
mongoDB

- Base de données oriéentée document
- open source
- Hautes performances
- Haute disponibilité
- Scaling automatique

BDD document

Un enregistrement dans MongoDB est un document

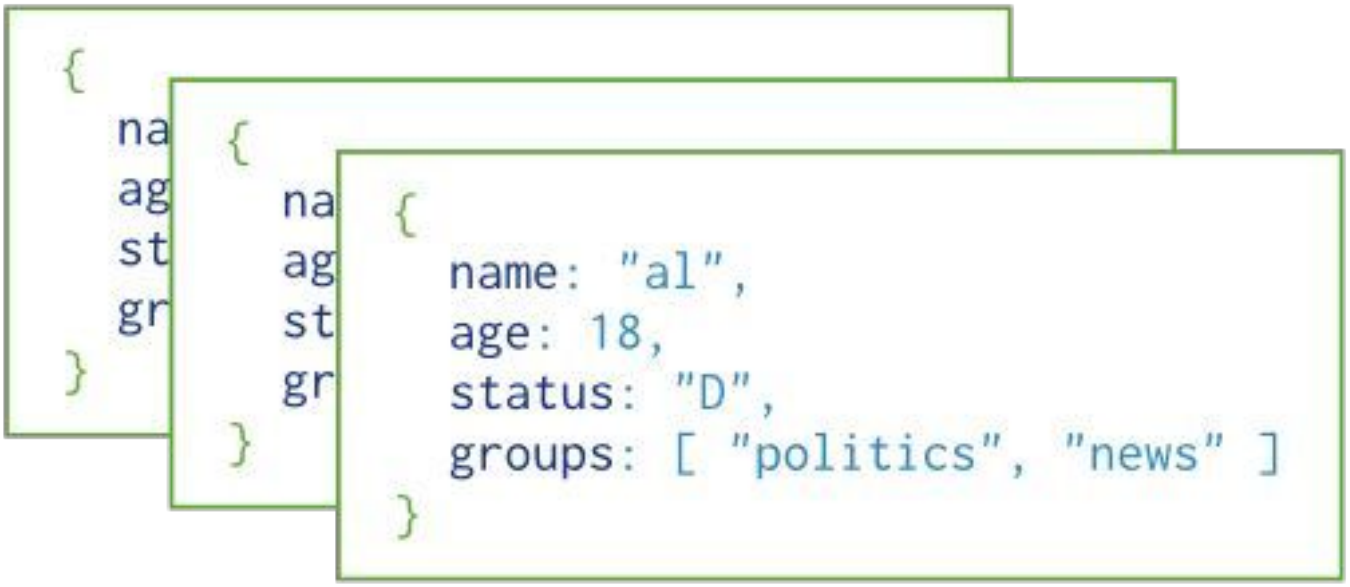
Une structure de données composée de clef/valeur

Un document est similaire à un objet Json

BDD document

Les documents sont stockés dans des collections

Les collections partagent des indexs communs

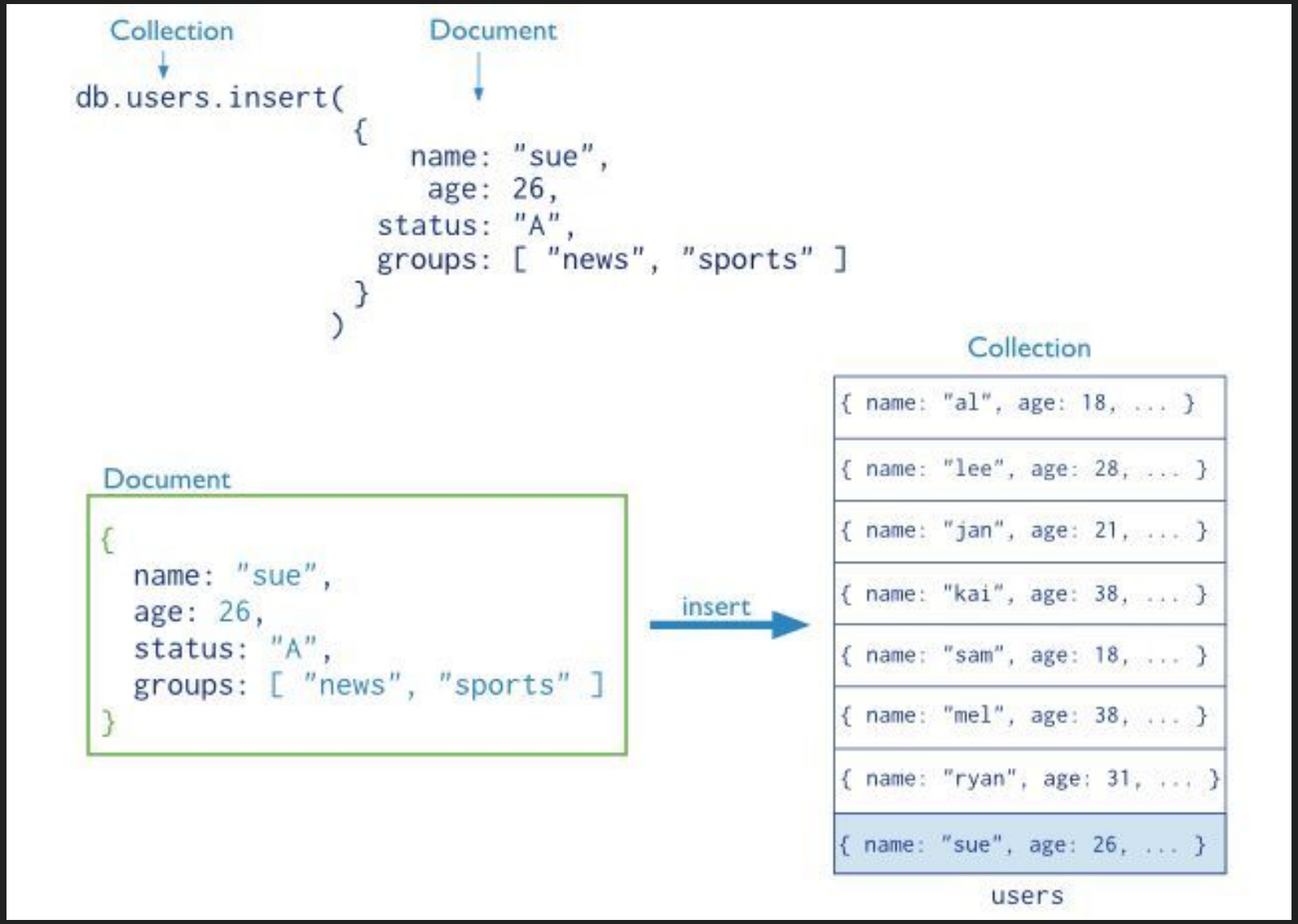


Collections et documents

- Pas de schéma pré-défini
  - Les documents d'une collection peuvent avoir des champs différents
  - Des champs peuvent être ajoutés, modifiés ou effacé à n'importe quel moment
- Les documents sont au format BSON (Json like)
  - Paires de clef/valeur (hashes)

```
{
        "_id": ObjectId("223EBC5477A124425"),
        "Last Name": "Marin",
        “firstname” : “Xavier”,
        “jedi_power” : 42
}
```

Insertion de données

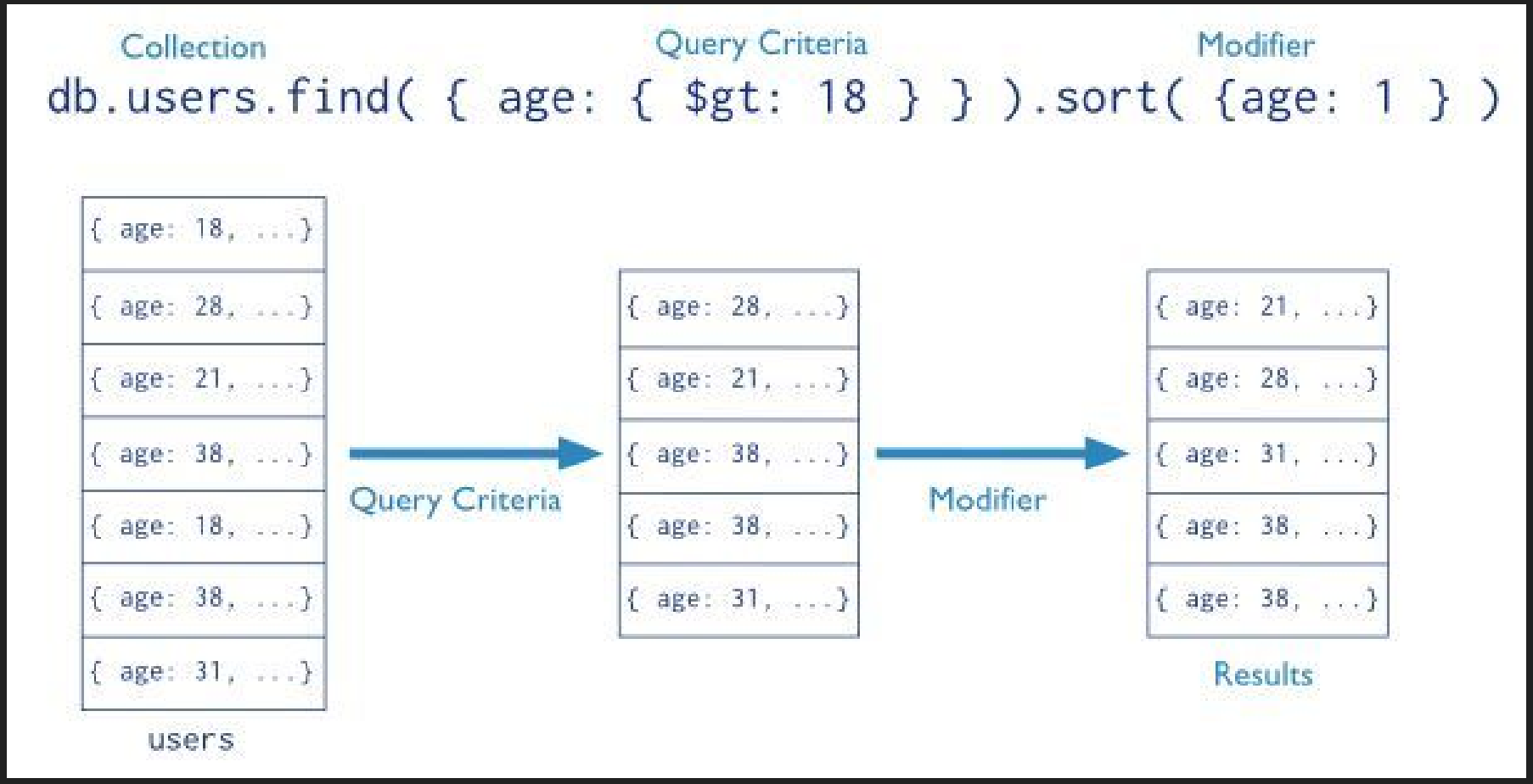


Requêter

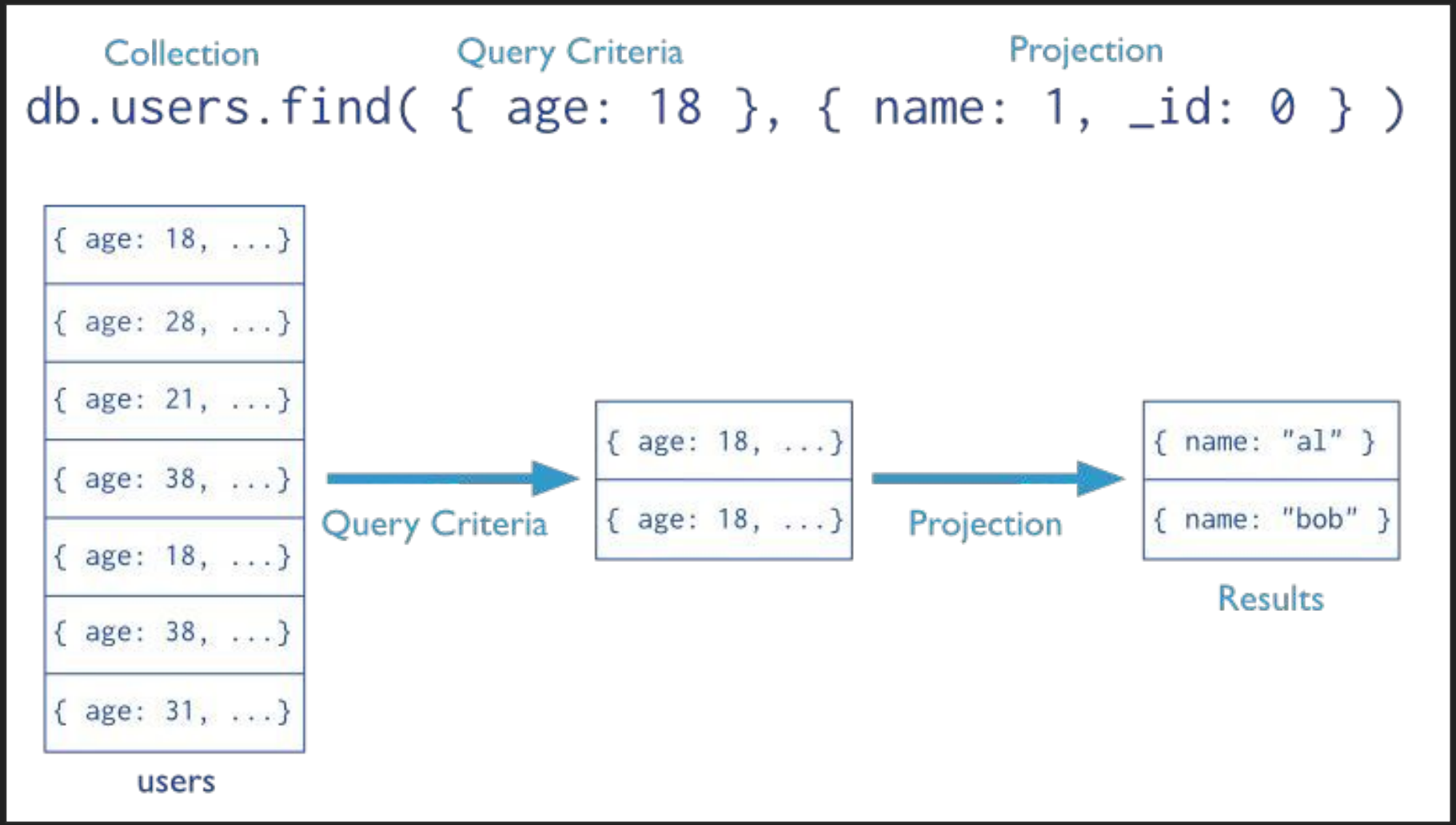
db.collection.find()

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

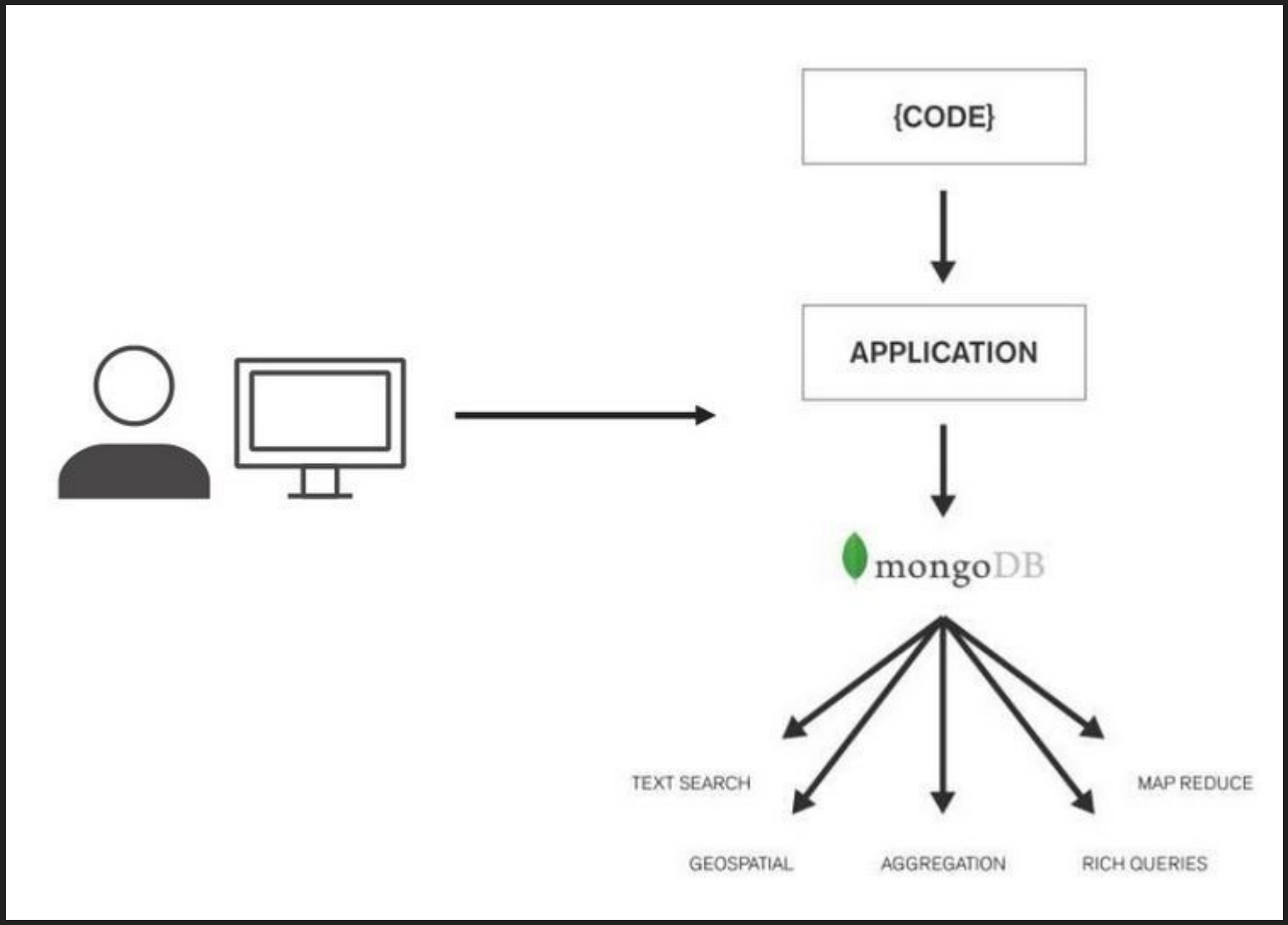
Requêter


Collection
Query Criteria
Modifier
db.users.find({ age: { $gt: 18 } }).sort({age: 1 })
{ age: 18, ...}
{ age: 28, ...}
{ age: 21, ...}
{ age: 38, ...}
{ age: 18, ...}
{ age: 38, ...}
{ age: 31, ...}
users
Query Criteria
{ age: 28, ...}
{ age: 21, ...}
{ age: 38, ...}
{ age: 38, ...}
{ age: 31, ...}
Modifier
{ age: 21, ...}
{ age: 28, ...}
{ age: 31, ...}
{ age: 38, ...}
{ age: 38, ...}
Results

Projections



MongoDB est bien doté



SGBDR vs MongoDB

SGBDR

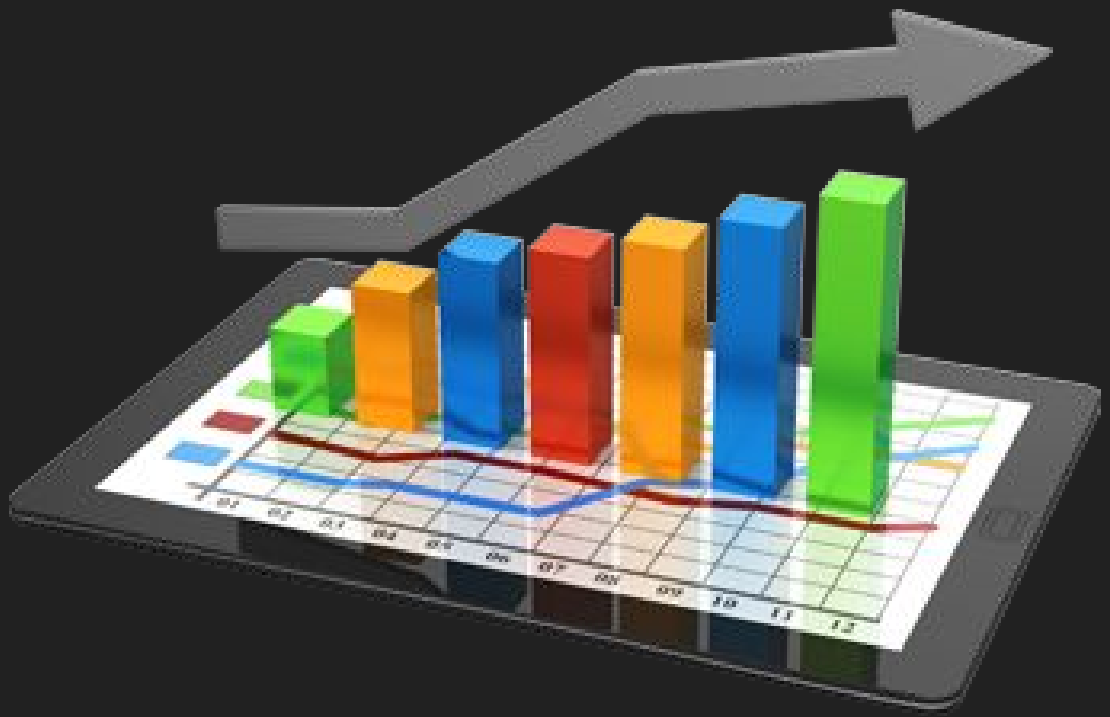
- Les bases ont des tables
- Les tables ont des tuples
- Les tuples ont des champs
- Les champs contiennent des types simples
- Le schéma est rigide

MongoDB

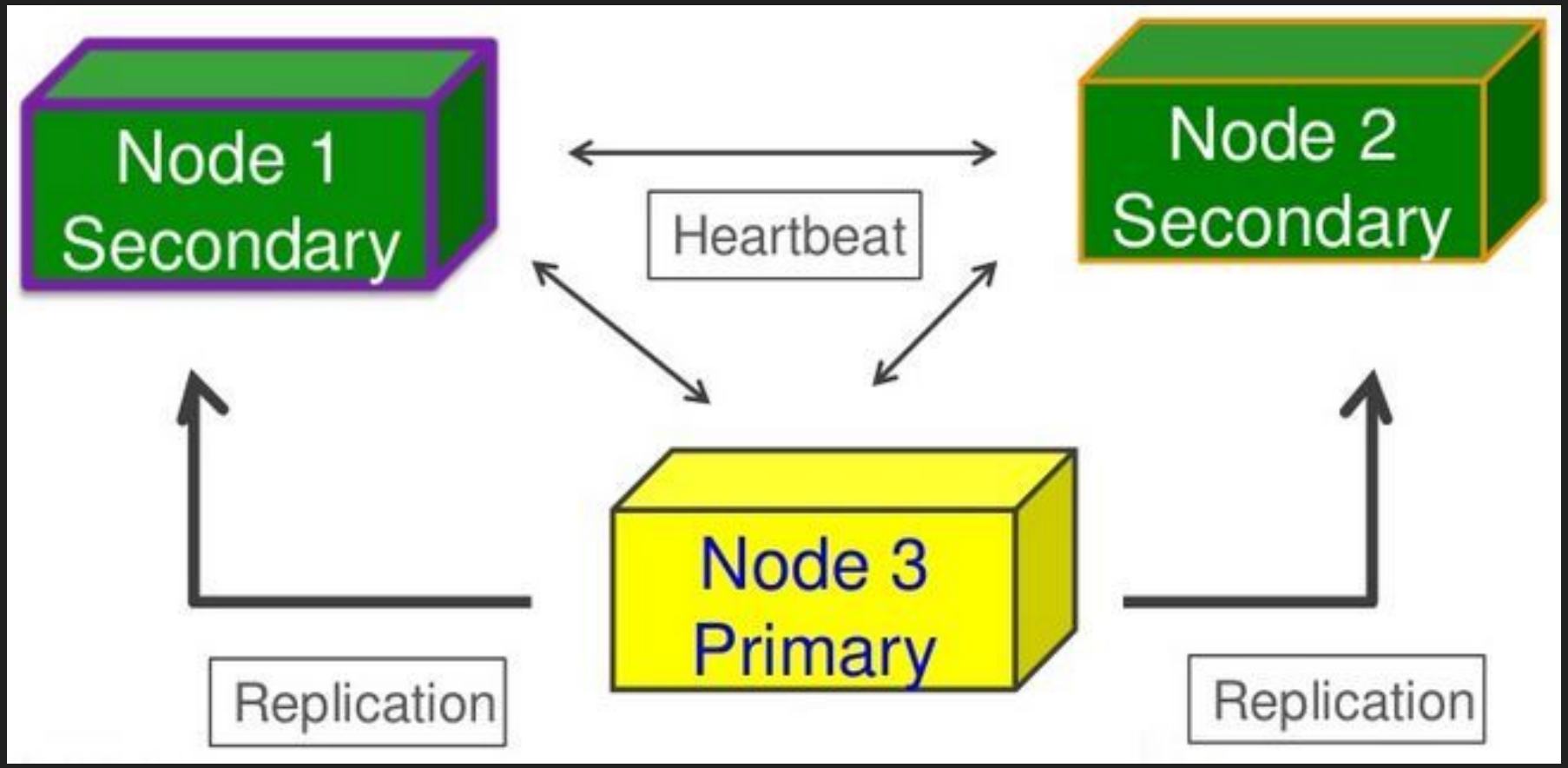
- Les bases ont des collections
- Les collections ont des documents
- Les documents ont des champs
- Les champs contiennent
  - des types simples
  - des tableaux
  - d'autres documents
- Le schéma est souple

Les points forts

- Performances
- haute disponibilité
- Scaliabité automatique



Haute disponibilité



MongoDB

1. Télécharger MongoDB http://www.mongodb.org/downloads
2. Installer (.msi, .tar.gz ou .deb)
3. Se connecter :
    a. mongo --host 127.0.0.1 --port 27017

Premiers pas

- Pour voire les bases disponibles
  - ```
    > show dbs
    local          0.078GB
    test            0.031GB
    ```
- Pour choisir une base
  - ```
    > use test
    switched to db test
    ```
- Pour savoir quelle est la base courante
  - ```
    > db
    ```
- Pour avoir de l'aide
  - ```
    > help
    ```
- Pour avoir la liste des collections d'une base
  - ```
    > show collections
    ```
    ```

Premiers pas

- Pour insérer des données
  - **> a = {"Last Name": "Hammet","First Name": "Kirk","Date of Birth": "1976-05-05" }**

    {
    "Last Name" : "Hammet",
    "First Name" : "Kirk",
    "Date of Birth" : "1976-05-05"
    }
  - **> db.test.insert(a)**

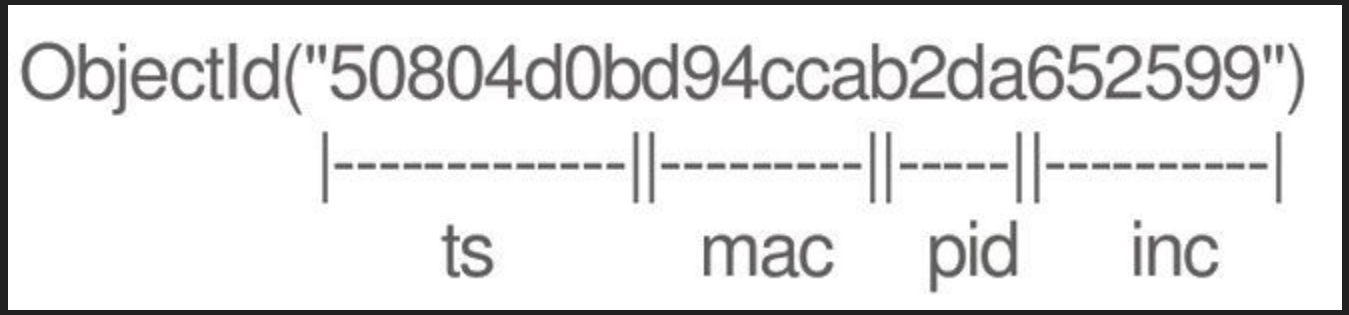
    WriteResult({ "nInserted" : 1 })
  - > b={"Field A": "Value A", "Field B": "Value B"}

    { "Field A" : "Value A", "Field B" : "Value B" }
  - **> db.test.insert(b)**

    WriteResult({ "nInserted" : 1 })

Premiers pas

- Pour requêter des données
- Tous les documents d'une collection
  - `> db.test.find()`
  - `{ "_id" : ObjectId("554a9944b5091037c44dddcc"), "Last Name" : "Hammet", "First Name" : "Kirk", "Date of Birth" : "1976-05-05" }`
    `{ "_id" : ObjectId("554a998eb5091037c44dddcd"), "Field A" : "Value A", "Field B" : "Value B" }`
- Clef primaire : _id
  - Automatiquement indexé
  - Généré sous forme d'ObjectId si non fourni
  - Doit être unique et immutable
  - Valeur 12 octets unique à travers le cluster



Utiliser Javascript dans 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" : "Hammet", "First Name"
: "Kirk", "Date of Birth" : "1976-05-05" }
{ "_id" : ObjectId("554a9944b5091037c44dddcc"), "Last Name" : "Hammet", "First Name"
: "kirk", "Date of Birth" : "1976-05-05" }
{ "_id" : ObjectId("554a998eb5091037c44dddcd"), "Field A" : "Value A", "Field B" :
"Value B" }
{ "_id" : ObjectId("554a9c21b5091037c44dddce"), "a" : 42, "b" : 0 }
{ "_id" : ObjectId("554a9c21b5091037c44dddcf"), "a" : 42, "b" : 1 }
{ "_id" : ObjectId("554a9c21b5091037c44dddd0"), "a" : 42, "b" : 2 }
{ "_id" : ObjectId("554a9c21b5091037c44dddd1"), "a" : 42, "b" : 3 }
{ "_id" : ObjectId("554a9c21b5091037c44dddd2"), "a" : 42, "b" : 4 }
```

Retrouver un document spécifique

```
> db.test.find({"Field A": "Value A"})
{ "_id" : ObjectId("554a998eb5091037c44dddcd"), "Field A" : "Value A", "Field B" :
"Value B" }

> db.test.find({ b: { $gt: 2 } }).sort({ b: -1 })
{ "_id" : ObjectId("554a9c21b5091037c44dddd2"), "a" : 42, "b" : 4 }
{ "_id" : ObjectId("554a9c21b5091037c44dddd1"), "a" : 42, "b" : 3 }
```

Opérateurs conditionnels :

- $all, $exists, $type, $mod,
- $or, $and, $not, $nor $size,
- $eq, $ne, $lt, $lte, $gt, $gte, $in, $nin…

Retrouver un document spécifique

Par expression régulière

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

Opérations

```
> db.test.insert(record)

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

> db.test.findOne(query)

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

Création d'index :

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

Questions ?