Dataswati
INDUSTRIES DATA-DRIVEN

ESGI école supérieure de génie informatique

Initiation au NoSQL

Etape 4 – BDD MongoDB – Fonctionnement

A propos

L'objectif de ce support de cours est de comprendre le fonctionnement de MongoDB et d'aborder un premier niveau de prise en main via le shell et l'outil RoboMongo.

Plan

- Définition
- Les propriétés de MongoDB
- Les standards
  - Les collections
  - Règles sur les clés
  - Fonctions d'écriture
  - Fonctions de lecture
  - Fonctions de mise à jour
  - Fonctions de suppression
  - Quelques fonctions utiles
  - Les Bulks
  - Pour aller plus loin

Contenu

Définition

Parmi les bases de données NoSQL, celles orientées « document » sont celles qui permettent de stocker, retrouver et gérer des données de type document.

Ce sont les principales BDD qui ont émergées avec le Big Data. En effet elles répondent à la règle des 3 V.

- **Volume**: De gros contenu stockable
- **Vélocité**: Rapidement
- **Variété**: Différent types de structures peuvent être intégrées.

Pour ce faire les bases de données documents doivent être scalables (possibilité de fonctionner en cluster), permettre une indexation rapide du contenu et un minimum d'interaction entre la commande et son traitement (Souvent au détriment que la garantie de transaction)

Quelques technologies

Nom	Provider	Avantages
CouchDB	Apache Software Foundation	Full http request (web compliant)
DocumentDB	Microsoft	?
Elasticsearch	Shay Banon	Orienté moteur de recherche (facettes)
MongoDB	Mongo DB Inc.	Interface facilitée dans beaucoup de langage. Shell dédié et efficace
Cassandra	Apache	Très efficace sur des volumes importants de données.

Les propriétés de MongoDB

Principe

Dans MongoDB, un enregistrement est un document composé d'un ou plusieurs éléments clés/valeurs.

Les document MongoDB sont similaire au objets JSON, il est ainsi possible de faire contenir un Array dans la valeur de clé.

Propriétés

High Performance
MongoDB provides high performance data persistence. In particular,

- Support for embedded data models reduces I/O activity on database system.
- Indexes support faster queries and can include keys from embedded documents and arrays.

Rich Query Language
MongoDB supports a rich query language to support read and write operations (CRUD) as well as:

- Data Aggregation
- Text Search and Geospatial Queries

High Availability
MongoDB's replication facility, called replica set, provides:

- automatic failover and
- data redundancy

A replica set is a group of MongoDB servers that maintain the same data set, providing redundancy and increasing data availability.

Horizontal Scalability
MongoDB provides horizontal scalability as part of its core functionality:

- Sharding distributes data across a cluster of machines.
- MongoDB 3.4 supports creating zones of data based on the shard key. In a balanced cluster,
- MongoDB directs reads and writes covered by a zone only to those shards inside the zone. See the Zones manual page for more information.

Support for Multiple Storage Engines
MongoDB supports multiple storage engines, such as:

- WiredTiger Storage Engine and
- MMAPv1 Storage Engine.

In addition, MongoDB provides pluggable storage engine API that allows third parties to develop storage engines for MongoDB.

Source https://docs.mongodb.com/manual/introduction/

Le schéma directeur n'existe pas

Une base MongoDB n'est pas portée par un schéma directeur, autrement dit, vous pouvez créer des schémas librement sans contrainte mais également sans maitrise.

Les performances peuvent s'en trouver limiter et pour obtenir plus de performance vous pouvez utiliser les collections 'déclarées' voir à partir de ce lien:

https://docs.mongodb.com/manual/reference/method/db.createCollection/#db.createCollection

Manipulation via le shell

Introduction

Le shell permet de réduire le temps de traitement en utilisant un langage très proche du résultat stocker.

Fonctionnement du Shell

Les commande du shell se décompose en 3 parties:

db.[collection].[function_command({contenu})]

Choix de la base

Il est possible d'utiliser plusieurs bases différentes dans lesquels seront stockées les collections. Le fonctionnement est identique au base SQL.

Use

Utilisé pour utiliser une base de données, si la base n'existe pas elle sera créée à la première collection insérée.

use db

Les collections

Certaines collections contiennent des informations sur le document. Elles peuvent être structuré (comme dans une base SQL standard ou non).

Les collections structurées

La déclaration de collection structurée permet de prédéfinir le format des contenus à insérer dans la collection. Les requêtes et le stockage en sera ainsi plus performant, mais au détriment de la souplesse d'utilisation.

```
db.createCollection(<name>, {
        capped: <boolean>,
        autoIndexId: <boolean>,
        size: <number>,
        max: <number>,
        storageEngine: <document>,
```

```
        validator: <document>,
        validationLevel: <string>,
        validationAction: <string>,
        indexOptionDefaults: <document>,
        viewOn: <string>,
        pipeline: <pipeline>,
        collation: <document>
} )
```

Option	Description
Capped	Limitation en taille de la collection
Size	Taille de la collection. Si de nouveaux documents sont créés au-delà de la limite, les documents les plus anciens sont supprimés.
Max	Nombre max de document

Validator	Donne une description des champs obligatoire et permet de renseigner un format pour les champs qui le sont.
	
	`{ $or:`
	`[`
	`{ phone: { $type: "string" } },`
	`{ email: { $regex: /@mongodb\.com$/ } },`
	`{ status: { $in: [ "Unknown", "Incomplete" ] } }`
	`]`
	`}`
	
	$or: Variable

Pour plus de détails sur les options non précisés dans le document voir ici :
https://docs.mongodb.com/manual/reference/method/db.createCollection/#db.createCollection

Les collections non structurées

Une collection non structurée se crée directement depuis une méthode d'insertion (voir ci-dessous) et ne nécessite pas de déclarations préalables.

Règles sur les clés

_id

L'éléments _id est l'attribut d'index obligatoire pour un item de collection. Lorsqu'il n'est pas renseigné par l'utilisateur, il se génère automatiquement. Il est unique par collection et doit être une chaine de caractère ou un int, le format array n'est pas accepté.

Attributs

Les attributs sont souvent libre, mais **ne peuvent pas contenir** les caractères spéciaux $ | .

Un attribut peut être de type: **Boolean | String | Int | Float | Array | JSON**.
Pour les 2 valeurs array et JSON, la profondeur n'est pas limitée.

Fonctions d'écriture

insertOne()

Insérer un item dans une collection, si la collection n'existe pas elle est créé.

```
db.collection.insertOne(
  <document>,
  {
    writeConcern: <document>
  }
)
```

Détail

Option	Syntaxe

<document>	{
	Cle : valeur,
	Cle : valeur,
	}

Pour plus d'informations sur writeConcern voir la page:
https://docs.mongodb.com/manual/reference/write-concern/

insertMany()

Idem à insertOne() à l'exception que l'on peut passer une collection de documents.

```
db.collection.insertMany(
  [<document 1>, <document 2>,...]
  {
    writeConcern: <document>
  }
)
```

Fonction de lecture

find()

Sert à chercher dans une collection et à afficher un résultat.

```
db.collection.find(
        <query>,
        <projection>
)
```

<query>
La query contient les informations permettant de trier les items de la collection

Option	Description
{}	Affiche tout le contenu

{clé : valeur}	Trouve uniquement les items ayant la valeur dans la clé. La valeur peut être numérique ou string.
Clé: { array }	Trouve les items ayant le tableau array dans la clé
{ "cle.sousclé": "valeur" }	Trouve les items ayant la valeur dans la sous-clé et clé.
cle: ["valeur1", "valeur2"] }	Trouve les items ayant la valeur1 et la valeur2 dans l'ordre dans la clé.
Pour plus de détails sur les options non précisés dans ce document voir ici : *https://docs.mongodb.com/manual/reference/method/db.collection.find/*	

<projection>
Identifie les clés à retourner dans la réponse. Même s'il n'est pas inclus, la clé _id sera toujours retourné depuis un find.

```
{
Clé 1 : 1,
Clé 2 : 1
}
```

Pour plus d'informations sur Find voir la page:
https://docs.mongodb.com/manual/reference/method/db.collection.find/

Fonction de mise à jour

Update()

La fonction update permet de modifier un ou plusieurs items identifié(s) dans une collection.

```
db.collection.update(
  <query>,
  <update>,
  {
    upsert: <boolean>,
    multi: <boolean>,
    writeConcern: <document>,
    collation: <document>
  }
)
```

<query>
Idem à la fonction find()

<update>

<other options>

Option	Description
upsert	Permet de créer l'élément si non identifié

multi	Modifie toutes les lignes identifiées si la valeur est TRUE. !!! Valeur défaut: FALSE !!!
writeConcern	*https://docs.mongodb.com/manual/reference/method/db.collection.update/#update-wc*
collation	*https://docs.mongodb.com/manual/reference/bson-type-comparison-order/#collation*

Pour plus de détails sur les options non précisés dans ce document voir ici :
https://docs.mongodb.com/manual/reference/method/db.collection.update/

Fonctions de suppression

drop()

Supprime l'ensemble de la collection.

remove()

Supprime les items identifiés dans une collection.

```
db.collection.remove(
  <query>,
  <justOne>
)
```

<query>
Idem find

<justOne>

```
{
   justOne: <boolean>,
   writeConcern: <document>,
   collation: <document>
}
```

Si justOne

Option	Description
justOne	Si à TRUE seule la première ligne identifiée est supprimée (par défaut à FALSE).
writeConcern	***https://docs.mongodb.com/manual/reference/method/db.collection.update/#update-wc***
collation	***https://docs.mongodb.com/manual/reference/bson-type-comparison-order/#collation***

Quelques fonctions utiles

findAndModify()

Ou comment tout faire en une commande.

Commande puissante mais régit par des règles précises

Pour plus de détails sur le fonctionnement de findAndModify :
https://docs.mongodb.com/manual/reference/method/db.collection.findAndModify/#db.collection.findAndModify

dropDatabase()

Au cas où vous souhaiteriez supprimer l'ensemble d'une DB.

Attention aucunes confirmations n'est demandées avant l'exécution

Aggregate()

Permet de réunir des éléments sur une base de conditions préalablement définis.

```
db.orders.aggregate([
                { $match: <query>},
                { $group: <group> },
                { $sort: { <sort> }
               ])
```

<query>
Idem au find

<group>

```
{
        _id: {clé: valeur de groupe},
        Nom_resultat : { $aggregation_function : $valeur à aggreger}
 }
```

Pour plus de détails sur le fonctionnement de group
https://docs.mongodb.com/manual/reference/method/db.collection.group/#db.collection.group

<sort>
Fonction de tri des résultats. La valeur: 1 correspond à ASC et la valeur -1 à DESC.

Plus d'informations sur le fonctionnement
https://docs.mongodb.com/manual/reference/method/db.collection.aggregate/#db.collection.aggregate

Disctinct()

Permet de grouper des résultats par clés.

```
db.collection.distinct(
        <field>,
        <query>
)
```

<field>
Le champ sur lequel appliquer le distinct

<query>
Les items sur lesquels faire la distinction. Fonctionnement idem au find

Plus d'informations sur le fonctionnement
https://docs.mongodb.com/manual/reference/method/db.collection.distinct/#distinct-method-options

Les BULKS

Le bulk actions est un moyen de lancer plusieurs requêtes différentes en 1 fois afin d'optimiser la requête à la BDD. Les bulks s'exécutent sur une seule collection à la fois.

Les bulk requests permettent la saisie en série de commandes et retourne une série de résultats. Cela à l'avantage de ne faire qu'une connexion auprès du serveur et d'éviter donc une sur sollicitation de celui-ci.
Si une Bulk Request, ne contient que des éléments d'insertion, elles seront mises dans la queue de traitement du serveur qui terminera la connexion.

- En cas de conflit entre 2 requêtes dernière requête du BULK l'emportera
- Bulk est à privilégier lorsque vous utilisez le serveur dans des conditions de charge importantes

Pour aller plus loin

Voir le site de MongoDB https://www.mongodb.com/