Introduction à MongoDB

Module BD et GL

Dr I Gueye

MongoDB

- MongoDB est une **Base de données Orientée Document et open-source.** C'est un des leaders du **NoSQL**. MongoDB est écrit en C++.

- Cette séance de TPE a pour objetcif de vous permettre de comprendre un peu plus les concepts de MongoDB, nécessaire pour créer des BD **orientée grande échelle et performances.**

MongoDB

- MongoDB est une BD cross-platform, orientée document qui offre une haute performance, une haute disponibilité, et un passage à l'échelle facile.
- MongoDB foncitonne avec les **concepts** de **collection** et **document**.

MongoDb : Database

- Database est un conteneur physique pour les collections.
- Chaque database reçoit son propre ensemble de fichiers sur le système de fichier.
- Un seul serveur MongoDB peut avoir (et l'a habituellement) plusieurs bases à gérer.

MongoDB : Collection

- Collection est un ensemble (un groupe)de documents de MongoDB. C'est l'équivalent d'une Table sur une une Base relationnelle.
- Une collection existes (est) une seule BD Les Collections ne requièrent pas un schema. Les Documents dans une collection peuvent avoir des champs différents.
- En général, tous les documents dans une collection sont là pour le même contexte ou un contexte similaire.

MongoDB: Document

- Un document est un ensemble de paires Clé-Valeur.
- Les Documents ont un schéma dynamique. Un schéma dynamic signifie que les documents dans la même collection n'ont pas besoin d'avoir les mêmes champs ou la même structure, et les champs communs dans les documents d'une collection peuvent contenir différents types de données.

MongoDB vs RDBMS

Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)
Database Server and Client	
Mysqld/Oracle	mongod
mysql/sqlplus	mongo

MongoDb : Exemple de doncument

- L’exemple suivant montre la structure (sous forme d’un document) d’un blog, qui est simplement en ensemble de pairs clé-valeur séparées par des virgules.

```
{
    _id: ObjectId(7df78ad8902c)
    title: 'MongoDB Overview',
    description: 'MongoDB est base de donnees NoSQL',
    by: ’Gueye Ibrahima',
    url: 'http://www.ibrahimagueye.com',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 100,
    comments: [
        {
            user:'user1',
            message: 'My first comment',
            dateCreated: new Date(2018,2,28,2,15),
            like: 0
        },
        {
            user:'user2',
            message: 'My second comments',
            dateCreated: new Date(2018,3,1,7,45),
            like: 5
        }
    ]
}
```

_id ???

- **_id** est un nombre hexadécimal de 12 bytes qui assure l'unicité de chaque document. Vous pouvez donner un _id en insérant un document. Si vous en donnez pas, MongoDB en génère un unique pour tout document.
- Les 4 premiers bytes des 12 bytes sont pour le timestamp current, les 3 bytes suivants pour l'id de la machine, les 2 suivants pour l'id du processus du serveur MongoDB et les 3 restants sont une valeur incrémentale simple.

- Toute BD relationelle a une conception d’un schéma qui montre le nombre de tables et les relations entre ces tables (voir MySQL et Oracle). Cependant, dans MongoDB il n’y aucune notion de relation.

Avantages de MongoDB par rapport aux SGBD relationnels

- Pas de schéma: MongoDB est une BD document dans laquelle une collection détient différents documents. Le nombre de champs, le contenu et la taille des documents peuvent changer d'un document à l'autre.
- La structure d'un objet est très claire.
- Pas de jointures complexes.
- Possibilité de requêtes profondes.
  - MongoDB supporte les requêtes dynamiques sur les documents en utilisant un langage de requête (basé-document) qui est presque aussi puissant que le SQL.
- Tuning (réglage/paramétrage avancés).
- Facilité de passage à l'échelle.
- Conversion/mapping d'objets des applications vers des objets BD pas nécessaire.
- Utilise la mémoire vive pour le stockage (par fenêtre temporelles) des données, permettant ainsi un accès plus rapide des données.

Pourquoi utiliser MongoDB?

- Stockage Orinté Document: Les données sont stockées sous la forme (du style) de document JSON.
- Index sur tous les attributs
- Réplication et haute disponibilité
- Auto-sharding
- « Requêtage » riche
- Mises à jour (écritures) très rapides
- Un support proffessionel assuré par MongoDB

Où utiliser MongoDB?

- Big Data
- Gestion de contenu (CMD:Content Management and Delivery)
- Plateforme Mobile et Social
- Gestion de données d'utilisateurs
- Data Hub
- Dans la plupart des applications web …

MondoDB: Modélisation des données

- Les données dans MongoDB ont un schéma flexible.
- Les documents dans collection n’ont pas forcément les ensembles de champs ou la même structure.

Quelques considérations pendant la conception d'un schéma pour MongoDB

- Faire un schéma en tenant compte des besoins des utilisateurs.
- Combiner les objets dans un même document si vous devez les utiliser ensemble. Autrement, les séparer (mais s'assurer qu'on aura pas besoin de jointures).
- Répliquer les données en tenant compte des contraintes d'espace disque et de CPU.
- Faire des jointures pendant les écritures et non pendant les lectures.
- Optimiser votre schéma pour les cas les plus fréquents.
- Faire les agrégations complexe dans le schéma.

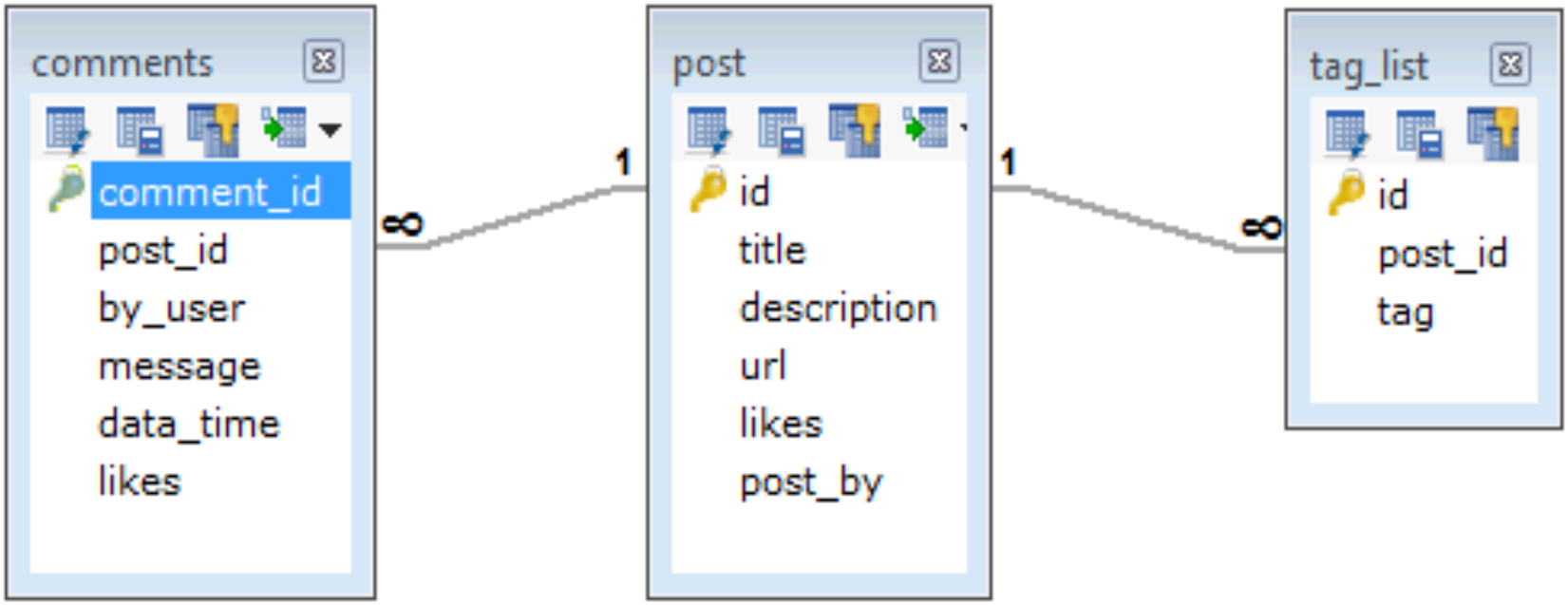
Exemple

Supposons qu'un client a besoin de concevoir une BD pour son blog/site. Regardez la différence entre les schémas RDBMS et MongoDB.

Le site web a les besoins suivants:

- Tout post a un titre unique, une description et une url.
- Tout post peut avoir un ou plusieurs tags.
- Tout post a le nom de celui qui l'a publié et le nombre de likes.
- Tout post a des commentaires laissés par les utilisateurs avec leur nom, message, datetime et les likes.
- Sur chaque epost, il peut y avoir zéro ou +sieurs commentaires.

Dans un schéma relationnel, la conception pour ces besoins au minimum 3 tables.



Tandis qu'un schéma MongoDB aura une seule collection post et la structure suivante:

```
{
   _id: POST_ID
   title: TITLE_OF_POST,
   description: POST_DESCRIPTION,
   by: POST_BY,
   url: URL_OF_POST,
   tags: [TAG1, TAG2, TAG3],
   likes: TOTAL_LIKES,
   comments: [
      {
         user:'COMMENT_BY',
         message: TEXT,
         dateCreated: DATE_TIME,
         like: LIKES
      },
      {
         user:'COMMENT_BY',
         message: TEXT,
         dateCreated: DATE_TIME,
         like: LIKES
      }
   ]
}
```

- Donc pour montrer les données dans une base relationnelle, vous aurez besoin de faire des jointures. Au contraire, dans MongoDB, les donnés seront lues depuis une seule collection.

MongoDB: Installation et Usage

- MongoDB: https://www.mongodb.com/fr
- Installation en local: télécharger la bonne version ici: https://docs.mongodb.com/manual/administration/install-community/

- Documentation du shell de MongoDB: https://docs.mongodb.com/manual/mongo/

- MongoDB Cloud Products: https://docs.mongodb.com/cloud/

MongoDB: Installation et Usage

- MongoDB Atlas: https://cloud.mongodb.com/
- Installation d'un cluster MongoDB via cloud. Rapide et simple
- Monitoring via interface web
- Utilisation via connexion distante depuis notre machine local. S'assurer qu'on a la bonne version du shell.
  - Mongo 'Url_du_cluster' --usernane 'login'
- Documentation Atlas: https://docs.atlas.mongodb.com/

Installation

- Installation sur machines locales et tests sur Cloud!
- Allez-y !

MongoDB: Create Database

- La commande ‘use’

- Syntaxe basique: use Database_Name.
- Cette commande crée une nouvelle bd nommée ‘Database_Name’ s’il n’existe pas, autrement elle retourne la bd existante.

MongoDB: showing the dbs

- Pour voir la BD en cours: commande ‘db’
- Vérifier la liste des bd: ‘show dbs’
  - Seules les bases contenant au moins un document seront listées.
- Dans MongoDB, la BD par défaut est ‘test’. Si vous n’en a avez pas créée, alors les collections seront stockées dans test.

MongoDB: Supprimer une BD

- La commande db.dropDatabase() supprime une BD qui existe.

- Elle supprime la base selectionnée. Si vous n'avez pas sélectionné de base, alors la base 'test' sera supprimée.

- Exemple: use mydb
- Db.dropDatabase() supprimera la base 'mydb'

MongoDB: Créer une Collection

- Syntaxe: db.createCollection(name, options)
- Dans cette commande, ‘name’ est le nom de la collection à créer. ‘options’ est un document et est utilisé pour spécifier la configuration de la collection.

Parameter	Type	Description
Name	String	Name of the collection to be created
Options	Document	(Optional) Specify options about memory size and indexing

MongoDB: Créer une Collection

Field	Type	Description
capped	Boolean	(Optional) If true, enables a capped collection. Capped collection is a fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size. **If you specify true, you need to specify size parameter also.**
autoIndexID	Boolean	(Optional) If true, automatically create index on _id field. Default value is false.
size	number	(Optional) Specifies a maximum size in bytes for a capped collection. **If capped is true, then you need to specify this field also.**
max	number	(Optional) Specifies the maximum number of documents allowed in the capped collection.

MongoDB: Créer une Collection

- Exemples:
  - Syntaxe basique de la méthode createCollection() sans options:

```
>use test
switched to db test
>db.createCollection("mycollection")
{ "ok" : 1 }
>
```

  - pour tester la collection créée utiliser la commande show collections .

MongoDB: Créer une Collection

- Exemples:
- syntaxe de createCollection() avec quelques options importantes:

```
>db.createCollection("mycol", { capped : true, autoIndexID : true, size :
6142800, max : 10000 } )
{ "ok" : 1 }
>
```

- Remarque: dans MongoDB, vous n'avez pas besoin de créer une collection. MongoDB crée automatiquement une collection, quand vous insérer un document.
- Exemple: > db.DIC1.insert({name:'Sanokho'})
- > show collections
- Que voyez-vous?

MongoDB: supprimer une Collection

- db.COLLECTION_NAME.drop()
- Exemple:
- Lister les collections existantes:

```
>use mydb
switched to db mydb
>show collections
mycol
mycollection
system.indexes
>
```

MongoDB: supprimer une Collection

- db.COLLECTION_NAME.drop()
- Exemple:
- Suppri;er mycollection:

```
>db.mycollection.drop()
true
>
```

drop() retourne true, si la collection sélectionnée est supprimée avec succès, sinon elle retourne false.

MongoDB: supprimer une Collection

- db.COLLECTION_NAME.drop()
- Exemple:
- Lister de nouveau les collections existantes:

>use mydb

switched to db mydb

>show collections

Mycol

system.indexes

>

MongoDB: Types de données

- **MongoDB** supporte beaucoup de types de données dont certains sont:
- **String** : Le type le plus utilisé pour stocker des données avec MongoDB. Doit respecter la norme UTF-8.
- **Integer** (32 bit ou 64 bit selon le serveur), **Boolean** (true/ false), **Double**, **Arrays** (tableaux ou listes ou valeurs multiples) à stocker dans une seule clé., **Timestamp**,
- **Object** : type utilisé pour les documents dits « embedded», **Null**, **Date**,
- **Code** (used to store JavaScript code into the document),
- **Regular expression** (expressions régulières)

MongoDB: Insérer un document

- **> db.COLLECTION_NAME.insert(document)**
- **Exemple:**

```
> db.mycol.insert({cle: ‘Valeur’})
```

MongoDB: Insérer un document

- **> db.COLLECTION_NAME.insert(document)**
- **Exemple:**

```
>db.mycol.insert({
title: ’Mr',
name: ’Ndao',
fonction: ’Responsqble’
})
```

MongoDB: Insérer un document

- **> db.COLLECTION_NAME.insert(document)**
- **Exemple:**

```
> db.mycol.insert({
_id: ObjectId(7df78ad8902c),
title: 'MongoDB Overview',
description: 'MongoDB is a No SQL database',
classe: ’dic1 GIT',
url: 'http://www.ept.sn',
tags: ['mongodb', 'database', 'NoSQL’, ‘EPT’],
likes: 100
})
```

MongoDB: requête de lecture

- **> db.COLLECTION_NAME.find()**

MongoDB: requête de lecture

Operation	Syntax	Example	RDBMS Equivalent
Equality	{<key>:<value>}	db.mycol.find({"by":"tutorials point"}).pretty()	where by = 'tutorials point'
Less Than	{<key>:{$lt:<value>}}	db.mycol.find({"likes":{$lt:50}}).pretty()	where likes < 50
Less Than Equals	{<key>:{$lte:<value>}}	db.mycol.find({"likes":{$lte:50}}).pretty()	where likes <= 50
Greater Than	{<key>:{$gt:<value>}}	db.mycol.find({"likes":{$gt:50}}).pretty()	where likes > 50
Greater Than Equals	{<key>:{$gte:<value>}}	db.mycol.find({"likes":{$gte:50}}).pretty()	where likes >= 50
Not Equals	{<key>:{$ne:<value>}}	db.mycol.find({"likes":{$ne:50}}).pretty()	where likes != 50

MongoDB: requête de lecture

- **> db.COLLECTION_NAME.findOne()**
  - Afficher une collection
- db.macollection.find({‘Sexe’:’Homme’})
  - Retourne tous les documents pour lesquels le champ sexe vaut ‘Homme’
  - Pour compter le nombre de document s dans le resutatil faut ajouter count()
    - db.macollection.find(‘Sexe’:’Homme’).count()

MongoDB: requête de lecture

- db.macollection.find (
  { ‘Sexe’ : ‘Homme’,
  ‘Classe’ : ‘DIC1’ })

  – Filtrage selon deux critères
    - Même principe pour plus de critères.

Filtrage avec des opérations

$gt, $gte	>, ≥	Plus grand que (*greater than*)	"a" : {"$gt" : 10}
$lt, $lte	<, ≤	Plus petit que (*less than*)	"a" : {"$lt" : 10}
$ne	≠	Différent de (*not equal*)	"a" : {"$ne" : 10}
$in, $nin	∈, ∉	Fait parti de (ou ne doit pas)	"a" : {"$in" : [10, 12, 15, 18] }
$or	∨	OU logique	"a" : {“$or” : [{"$gt" : 10}, {“$lt” : 5}] }
$and	∧	ET logique	"a" : {“$and” : [{"$lt" : 10}, {“$gt” : 5}] }
$not	¬	Négation	“a” : {“$not” : {"$lt" : 10} }
$exists	∃	La clé existe dans le document	“a” : {“$exists” : 1}
$size		test sur la taille d'une liste (uniquement par égalité)	“a” : {“$size” : 5}

Exercice

- Creer une BD TP2
  - Creer deux collections
    - GIT
      - Avec des documents ayant la structure suivants
        - » {numero:, nom:, prenom: , sexe:, classe:, annee_arrivee:}
    - EPT
      - Avec des documents ayant la structure suivants
        - » {numero:, nom:, prenom:, Filiere:, annee:}
  - Inserer au moins 10 documents dans chaque collection.

Exercice

- Trouver les documents contenant les eleves du GIT qui sont au departement depuis 2017

- Trouver les documents contenant des filles en DIC1

Complement

- sélectionner les documents dont un array contient une valeur x
- db. colName.find({ tags: 'business' })
- ou un élément parmi plusieurs
- db. colName.find({ tags: { $in: ['business', 'french'] } })
- ou la combinaison de plusieurs
- db. colName.find({ tags: { $all: ['business', 'french'] } })
- sélectionner les documents dont la valeur est différente de
- db. colName.find({ birthyear: { $ne: 2000 } })

Complement

- db.oldname.renameCollection("newname")
- db.dropDatabase()
  - Efface la bd courante

Complement

- Mise à jour dun document
- db.colName.update({ _id: x }, { $set: { "firstname": ”NeWFirstName" } })

- # effacer un document
- db.colName.remove({ _id: ObjectId("55accc6c039c97c5db42f192") })

- # effacer tous les documents d'une collection
- db.colNmae.remove({})