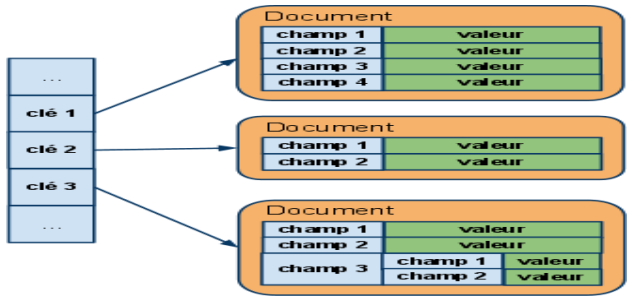
BD ORIENTÉE DOCUMENT (1)

- Elles stockent une collection de « **documents** »
- elles sont basées sur le modèle « clé-valeur » mais la valeur est un document en **format semi-structuré hiérarchique** de type **JSON** ou **XML** (possible aussi de stocker n'importe quel objet, via une sérialisation)
- les **documents** n'ont **pas de schéma**, mais une **structure arborescente** : ils contiennent une liste de champs, un champ a une valeur qui peut être une liste de champs, ...
- elles ont généralement une **interface d'accès HTTP REST** permettant d'effectuer des requêtes (plus complexe que l'interface CRUD des BD clés/valeurs)
- Implémentations les plus connues :
  - CouchDB (fondation Apache)
  - RavenDB (pour plateformes « .NET/Windows » - LINQ)
  - MongoDB, Terrastore, …

BD ORIENTÉE DOCUMENT (2)

- Un **document** est composé de **champs** et **des valeurs associées**
- ces **valeurs** :
  - peuvent être **requêtées**
  - sont soit d'un **type simple** (entier, chaine de caractère, date, ...)
  - soit elles mêmes **composées** de plusieurs couples clé/valeur.
- bien que les documents soient structurés, ces BD sont dites "**schemaless**" : il n'est **pas nécessaire de définir au préalable les champs** utilisés dans un document.
- les documents peuvent être très **hétérogènes** au sein de la BD
- permettent **d'effectuer des requêtes sur le contenu** des documents/objets : pas possible avec les BD clés/valeurs simples
- Elles sont principalement utilisées dans le **développement de CMS** (Content Management System - outils de gestion de contenus).

BD ORIENTÉE DOCUMENT: « EXEMPLE »



BD ORIENTÉE DOCUMENT: « FORCES ET FAIBLESSE »

- **Forces :**
  - modèle de données simple mais puissant (expression de tructures imbriquées)
  - bonne mise à l'échelle (surtout si sharding pris en charge)
  - pas de maintenance de la BD requise pour ajouter/supprimer des «colonnes»
  - forte expressivité de requêtage (requêtes assez complexes sur des structures imbriquées)
- **Faiblesses :**
  - inadaptée pour les données interconnectées
  - modèle de requête limitée à des clés (et indexes)
  - peut alors être lent pour les grandes requêtes (avec MapReduce)

BD ORIENTÉE DOCUMENT: «UTILISATIONS PRINCIPALES »

- Les BD NoSQL de type « Document » principalement utilisées pour
  - •Enregistrement d'événements
  - •Systèmes de gestion de contenu
  - •Web analytique ou analytique temps-réel
  - •Catalogue de produits
  - •Systèmes d'exploitation
  - •…

BD MONGODB

QU'EST-CE QUE MONGODB?

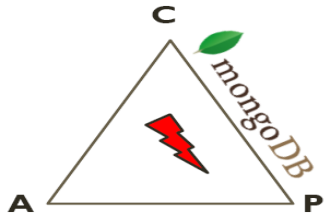
MongoDB, un SGBD "NoSQL", l'un des plus populaires

- fait partie des NoSQL dits "documentaires" (avec CouchDB)
- s'appuie sur un modèle de données semi-structuré (encodage JSON) ;
- pas de schéma (complète flexibilité) ;
- **un langage d'interrogation** original (et spécifique) ;
- pas (ou très peu) **de support transactionnel**.

Construit dès l'origine comme un système **scalable** et **distribué**.

- distribution par partionnement (sharding) ;
- technique adoptée : découpage par intervalles (type BigTable, Google) ;
- tolérance aux pannes par réplication.

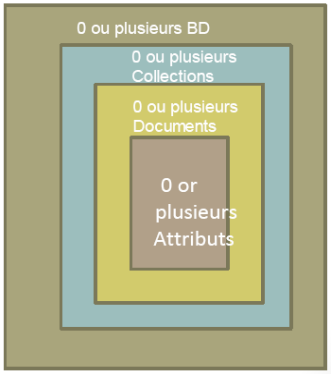
MONGODB :THÉORÈME DE CAP?



MongoDB mettre l'accent sur la consistance et la tolérance aux pannes

MONGODB: HIÉRARCHIE DES OBJETS

- Une instance MongoDB peut avoir zéro ou plusieurs « bases de données »
- Une base de données peut contenir zéro ou plusieurs «collections».
- Une collection peut contenir zéro ou plusieurs «documents».
- Un document peut avoir un ou plusieurs «champs».
- Les index de MongoDB fonctionnent comme leurs homologues SGBDR



COMPARAISON BASES RELATIONNELES ET MONGODB

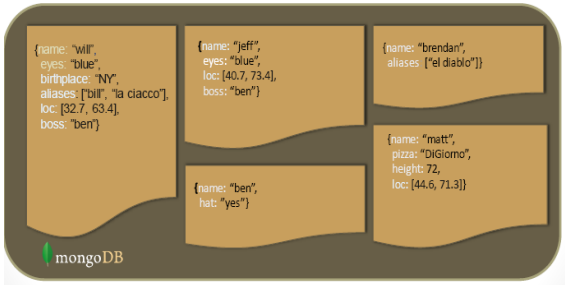
BDR		MongoDB
Base de donnée	➡	Base de données
Table, Vue	➡	Collection
Tuple	➡	Document (JSON, BSON)
Colonne	➡	Champs
Index	➡	Index
Jointure	➡	Document imbriqués
Clé étrangère	➡	Référence
Partition	➡	Shard

MONGODB: « PROCESSUS ET CONFIGURATION»

- Le processus Mongod: instance de la base de données
- Le processus Mongos: Sharding processus
  - •Analogue à un routeur de base de données.
  - •Traite toutes les requêtes.
  - •Décide le nombre de processus mongos qui doivent recevoir la requête.
  - •Mongos rassemble les résultats et les renvoie au client.
- Le processus Mongo: un shell interactif (le client)
  - •Un environnement JavaScript pour l'utiliser avec MongoDB.
- Vous pouvez avoir un processus mongos pour tout le système, peu importe le nombre d'instance « mongod».
- Ou vous pouvez avoir unprocessus mongos local pour chaque client si vous voulez minimiser la latence du réseau.

MONGODB: « PAS DE SCHÉMA »

- MongoDB n'a pas besoin de schéma de données prédéfini
- Chaque document d'une collection peut avoir des données différentes



LE FORMAT JSON

- Les données sont dans des paires nom / valeur
- Une paire nom / valeur consiste en un nom de champ suivi de deux points, suivi d'une valeur:
  - Exemple: nom: "MERAD"
- Les données sont séparées par des virgules:
  - Exemple: nom: "MERAD", prénom: "Manel"
- Les accolades contiennent des objets
  - Exemple: {nom: "MERAD", prénom: "Manel" , pays : " Algérie"}
- Un tableau est stocké entre crochets[ ]
  - Exemple [{nom: "MERAD", prénom: "Manel",, pays : " Algérie"}, {nom: "LAMI", prénom:"Yanis" ,pays : " Algérie"}]

LES OPÉRATIONS CRUD

- **Create**
  - db.collection.insert(<document>)
  - db.collection.save(<document>)
  - db.collection.update(<query>, <update>, { upsert: true })
- **Read**
  - db.collection.find(<query>, <projection>)
  - db.collection.findOne(<query>, <projection>)
- **Update**
  - db.collection.update(<query>, <update>, <options>)
- **Delete**
  - db.collection.remove(<query>, <justOne>)

L'OPÉRATION CREATE

DB.collection spécifie la collection ou seront enregistrés les documents.

- db.nom_collection.insert (<document>)
  - Omettre le champ _**id** pour que MongoDB génère automatiquement la clé
  - Exemple db.parts.insert ({{type: "tournevis", quantité: 15})
  - db.parts.insert ({_ **id**: 10, type: "marteau", quantité: 1})
- db.nom_collection.update (<requête>, <mise à jour>, {upsert: true})
  - Mettra à jour un ou plusieurs enregistrements dans une collection satisfaisant la requête
- db. nom_collection.save (<document>)
  - Met à jour un enregistrement existant ou crée un nouvel enregistrement

L'OPÉRATION READ

- db.collection.find(<query>, <projection>).cursor modified
  - Fournit des fonctionnalités similaires à la commande **SELECT**
  - <query> where condition , <projection> attributs dans le résulat.
  - Exemple: varPartsCursor= db.parts.find({parts: "marteau"}).limit(5)
  - la requête à un curseur pour limiter le nombre de lignes affichés par page
  - on peut modifier la requêtes en ajoutant limits,skips, and sort orders(Tri).
- Pour avoir le première ligne de résultat on utilise:
  - db.collection.findOne(<query>, <projection>)

L'OPÉRATION REMOVE

- db.nom_collection.remove(<query>, <justone>)
  - •Supprimer tous les enregistrements d'une collection qui correspondent au critère
  - •<justone> -spécifie de supprimer seulement I enregistrement correspondant au critère
  - •Exemple: db.parts.remove (type: / ^ h /}) - supprime toutes les pièces commençant par h
  - • Db.parts.remove () -dsupprime tous les documents dans la collections de part

EXEMPLES- CRUD

```
> db.user.insert({_id:51
      nom: "LAIDI",
      prénom : "Ahmed",
      age: 39
})
```

```
> db.user.find ()
{ "_id" : ObjectId("51"),
      "nom" : "LAIDI",
      "prénom" : "Ahmed",
      "age" : 39
}
```

```
> db.user.update(
      {"_id" : ObjectId("51")},
      {
          $set: {
              age: 40,
              Salaire: 7000}
      }
)
```

```
> db.user.remove({
      "nom": /^L/
})
```

INDEX

- On peut utiliser des index pour optimiser les requêtes les plus fréquentes. L'accès aux documents indexés est ainsi bien plus rapide. Un index peut être créé sur n'importe quel attribut de documents.
- Par défaut dans MongoDB un index est positionné sur le champ _id.
- La syntaxe de création d'index est la suivante : db.collection.ensureIndex({champs}, {options})
- Parmi les options possibles : unique : pas de doublon ; sparse : le champ doit être présent ;

  TTL : le document indexé a une durée de vie ; name : le nom de l'index ; background : l'indexation s'effectue en tâche de fond.
- Pour les champs, la valeur à positionner est 1 ou -1 selon l'ordre voulu (ascendant ou descendant) des résultats.
- Exemple: db.parts.ensureIndex({"type": 1}, {"unique": true})

RÉFÉRENCES

- Bernard ESPINASSE - Introduction aux systèmes NoSQL
- Kathleen Durant-Introduction to NoSQL and MongoDB