MongoDB : Base de donnée orientée documents.

UE « Systèmes d'Information Tactiques »
Master 2 Génie Logiciel
ISTIC, Université de Rennes 1
Année 2015-2016, second semestre

Emmanuel Caruyer
CR CNRS
Emmanuel.Caruyer@irisa.fr

Plan du cours

MongoDB : Présentation générale

- SQL ou *NoSQL* ?
- L'organisation en documents
- Un modèle sans schéma mais pas sans conception
- Améliorer les performances avec un index

Développer avec MongoDB

- Utilisation de la console Javascript
- Interaction à l'aide de l'API Java
- Écriture d'une requête
- Améliorer les performances avec un index

Données géographiques

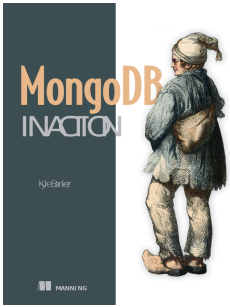
- Requêtes géographiques

MongoDB : Bibliographie

- Kyle Banker, MongoDB in Action (Dec. 2011), Ed. Hanning.

MongoDB Reference

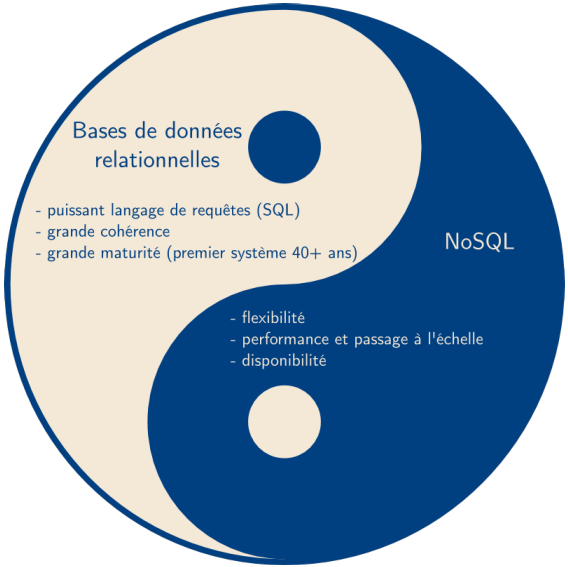
- https://docs.mongodb.org/manual/reference/database-references/
- https://docs.mongodb.org/ecosystem/drivers/java/



MongoDB : Présentation générale

- SQL ou *NoSQL* ?
- L'organisation en documents
- Un modèle sans schéma mais pas sans conception

SQL ou *NoSQL* ?


Bases de données relationnelles
- puissant langage de requêtes (SQL)
- grande cohérence
- grande maturité (premier système 40+ ans)
NoSQL
- flexibilité
- performance et passage à l'échelle
- disponibilité

MongoDB : Présentation générale

Un SGBD innovant

- Base de donnée de type *NoSQL* (Not only SQL)
- L'élément de base est le document

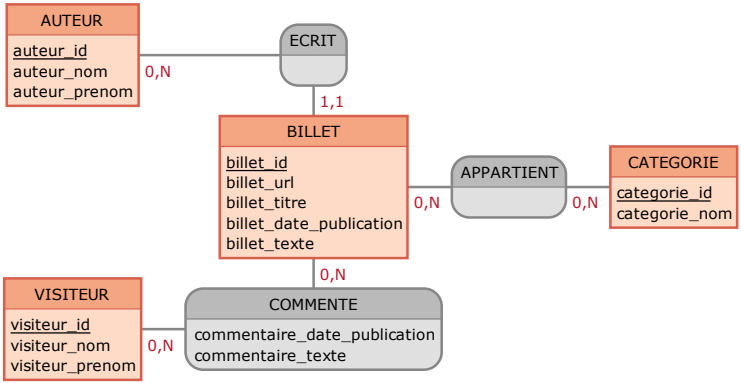
Un système ouvert et pensé Web

- Distribué sous licence *AGPL* (libre et ouvert)
- Format BSON, interactions Javascript, APIs via drivers

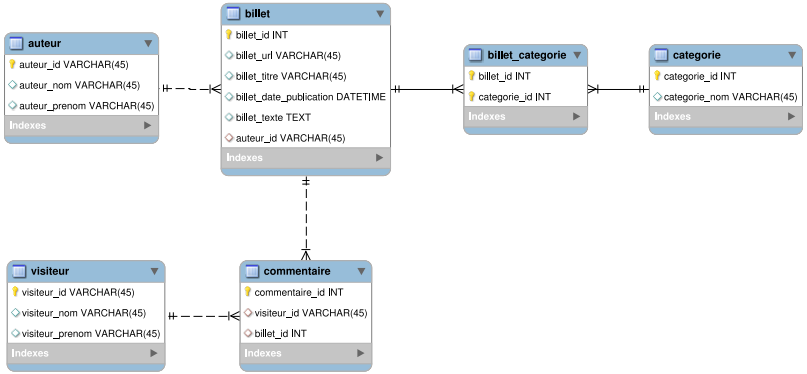
Optimisé pour une disponibilité maximale

- Faible structuration relationelle au profit de la rapidité d'accès.
- Écrit en C++
- Montée en charge par multiplication du nombre de serveurs

Étude d'un exemple : Blog (1)



Étude d'un exemple : Blog (2)



Étude d'un exemple : Blog (3)

```
{
  _id: ObjectID('4bd9e8e17cefd644108961bb'),

  titre: "La Renault Twizy, entre scooter et citadine",
  auteur: "ecaruyer",

  categories: ["automobile", "écologie", "zéro émission"],

  commentaires: [
    {
      visiteur: "david35",
      texte: "Merci pour cet article !"
    },
    {
      visiteur: "melody_nelson",
      texte: "Cette voiture est top, mais pêche encore par son autonomie..."
    }
  ],

  texte: "La citadine <strong>Renault Twizy</strong> [...]"
}
```

Les documents contiennent des données typées

```
{
  nom: "ISTIC",
  description: "UFR Informatique - Électronique, Université de Rennes 1",
  categories: ["école", "université", "centre de recherche"],
  localisation: [48.115374, -1.638845],
  adresse: {
    numéro: 263,
    libellé: "Avenue",
    adresse1: "Général Leclerc",
    code_postal: 35000,
    ville: "Rennes"
  }
}
```

chaîne de caractères

tableau

données géospatiales

document intégré

Un modèle sans schéma *a priori*

L'organisation sous forme clé-valeurs

- C'est l'application qui garantit la cohérence des données au sein d'une collection
- Ce type de SGBD est particulièrement adapté au développement agile
- Les valeurs peuvent représenter des objets complexes ou des listes d'objets

Un SGBD sans jointure ni transaction

- Toutes les informations recueillies par une requête se trouvent dans le document
- Favorise la rapidité et la simplicité au prix d'une possible redondance dans les données

L'importance du design

Compromis redondance/rapidité d'exécution

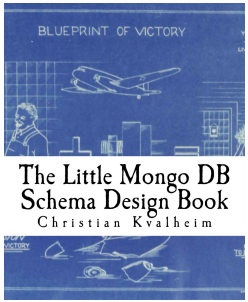
- Les données d'une requête se trouvent généralement toutes dans le document
- La taille maximale des documents est plafonnée (16Mo)
- Nécessité de trouver un compromis entre redondance et rapidité

Remarques générales

- Importance des scenarii d'utilisation (lectures/écritures) sur la façon d'organiser les données
- Ne pas se limiter à un enregistrement par document
- Quand le volume de données augmente, le nombre de documents doit augmenter

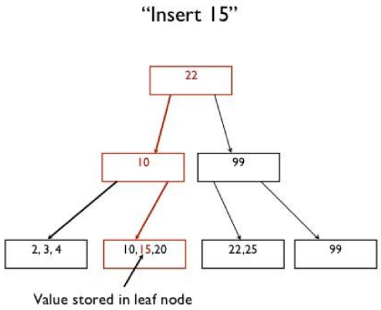
L'importance du design

Bibliographie complémentaire



Améliorer les performances avec un index (1)

Comme les SGBD relationels, MongoDB utilise un B-Tree



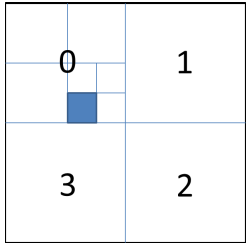
Améliorer les performances avec un index (2)

Les indexations "full text"

- Un index inversé de tous les mots présents dans un champs
- Permet de faire des recherches par le contenu

L'indexation géospatiale

- Indexation basée sur un quadtree



Développer avec MongoDB

- Utilisation de la console Javascript
- Interaction à l'aide de l'API Java
- Écriture d'une requête
- Améliorer les performances avec un index

Utilisation de la console Javascript (1)

Lancement du serveur et de la console

- Le serveur se lance à l'aide de la commande `mongod.exe`
- L'interpréteur de commandes Javascript correspond à la commande `mongo.exe`

Connexion ou création d'une base de donnée

```
use <nom_de_la_base>
```

Insertion d'un document dans une collection

```
db.<nom_de_la_collection>.insert(<document>)
```

Lecture d'une collection

```
db.<nom_de_la_collection>.find([paramètres])
```

Utilisation de la console Javascript (2)

```
ecaruyer@victoria ~ $ mongo
MongoDB shell version: 2.4.9
connecting to: test

> use bibliographie
switched to db bibliographie

> document = {
... auteurs: ["Antoine Dupont", "Frédéric Petit"],
... titre: "Databases for the next generation",
... editeur: "University Press",
... pages: 525,
... categories: ["informatique", "vulgarisation", "technologies web"]
... }

> db.articles.insert(document)

> db.articles.find()
{ "_id" : ObjectId("569c5dac1dd328fd9d406fb0"), "auteurs" : [  "Antoine
Dupont",  "Frédéric Petit" ], "titre" : "Databases for the next generation",
"editeur" : "University Press", "pages" : 525, "categories" : [
"informatique",  "vulgarisation"] }
```

Utilisation de la console Javascript (3)

Modification d'une entrée dans la base

```
db.articles.find({auteurs: "Antoine Dupont"})
  .update({"$addToSet": {"categories": "technologies web"}})
```

Modification de tutes les résultats d'une requête

```
db.articles.find({auteurs: "Antoine Dupont"})
  .update({"$addToSet": {"categories": "technologies web"}}, false, true)
```

Suppression d'une entrée

```
db.articles.remove({auteurs: "Antoine Dupont"})
```

Intéraction à l'aide de l'API Java

```
// Create a client to connect to the MongoDB server on localhost
MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

// Connect to a specific database.
// Note: if "mydb" does not exist, it'll be create on the first write.
MongoDatabase database = mongoClient.getDatabase("bibliographie");

// Retrieve (or create) the collection "articles".
MongoCollection collection = database.getCollection("article");

// Make a document and insert it
Document doc = new Document("auteurs", Arrays.asList("Antoine Dupont", "Frédéric Petit"))
               .append("titre", "Databases for the next generation")
               .append("editeur", "University Press")
               .append("categories", Arrays.asList("informatique", "vulgarisation", "technologies web"));

collection.insertOne(doc);

// Query the collection
Document myDoc = collection.find().first();
```

Écriture d'une requête (1)

Une requête est représentée par un document

```
db.articles.find({editeur: "University Press"})
```

On peut enrichir une requête à l'aide d'opérateurs

```
db.articles.find({pages: {"$gt": 500}})
```

```
db.articles.find({auteurs: {"$in": ["Eric Legrand", "Antoine Dupont"]}})
```

```
db.articles.find({auteurs: {$not: /^F/}})
```

```
db.billets.find({"categories": "automobile",
                 "commentaires.visiteur": "melody_nelson"})
```

```
db.billets.find({"$or": [{"categories": "automobile"},
                         {"commentaires.visiteur": "melody_nelson"}]})
```

Écriture d'une requête (2)

Récupération d'un sous-ensemble du document (projection)

- Par soucis d'optimisation, on peut ne récupérer que les éléments nécessaires d'un document

```
db.articles.find({editeur: "University Press"},
                 {auteurs: 1})
```

Tri des résultats

- Il est possible de trier les résultats d'une recherche

```
db.articles.find({}).sort({titre: -1})
```

Améliorer les performances avec un index (1)

Comprendre l'exécution d'une requête : `explain()`

```
> db.articles.find({"auteurs.nom": "Dupont"}).explain()
{
        "cursor" : "BasicCursor",
        "isMultiKey" : false,
        "n" : 1,
        "nscannedObjects" : 2,
        "nscanned" : 2,
        "nscannedObjectsAllPlans" : 2,
        "nscannedAllPlans" : 2,
        "scanAndOrder" : false,
        "indexOnly" : false,
        "nYields" : 0,
        "nChunkSkips" : 0,
        "millis" : 0,
        "indexBounds" : {

        },
        "server" : "victoria:27017"
}
```

Améliorer les performances avec un index (2)

Créer un index avec `createIndex`

```
> db.articles.createIndex({"auteurs.nom": 1})
> db.articles.find({"auteurs.nom": "Dupont"}).explain()
{
        "cursor" : "BtreeCursor auteurs.nom_1",
        "isMultiKey" : true,
        "n" : 1,
        "nscannedObjects" : 1,
        "nscanned" : 1,
        "nscannedObjectsAllPlans" : 1,
        "nscannedAllPlans" : 1,
        "scanAndOrder" : false,
        "indexOnly" : false,
        "nYields" : 0,
        "nChunkSkips" : 0,
        "millis" : 0,
        "indexBounds" : {
                "auteurs.nom" : [
                        [
                                "Dupont",
                                "Dupont"
                        ]
                ]
        },
        "server" : "victoria:27017"
}
```

Données géographiques

Requêtes géographiques (1)

Exemple de collection contenant des données de géolocalisation

```
{
  "_id" : ObjectId("4d291187888cec7267e55d24"),
  "city" : "New York City",
  "loc" : {
    "lon" : -73.9996
    "lat" : 40.7402,
  },
  "state" : "New York",
  "zip" : 10011
}
```

Création d'un index géographique

```
> db.zips.createIndex({loc: '2dsphere'})
```

Requêtes géographiques (2)

Récupération des *k* plus proches voisins

```
> db.zips.find({'loc': {$near: [ -73.977842, 40.752315 ]}}).limit(3)
```

Récupération des enregistrements à l'intérieur d'un disque

```
> center = [-73.977842, 40.752315]
> radius = 0.011
> db.zips.find({loc: {$within: {$center: [ center, radius ] }}})
```

Récupération des enregistrements à l'intérieur d'un rectangle

```
> lower_left = [-73.977842, 40.752315]
> upper_right = [-73.923649, 40.762925]
> db.zips.find({loc: {$within:
{$box: [ lower_left, upper_right ] }}})
```

Questions ?