Bases de données documentaires et distribuées
Cours NFE04
Introduction ‘a MongoDB

Auteurs : Raphaël Fournier-S’niehotta, Philippe Rigaux, Nicolas Travers
prénom.nom@cnam.fr

Département d’informatique
Conservatoire National des Arts & Métiers, Paris, France

Qu'est-ce que MongoDB, ?

MongoDB, un système "NoSQL", l'un des plus populaires

- fait partie des NoSQL dits "documentaires" (avec CouchDB)
- s'appuie sur un modèle de données semi-structuré (encodage JSON) ;
- pas de schéma (complète flexibilité) ;
- un **langage d'interrogation** original (et spécifique) ;
- pas (ou très peu) de **support transactionnel**.

Construit dès l'origine comme un système **scalable** et **distribué**.

- distribution par partionnement (*sharding*) ;
- technique adoptée : découpage par intervalles (type BigTable, Google) ;
- tolérance aux pannes par réplication.

Objectifs de ce cours

Découvrir un système NoSQL $\Rightarrow$ session interactive commentée.

Modélisation. Comment modélise-t-on une application ?
Avantages/inconvénients vs. un système relationnel.

Un aperçu du **langage d'interrogation** de MongoDB.

A faire : récupérer les documents JSON sur le site BDPédia.

Le minimum à savoir

Vous connaissez déjà JSON ?

- Système client/serveur ; le serveur est `mongod`
- Open-source, multi plate-formes : installez-le chez vous !
- Un client en mode terminal : `mongo` ; langage Javascript ;
- D'autres clients plus ergonomiques, dont `RockMongo`, une interface Web.

Conçu et développé par 10gen.

Découvrons (avec le client `mongo`)

On se place dans une **base** `nfe204` (cd. MySQL)

```
use nfe024
```

On insère un **document** JSON dans une **collection**

```
> db.test.insert ({"nom": "nfe024"})
```

On affiche le contenu de la collection

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

On insère n'importe quel autre document (pas de **schéma**)

```
> db.test.insert ({"produit": "Grulband", prix: 230,
enStock: true})
```

On peut donner un identifiant explicite :

```
> db.test.insert ({_id: "1", "produit": "Kramölk", prix:
10, enStock: true})
> db.test.count()
```

Quelques documents plus sérieux

Sur le site `http://webscope.bdpedia.fr`, récupérer des documents JSON pour les films.
Exemple

```
{
  "_id": "movie:100",
  "title": "The Social network",
  "summary": "On a fall night in 2003, Harvard undergrad and
       programming genius Mark Zuckerberg sits down at his
       computer and heatedly begins working on a new idea. (...)",
  "year": 2010,
  "director": {"last_name": "Fincher",
               "first_name": "David"},
  "actors": [
       {"first_name": "Jesse", "last_name": "Eisenberg"},
       {"first_name": "Rooney", "last_name": "Mara"}
    ]
}
```

Aperçu du langage de requêtes

On peut exprimer l'équivalent du `select-from-where`, **sur une seule collection** (pas de **jointure**).

```
db.movies.find ({"_id": "movie:2"})
```

Avec des **expressions de chemin** (simples !)

```
db.test.find ({"director.last_name": "Scott"} )
```

Fonctionne **aussi** pour les acteurs (même si c'est un **tableau**).

```
db.test.find ({"actors.last_name": "Weaver"})
```

Mise à jour :

```
db.test.update ({"_id": "movie:2"},
                    {$set: {sous_titre: "Le passager"}} )

db.test.update ({"_id": "movie:2"}, {$unset: {country: 1}} )
```

Et on supprime tout :

```
db.test.remove()
```

Créer une petite base

Aller sur le site Webscope (`http://wbescope.bdpedia.fr`) et exporter les films et les artistes séparément en JSON.

Importer les données JSON avec `mongoimport`.

```
mongoimport -d nfe204 -c movies --file movies.json --jsonArray
```

Idem pour les artistes

```
mongoimport -d nfe204 -c artists --file artists.json --jsonArray
```

Les requêtes

Recherche par clé (très rapide).

```
db.artists.find({_id: "artist:99"})
```

Remarquez l'utilisation de `findOne`. Avec un chemin.

```
db.movies.find({"actors._id": "artist:99"})
```

Avec pagination

```
db.movies.find({"actors._id": "artist:99"}).skip(0).limit(2)
```

Avec une **projection**

```
db.movies.find({"actors._id": "artist:99"}, {"title": 1})
```

Avec une expression régulière (pas de guillemets !) :

```
db.movies.find({"title": /^Re/}, {"title": 1})
```

Par intervalle :

```
db.movies.find({"year": {$gte: 2000, $lte: 2005}, {"title": 1})
```

Les requêtes, suite

Quelques clauses ensemblistes

```
db.artists.find({_id: {$in: ["artist:34", "artist:98", "artist:1"]}, {
```

Autres possibilités : `$nin`, `$all`, `$exist`

```
db.movies.find({'summary': {$exists: true}}, {"title": 1})
```

Et les jointures

Il faut les effectuer du **côté client**.

Exemple (en supposant un schéma uniquement avec références) : tous les films réalisés par Clint Eastwood.

```
eastwood = db.artists.findOne({"first_name": "Clint", "last_name": "Ea
db.movies.find({"director._id": eastwood['_id']}, {"title": 1})
```

Notez le `findOne` !

Revient à effectuer une jointure par boucle imbriquées avec l'application client : très peu efficace à grande échelle.

Exemple : noms des réalisateurs (toujours dans un schéma avec références)

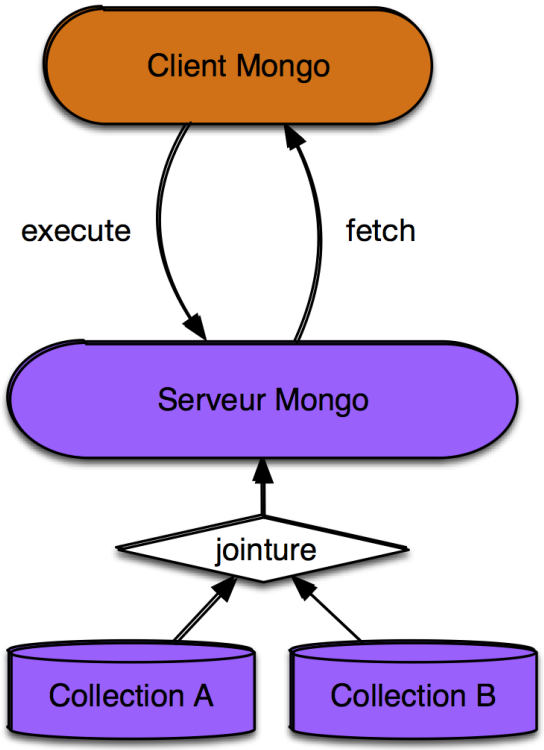
- Sélectionner dans `movies` tous les ids des réalisateurs

  ```
  directors = db.movies.find({}, {"director._id": 1})
  ```

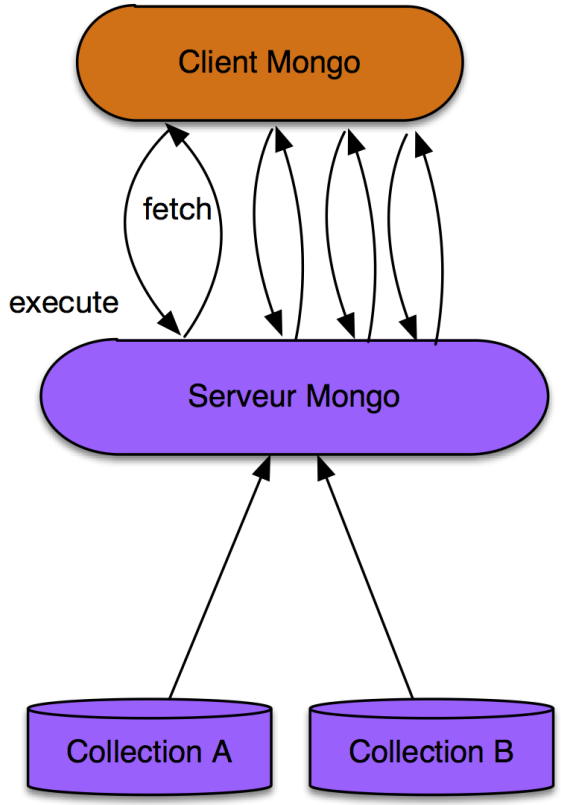
- Faire une boucle sur `directors`, pour chaque `director`

  ```
  noms = db.artists.find({"_id": director['_id']}, {"last_name": 1})
  ```

Comparaison des techniques de jointure



Scénario 1: la jointure s'effectue côté serveur



Scénario 2: la jointure s'effectue côté client