mongoDB

INTRODUCTION

Mme Ines Slimene
Ines_slimene@yahoo.fr

Plan

Modèle JSON et Organisation

Modèlisation

Replica Set

Sharding

Sauvegarde et restauration des données

Modèle et organisation

Introduction

Un document est la représentation d'une donnée en BSON.

BSON = Binary JSON.

Extension du JSON (support officiel du type Date, ...).

Pas de schéma, pas de relation, pas de transaction.

Modèle JSON

JSON (Javascript Object Notation) est le format de sérialisation des objets Javascript.

Avantage principal : il est possible de le parser directement sous la forme d'un objet du langage (javascript, mais aussi tous les langages de scripts, voire Java ou C++)

Une valeur est soit une chaîne de caractère, un numérique, un Booléen, un objet ou un tableau.

Un objet est une liste non-ordonnée de paires (clé, valeur).

Exemple: {first_name: 'Joe', last_name: 'Doe'}

Un tableau est une liste ordonnée de valeurs.

["doe@cnam.fr", "john.doe@cnam.fr","doej@cnam.fr"]

Exemple avec JSON

Point de départ : listes associatives, i.e., des enregistrements avec des paires (clé, valeur).

{nom: "Alan", tél: 2157786, email: "agb@abc.com"}

Extension naturelle : les valeurs sont elles-mêmes d'autres structures.

{nom: {prénom: "Alan", famille: "Black"},tél: 2157786, email: "agb@abc.com"}

Autre exemple: imbrication de listes.

{name: "Alan", téls: [2157786, 2498762] }

Organisation

Un serveur MongoDB est composé de bases de données.

Une base de données contient des collections.

Les collections possèdent les documents.

Chaque document possède un identifiant unique généré par MongoDB, le champ _id.

Organisation

RDBMS	MongoDB
Database	Database
Table, View	Collection
Row	Document (JSON, BSON)
Column	Field
Index	Index
Join	Embedded Document
Foreign Key	Reference
Partition	Shard

Documents

Un document est l'équivalent d'une ligne dans une base de données SQL avec quelques différences :

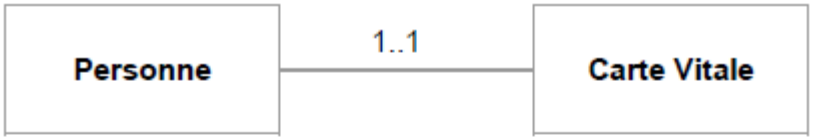
- Les documents sont sans schéma
- Les documents utilise le syntaxe BSON
- La taille max d'un document est 16 mégabytes
- Non Relationnel

Documents

Exemple :

```
{
  name : ‘MongoDB’,
  type  : ‘database’,
  count : 1,
  info   :{
            x : 203,
            y : 102
           }
}
```

Modélisation: quelques règles simples

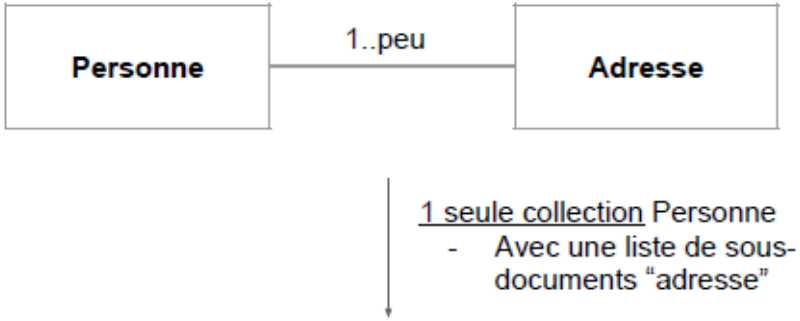


1 seule collection Personne
- Avec un sous-document "carte_vitale"

Collection "Personne"

```
{
  nom: "...",
  carte_vitale: {
     date_emission: "...",
     cnam: "..."
   }
}
```

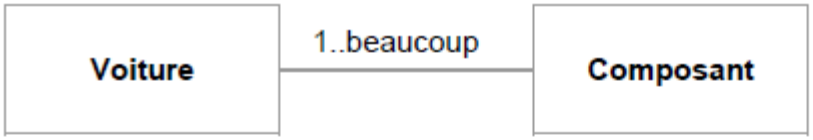
Modélisation: quelques règles simples



Collection "Personne"

```
{
  nom: "Séraphin Lampion",
  adresses: [
     {
        rue: "rue de la Roquette",
        ville: "Paris"
     },
     {
        rue: "chemin du chateau",
        ville: "Moulinsart"
     }
  ]
}
```

Modélisation: quelques règles simples



2 collections Voiture et Composant
- C'est Voiture qui référence les composants

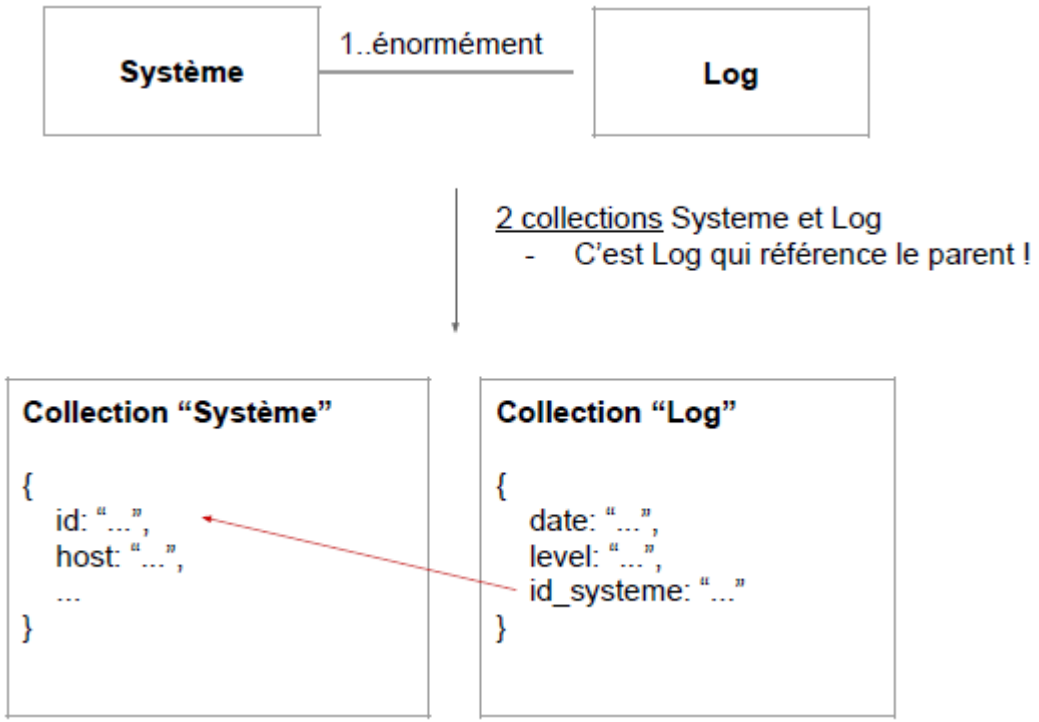
Collection "Voiture"

```
{
   marque: "...",
   modele: "...",
   composants: [
      "Id_101",
      "Id_22",
      "Id_345",
      ...
   ]
}
```

Collection "Composant"

```
{
   id: "...",
   type: "...",
   prix: ...
}
```

Modélisation: quelques règles simples



Modélisation: quelques règles simples

Ne pas oubliez d'en tenir compte !

- "J'ai besoin de l'ensemble des données à chaque requête"
  - Mettez tout dans une seule collection
- "J'ai besoin d'en avoir seulement une partie"
  - Faites plusieurs collections et des références
  - Ex: les posts d'un blog et leurs commentaires :
    - 2 besoins : affichage liste des posts + affichage post avec commentaires
    - Modélisation avec 2 collections (posts, comments)

Références entre docs

Avantages à séparer dans plusieurs docs ?

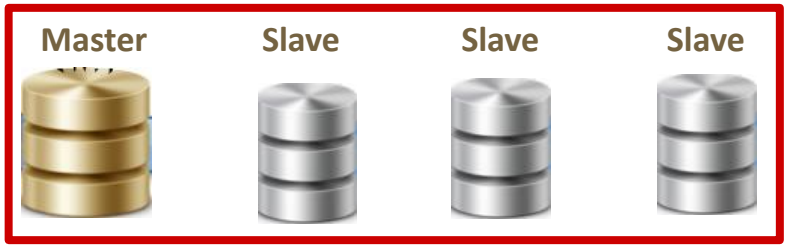
- **Flexibilité**
  - Changement sur une collection sans impact sur les autres
- **Cardinalité forte / gros document**:
  - Attention à la place mémoire
  - Taille max des docs: 16Mo

Désavantages à séparer dans plusieurs docs ?

- On risque d'avoir à gérer des jointures dans l'application, compliqué !

Sharding et Réplica Set

Réplica Set : cohérence éventuelle



La lecture des données est effectuée dans le master.

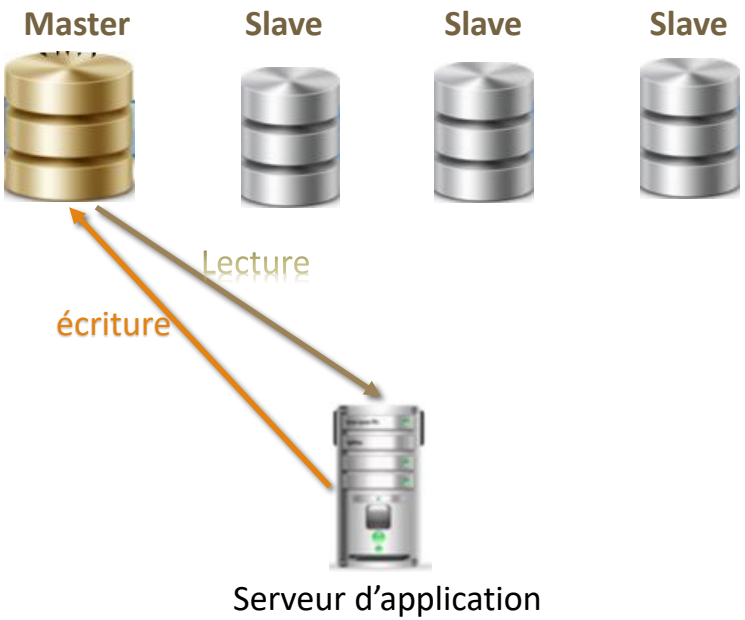
La lecture peut être effectuée a partir du slave mais problème de consistance (la réplication est asynchrone) => cohérence éventuelle

Réplica set : Élection

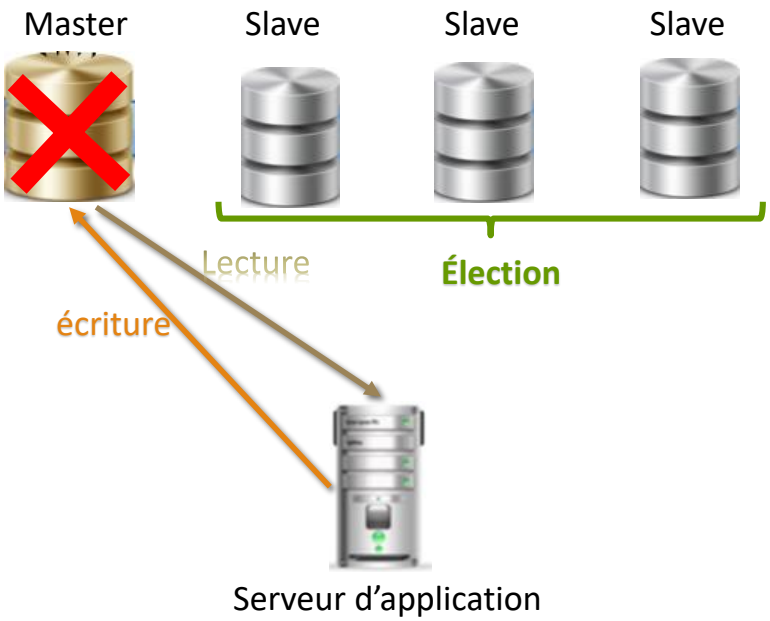
Type de nœud:

- regular : a full copy of data & voting & ready to be primary
- Passive : a full copy of data & voting
- arbiter : voting & no data replicated (ne peut pas être un master)
- Hidden : ne peut pas être un master

Réplica set : Failover

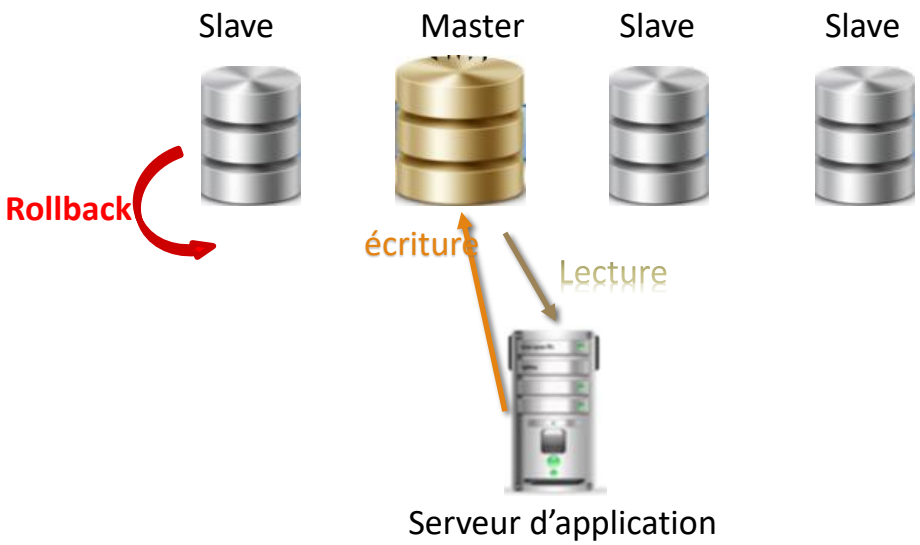


Réplica set : Failover



Réplica set : Failover

Suite a la reprise de l'ancien master, il essaye de synchroniser avec le nouveau master. S'il trouve qu'il a des données en plus il doit effectuer un Rollback pour qu'il soit a niveau avec les autres nœuds.



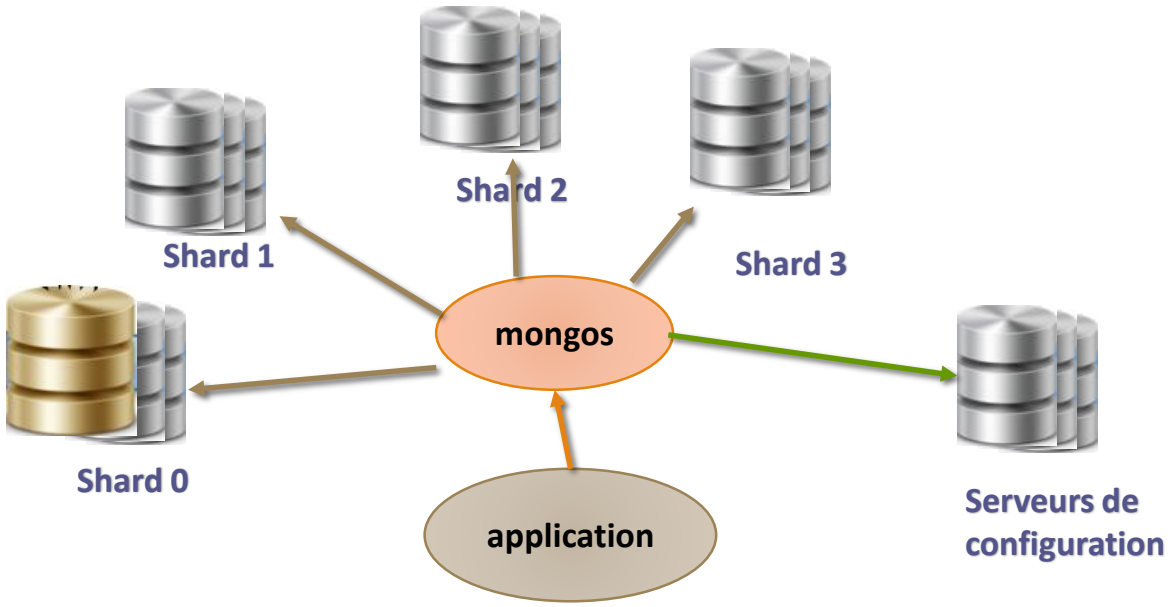
Sharding (segmentation)

Répartition des données en segments et sauvegarde de chaque segment dans un serveur (shard)

La répartition peut être effectuée de façon arbitraire ou bien selon un sharding_key

- Arbitraire : parcourir tous les serveurs pour trouver une information.
- Shard_key: tableau de correspondance entre shard_key et segment de donnée (recherche facile de l'information)
- Exemple : collection orders, order_id peut être considérer comme shard_key

Sharding


Shard 1
Shard 2
Shard 3
Shard 0
mongos
application
Serveurs de configuration

Implication du sharding

Tout document doit contenir un shard key

Le shard key est immuable (une fois défini on ne peut plus le modifier)

Choix de shard key:

- cardinalité suffisante (une valeur qui ne figure que 3 fois ne peut pas être repartie sur 100 shard)
- Incrémentation continue (id ou timestamp): plage de valeur de min au max. pour les valeurs qui dépassent le max elles vont être insérer dans le dernier shard d'où problème de répartition des données.

Index doit commencer par le shard key (exemple : supposant que student_id est le shard key index(student_id,class_id)

Shard key doit être spécifié comme multiple (cas update multi)

Pas de shard key : problème de recherche dans tout les segments

Transfert de données

Import/export

mongoexport et mongoimport sont deux commandes utilisées pour l'import et l'export de données a partir de fichiers JSON ou CSV.

Exemple :

- Export sous format json

mongoexport --db test --collection people –out people.json

- Export sous format csv

mongoexport --db test --collection people --csv --fields nom,salaire,dept --out people.csv

Import/export

Importer à partir d'un fichier json

mongoimport --db base --file people.json

Importer a partir d'un fichier csv

mongoimport --db base --type csv --file people.csv --headerline

Sauvegarde et récupération

Sauvegarde :

Mongodump : permet de sauvegarder une partie ou la totalité de la base dans un dossier dump.

mongodump –help : visualiser les options de mongodump

- --db DBNAME sauvegarde de la base DBNAME
- --collection COLLECTIONNAME sauvegarde la collection COLLECTIONNAME

Exemple :

mongodump --db test --out backup

→ sauvegarder la base **test** dans le répertoire **backup**

Sauvegarde et récupération

Récupération :

mongorestore : récupère les données a partir d'un fichier BSON

--db et--collection permettent de récupérer une base et une collection spécifique.

Exemple:

mongorestore --db test --collection people backup/test/people.bson

→récupérer la collection **people** dans la base **test** a partir du fichier **people.bson**