

# Conception d'Applications Interactives : Applications Web et JEE

Séance #4  
Une forge JavaScript

2/4 - MongoDB and NoSQL





# **Introduction à MongoDB**



# What's MongoDB



open-source document database  
that provides high performance,  
high availability,  
and automatic scaling



# Document database

A record in MongoDB is a document, which is a data structure composed of field and value pairs

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

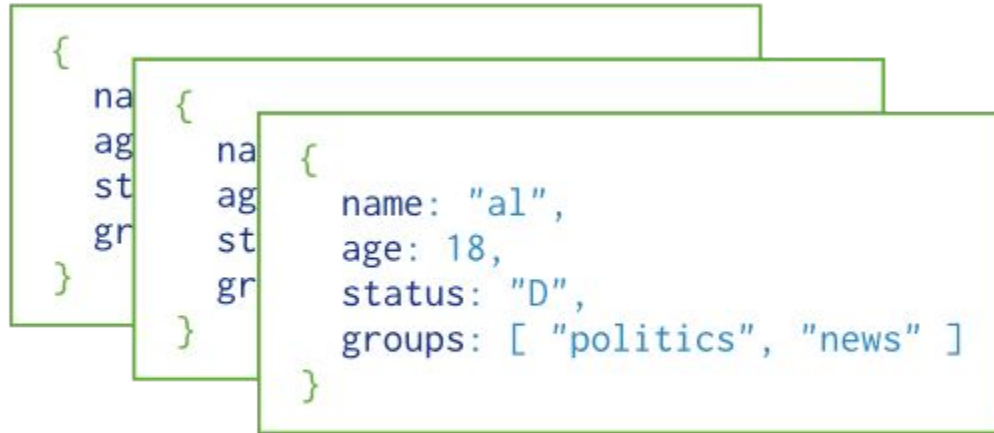
← field: value  
← field: value  
← field: value  
← field: value

Documents are similar to JSON objects



# Document database

Documents are stored in collections



Collection

Collections share common indexes



# Collections & documents

- No predefined schema
  - Documents in a collection can have different fields
  - Fields can be added, modified or deleted at any time
- Documents follow BSON (JSON-like) format
  - Key-value pairs (hashes)

```
{
```

```
  "_id": ObjectId("223EBC5477A124425"),  
  "Last Name": "Gonzalez",
```



# Insert data

Collection  
↓  
`db.users.insert(`

Document  
↓

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

)

Document

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

insert →

Collection

{ name: "al", age: 18, ... }
{ name: "lee", age: 28, ... }
{ name: "jan", age: 21, ... }
{ name: "kai", age: 38, ... }
{ name: "sam", age: 18, ... }
{ name: "mel", age: 38, ... }
{ name: "ryan", age: 31, ... }
{ name: "sue", age: 26, ... }

users



# Query data

- `db.collection.find()`

```
db.users.find(  
  { age: { $gt: 18 } },  
  { name: 1, address: 1 }  
) .limit(5)
```

← collection  
← query criteria  
← projection  
← cursor modifier

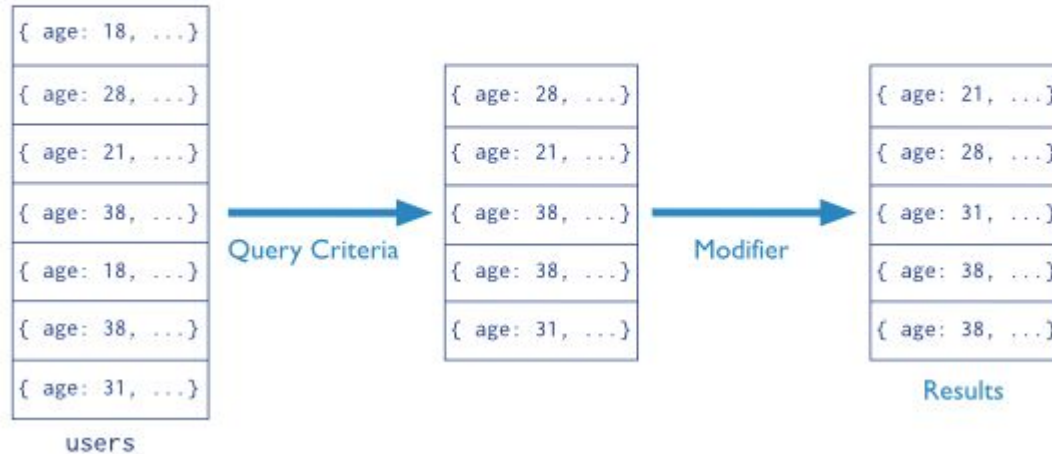
It returns a cursor, an iterable object





# Query data

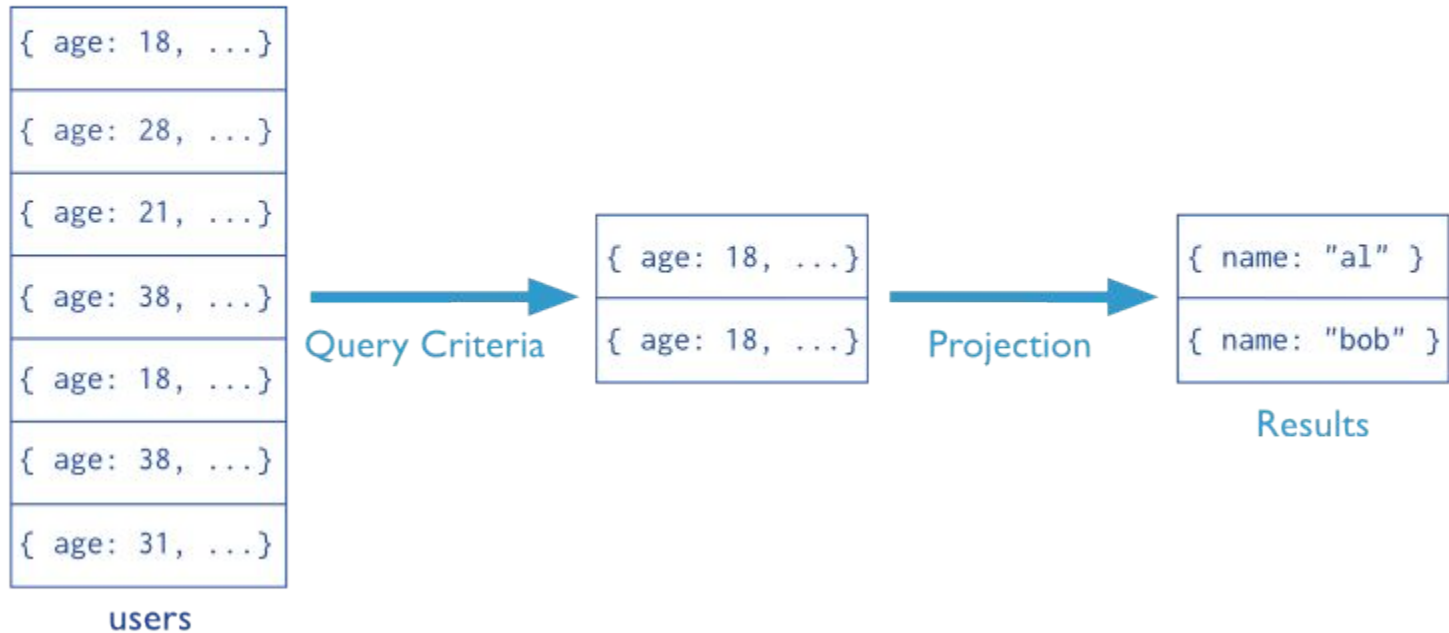
Collection                      Query Criteria                      Modifier  
`db.users.find( { age: { $gt: 18 } } ).sort( {age: 1 } )`



# Projections

Collection                      Query Criteria                      Projection

```
db.users.find( { age: 18 }, { name: 1, _id: 0 } )
```



# RDBMS vs MongoDB

## RDBMS

- Databases have tables
- Tables have rows
- Rows have cell
- Cells contain types simples
- Schemas are rigid

## MongoDB

- Database have collections
- Collections have documents
- Documents have fields
- Fields contain
  - Types simples
  - Arrays
  - Other documents
- Schemas are fluid



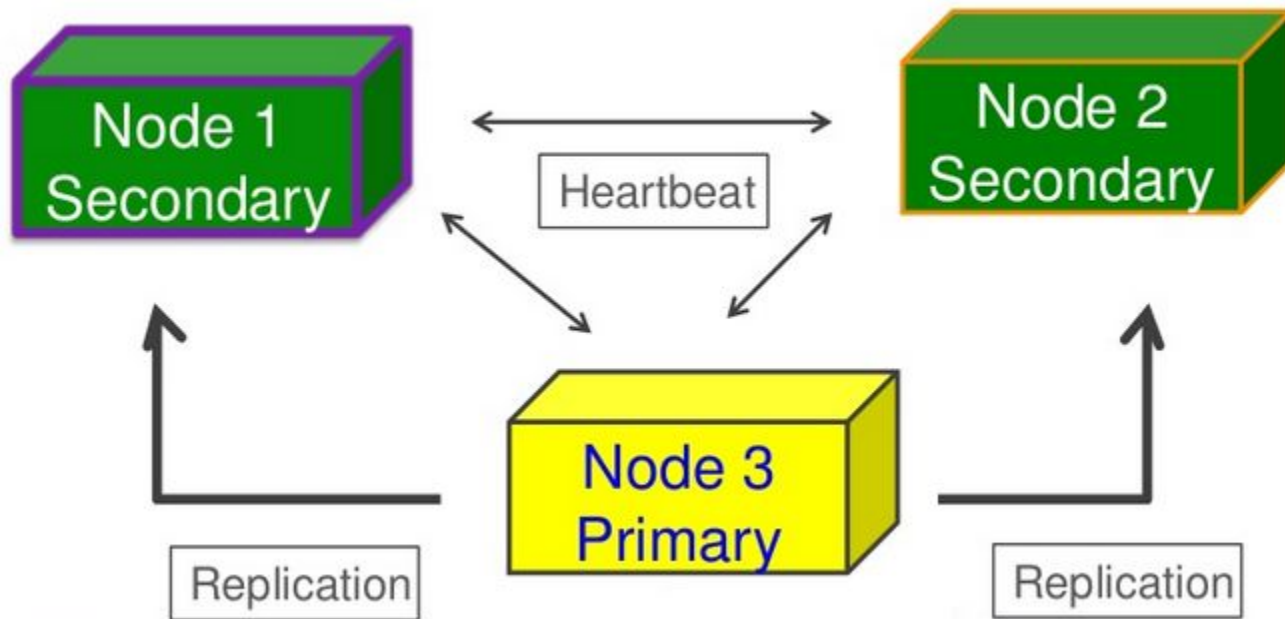
# Key features

- High performance
- High availability
- Automatic scaling



# High availability

- Election at initialization or when primary lost



# Installing MongoDB

1. Download MongoDB <http://www.mongodb.org/downloads>

2. Install the msi (win) or uncompress the tgz (linux)

e.g. `C:\mongodb` or `/opt/mongodb`

3. Create a data directory

e.g. `/opt/mongodb_data` or `C:\mongodb_data`

4. Run mongo demon

`C:\mongodb\bin\mongod.exe --dbpath C:\mongodb_data` or



# Running MongoDB

```
horacio@horacio-xps:~$ mongod --dbpath /opt/data/
2015-05-06T23:46:14.392+0200 I JOURNAL [initandlisten] journal dir=/opt/data/journal
2015-05-06T23:46:14.394+0200 I JOURNAL [initandlisten] recover : no journal files present, no recovery needed
2015-05-06T23:46:14.431+0200 I CONTROL [initandlisten] MongoDB starting : pid=5493 port=27017 dbpath=/opt/data/ 64-bit
host=horacio-xps
2015-05-06T23:46:14.431+0200 I CONTROL [initandlisten] db version v3.0.2
2015-05-06T23:46:14.431+0200 I CONTROL [initandlisten] git version: 6201872043ecbbc0a4cc169b5482dcf385fc464f
2015-05-06T23:46:14.431+0200 I CONTROL [initandlisten] build info: Linux build6.nj1.10gen.cc 2.6.32-431.3.1.el6.x86_64 #1 SMP
Fri Jan 3 21:39:27 UTC 2014 x86_64 BOOST_LIB_VERSION=1_49
2015-05-06T23:46:14.431+0200 I CONTROL [initandlisten] allocator: tcmalloc
2015-05-06T23:46:14.431+0200 I CONTROL [initandlisten] options: { storage: { dbPath: "/opt/data/" } }
2015-05-06T23:46:14.433+0200 I STORAGE [initandlisten] info openExisting file size 16777216 but mmapv1GlobalOptions.
smallfiles=false: /opt/data/startupweekendbrest.0
2015-05-06T23:46:14.440+0200 I INDEX [initandlisten] allocating new ns file /opt/data/local.ns, filling with zeroes...
2015-05-06T23:46:14.532+0200 I STORAGE [FileAllocator] allocating new datafile /opt/data/local.0, filling with zeroes...
2015-05-06T23:46:14.532+0200 I STORAGE [FileAllocator] creating directory /opt/data/_tmp
2015-05-06T23:46:14.572+0200 I STORAGE [FileAllocator] done allocating datafile /opt/data/local.0, size: 64MB, took 0.023 secs
2015-05-06T23:46:14.585+0200 I NETWORK [initandlisten] waiting for connections on port 27017
```



# Tools in MongoDB

- **mongod** - primary daemon process
- **mongo** - command-line client
- **mongostat** - command-line stats summary
- **mongotop** - command-line performance tracker





# Connecting to a MongoDB instance

```
mongo --host 127.0.0.1 --port 27017
```

or using default parameters

```
mongo
```

- Default host: 127.0.0.1
- Default port: 27017

```
horacio@horacio-xps:~$ mongo
MongoDB shell version: 3.0.2
```



# First steps

- To view available databases:

```
> show dbs
```

```
local          0.078GB
```

```
test           0.031GB
```

- To choose a database:

```
> use test
```

```
switched to db test
```



# First steps

- To check what's the current database:

```
> db
```

- To get some help:

```
> help
```

- To show the collections within a database:

```
> show collections
```



# Enter some data

```
> a = {"Last Name": "Gonzalez", "First Name": "Horacio", "Date of Birth": "1976-05-05" }
{
  "Last Name" : "Gonzalez",
  "First Name" : "Horacio",
  "Date of Birth" : "1976-05-05"
}
> db.test.insert(a)
WriteResult({ "nInserted" : 1 })
> b={"Field A": "Value A", "Field B": "Value B"}
{ "Field A" : "Value A", "Field B" : "Value B" }
> db.test.insert(b)
WriteResult({ "nInserted" : 1 })
```



# Query data

- Find all the elements in a collection

```
> db.test.find()
```

```
{ "_id" : ObjectId("554a9944b5091037c44dddcc"), "Last Name" : "Gonzalez",  
  "First Name" : "Horacio", "Date of Birth" : "1976-05-05" }
```

```
{ "_id" : ObjectId("554a998eb5091037c44dddc"), "Field A" : "Value A", "Field  
B" : "Value B" }
```



# \_id

- Primary key
- Automatically indexed
- Generated as an ObjectId if not provided
- Must be unique and immutable

Obj(ObjectId("50804d0bd94ccab2da652599")) cluster

-----||-----||-----||-----|

ts mac pid inc



# Using JavaScript in mongo

```
> for(var i=0; i<5; i++) db.test.insert({a:42, b:i})
```

```
WriteResult({ "nInserted" : 1 })
```

```
> db.test.find()
```

```
{ "_id" : ObjectId("554a990f0ebf783b63a57776"), "Last Name" : "Gonzalez", "First Name" : "Horacio", "Date of Birth" : "1976-05-05" }
```

```
{ "_id" : ObjectId("554a9944b5091037c44dddcc"), "Last Name" : "Gonzalez", "First Name" : "Horacio", "Date of Birth" : "1976-05-05" }
```

```
{ "_id" : ObjectId("554a998eb5091037c44dddc"), "Field A" : "Value A", "Field B" : "Value B" }
```

```
{ "_id" : ObjectId("554a9c21b5091037c44dddce"), "a" : 42, "b" : 0 }
```

```
{ "_id" : ObjectId("554a9c21b5091037c44dddcf"), "a" : 42, "b" : 1 }
```

```
{ "_id" : ObjectId("554a9c21b5091037c44ddd0"), "a" : 42, "b" : 2 }
```

```
{ "_id" : ObjectId("554a9c21b5091037c44ddd1"), "a" : 42, "b" : 3 }
```

```
{ "_id" : ObjectId("554a9c21b5091037c44ddd2"), "a" : 42, "b" : 4 }
```



# Query for specific documents

```
> db.test.find({"Field A": "Value A"})
```

```
{ "_id" : ObjectId("554a998eb5091037c44ddcd"), "Field A" : "Value A", "Field B" :  
"Value B" }
```

```
> db.test.find({ b: { $gt: 2 } }).sort({ b: -1 })
```

```
{ "_id" : ObjectId("554a9c21b5091037c44ddd2"), "a" : 42, "b" : 4 }
```

```
{ "_id" : ObjectId("554a9c21b5091037c44ddd1"), "a" : 42, "b" : 3 }
```

## Conditional operators:

`$all`, `$exists`, `$type`, `$mod`,

`$or`, `$and`, `$not`, `$nor` `$size`,

`$eq`, `$ne`, `$lt`, `$lte`, `$gt`, `$gte`, `$in`, `$nin`...





# Querying with RegEx

```
> db.test.findOne({ "Last Name": /Gon/})  
{  
  "_id" : ObjectId("554a990f0ebf783b63a57776"),  
  "Last Name" : "Gonzalez",  
  "First Name" : "Horacio",  
  "Date of Birth" : "1976-05-05"  
}
```



# Operations

> `db.test.insert(record)`

> `db.test.find(query)[.skip(X)][.limit(Y)]`

> `db.test.findOne(query)`

> `db.test.remove(query[, justone=false])`

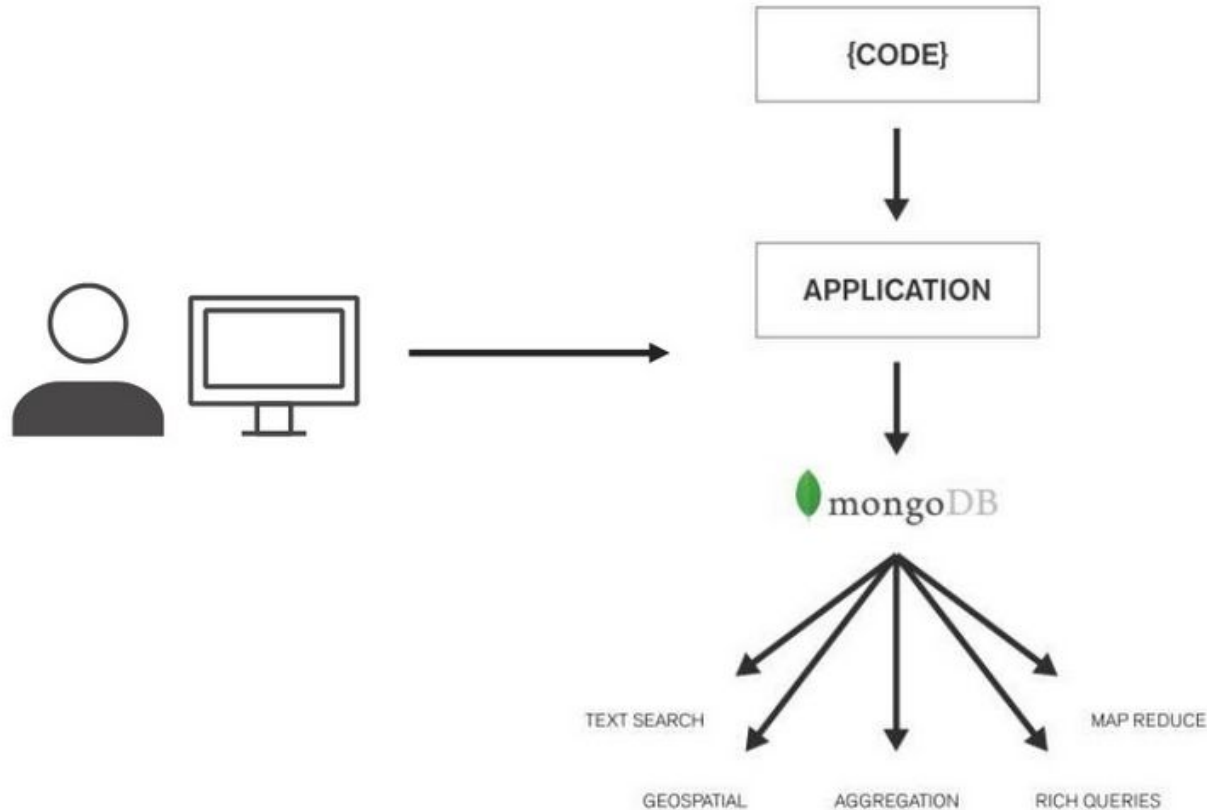


# Creating index

```
> db.test.ensureIndex({ b: 1 })  
{  
  "createdCollectionAutomatically" : false,  
  "numIndexesBefore" : 1,  
  "numIndexesAfter" : 2,  
  "ok" : 1  
}
```



# MongoDB is fully featured



# Exercise 1

- Install MongoDB
- Run mongod
- Connect using mongo
- Create a collection, add some object, tests some queries

## References

Methods: <http://docs.mongodb.org/manual/reference/method/>



## Exercise 2

Model and create a collection to store the beer data  
for the Angular-Beers project

<https://github.com/LostInBrittany/angular-beers>

- Tests some queries

