Imbrication avec Json et Mongo (base de données orientée document)

Table des matières

Cours

1. Exemple de base de données orientée document avec MongoDB

1.1. Présentation de MongoDB

MongoBD est une base de données *open source* NoSQL orientée document. Elle stocke des données au format JSON (en fait BSON, qui est une version binaire de JSON).

Le serveur MongoDB est organisé en plusieurs *databases* :

- Chaque *database* contient des *collections*.
- Chaque *collection* contient des *documents*.
- Chaque *document* est au format JSON et contient donc des propriétés.

Comparaison SQL / MongoDB

SQL	MongoDB
base de données et/ou schéma	base de données
table	collection
enregistrement	document
attribut (atomique)	propriété (chaîne, entier, tableau, structure)

Schema-less

C'est une base *schema-less*, aussi une collection peut contenir des documents de structures différentes et il n'est pas possible de définir la structure a priori d'une collection. La structure d'une collection n'est donc définie que par les document qui la compose, et elle peut évoluer dynamiquement au fur et à mesure des insertions et suppressions.

Identification clé / valeur

Chaque document est identifié par un identifiant nommé *_id* unique pour une collection, fonctionnant comme une **clé primaire artificielle**.

Architecture

MongoDB fonctionne a minima sous la forme d'un serveur auquel il est possible de se connecter avec un client textuel (*mongo shell*).

MongoDB peut être distribuée sur plusieurs serveurs (partitionnement horizontal ou *sharding*) et accédée à travers de multiples couches applicatives (langages, API...)

Complément

https://docs.mongodb.org/manual

1.2. Installation et démarrage de MongoDB

MongoDB est disponible sur Windows, Mac OS X et Linux : https://docs.mongodb.com/manual/installation

L'installation présentée ici est uniquement destinée à un contexte d'apprentissage, elle permet l'installation d'un serveur sur une machine Linux (ou Mac OS X) sans privilèges utilisateurs et l'exploitation de ce serveur avec un client textuel *CLI* ∗ situé sur la même machine.

Installation du serveur et du client

Installer *MongoDB Community Edition* sur son système.

Exemple sous Debian : `apt-get install mongodb-org` après avoir déclaré le dépôt MongoDB.

https://docs.mongodb.com/manual/tutorial/install-mongodb-on-debian/

Démarrage d'un serveur Mongo

Créer un espace de stockage en lecture écriture pour la base de données (exemple : `mkdir ~/mongodata`)

Lancer le serveur MongoDB sur le port standard 27017 : `mongod --dbpath ~/mongodata`

Démarrage du client CLI Mongo (mongo shell)

Pour se connecter à un serveur MongoDB sur le port standard : `mongo --host nom-du-serveur` (par exemple : `mongo --host localhost`)

Test

Une fois connecté exécuter le code suivant pour vérifier le bon fonctionnement de la base :

```
1 db.test.insert({ "test":"Hello World !" })
2 db.test.find({}, {_id:0})
```

Le résultat attendu est le suivant, la clé générée étant bien entendu différente :

```
{ "test" : "Hello World !" }
```

Complément

https://docs.mongodb.com/manual/mongo/[1]

1.3. Créer des bases de données et des collections

Créer une base et une collection *Fondamental*

MongoDB est une base *schema-less*, la création des bases et des collections est dynamique lors d'une première **insertion** de données.

Méthode

Pour créer une base de données il faut exécuter une instruction `use` sur une nouvelle base de données, puis donner un ordre d'insertion d'un premier document JSON avec `insert`.

Exemple

```
use db1
db.col1.insert( { "x":1 } )
```

[1] https://docs.mongodb.com/manual/mongo

Catalogue de données *Syntaxe*

- On peut voir la liste des bases de données avec :

  `show dbs`
- On peut voir la liste des collections de la base de données en cours avec :

  `show collections`
- On peut voir le contenu d'une collection avec :

  `db.col1.find()`

Complément

`https://docs.mongodb.com/manual/mongo`

1.4. Insertion des documents

L'insertion de données dans une base MongoDB se fait avec l'instruction `db.collection.insert(`*`Document JSON`*`)`.

Si la collection n'existe pas, elle est créée dynamiquement.

L'insertion associe un identifiant (*_id*) à chaque document de la collection.

Exemple

```
db.Cinema.insert(
{
   "nom":"Honkytonk Man",
   "realisateur":{
      "nom":"Eastwood",
      "prenom":"Clint"
   },
   "annee":1982,
   "acteurs":[
      {
         "nom":"Eastwood",
         "prenom":"Kyle"
      },
      {
         "nom":"Eastwood",
         "prenom":"Clint"
      }
   ]
}
)
```

Complément

https://docs.mongodb.org/manual/core/crud[1]

https://docs.mongodb.com/manual/tutorial/insert-documents[2]

[1] https://docs.mongodb.org/manual/core/crud/
[2] https://docs.mongodb.com/manual/tutorial/insert-documents/

1.5. Trouver des documents

La recherche de données dans une base MongoDB se fait avec l'instruction `db.collection.find(`*`Document JSON, document JSON`*`)`, avec :

- le premier document JSON définit une restriction ;
- le second document JSON définit une projection (ce second argument est optionnel).

Restriction — *Exemple*

```
db.Cinema.find({"nom":"Honkytonk Man"})
```

retourne les document JSON tels qu'ils ont à la racine un attribut "nom" avec la valeur "Honkytonk Man".

Restriction et projection — *Exemple*

```
db.Cinema.find({"nom":"Honkytonk Man"}, {"nom":1, "realisateur":1} )
```

retourne les document JSON tels qu'ils ont à la racine un attribut "nom" avec la valeur "Honkytonk Man", et seul les attributs situés à la racine "nom" et "realisateur" sont projetés (en plus de l'attribut "_id" qui est projeté par défaut).

Complément

https://docs.mongodb.org/manual/tutorial/query-documents/

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

2. Interroger Mongo en JavaScript

2.1. Interroger Mongo en JavaScript

Le console `mongo` permet d'exécuter des programme JavaScript avec instruction `load`.

```
//test.js
print("Hello world");
```

```
> load("test.js")
```

Parcours d'un résultat de requête Mongo

```
//query.js
conn = new Mongo();
db = conn.getDB("db1");

recordset = db.User.find({"liked":{$elemMatch:{"star":3}}}, {"_id":0,
"liked.film":1})

while ( recordset.hasNext() ) {
   printjson( recordset.next() );
}
```

Complément

https://docs.mongodb.com/manual/tutorial/write-scripts-for-the-mongo-shell/

Exercice

1. Exercice : Au ciné avec Mongo I

[1h30]

Soit les données suivantes représentant des films de cinéma.

```
db.Cinema.drop()

db.Cinema.insert(
{
nom:"Goodfellas",
annee:1990,
realisateur:{nom:"Scorsese", prenom:"Martin"},
acteurs:
  [
  {nom:"De Niro", prenom:"Robert"},
  {nom:"Liotta", prenom:"Ray"},
  {nom:"Pesci", prenom:"Joe"}
  ]
})

db.Cinema.insert(
{
nom:"The Godfather",
annee:1972,
realisateur:{nom:"Coppola", prenom:"Francis Ford"},
acteurs:
  [
  {nom:"Pacino", prenom:"Al"},
  {nom:"Brando", prenom:"Marlon"},
  {nom:"Duvall", prenom:"Robert"}
  ]
})

db.Cinema.insert(
{
nom:"Million Dollar Baby",
realisateur:{nom:"Eastwood", prenom:"Clint"},
acteurs:
  [
  {nom:"Swank", prenom:"Hilary"},
  {nom:"Eastwood", prenom:"Clint"}
  ]
})

db.Cinema.insert(
{
nom:"Gran Torino",
annee:2008,
realisateur:{nom:"Eastwood", prenom:"Clint"},
```

```
acteurs:
  [
  {nom:"Vang", prenom:"Bee"},
  {nom:"Eastwood", prenom:"Clint"}
  ]
})

db.Cinema.insert(
{
nom:"Unforgiven",
realisateur:{nom:"Eastwood", prenom:"Clint"},
acteurs:
  [
  {nom:"Hackman", prenom:"Gene"},
  {nom:"Eastwood", prenom:"Clint"}
  ]
})

db.Cinema.insert(
{
nom:"Mystic River",
realisateur:{nom:"Eastwood", prenom:"Clint"},
acteurs:
  [
   {nom:"Penn", prenom:"Sean"},
   {nom:"Bacon", prenom:"Kevin"}
  ]
})

db.Cinema.insert(
{
nom:"Honkytonk Man",
realisateur:{nom:"Eastwood", prenom:"Clint"},
annee:1982,
acteurs:
  [
   {nom:"Eastwood", prenom:"Kyle"},
   {nom:"Bloom", prenom:"Verna"}
  ]
})

db.Cinema.find()
```

L'objectif est d'initialiser une base MongoDB avec ce script, puis d'écrire les requêtes MongoDB permettant de répondre aux questions suivantes.

Question 1

Créer une nouvelle base MongoDB et exécuter le script. Nommez votre base par votre nom de famille ou votre login sur la machine par exemple.

Indice :

Pour créer une base de données, utiliser l'instruction `use myNewDatabase`, puis exécuter au moins une instruction d'insertion.

Indice :

Créer des bases de données et des collections (cf. p.4)

Question 2

Quels sont les films sortis en 1990 ?

Indice :

Trouver des documents *(cf. p.6)*

Indice :

`db.Cinema.find(`*`document JSON`*`)`

Indice :

La syntaxe JSON *(cf. p.13)*

Indice :

```
db.col.find({"attribute":"value"})
```

Question 3

Quels sont les films sortis avant 2000 ?

Indice :

https://docs.mongodb.com/manual/tutorial/query-documents

Indice :

On utilisera l'objet `{$lt:value}` à la place de la valeur de l'attribut à tester (*$lt* pour *lesser than*).

Question 4

Quels sont les films réalisés par Clint Eastwood ?

Indice :

On utilisera un objet comme valeur.

Question 5

Quels sont les films réalisés par quelqu'un prénommé Clint ?

Indice :

Utiliser le navigateur de propriété des objets *point* : *object.attribute*.

Question 6

Quels sont les films réalisés par quelqu'un prénommé Clint avant 2000 ?

Indice :

Utiliser une liste de conditions *attribut:valeur* pour spécifier un *AND* (et logique) :

```
db.col.find({"attribute1":"value1", "attribute2":"value2"})
```

Question 7

Quels sont les films dans lesquels joue Clint Eastwood ?

Indice :

https://docs.mongodb.com/manual/tutorial/query-array-of-documents/

Question 8

Quels sont les films dans lesquels joue un Eastwood ?

Question 9

Quels sont les noms des films dans lesquels joue un Eastwood ?

Indice :

Pour gérer la **projection**, utiliser un second argument de la clause *find()* :

`db.Cinema.find({`*document JSON de sélection*`}, {`*document JSON de projection*`})`

avec document JSON de projection de la forme : `{"attribut1":1, "attribut2":1...}`

Indice :

Les identifiants sont toujours affichés par défaut, si on veut les supprimer, on peut ajouter la clause `_id:0` dans le document de projection.

Question 10

Compléter le programme JavaScript suivant afin d'afficher les titre selon le format suivant :

```
- Million Dollar Baby
- Gran Torino
- Unforgiven
- Honkytonk Man
```

Indice :

```
conn = new Mongo();
db = conn.getDB("...");

recordset = ...

while ( recordset.hasNext() ) {
   film = recordset.next() ;
   print("- ", ...);
}
```

On veut à présent ajouter une nouvelle collection permettant de gérer des utilisateurs et leurs préférences. Pour chaque utilisateur on gérera un pseudonyme, et une liste de films préférés avec une note allant de une à trois étoiles.

Question 11

Ajouter trois utilisateurs préférant chacun un ou deux films.

On utilisera les identifiants des films pour les référencer.

Indice :

Insertion des documents *(cf. p.5)*

Indice :

https://docs.mongodb.com/manual/reference/method/ObjectId[1]

[1] https://docs.mongodb.com/manual/reference/method/ObjectId/

2. Exercice : Au ciné avec Mongo II

[30 min]

Soit les données suivantes représentant des films de cinéma et des avis d'utilisateurs concernant ces films.

```
db.Cinema.drop()

db.Cinema.insert(
{
nom:"Goodfellas",
annee:1990,
realisateur:{nom:"Scorsese", prenom:"Martin"},
})

db.Cinema.insert(
{
nom:"The Godfather",
annee:1972,
realisateur:{nom:"Coppola", prenom:"Francis Ford"},
})

db.Cinema.insert(
{
nom:"Million Dollar Baby",
realisateur:{nom:"Eastwood", prenom:"Clint"},
})

db.Cinema.insert(
{
nom:"Gran Torino",
annee:2008,
realisateur:{nom:"Eastwood", prenom:"Clint"},
})

db.Cinema.find()
```

```
db.User.drop()

db.User.insert(
{
 "pseudo":"Stph",
 "liked" :
   [
     {"film":ObjectId("590c366d70f50381c920ca71"),"star":3},
     {"film":ObjectId("590c366d70f50381c920ca72"),"star":1}
   ]
}
)

db.User.insert(
{
 "pseudo":"Luke",
 "liked" :
   [
     {"film":ObjectId("590c366d70f50381c920ca71"),"star":2}
   ]
}
)

db.User.insert(
```

```
{
 "pseudo":"Tuco",
 "liked" :
   [
     {"film":ObjectId("590c366d70f50381c920ca73"),"star":3}
   ]
}
)

db.User.find()
```

Question 1

Critiquer cette insertion mobilisant les identifiants des films ? Pourquoi n'est ce pas reproductible ? Imaginez deux solutions.

Question 2

Pourquoi trouver les titres de ces films n'est pas trivial avec Mongo (si, comme dans l'énoncé initial on a pas stocké le nom du film comme clé de référencement dans la collection User) ?

Question 3

On cherche à présent les identifiants des films qui sont aimés au moins une fois avec 3 étoiles.

On essaie cette requête : `db.User.find({"liked.star":3}, {_id:0, "liked.film":1})`, mais elle ne renvoie pas le résultat escompté.

Expliquez pourquoi ? Proposez des solutions.

Contenus annexes

1. La syntaxe JSON en bref

Règles syntaxiques — *Syntaxe*

- Il ne doit exister qu'un seul élément père par document contenant tous les autres : **un élément racine**.
- Tout **fichier JSON bien formé** doit être :
  - **soit un objet** commençant par { et se terminant par },
  - **soit un tableau** commençant par [et terminant par].

  Cependant ils peuvent être vides, ainsi [] et {} sont des JSON valides.
- Les **séparateurs** utilisés entre deux paires/valeurs sont des **virgules**.
- Un objet JSON peut contenir d'autres objets JSON.
- Il ne peut pas y avoir d'éléments croisés.

Éléments du format JSON — *Fondamental*

Il existe deux types d'éléments :

- Des couples de type **"nom": valeur**, comme l'on peut en trouver dans les tableaux associatifs.
- Des listes de valeurs, comme les tableaux utilisés en programmation.

Valeurs possibles — *Définition*

- Primitifs : nombre, booléen, chaîne de caractères, null.
- Tableaux : liste de valeurs (tableaux et objets aussi autorisés) entrées entre crochets, séparées par des virgules.
- Objets : listes de couples "nom": valeur (tableaux et objets aussi autorisés) entrés entre accolades, séparés par des virgules.

Exemple

```
{
   "nom cours" : "NF29",
   "theme" : "ingenierie documentaire",
   "etudiants" : [
                 {
                   "nom" : "Norris",
                   "prenom" : "Chuck",
                   "age" : 73,
                   "pays" : "USA"
                 },
```

```
                {
                  "nom" : "Doe",
                  "prenom" : "Jane",
                  "age" : 45,
                  "pays" : "Angleterre"
                },
                {
                  "nom" : "Ourson",
                  "prenom" : "Winnie",
                  "age" : 10,
                  "pays" : "France"
                }
              ]
}
```

Glossaire

CLI (Interface en ligne de commande)

Une interface en ligne de commande (en anglais *command line interface*) est une interface homme-machine dans laquelle la communication entre l'utilisateur et l'ordinateur s'effectue en mode texte.

https://fr.wikipedia.org/wiki/Interface_en_ligne_de_commande

Index