet de filtrer les documents AVANT le map, utilise l'API MongoDB. Utile pour l'utilisation des indexes

//out : collection de stockage du résultat

db.publis.mapReduce(mapFunction, reduceFunction, queryParam);

//Consulter le résultat

db.**result_set.find();**

MongoDB - MapReduce

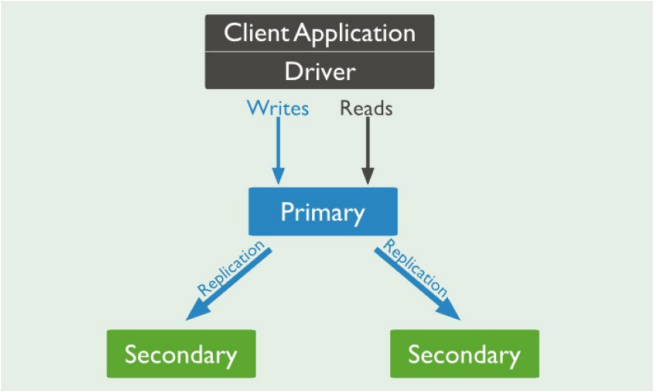
- Exemple :

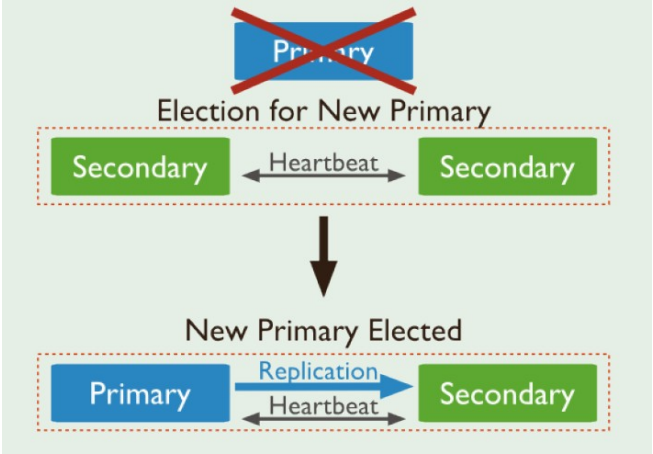
```
var mapFunction = function () {
  if(this.genres.contains(“Action”)) emit(this.directors[0], this.rank);
  if(this.actors.contains(“Harrison Ford”)) emit(this.directors[0], 1);
}
var reduceFunction = function (key, values) {return Array.sum(values);}
```


```
{
"title" : "Star Wars: Episode VII",
"directors" : ["J.J. Abrams"],
"release_date" : "2015-12-16",
"genres" : ["Action","Adventure","Fantasy","Sci-Fi"],
"plot" : "A continuation of the saga created by George Lucas.",
"rank" : 168,
"year" : 2015,
"actors" : ["Mark Hamill","Harrison Ford","Carrie Fisher"]
}
```

MongoDB – Replica Set

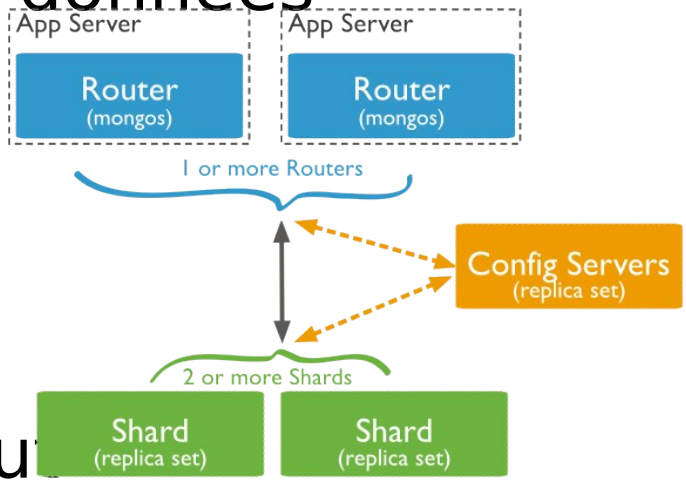
- Réplication d'un serveur
  - Asynchrone
    - Primary server : écritures
    - Secondary servers : lectures
      - Mise à jour via oPlog (fichier de log)
  - Distribue la charge de lecture (requêtes)
  - Tolérance aux pannes
    - Élection d'un nouveau serveur primaire
    - Besoin d'un serveur arbitre

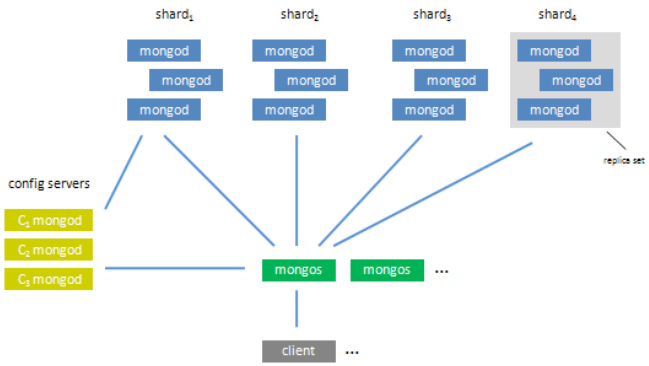




MongoDB - Sharding

- Distribution sur un cluster de machines
  - Equilibrage de charge des données
  - Combinaison avec *Replica*
  - Nécessite :
    - *Config Servers*
    - *Mongos (routeur)*
  - Partitionnement sur attribu
    - Ranged-based
    - Hash-based





MongoDB - Monitoring

- Commande : mongostat
- Commande : mongotop
- ~~Interface HTTP : http://localhost:28017~~

MongoDB - cohérence

- Unacknowledged
  - On ne sait pas si le requête d'écriture est reçue
- Acknowledged
  - Mode **par défaut :**
  - On ne sait que la requête d'écriture est reçue, on ne sait pas si elle est persistée
- Journaled
  - La requête est reçue et persistée dans le journal de la DB, elle est donc persistée, mais pas sur tout le cluster
- Replica Acknowledged
  - La requête d'écriture est forcément propagée aux noeuds du cluster répliqué

MongoDB – Outils

- Répertoire bin/
- **mongod : Lancement du serveur (daemon)**
- **mongo : shell pour exécuter les commandes**
- **mongoimport *: importation d'un fichier de données***
- **mongostat : Status du process en cours (inserts,** updates, queries, commands, memory…)
- **mongotop : Temps réparti en lecture vs écriture**
- **mongos : Service de routage (sharding)**