Bases de Données NoSQL

Prénom: Abdelillah

Nom: SELHI

N°Étudiant: 12206456

Groupe: INFOA3

Introduction à MongoDB

Qu'est-ce que MongoDB ?

MongoDB est une base de données **NoSQL** (Not Only SQL) de type **document-orientée**. Contrairement aux bases de données relationnelles traditionnelles (SQL), MongoDB stocke les données sous forme de documents JSON, offrant une flexibilité et une scalabilité.

Caractéristiques Principales

1. Architecture Document-Orientée

- Les données sont stockées en documents JSON/BSON (Binary JSON)

2. Scalabilité Horizontale

- **Sharding** : distribution des données sur plusieurs serveurs
- Croissance sans limite du volume de données

3. Performance

- Lecture rapide depuis la mémoire
- Indexation pour accélérer les requêtes

4. Haute Disponibilité

- Réplication des données sur plusieurs serveurs
- Récupération automatique en cas de défaillance

Utilité de MongoDB

Exemples d’utilisation

- **Gestion de Catalogs Dynamiques** (e-commerce)
    - Chaque produit peut avoir des attributs différents
    - Modifications rapides sans migration de schéma
- **Réseaux Sociaux**
    - Stockage d'énormes volumes de contenus utilisateurs
    - Profiles, messages, interactions

Concepts Fondamentaux

Structure des Données

Database (Base de données)
├── Collection 1 (Type de document)
│ ├── Document 1 { _id, field1, field2, ... }
│ ├── Document 2 { _id, field1, field2, ... }
│ └── Document N { _id, field1, field2, ... }
│
└── Collection N (Type de document)
└── Documents...

Exemple de document MongoDB :

```
{
"_id": ObjectId("507f1f77bcf86cd799439011"),
"title": "Inception",
"year": 2010,
"genres": ["Sci-Fi", "Thriller"],
"directors": ["Christopher Nolan"],
"imdb": {
"rating": 8.8,
"votes": 2000000
},
"countries": ["USA", "UK"],
"rated": "PG-13"
}
```

Commandes MongoDB

1. Commandes de Base - find()

db.collection.find(filter, projection)

- `filter` : Critères de filtrage (optionnel, objet vide = tous les documents)
- `projection` : Champs à afficher/masquer (optionnel)

1.1 - Trouver Tous les Documents

```
db.movies.find()
```

1.2 - Filtrer par Égalité: Trouver les films dont le genre contient "Comedy"

```
db.movies.find({ genres: "Comedy" })
```

1.3 - Filtrer par comparaison: Films sortis depuis 2015

db.movies.find({ year: { $gte: 2015 } })

- **Opérateurs** :

  $gte : Plus grand ou égal à (≥)

  $gt : Strictement plus grand (>)

  $lte : Plus petit ou égal à (≤)

  $lt : Strictement plus petit (<)

  $eq : Égal à

  $ne : Différent de

1.4 - Filtrer par Double comparaison: Films entre 2000 et 2005 (inclus)

db.movies.find({ year: { $gte: 2000, $lte: 2005 } })

1.5 - Filtrer avec Opérateur $all: Films qui ont TOUS les genres spécifiés

db.movies.find({ genres: { $all: ["Drama", "Romance"] } })

1.6 - Filtrer par Existence de Champ: Films n'ayant pas le champ rated

db.movies.find({ rated: { $exists: false } })

1.7 - Afficher Uniquement Certains Champs

db.movies.find({ year: { $gte: 2000, $lte: 2005 } }, { title: 1, year: 1 })

1.8 - Afficher Tous Sauf Certains Champs: Afficher tous les champs sauf poster

db.movies.find({}, { poster: 0 })

1.9 - Limiter le Nombre de Résultats

db.movies.find({ year: { $gte: 2015 } }).limit(5)

1.10 - Sauter des Documents: Sauter les 10 premiers documents, retourner les 5 suivants

db.movies.find({}).skip(10).limit(5)

1.11 - Trier les Résultats: Trier par année décroissante

db.movies.find({}).sort({ year: -1 }) => 1 pour l'ordre croissant

1.12 - Affichage Lisible db.movies.find({}).pretty()

Recherche par Expressions Régulières

1.13 - Correspondance de Chaîne au Début: Films dont le titre commence par "Star"

db.movies.find({ title: /^Star/ }, { title: 1 })

1.14 - Correspondance Insensible à la Casse: Recherche "star" peu importe la casse

db.movies.find({ title: /star/i }, { title: 1 })

Requêtes Avancées

1.15 - Vérifier la Longueur d'un Tableau: Films avec plus de 2 genres

db.movies.find({ $where: "this.genres.length > 2" }, { title: 1, genres: 1 })

2. Commandes d'Agrégation - aggregate()

db.collection.aggregate([
{ $stage1: {...} },
{ $stage2: {...} },
...
])

Étapes d'Agrégation Principales

2.1 - $match : Filtrer les Documents: Filtrer au début du pipeline (comme WHERE en SQL)

db.movies.aggregate([
{ $match: { year: { $gte: 2000 } } }
])

2.2 - $group : Grouper et Calculer: Grouper par année et compter les films

db.movies.aggregate([
{ year", total: { $sum: 1 } } },
{ $sort: { _id: 1 } }
])

Opérateurs d'accumulation :

$sum : Somme

$avg : Moyenne

$min : Minimum

$max : Maximum

2.3 - $unwind : Créer un document par élément du tableau

```
db.movies.aggregate([
{ $unwind: "genres" }
])
```

exemple avant : { _id: 1, genres: ["Drama", "Romance"] }

exemple après :
{ _id: 1, genres: "Drama" }
{ _id: 1, genres: "Romance" }

2.4 - Calculs Statistiques par Genre: Note IMDb moyenne par genre, trié décroissant

```
db.movies.aggregate([
{ $unwind: "genres" },
{ genres", moyenne: { $avg: "imdb.rating" } } },
{ $sort: { moyenne: -1 } }
])
```

2.5 - $project : Réorganiser et Calculer les Champs: Créer de nouveaux champs

```
db.movies.aggregate([
{ $project: {
title: 1,
decade: { $subtract: ["year", { $mod: ["year", 10] }] },
"imdb.rating": 1
}}
])
```

- **Opérateurs** :

  $subtract : Soustraction

  $mod : Modulo

  $add, $multiply, $divide

  $toUpper, $toLower

2.6 - $sort : Trier dans le Pipeline: Trier les films par note IMDb décroissante

```
db.movies.aggregate([
{ $sort: { "imdb.rating": -1 } },
{ $project: { title: 1, "imdb.rating": 1 } }
])
```

2.7 - $limit : Limiter les Résultats

```
db.movies.aggregate([
{ $unwind: "directors" },
{ directors", total: { $sum: 1 } } },
{ $sort: { total: -1 } },
{ $limit: 5 }
])
```

2.8 - Exemple Complexe : Films Mieux Notés par Décennie

```
db.movies.aggregate([
{ $match: { "imdb.rating": { $exists: true } } },
{ $project: {
title: 1,
decade: { $subtract: ["year", { $mod: ["year", 10] }] },
"imdb.rating": 1
}},
{ decade", maxRating: { $max: "imdb.rating" } } },
{ $sort: { _id: 1 } }
])
```

3. Commandes de Mise à Jour - update()

updateOne() - Modifier UN Document

3.1 - Ajouter/Modifier un Champ : $set

```
db.movies.updateOne(
{ title: "Jaws" },
{ $set: { etat: "culte" } }
)
```

3.2 - Incrémenter une Valeur : $inc

```
db.movies.updateOne(
{ title: "Inception" },
{ $inc: { "imdb.votes": 100 } }
)
```

3.3 - Remplacer un Tableau : $set

```
db.movies.updateOne(
{ title: "Titanic" },
{ $set: { directors: ["James Cameron"] } }
)
```

updateMany() - Modifier PLUSIEURS Documents

3.4 - Supprimer un Champ : $unset

```
db.movies.updateMany(
{},
{ $unset: { poster: "" } }
)
```

3.5 - Ajouter à champ

```
db.movies.updateMany(
{},
{ $set: { viewed: false } }
)
```

4. Commandes d'Indexation

4.1 - Créer un Index Simple

```
db.movies.createIndex({ year: 1 })
```

Utilité dans cet exemple: Accélérer les requêtes filtrées sur `year`

Impact

```
db.movies.find({ year: 1995 }).explain("executionStats")
```

Sans index : parcourt TOUS les documents
Résultat : "totalDocsExamined": 45000 documents

Avec index : parcourt uniquement les documents correspondants
Résultat : "totalDocsExamined": 1200 documents

4.2 - Créer un Index Composé (Important ! Le champ le plus sélectif en premier)

```
db.movies.createIndex({ year: 1, "imdb.rating": -1 })
```

- **Utilité dans cet exemple**: Optimiser les requêtes filtrées par année ET notées par rating

4.3 - Lister Tous les Index

db.movies.getIndexes()

4.4 - Analyser une Requête : explain()

db.movies.find({ year: 1995 }).explain("executionStats")

Utilité : Analyser la performance d'une requête

- **Champs importants** :
  `totalDocsExamined` : Documents examinés
  `executionTimeMillis` : Temps d'exécution

4.5 - Supprimer un Index

db.movies.dropIndex({ year: 1 })

5. Documentation: www.mongodb.com