FORMATION

IT - Digital - Management

m2iformation.fr

MongoDB

Nehemie “Alfred” BALUKIDI
CTO/Software Engineering & Data|ML/AI

Introduction à MongoDB

Qu'est-ce que MongoDB ?

- MongoDB est une base de données NoSQL populaire qui stocke les données dans des documents flexibles, similaires à JSON.
- Elle est conçue pour l'évolutivité, les performances et la facilité d'utilisation.

Principales caractéristiques

Cas d'utilisation

Internet of Things

Applications Web et mobiles

Analyse en temps réel

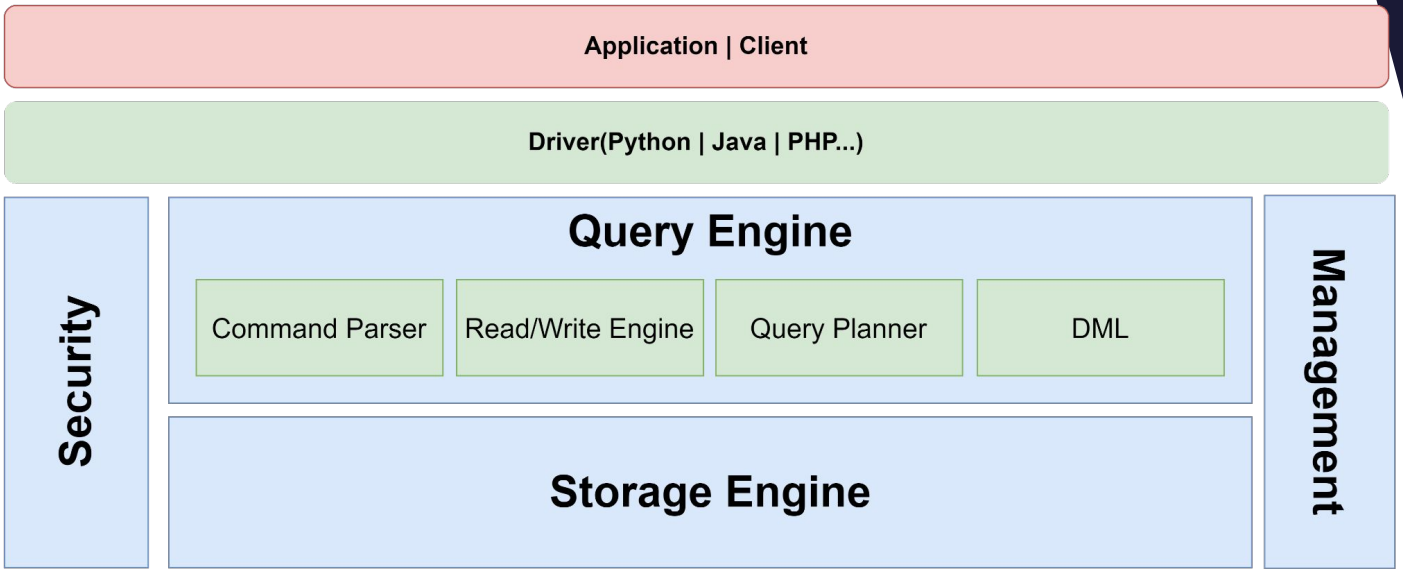
Mise en cache et journalisation

Gestion du catalogue

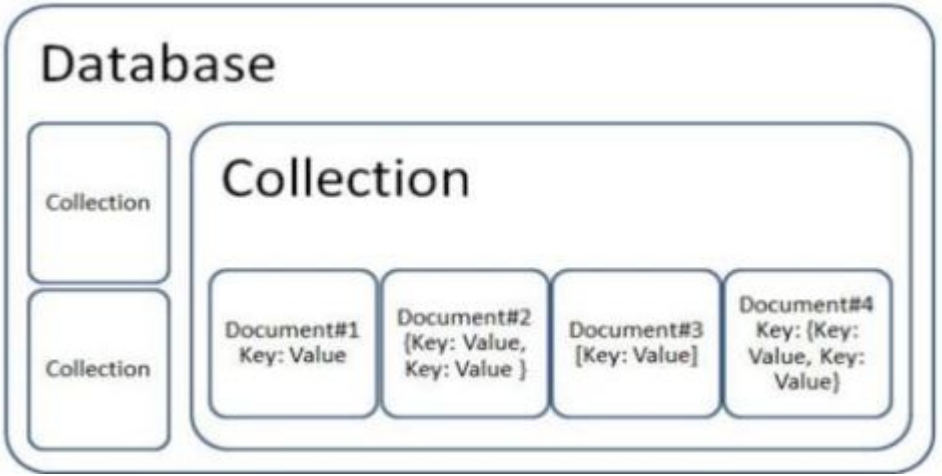
Gestion de contenu

Architecture MongoDB

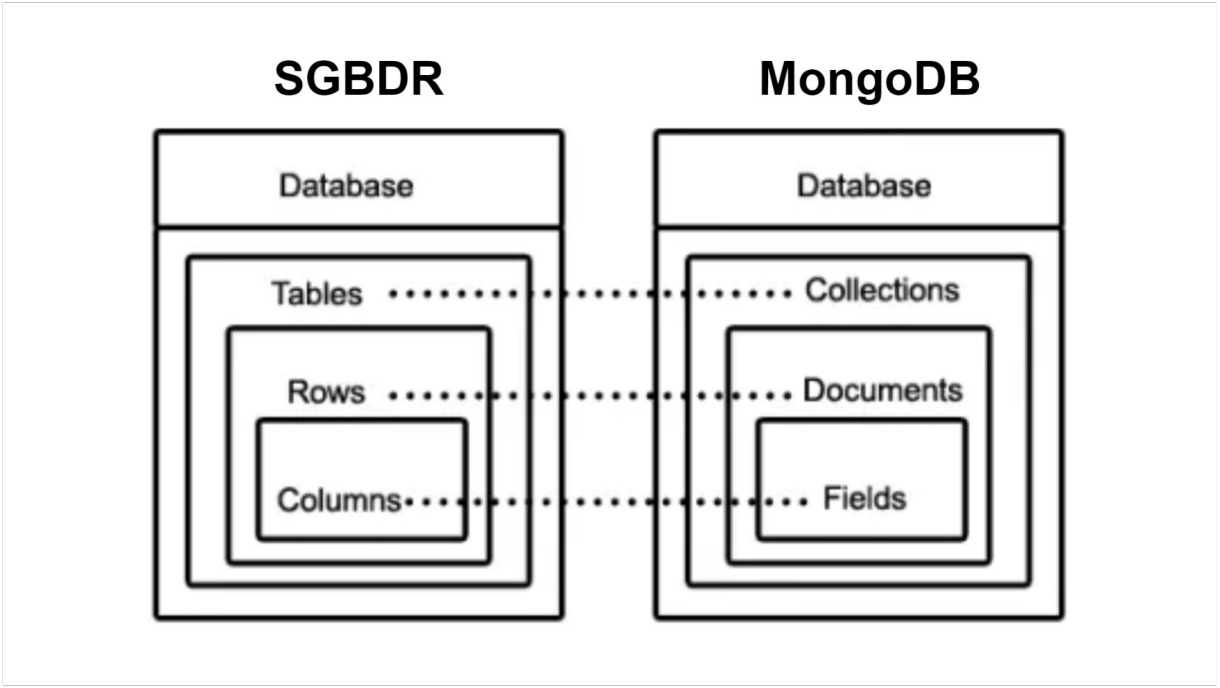
Architecture de l'utilitaire mongod



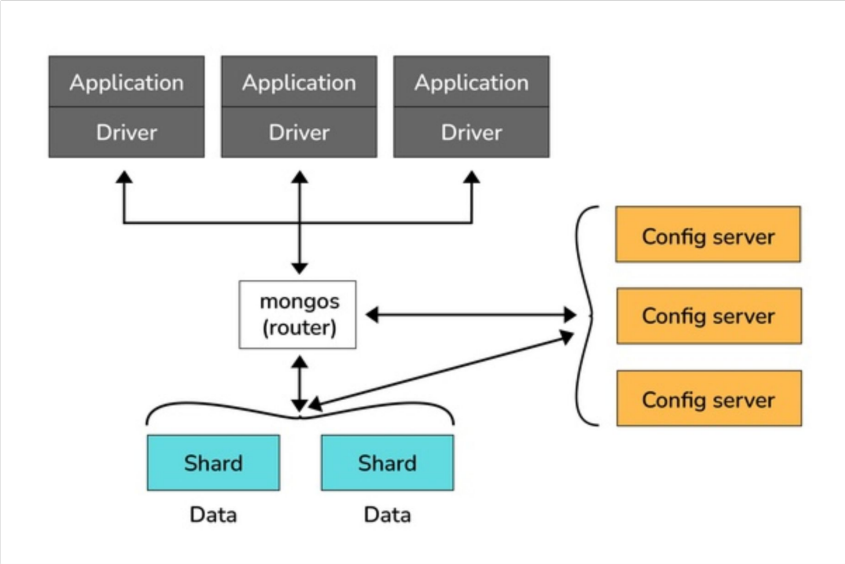
Stockage des données



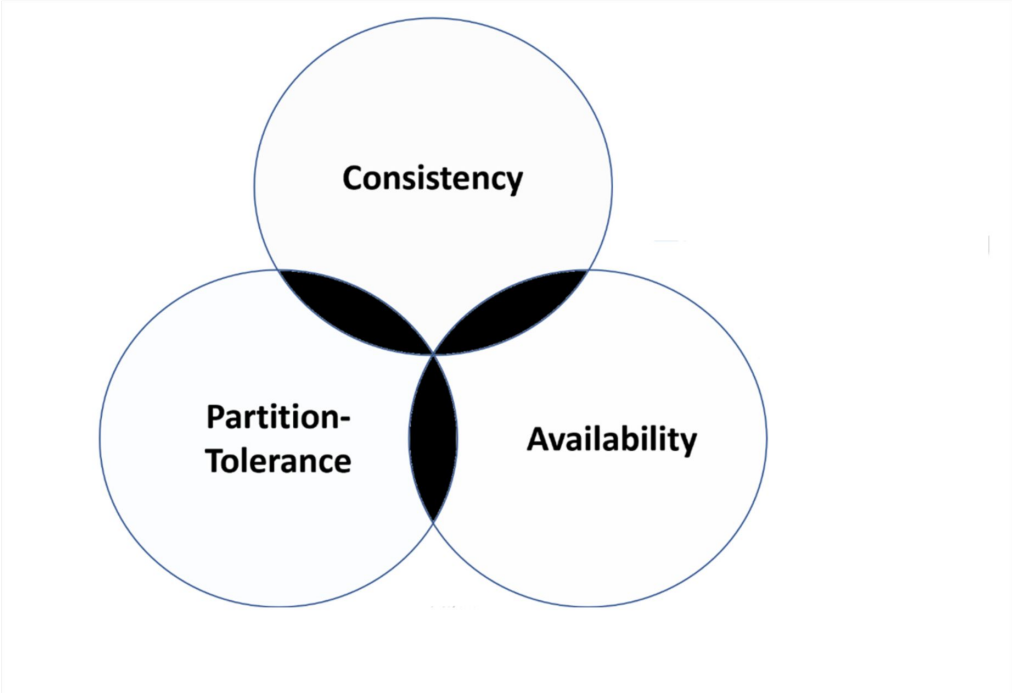
SGBDR vs MongoDB



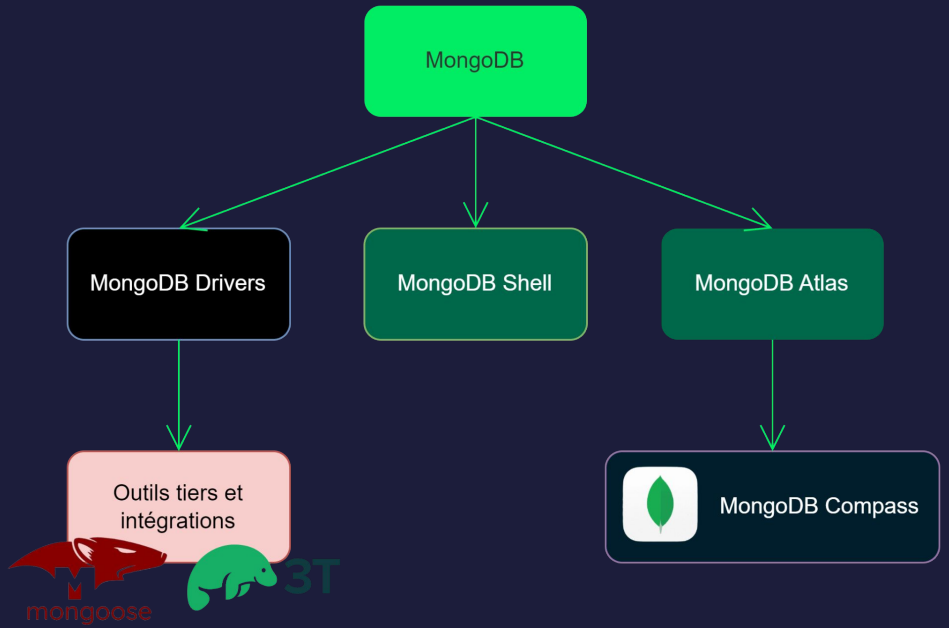
Architecture distribuée MongoDB



Le théorème CAP



Installations & Configuration



Un mot sur BSON

Le JSON Binaire

Qu'est-ce que BSON ?

```
{"hello", "world"} ->
\x16\x00\x00\x00\x00          // taille totale du document
\0x02                         // 0x02 = type String
hello\x00                     // nom du champ
\x86\x00\x00\x00\x00world\x00 // valeur du champ
\x00                          // 0x00 = type EOD ('fin de l'objet')
```

- BSON est une représentation binaire des documents similaires à JSON
- Il étend le modèle JSON pour fournir des types de données supplémentaires et un encodage.

Certains des types inclus dans BSON

- byte => 1 octet (8 bits)
- int32 => 4 octets (entier signé 32 bits, complément à deux)
- int64 => 8 octets (entier signé 64 bits, complément à deux)
- uint64 => 8 octets (entier non signé 64 bits)
- double => 8 octets (virgule flottante binaire 64 bits)
- decimal128 => 16 octets (virgule flottante décimale 128 bits)
- date | 8 octets (entier 64 bits)
- array => Le stockage est basé sur les données
- objectId | 12 octets (valeur d'horodatage 4 octets, valeur aléatoire 5 octets et compteur incrémental 3 octets)

Avantages par rapport à JSON

- BSON est léger et traversable
- Efficace pour le stockage et le traitement des données, et
- Supporte des types de données plus variés que JSON

Retrouvez un tableau comparatif complet sur:
https://gist.github.com/Olfredos6/32154c7fe3a202e076a027895af04f64

Bases des interactions avec MongoDB

Se connecter et interroger les collections d'une instance MongoDB

Connexion à un MongoDB

La connexion à un MongoDB se réalise facilement à l’aide des différents outils. Nous travaillerons avec MongoDB Shell(mongosh).

Veuillez utiliser la ressource à l'adresse:
https://gist.github.com/Olfredos6/7f2c6457d78c9cf93d0ffb568aebd28d#interagir-avec-mongodb

Interroger une collection MongoDB

MongoDB prend en charge les opérations CRUD (Create, Read, Update, Delete) pour manipuler les documents dans une collection.

Veuillez utiliser la ressource à l'adresse:
https://gist.github.com/Olfredos6/dc4c5fd0a65572932101f85f065747b5#file-mongodb_basic_queries-md

Indexation dans MongoDB

Rôle des index dans MongoDB

- Amélioration des performances de requête
- Prise en charge de requêtes efficaces sur de grands volumes de données

Types d'index dans MongoDB

- Index par défaut (_id)
- Index uniques
- Index composés
- Index géospatiaux
- Index de texte
- Index TTL (Time-To-Live)

ObjectId dans MongoDB

- ***ObjectId*** est un identifiant unique utilisé comme clé primaire par défaut (_id) dans les documents MongoDB.

```
_id: ObjectId('661f3899df489409dfa1f242')
```

- Il s'agit d'un type BSON de 12 octets, composé de :
  - 4 octets : Timestamp (secondes depuis l'epoch Unix)
  - 3 octets : Identifiant de la machine
  - 2 octets : ID du processus
  - 3 octets : Compteur

ObjectId dans MongoDB(suite)

- **Unique** : Garantit l'unicité à travers tous les documents d'une collection.
- **Ordonné** : Permet le tri chronologique basé sur le temps d'insertion.
- **Léger** : Minimise l'overhead de stockage par rapport à d'autres identifiants uniques.
- **Décentralisé** : Généré localement sans coordination avec d'autres systèmes.
- **Immuable** : Une fois généré, un ObjectID ne peut pas être modifié.

Création et gestion des index

```
// Créer un index unique sur le champ "email"
db.users.createIndex({ email: 1 }, { unique: true })

// Créer un index composé sur les champs "lastName" et "firstName"
db.users.createIndex({ lastName: 1, firstName: 1 })

// Créer un index géospatial sur le champ "location"
db.restaurants.createIndex({ location: "2dsphere" })

// Créer un index de texte sur les champs "title" et "description"
db.articles.createIndex({ title: "text", description: "text" })
```

- Méthode ***createIndex()***
- Options d'index (unique, background, sparse, etc.)
- Affichage des index existants
- Suppression des index avec ***dropIndex()***

Modélisation

Principes de modélisation des données dans MongoDB et relations

Principes de modélisation dans MongoDB

- Approche orientée document
- Dénormalisation et imbrication des données
- Équilibre entre performances et maintien de la cohérence

Approches de modélisation des relations

Approche par référence

- On stocke une référence d'un document dans un autre document.
- Les documents liés peuvent être récupérés à l'aide de requêtes distinctes.
- Adaptée pour les données liées qui sont fréquemment mises à jour

Approches de modélisation des relations

Approche par imbrication

- On imbrique les documents liés directement dans le document principal
- La récupération de toutes les informations nécessaire se fait en une seule requête
- Pour les données liées qui sont fréquemment accédées ensemble ou pour besoin de performances de lecture optimales

Types de relations: One-To-One

One-To-One(1-à-1) implique un document est lié à un seul autre document.

Ex: un utilisateur peut avoir un seul profil associé

```
// Collection "utilisateurs"
{
  "_id": ObjectId("user1"),
  "nom": "John Doe",
  "email": "john@example.com",
  "profileId": ObjectId("profile1")
}

// Collection "profils"
{
  "_id": ObjectId("profile1"),
  "userId": ObjectId("user1"),
  "bio": "Je suis un développeur passionné."
}
```

Types de relations: One-To-Many

One-To-Many(1-à-N, 1-à-plusieurs) implique un document est lié à plusieurs autres documents. Par exemple, un article de blog peut avoir plusieurs commentaires associés.

```
// Collection "articles"
{
  "_id": ObjectId("article1"),
  "title": "Article 1",
  "content": "Contenu de l'article 1",
  "commentaires": [
    {
      "_id": ObjectId("comment1"),
      "content": "Commentaire 1 pour l'article 1"
    },
    {
      "_id": ObjectId("comment2"),
      "content": "Commentaire 2 pour l'article 1"
    }
  ]
}
```

Types de relations: Many-To-Many

Cette relation implique que plusieurs documents sont liés à plusieurs autres documents. Par exemple, des étudiants peuvent suivre plusieurs cours, et chaque cours peut avoir plusieurs étudiants inscrits.

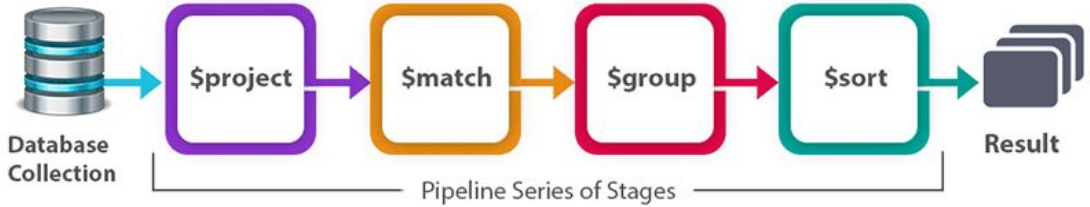
```
// Collection "etudiants"
{
  "_id": ObjectId("etudiant1"),
  "nom": "Alice",
  "coursIds": [
    ObjectId("cours1"),
    ObjectId("cours2")
  ]
}

// Collection "cours"
{
  "_id": ObjectId("cours1"),
  "nom": "Mathématiques",
  "etudiantIds": [
    ObjectId("etudiant1"),
    ObjectId("etudiant2")
  ]
}
```

Agrégations et pipelines

Traiter et de transformer les documents d'une collection

L’agrégation dans MongoDB

Un puissant outil d'analyse de données qui permet de traiter et de transformer les documents d'une collection



Une pipeline d'agrégation est une série d'étapes où les documents passent séquentiellement pour produire un résultat agrégé.

Étapes courantes du pipeline d'agrégation

- ***$match*** : Filtrage des documents
- ***$group*** : Regroupement des documents
- ***$project*** : Projection des champs
- ***$sort*** : Tri des documents
- ***$limit*** et ***$skip*** : Limitation et saut de documents
- ***$lookup***: jointure du type LEFT OUTER JOIN à une collection

Opérateurs d'agrégation

- ***$sum, $avg, $max, $min*** : Calculs mathématiques
- ***$push, $addToSet*** : Création de tableaux
- ***$first, $last*** : Sélection de valeurs
- ***$unwind*** : Déroulement de tableaux