Bases de données NoSQL

P.Mathieu

Université de Lille

LP DA2I Lille
http://www.iut-a.univ-lille.fr
prenom.nom@univ-lille.fr

8 septembre 2019

NoSQL

NoSQL (Not Only SQL)

- De plus en plus d'applications créent des volumes massifs de données
- Les structures doivent évoluer rapidement
- Les données sont souvent semi-structurées voir non structurées
- Faciliter le lien avec la programmation orientée objets

Différents types de SGBD

- Orientés documents
  chaque clé est associée à une structure complexe appelée Document (MongoDB, CouchDB)
- Orientés Graphes
  adaptées au stockage de graphes sociaux (Neo4J, Giraph)
- Orientés Clé-Valeurs
  Chaque information est stockée selon un modèle clé-valeur (Redis, Berkeley DB, DynamoDB)
- Orientés colonnes
  Spécialement optimisées pour de larges ensembles de données (Cassandra, HBase)

NoSQL
JSON

JSON (JavaScript Object Notation) est un format de données textuelles dérivé de la notation des objets du langage JavaScript.

Utilisé principalement pour l'échange de donnnées entre applications (web-services)

Un document JSON comprend deux types d'éléments structurels :

- des ensembles de paires `clé:valeur` , entre accolades `{}`
- des listes ordonnées de valeurs, entre crochets `[]`

Types de données : chaînes (entre guillemets), nombres, booléens (true, false), null, objet Json

NoSQL
Exemple

```
{
   _id: ObjectId(7df78ad8902c)
   title: 'MongoDB Overview',
   description: 'MongoDB is no sql database',
   by: 'tutorials point',
   url: 'http://www.tutorialspoint.com',
   tags: ['mongodb', 'database', 'NoSQL'],
   likes: 100,
   comments: [
      {
         user:'user1',
         message: 'My first comment',
         dateCreated: new Date(2011,1,20,2,15),
         like: 0
      },
      {
         user:'user2',
         message: 'My second comments',
         dateCreated: new Date(2011,1,25,7,45),
         like: 5
      }
   ]
}
```

Seule contrainte : pas 2 fois la même clé dans un même document

MongoDB
Principe

- Base de données NoSQL **orientée Documents**, open-source, écrite en C++
- Mode Client Serveur
- Téléchargeable sur mongodb.org
- Manuel sur https://docs.mongodb.com/manual/
- Données (document) représentées au format BSON
- Langage d'interrogation "maison"
- Pas de système transactionnel ni d'intégrité référentielle
- Case sensitive

```
{"foo" : 3} est différent de {"foo" : "3"}
{"foo" : 3} est différent de {"Foo" : 3}
```

MongoDB
Vocabulaire

Université de Lille

RDBMS	MongoDB
Database	Database
Table	Collection
Ligne	Document
colonne	Field
Jointure	`$lookup` , ou Embedded Documents
Primary Key	`_id`

MongoDB
Exemple de collection

Université de Lille

```
{nom: "paul", naiss: 1989, ville: Lille,
  tel : {fixe: 0311111111 port:0611111111}}  email:["paul@gmail.com","paul@ovh.com"]}
{nom: "pierre", naiss: 1997, ville: Lille,
  tel : {fixe: "0311111111", port:"0611111111"},  email:["pierre@gmail.com"]}
{nom: "jean", naiss: 1999, ville: Lille,
  tel : {fixe: "0355555555", port:"0655555555"},  email:["jean@gmail.com"]}
{nom: "lucie", naiss: 1998, ville: Lille,
  tel : {fixe: "0333333333", port:"0633333333"},  email:["lucie@gmail.com"]}
{nom: "lucas", naiss: 2000, ville: Lille,
  tel : {fixe: "0399999999", port:"0699999999"},  email:["lucas@gmail.com"]}
{nom: "iris", naiss: 1999 , ville: Lille,
  tel : {fixe: "0322222222", port:"0622222222"},  email:["iris@gmail.com"]}
{nom: "elsa", naiss: 2002, ville: Lille,
  tel : {fixe: "0377777777", port:"0677777777"},  email:["elsa@gmail.com","elsa@yahoo.fr"]}
{nom: "farid", naiss: 2002, ville: Lille,
  tel : {fixe: 0344444444 port:0644444444},  email:["farid@yahoo.fr"]}
```


MongoDB
Pros and Cons

Université de Lille

- L'objectif d'un SGBDR est d'éviter toute redondance et de faciliter aussi bien la lecture que la mise à jour (donc MCD, Tables et intégrité référentielle)
- L'objectif qu'un SGBD NoSQL est d'éviter toute opération couteuse (jointure), et de faciliter les recherches
- Un SGBDR nécessite un schéma (éventuellement modifiable)
- Un SGDB NoSQL ne possède aucun schéma. Une même collection peut contenir des documents de schémas complètement différents

les avantages des uns sont les inconvénients des autres !

MongoDB
Installation

Université de Lille

- Download : `https://www.mongodb.org/downloads`
- Serveur :

```
mongod --dbpath /tmp
mongod --config /usr/local/etc/mongod.conf
```

- Fichiers :

```
configuration file (/usr/local/etc/mongod.conf)
log directory path (/usr/local/var/log/mongodb)
data directory path (/usr/local/var/mongodb)
```

- port utilisé par défaut : 27017
- Client :

```
mongo -v
mongo --quiet
```

MongoDB

Démarrage

Université de Lille

- Par défaut 3 bases sont déjà créées (`admin`, `config`, `local`) et on se trouve dans une base `test` vide

```
show dbs
db
```

- Changer de base : `use mabase`
- Le shell est un interpreteur Javascript complet
- Au lancement, exécution du fichier `/.mongorc.js`
- par défaut mono-utilisateur et uniquement connexions locales.

MongoDB

Le CRUD en un slide !

Université de Lille

les commandes générales sont préfixées par `db.` Celles d'une collection `matable` sont préfixées par `db.matable.`

```
use mabase
db.createCollection("matable")
show collections
db.matable.insert({nom:'dupond' , prenom:'paul' , age:15})
db.matable.insert({nom:'durand' , prenom:'paul' , age :25})
db.matable.find()
db.matable.find({'prenom':'paul'},{'nom':1})
db.matable.count()
db.matable.update({nom:'durand'}, {$set:{prenom:"lucie"}})
db.matable.remove({nom:'durand'})
db.matable.remove({})
db.matable.drop()
quit()
```


MongoDB

La recherche avancée

Université de Lille

```
db.matable.find({prenom: 'paul'})
db.matable.find({prenom:'paul'},{nom:1})   (uniquement le no
db.matable.find({prenom:'paul'},{nom:0})   (tout sauf le nom
db.matable.find({age: {$gt:18}})           (lt, lte,gt, gte,

db.matable.find({$or: [{key1: value1}, {key2:value2}]})
```

Le `$and` est équivalent à une virgule

`find` peut aussi etre accolé à `.sort({KEY:1})` ou `.limit(n)` ou `.skip(n)` ou `.pretty()` ou `milli`

MongoDB

Le pipieline Aggregate

Université de Lille

`aggregate` est l'opération principale pour les interrogations complexes

SQL op	Aggregate op
where	$match
group by	$group
having	$match
select	$project
order by	$sort
limit	$limit
sum	$sum
count	$count

Voir
https://docs.mongodb.com/manual/reference/sql-aggregation-comparison

MongoDB
Le pipeline Aggregate

Université de Lille

```
db.matable.aggregate([
   { $match: { nom: "durand" } },
   { $group: { _id: "$prenom", total: { $sum: "$age" } } },
   { $sort: { total: -1 } }
   ])
```

Opérations d'agrégation : `$sum`, `$avg`, `$min`, `$max` , `$first`, `$last`

```
db.matable.aggregate([{$group: {_id: "$nom", result: {$sum : 1 }}}])
db.matable.aggregate([{$group: {_id: "$nom", result: {$sum : "$age" }}}])
```

Utiliser `db.matable.explain().aggregate` pour avoir des détails sur l'exécution

MongoDB
Créer des relations entre les collections

Université de Lille

En mongo il y a 2 méthodes pour créer une relation

- par imbrication
- par référence (avec `aggregate` et `lookup` pour la jointure)

```
db.etudiants.aggregate(
[ {$lookup: {from: "notes" , localField: "_id" ,
             foreignField: "etu_id" , as :"notes"} }]
```


MongoDB
Une collection volumineuse !

Université de Lille

Comme on est dans un interpréteur Javascript, il est très facile de générer des millions de documents !

```
db.createCollection("users")

for (i=0; i<1000000; i++) {
    db.users.insert({"i" : i,
                     "username" : "user"+i,
                     "age" : Math.floor(Math.random()*120),
                     "created" : new Date()
                    } );
}

db.stats()
db.users.find({username: "user101101"})
db.users.ensureIndex({"username" : 1})
```

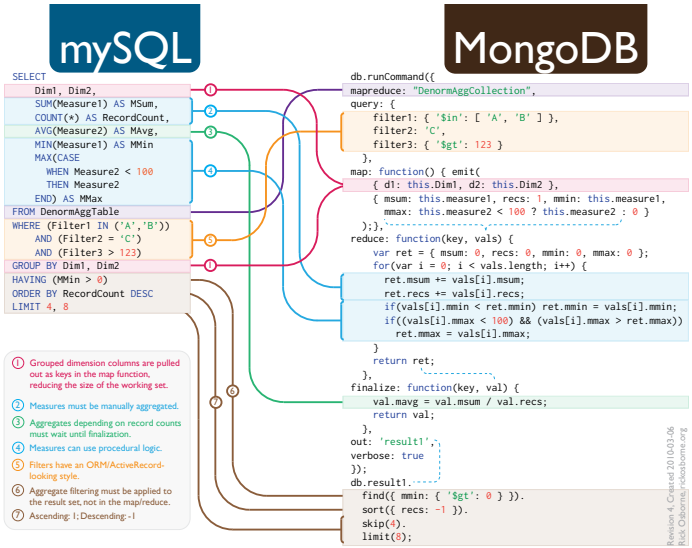

MongoDB
Map-Reduce

Université de Lille

- MapReduce est un patron de conception de requetes qui consiste à partitionner les données pour ensuite les traiter en parallèle
- Patron notamment mis en place par Google dans son framework Hadoop
- MapReduce s'appuie principalement sur deux fonctions : `map` qui associe des données à des clés et `reduce` qui collecte les données d'une clé particulière pour effectuer les calculs

MongoDB

Equivalence entre SQL et MapReduce



MongoDB

Java et Mongo

```java
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;

import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;

public class Test {

   public static void main( String args[] ) {

      // Creating a Mongo client
      MongoClient mongo = new MongoClient( "localhost" , 27017 );

      // Creating Credentials
      MongoCredential credential;
      credential = MongoCredential.createCredential("sampleUser", "myDb",
         "password".toCharArray());
      System.out.println("Connected to the database successfully");

      // Accessing the database
      MongoDatabase database = mongo.getDatabase("myDb");

      // Creating a collection
      database.createCollection("sampleCollection");
      System.out.println("Collection created successfully");

      // Retieving a collection
      MongoCollection<Document> collection = database.getCollection("myCollection");
      System.out.println("Collection myCollection selected successfully");
   }
```


MongoDB

Gestion des utilisateurs

Par défaut Mongo est accessible sans authentification, sans utilisateur et uniquement de la machine locale. Aucun utilisateur n'est créé.

Pour que l'authentification soit necessaire, lancer `mongod` avec `-auth` ou mettre dans le fichier de config `mongod.conf`

```
security:
authorization: enabled
```

Less utilisateurs doivent ensuite être créés dans la ou les bases concernées

MongoDB

Gestion des utilisateurs

```
use admin
db.createUser( {user: "phil", pwd: "phil", roles: [{"role:r
db.getUsers()    (ou show users)
db.changeUserPassword("username", "newPass")
db.revokeRolesFromUser("phil",[{role:"dbAdmin", db:"test"}]
db.grantRolesToUser("phil",["dbAdmin"])

db.dropUser("phil")
show roles
```

Si un utilisateur est ajouté à la base `admin`, il hérite automatiquement de ses permissions pour toutes les bases

MongoDB
Authentification

Deux manières différentes d'activer l'authentification

- En ligne de commande

```
mongo --authenticationDatabase "admin" -u "phil" -p
show collections
```

- Une fois connecté

```
mongo
use admin
db.auth("phil","phil")
use test
show collections
```


MongoDB
Accès à distance

Les adresses ip autorisées doivent être déclarées dans le `mongod.conf`

```
net:
  bindIp: 127.0.0.1 , 0.0.0.0
```

L'accès se fait par

```
mongo 134.206.153.87 --authenticationDatabase "admin"
```


MongoDB
Import - Export

- mongodump (dump en bson, binaire !)
- mongorestore
- mongoimport

```
mongoimport -d mabase -c matable --type csv --file fichier.csv --head
```

- mongoexport

```
mongoexport --db music --collection artists --out /data/dump/fichier.
```