

## MongoDB - TP

Raja CHIKY
Raja.chiky@isep.fr





#### Modèle document



- Collection de « documents »
- Modèle « clé/valeur », la valeur est un document semi-structuré hiérarchique de type JSON ou XML
- Pas de schéma pour les documents
- Structure arborescente : une liste de champs, un champ a une valeur qui peut être une liste de champs, ...
- CRUD
- BD les plus connues:
  - CouchDB, MongoDB, Terrastore, ...







- SGBD open source, orienté documents. Il est scalable, sans schéma, et non relationnel
- Utilisé par des sites web très connus et à fort trafic comme Sourceforge, Foursquare, Craiglist, Disqus.
- Cohérence des données:
  - Modification atomique des documents
    - L'opération est indivisible et une fois engagée elle sera menée à bien.
  - Pas de transaction native
    - Pour des raisons de performances et de scalabilité, mongoDB ne gère pas les transactions.
  - Possibilité de levée d'une exception
    - Il est possible de demander la levée d'une exception si une opération se passe mal, il est donc possible de s'assurer que les opérations s'effectuent correctement.



## **En Production**













theguardian



















The New Hork Times

http://www.mongodb.org/about/production-deployments/





## **BD Document**



SGBDR		MongoDB
Database	$\Rightarrow$	Database
Table, View	$\Rightarrow$	Collection
Row	<b>→</b>	Document (JSON, BSON)
Column		Field
Index		Index
Join		Embedded Document
Partition		Shard





#### **BD** Document



```
SGBDR
                MongoDB
                              > db.user.findOne({age:39})
Database
                Database
Table,
                Collection
                                  "_id" : ObjectId("5114e0bd42..."),
View
                                  "first": "John",
                Document
Row
                                  "last": "Doe",
                BSON)
                                  "age": 39,
Column
                Field
                                 "interests" : [
                Index
Index
                                      "Reading",
                Embedded
Join
                                      "Mountain Biking]
                                 "favorites": {
                Shard
Partition
                                      "color": "Blue",
                                      "sport": "Soccer"}
```





## **CRUD** example

```
> db.user.insert({
    first: "John",
    last : "Doe",
    age: 39
})
```

```
> db.user.find ()
{
    "_id": ObjectId("51..."),
    "first": "John",
    "last": "Doe",
    "age": 39
}
```

```
> db.user.remove({
    "first": /^J/
})
```





## MongoDB- Stockage des données

- Stockage au format BSON ("binary JSON")
- Types supportés:
  - String, Integer, Double, Date, Byte Array, Booléen, Null, BSON object et BSON Array
- Exemple dans une collection "users":

```
{
"_id": "c9167a15625045fb439c7078 ",
"username": "rchiky",
"firstname": "Raja",
"lastname": "Chiky"
}
```

 Drivers existant dans la plupart des langages de programmation

C, C++, JavaScript, Python, Perl, C# / .NET, Java, PHP, ...

- Flexibilité:
  - Authentification facultative
  - Génération automatique des collections
  - Accès à des fonctionnalités avancées



# Installation et configuration (Windows)



Ouvrir un shell cmd

>wmic os get osarchitecture (pour connaître l'architecture de son système d'exploitation (32 ou 64 bits)

Extraire l'archive dans « C:\ » par exemple

>cd /

>md data

(crée un répertoire data)

>md data\db

(crée un répertoire db)

>C:\mongodb\bin\mongod.exe

(lance le serveur)

• Ou

>C:\mongodb\bin\mongod.exe --dbpath "dossier data" (en spécifiant le dossier si c'est différent de c:\data\db)

Ouvrir un autre shell pour lancer le client

>C:\mongodb\bin\mongo.exe



## Installation et configuration (LINUX/MAC OS)



- www.mongodb.org/downloads
- tar zxvf mongodb-xxx.tgz (xxx:version choisie)
- mv mongodb-xxx /votredossier
- In -s mongodb-xxx mongodb
   (crée un lien symbolique vers le dossier contenant mongodb)
- sudo mkdir /data/db
   (par défaut, mongodb utilise ce dossier, il faut donc le créer)
- Sinon lui fournir le chemin de votre dossier au lancement

bin/mongod --dbpath /votreDossier





#### Lancement



```
macbook-de-isep:mongodb isep$ ./bin/mongod
./bin/mongod --help for help and startup options
Sun Nov 27 17:00:10
Sun Nov 27 17:00:10 warning: 32-bit servers don't have journaling enabled by default.
Please use -- journal if you want durability.
Sun Nov 27 17:00:10
Sun Nov 27 17:00:10 [initandlisten] MongoDB starting: pid=34382 port=27017 dbpath=/da
ta/db/ 32-bit host=macbook-de-isep.local
                                     isten] waiting for connections on port 27017
Sun Nov 27 17:00:10 [initandlisten]
Sun Nov 27 17:00:10 [initandlisten] ** NOTE: when using MongoDB 32 bit, you are limite
d to about 2 gigabytes of data
Sun Nov 27 17:00:10 [initandlisten] ** a Win see http://blog.mongodb.org/post/13778896
7/32-bit+limitationsll. Depending on your environment your output may vary.
Sun Nov 27 17:00:10 [initandlisten] **
                                     ** with --journal, the limit is lower and running, use the mongo JavaScript shell to connect t
Sun Nov 27 17:00:10 [initandlisten]
Sun Nov 27 17:00:10° [initandlisten] db version v2.0.1, pdfile version 4.5
Sun Nov 27 17:00:10 [initandlisten] git version: 3a5cf0e2134a830d38d2d1aae7e88cac31bdd
Sun Nov-27-17:00:10 [initandlisten] build info: Darwin broadway.local 9.8.0 Darwin Ker
nel Version 9.8.0: Wed-Jul 15 16:55:01 PDT 2009; root:xnu-1228.15.4~1/RELEASE_1386 138
6 BOOST_LIB_VERSION=1_40
Sun Nov 27 17:00:10 [initandlisten] options: {}
SunyNovi 27:17:00:10 [websyr] admins web console waiting for connections on port 28017 or
SunhNove 27:17:00:40 [initandlisten]ewaiting for connections on porth 27017hgoDB server
```





#### Lancement du client



bin/mongo

```
macbook-de-isep:mongodb isep$ bin/mongo
MongoDB shell version: 2.0.1
connecting to: test
>
```

- Pour vous familiariser avec l'environnement essayez ces quelques commandes
- >help
- >db.help()





### **Exemple introductif (« hello world)**

```
> use prefs
switched to db prefs
> w={name:"Raja Chiky", zip:75014};
{ "name" : "Raja Chiky", "zip" : 75014 }
> x={name: "Sylvain Lefebvre", zip:75015};
{ "name" : "Sylvain Lefebvre", "zip" : 75015 }
> y={name:"Olivier hermant", zip:75005};
{ "name" : "Olivier hermant", "zip" : 75005 }
> z={name:"Matthieu Manceny", zip:92100};
{ "name" : "Matthieu Manceny", "zip" : 92100 }
> db.location.save(w);
> db.location.save(x);
> db.location.save(y);
> db.location.save(z);
```





## Requêtage



- Pour afficher les données
  - db.location.find()

```
> db.location.find()
{ "_id" : ObjectId("4ed2654d8d9ba30f9a2bba5f"), "name" : "Raja Chiky", "zip" : 75014 }
{ "_id" : ObjectId("4ed265548d9ba30f9a2bba60"), "name" : "Sylvain Lefebvre", "zip" : 75015 }
{ "_id" : ObjectId("4ed265578d9ba30f9a2bba61"), "name" : "Olivier hermant", "zip" : 75005 }
{ "_id" : ObjectId("4ed2655a8d9ba30f9a2bba62"), "name" : "Matthieu Manceny", "zip" : 92100 }
>
```

- ObjectID: identifiant unique de chaque document.
   Déclaré explicitement par le développeur ou implicitement par mongoDB
- Format: BSON (binary JSON): sérialisation binaire de documents « JSON-Like » avec une extension pour d'autres types (date, données binaires, etc.)
- Spécification de BSON: <a href="http://bsonspec.org/">http://bsonspec.org/</a>





## Requêtage



 Ajouter d'autres enregistrements à la collection « location »

```
> a= name:"Zakia Kazi", zip:75014};
{ "name" : "Zakia Kazi", "zip" : 75014 }
> b= name:"Bernard Hugueney", zip:75005};
{ "name" : "Bernard Hugueney", "zip" : 75005 }
> db.location.save(a);
> db.location.save(b);
```

■ Trouver les enregistrements pour zip:75005 (même chose pour name : 'Raja Chiky')

```
> db.location.find({zip: 75005});
> db.location.find({zip: 75005});
{ "_id" : ObjectId("4ed265578d9ba30f9a2bba61"), "name" : "Olivier hermant", "zip" : 75005 }
{ "_id" : ObjectId("4ed267f68d9ba30f9a2bba64"), "name" : "Bernard Hugueney", "zip" : 75005 }
>
```





#### A vous...

```
>use library
>document = ( { Type : "Book", Title : "Definitive Guide to
   MongoDB", ISBN: "987-1-4302-3051-9", Publisher:
   "Apress", Author: ["Membrey, Peter", "Plugge, Eelco",
   "Hawkins, Tim" ] } )
>db.media.insert(document)
>db.media.insert( { Type : "CD" ,Artist : "Nirvana",Title :
   "Nevermind",
Tracklist:[
{ Track: "I ", Title: "Smells like teen spirit", Length: "5:02"},
{ Track : "2 ", Title : "In Bloom ", Length : "4:15 " }
]})
```





## Requêtage: que font ces commandes?



```
>db.media.find()
>db.media.find ( { Artist : "Nirvana" } )
>db.media.find ( { Artist : "Nirvana"}, {Title: I} )
>db.media.find ( { Artist : "Nirvana"}, {Title: 0} )
>db.media.find( { "Tracklist.Title" : "In Bloom" } )
>db.media.findOne()

Ajoutez la fonction pretty() pour l'indentation
>db.media.find().pretty()
```





#### Fonctions: Sort, Limit et Skip

```
>db.media.find().sort( { Title: I })
>db.media.find().sort( { Title: -I })
>db.media.find().limit( I0 )
>db.media.find().skip( 20 )
>db.media.find().sort ( { Title : -I } ).limit ( I0 ).skip ( 20 )
```





## **Agrégations**









#### Ajouter une nouvelle donnée





## Groupement de résultats

```
>db.media.group ( {
  key: {Title : true},
  initial: {Total : 0},
  reduce : function (items,prev) {
  prev.Total += I }
} )
```

- key: paramètre de regroupement
- initial: valeur initiale (0 par défaut)
- reduce: prend 2 arguments, le document (items) et le compteur (prev) et effectue l'agrégation
- cond: condition que doit respecter les attributs du documents





## Ajout d'autres données

```
>dvd = ( { Type : "DVD", Title : "Matrix, The", Released : 1999, Cast: ["Keanu Reeves", "Carry-Anne Moss", "Laurence Fishburne", "Hugo Weaving", "Gloria Foster", "Joe Pantoliano"] } )
>db.media.insert(dvd)
>dvd = ( { "Type" : "DVD", "Title" : "Toy Story 3", "Released" : 2010 } )
>db.media.insert(dvd)
     Insertion avec JavaScript
>function insertMedia( type, title, released
  db.media.insert({
    "Type":type,
   "Title":title,
   "Released":released
 });
>insertMedia("DVD", "Blade Runner", 1982)
```





#### TELECO Evoluti

## Opérateurs de comparaison

- \$gt, \$lt, \$gte, \$lte, \$ne, \$in, \$nin (resp. >,<,>=,<=,!=,IN, NOT IN)</li>
- Que font ces requêtes?

\$or

```
>db.media.find({ $or : [ { "Title" : "Toy Story 3" }, { "ISBN" : "987-1-4302-3051-9" } ] } )
>db.media.find({ "Type" : "DVD", $or : [ { "Title" : "Toy Story 3" }, { "ISBN" : "987-1-4302-3051-9" } ] })
```







- \$slice: permet de rassembler les capacités de limit() et skip()
  - \$slice: [20, 10] // skip 20, limit 10
  - \$slice: 5 // les 5 premiers
  - \$slice:-5 //les 5 derniers

```
>db.media.find({"Title": "Matrix, The"}, {"Cast": {$slice: 3}})
>db.media.find({"Title": "Matrix, The"}, {"Cast": {$slice: -3}})
```





#### \$size et \$exists



```
>db.media.find ( { Tracklist : {$size : 2} } )
>db.media.find ( { Author : {$exists : true } } )
>db.media.find ( { Author : {$exists : false } } )
```





#### Création d'un index

```
Index ascendant
>db.media.ensureIndex({Title:I})
   Index descendant
>db.media.ensureIndex({Title:-I})
   Index pour les objets incrustés (enbed object)
>db.media.ensureIndex( { "Tracklist.Title" : I } )
   Forcer l'utilisation d'un index: hint()
>db.media.find( { ISBN: "987-I-4302-305I-9"} ) . hint ( { ISBN:
   -I } )
   error: { "$err" : "bad hint", "code" : 10113 }
>db.media.ensureIndex({ISBN: I})
>db.media.find( { ISBN: "987-1-4302-3051-9"} ) . hint ( { ISBN:
>db.media.getIndexes()
```





## Mise à jour des données



#### update(critere,nouvelObjet,upsert,multi)

- upsert=true //créer l'objet s'il n'existe pas
- Multi spécifie si la modification se fait sur un seul objet (par défaut) ou sur tous les objets respectant le critère

```
>db.media.update( { "Title" : "Matrix, the"}, {"Type" : "DVD", "Title" : "Matrix, the", "Released" : "1999", "Genre" : "Action"}, true)
```

Ajout/suppression d'un attribut

```
>db.media.update ( { "Title" : "Matrix, the" }, {$set : { Genre : "Sci-Fi" } } ) >db.media.update ( {"Title": "Matrix, the"}, {$unset : { "Genre" : I } } )
```

#### Suppression

Documents respectant un critère

```
>db.media.remove( { "Title" : "Different Title" } )
```

- Tous les documents
- >db.media.remove({})
- Toute la collection
- >db.media.drop()





## **Exercices (1)**



- Créer une base de données « lapins » et l'activer
- Lister les bases du serveur
- Insérer dans la collection « »

nom	genre	ville	regime	poids	taille
leny	f	Lyon	carotte, courgette	4	20
bunny	h	Paris		3	
olto	h	Paris	raisin, carotte, salade	5	25





### Exercices (2) - requêtes



- Trouvez tous les lapins mâles?
- Nombre de lapins qui aiment les carottes et qui pèsent plus de 4kg
- Tous les lapins qui aiment les courgettes ou les raisons ou qui n'ont pas de champ « ville »
- Tous les lapins qui n'aiment pas la salade
- Nous savons que Bunny se trouve en France, rajouter un champ pays à ce document
- Supprimer le champ « taille », s'il existe, de tous les documents
- Supprimer la base de données





## Base de données géographique

#### Importer les données

>./bin/mongoimport --type json -d geodb -c earthquakes -- file eartquakes.json

#### Modification des données

 Ajouter une colonne iso\_date dont la valeur est la conversion du timestamp contenu dans properties.time









 Convertir la chaîne de caractère du champ properties.types en tableau et le mettre dans un champ types\_as\_array

Utiliser la fonction ch.split(",") qui permet de séparer une chaine de caractère ch en plusieurs mots selon le séparateur ", »

```
db.earthquakes.find().forEach
( function(eq){
     var str = new String(eq.properties.types);
     eq.properties.types_as_array = str.split(",");
     db.earthquakes.save(eq); }
);
```







## Nettoyage des données

Nettoyer les éléments vides ("") du tableau properties.types\_as\_array
 >db.earthquakes.update(

 {},
 { \$pullAll: { "properties.types\_as\_array" : [""] } },
 { multi: true }





### Requêtes



 Donnez le nombre de documents dont la liste de type contient "geoserves" et "tectonic-summary »

 Ecrire une requête qui donne le nombre de tremblement terre en California (Indice : RegExp)





## Indexation géographique



- Nous allons maintenant modifier les données afin d'adapter les coordonnées géographiques au format qui nous permettra construire un index 2dsphere.
- Normalisez les données en supprimant le dernier élément du tableau 'geometry.coordinates' pour le copier dans un champ 'depth'.
- Exemple:

```
geometry : {
    "type" : "Point",
    "coordinates" : [-147.35, 63.59, 0.1]
}

// devient ....

depth : 0.1
geometry : {
    "type" : "Point",
    "coordinates" : [-147.35, 63.59]
}
```





#### Création de l'index



- Créer un index de type 2d sur les attributs « geometry.coordinates »
- Requête
  - Exécuter une requête qui cherche les tremblements de terre proche de la position -3.984,48.724 (1000km)

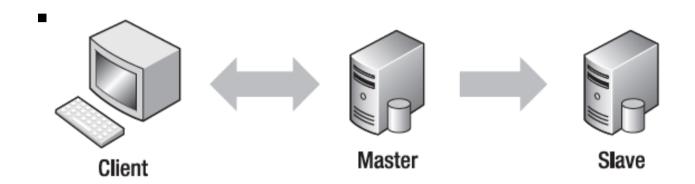
**Documentation**: http://docs.mongodb.org/manual/reference/operator/query-geospatial/





## Réplication : 1 maître et 1 ou plusieurs esclaves





- >mkdir /data/master
- >./bin/mongod --master --dbpath /data/master
- >mkdir /data/slave
- >./bin/mongod --slave --source localhost:27017 -- dbpath /data/slave -port 27018





## **Testons la réplication**



- 1. Connexion au maître
- >./bin/mongo
- >show dbs
- >use local
- >show collections
- >db.slaves.find()
- 3. Test
- >use testdb
- >db.testcollection.insert({nam
  e:'tim',surname:'hawkins'})
- > db.testcollection.find()

- 2. Examinons l'esclave
- >mongo --port 27018
- >show dbs
- >use local
- >show collections
- 4. Examinons le test (côté esclave)
- >show dbs
- >use testdb
- >show collections
- > db.testcollection.find()
- 5. Ajouter d'autres esclaves:
- >mkdir –p /data/slave2
- >mongod --slave --source localhost:27017 --dbpath / data/slave2 --port 27019





## Réplication : plusieurs maître et 1 esclave



>mkdir /data/master1

>./bin/mongod --master --dbpath /data/ master1 -port 27021

**Ouvrir un autre terminal:** 

>mongo localhost:27021

>use foo

>db.foocollection.insert({foodata:"our first foo document"})

>quit()

>mkdir /data/master2

>./bin/mongod --master --dbpath /data/ master1 --port 27022

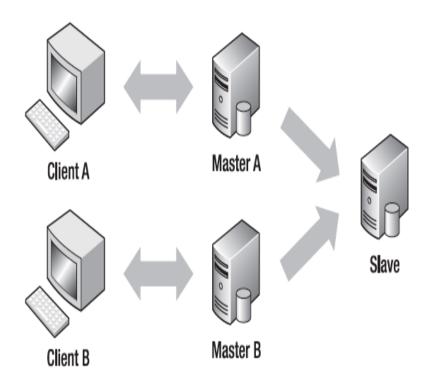
>mongo localhost:27022

>use bar

>db.barcollection.insert({bardata:"our first bar document"})

>quit()

>mongod --slave --dbpath /data/slave - port 27023







## Réplication : plusieurs maître et 1 esclave



```
Côté esclave
```

>mongod --slave --dbpath /data/slave --port 27023

>mongo localhost:27023

>use local

>db.sources.insert({host:'localhost:27021'})

>db.sources.insert({host:'localhost:27022'})

>db.sources.find()

#### >show dbs

- Exercice: Vérifier les contenus des collections bar et foo
- On peut décider de ne répliquer qu'une collection par les commandes suivantes:

>db.sources.insert({ host:'localhost:27021', only:'foo' }) >db.sources.insert({ host:'localhost:27022', only:'bar' })





## **Testons la réplication**



- 1. Connexion au maître
- >./bin/mongo
- >show dbs
- >use local
- >show collections
- >db.slaves.find()
- 3. Test
- >use testdb
- >db.testcollection.insert({nam e:'tim',surname:'hawkins'})
- > db.testcollection.find()

- 2. Examinons l'esclave
- >mongo --port 27018
- >show dbs
- >use local
- >show collections
- 4. Examinons le test (côté esclave)
- >show dbs
- >use testdb
- >show collections
- > db.testcollection.find()
- 5. Ajouter d'autres esclaves:
- >mkdir –p /data/slave2
- >mongod --slave --source localhost:27017 --dbpath / data/slave2 --port 27019





#### **Partitionnement**



Nous allons construire l'architecture suivante:

Service	Port	dbpath
Config server	27022	/data/config
Shard0	27023	/data/shard0
Shard1	27024	/data/shard1

Créer les répertoires nécessaires>mongod --port 27022 --dbpath /data/config -configsvr

Ensuite lancez le contrôleur

>mongos --configdb localhost:27022 --port 27021 -- chunkSize 1

>mongod --port 27023 --dbpath /data/shard0 --shardsvr

>mongod --port 27024 --dbpath /data/shard1 --shardsvr





#### **Partitionnement**



 Se connecter au contrôleur >mongo localhost:27021 >use admin >db.runCommand( { addshard : "localhost:27023", allowLocal: true }) >db.runCommand( { addshard : "localhost:27024", allowLocal: true }) Ajout d'une DB >phones=db.getSisterDB("phones") >db.runCommand({enablesharding:"phones"}) >db.runCommand({ shardcollection : "phones.testcollection", key : {\_id : 1}}) >use phones





## **Exercice: Remplir les partitions**

- Récupérer le fichier populate\_phones.js
- Analyser le code, à votre avis que fait-il?
- Exécuter le code avec les paramètres suivants
   >populatePhones(800,5550000,5750000)
- Consultez les 2 premiers enregistrements
- A chaque nouvelles collection, MongoDB crée automatiquement un index suivant l'identifiant \_id. Ces index se trouvent dans system.indexes

>db.system.indexes.find()

Tester les trois lignes suivantes et expliquez les résultats

```
>db.phones.find({display: "+1 800-5650001"}).explain()
>db.phones.ensureIndex(
{ display : 1 },
{ unique : true, dropDups : true }
)
>db.phones.find({ display: "+1 800-5650001" }).explain()
```





### populate\_phone.js



```
populatePhones = function(area,start,stop) {
for(var i=start; i < stop; i++) {</pre>
var country = 1 + ((Math.random() * 8) << 0);</pre>
var num = (country * 1e10) + (area * 1e7) + i;
db.testcollection.insert({
id: num,
components: {
country: country,
area: area.
prefix: (i * 1e-4) << 0,
number: i.
testtext: "Because of the nature of MongoDB, many of the more traditional functions that
a DB Administrator would perform are not required. Creating new databases collections
and new fields on the server are no longer necessary, as MongoDB will create these
elements on-the-fly as you access them. Therefore, for the vast majority of cases
managing databases and schemas is not required."
display: "+" + country + " " + area + "-" + i
});
```





#### **Partitionnement**



Vérification

>mongo localhost:27021

>use phones

>db.testcollection.count()

>mongo localhost:27023

>use phones

>db.testcollection.count()

>mongo localhost:27024

>use phones

>db.testcollection.count()





## Un peu d'admin...



#### Backup

- >mkdir testmongobackup
- >cd testmongobackup
- >../mongodb/bin/mongodump --help
- >../mongo/bin/mongodump
- >../mongodump --db library --collection media
- →./dump/[databasename]/[collectionname].bson

#### Restore

- >cd testmongobackup
- >../mongo/bin/mongorestore —help
- Tout restaurer
- >../mongo/bin/mongorestore —drop
- Restaurer une seule collection
- >../mongo/bin/mongorestore -d library -c media -drop





#### Sécurité



#### Authetification

- Côté client
- > use admin
- >db.addUser("admin", "adminpassword")
- Côté serveur
- > use admin
- >db.addUser("admin", "adminpassword")
- Shell (redémarrer le serveur)
- >sudo service mongodb restart

#### Ou

- >db.shutdownServer()
- S' authetntifier
- >use admin
- >db.auth("admin", "adminpassword")
- >use library
- >db.addUser("raja", "rajapassword")
- >db.addUser("sylvain", "sylvainpassword",true) //read only
- >db.removeUser("raja")

