Rapport TP

Prise en main de MongoDB - Partie 1

Addou Aicha Amira

Bombay Marc

3ème Année Ingénieur - INFO - Apprentissage

Environnement utilisé

Dans ce travail pratique dédié à la découverte et à la manipulation de MongoDB, nous avons mis en place un environnement basé sur **Docker** afin d'exécuter le serveur MongoDB dans un conteneur isolé.

Outil d'exploration des données

Pour interagir avec la base de données, nous avons utilisé **MongoDB Compass**, l'interface graphique fournie par MongoDB. Cet outil nous a permis d'afficher les collections, d'inspecter les documents et d'exécuter des requêtes facilement à travers une interface simplifiée.

Chargement des données

Les données utilisées proviennent du jeu *sample_mflix*, fourni au format BSON. Afin de l'importer correctement dans le serveur MongoDB, il a fallu utiliser `mongorestore` au lieu `mongoimport` qui traite les fichiers JSON.

PARTIE 1 – Filtrer et projeter les données

0. Structure d'une requête

```
db.<collection>.find(<filtre>, <projection>)
```

- `filtre` : critère pour sélectionner les documents
- `projection` : champs à afficher (1 = afficher, 0 = masquer)

1. Afficher les 5 films sortis depuis 2015

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

- affiche les films dont l'année est supérieure ou égale à 2015, limités aux 5 premiers.
- ici on utilise l'opérateur de comparaison $gte (GREATER OR EQUAL)

2. Trouver tous les films dont le genre est "Comedy"

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

- Recherche les films contenant "Comedy" dans le tableau genres.
- Ici on a un filtre simple sur l'attribut "genres".

3. Afficher les films sortis entre 2000 et 2005

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

- Filtre les films dont l'année est comprise entre 2000 et 2005 et affiche seulement le titre et l'année.
- On a utilisé les opérateur $gte et $lte pour encadrer la sortie du film
- On veut seulement afficher le "title" et "year" d'où {title: 1, year: 1} les autres attributs sont à 0 par défaut

4. Afficher les films de genres “Drama” ET “Romance”

```
db.movies.find({genres: {$all: ["Drama", "Romance"]}}, {title: 1, genres: 1})
```

- Recherche les films contenant obligatoirement les deux genres “ Drama” et “Romance” dans leur tableau.
- Voici un exemple de deux objets obtenus par l'exécution de la commande :

5. Afficher les films sans champ rated

```
db.movies.find({rated: {$exists: false}}, {title: 1})
```

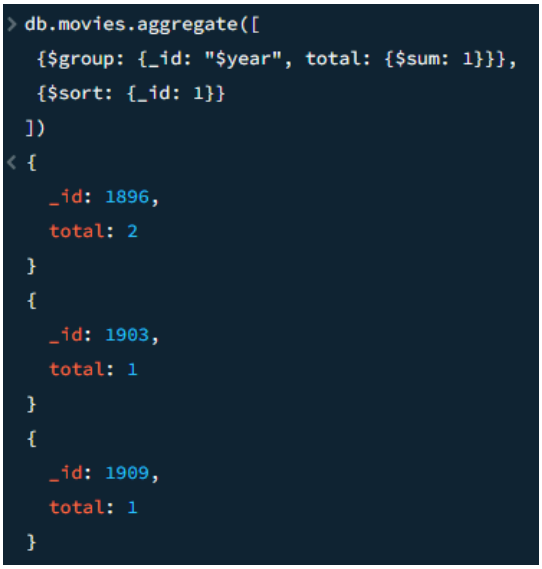
- Sélectionne les documents où le champ rated n’existe pas.
- L’attribut **`$exits`** permet d’afficher tous les film qui ne sont pas noté “rated”
- On évalue si le champs rated existe ou pas pour chaque film

PARTIE 2 – Agrégation

6. Afficher le nombre de films par année

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

- on groupe les films par année “year”, compte le total par année, puis trie de la plus ancienne à la plus récente.
- on utilise l’opérateur **`$sum :1`** pour incrémenter le total (+1) à chaque document appartenant à cette année. Si on avait mis **`$sum:2`** ça aurait incrémenté de +2
- on utilise l’opérateur **`$sor`**t sur l’id **`{$sort: {_id: 1}}`** ici le “1” signifie ordre croissant (-1 si on veut l’ordre décroissant)
- Voici une partie du résultat de la commande :



7. Afficher la moyenne des notes IMDb par genre

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

- **$unwind** déplie un tableau : s'il y a plusieurs genres dans un film, MongoDB crée autant de documents que d'éléments dans le tableau.
- **_id: "$genres"** : pour préciser que l'on agrège sur l'attribut genres.
- **moyenne: { $avg: "$imdb.rating" }** : MongoDB calcule la moyenne des notes IMDb pour tous les films du genre en question.
- **{ $sort: { moyenne: -1 } } :** Trie les genres par note décroissante

8. Afficher le nombre de films par pays

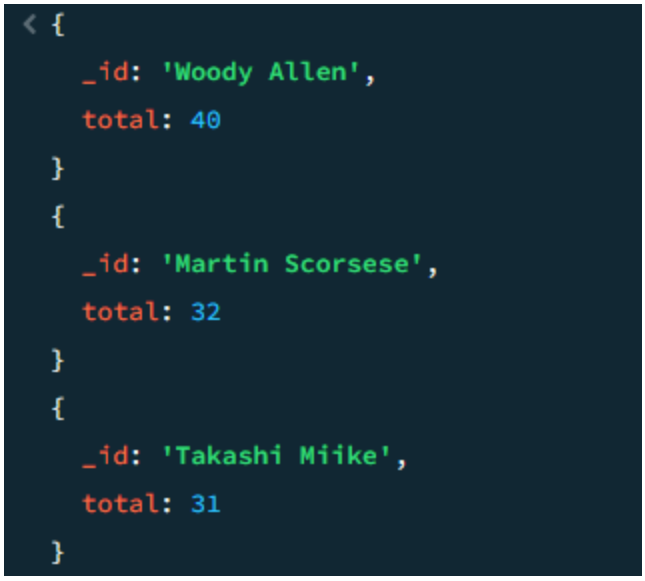
```
db.movies.aggregate([
 {$unwind: "$countries"},
 {$group: {_id: "$countries", total: {$sum: 1}}},
 {$sort: {total: -1}}
])
```

Se base sur la même structure que le requête précédente, on utilise ici l'opérateur **$sum** pour l'agrégation à la place de **$avg**

9. Afficher les top 5 réalisateurs

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

On compte le nombre de fois où le réalisateur apparaît, puis on renvoie les 5 premiers avec `{$limit : 5}`



10. Afficher les films triés par note IMDb

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

Trie les films par note IMDb décroissante et affiche uniquement le titre et la note.

PARTIE 3 – Mises à jour

Eléments de syntaxe

La syntaxe générale des mises à jour dans MongoDB est la suivante :

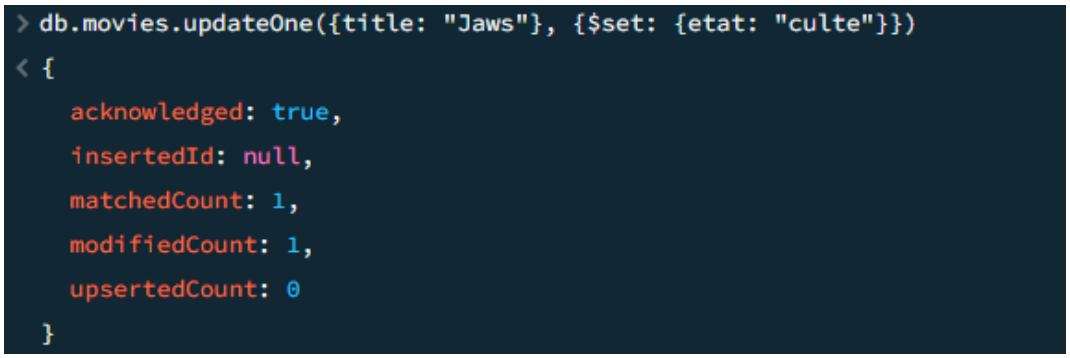
```
.updateOne(<filtre>, <mises a jour>)
.updateMany(<filtre>, <mises a jour>)
```

où:
`<mises a jour> : {<$op> : <quoi>}`

- L'instruction **updateOne()** ne met à jour qu'un seul document, même si le filtre correspond à plusieurs documents.
- À l'inverse, **updateMany()** applique la mise à jour à tous les documents correspondant au filtre.

11. Ajouter un champ etat

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



- Cette commande ajoute, ou modifie, le champ etat dans le document correspondant au film "Jaws".
- L'opérateur $set permet d'écrire la valeur indiquée dans le document :
- A noter que s'il s'agissait d'un nouveau champ, il est créé, sinon la valeur et mise à jour comme ça l'est dans notre cas.

12. Incrémenter les votes IMDb

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

```
> db.movies.updateOne({title: "Inception"}, {$inc: {"imdb.votes": 100}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

- Ici on augmente la valeur imdb.votes de 100.
- Cela se fait par **`{$inc: {<nom>: <de combien>}}`**

13. Supprimer le champ poster

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

```
> db.movies.updateMany({}, {$unset: {poster: ""}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 21349,
    modifiedCount: 18044,
    upsertedCount: 0
  }
```

- Ici on supprime le champ **poster** dans tous les films.
- Le filtre {} correspond à tous les documents.
- L'opérateur $unset permet de retirer un champ d'un document. La valeur associée (ici "") n'a pas d'importance : c'est la présence de **`$unset`** et du nom du champ qui déclenche sa suppression.

14. Modifier le réalisateur

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

```
> db.movies.updateOne({title: "Titanic"}, {$set: {directors: ["James Cameron"]}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

- Cette commande remplace entièrement la liste des réalisateurs du film **Titanic**.
- On note que `$set` marche bien avec les listes.
- On fournit une nouvelle liste qui remplace l'ancienne.

PARTIE 4 – Requêtes complexes

15. Afficher les films les mieux notés par décennie

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

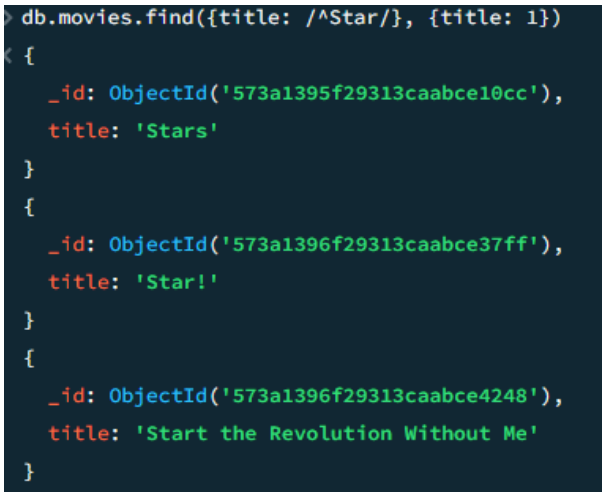
Calcule la décennie d'un film, regroupe par décennie et affiche la meilleure note IMDb trouvée dans chaque groupe.

N'avait pas fonctionné.

16. Afficher les films dont le titre commence par "Star"

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

Voici une partie du résultat de la requête :

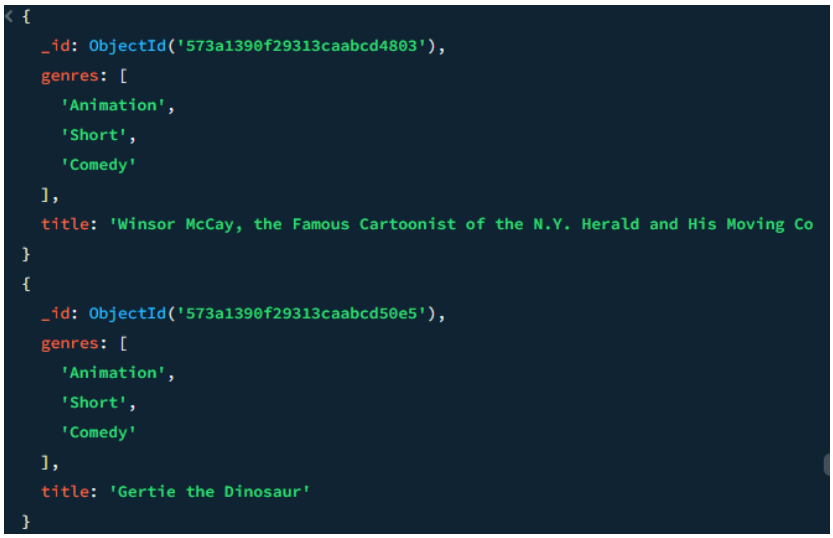


- Utilise une expression régulière pour sélectionner les titres débutant par "Star".
- Le symbole ^ indique le début de la chaîne, ce qui permet d'écarter les titres contenant "Star" ailleurs que dans la première position.

17. Afficher les films avec plus de 2 genres

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

- Filtre les films contenant strictement plus de deux genres
- L'expression passée à $where est du JavaScript exécuté par MongoDB
- L'opérateur $where d'évaluer une condition personnalisée qui n'est pas possible avec les opérateurs classiques de MongoDB (utilisation déconseillé en prod car lenteur)



18. Afficher les films de Christopher Nolan

```
db.movies.find({directors: "Christopher Nolan"}, {title: 1, year: 1, "imdb.rating": 1})
```

- Sélectionner les films dont le tableau directors contient Christopher Nolan.
- Requête classique avec filtre et sélection des champs visibles

```
{
  _id: ObjectId('573a13c4f29313caabd6cc80'),
  imdb: {
    rating: 8.5
  },
  year: 2012,
  title: 'The Dark Knight Rises'
}
{
  _id: ObjectId('573a13c5f29313caabd6ee61'),
  imdb: {
    rating: 8.8
  },
  year: 2010,
  title: 'Inception'
}
```

PARTIE 5 – Indexation

19. Créer un index sur year

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

Créer un index croissant sur le champ year pour optimiser les recherches et tris sur ce champ.

```
> db.movies.createIndex({year: 1})
< year_1
```

L'objectif est d'accélérer les requêtes qui :

- filtrent sur l'année (ex : `{year: 1995}`)
- trient par année
- utilisent des comparaisons (`$gt, $lte, etc.`)

On pourrait également utiliser -1 pour un index décroissant.

20. Vérifier les index existants

`db.movies.getIndexes()`

```
> db.movies.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    {
      v: 2,
      key: { _fts: 'text', _ftsx: 1 },
      name: 'cast_text_fullplot_text_genres_text_title_text',
      weights: { cast: 1, fullplot: 1, genres: 1, title: 1 },
      default_language: 'english',
      language_override: 'language',
      textIndexVersion: 3
    },
    { v: 2, key: { year: 1 }, name: 'year_1' }
  ]
```

On affiche tous les index présents sur la collection movies.

On y retrouve :

- l'index `_id` créé automatiquement
- les index ajoutés par l'utilisateur
- leur structure
- lleur type (simple, composé, texte, etc.)

21. Comparer deux requêtes avec explain()

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

Affiche le plan d'exécution détaillé, notamment le nombre de documents parcourus pour la requête.

- `executionStages`: structure du plan d'exécution
- `totalDocsExamined`: nombre de documents analysés
- `nReturned`: nombre de résultats
- le type d'accès utilisé (index scan ou collection scan)

22. Supprimer l'index sur year

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

Supprime l'index créé précédemment sur le champ year.

```
> db.movies.dropIndex({year: 1})
< { nIndexesWas: 3, ok: 1 }
```

On pourrait faire cela quand l'index :

- n'est plus utile
- coûte trop de place mémoire
- ralentit les opérations d'écriture (chaque index doit être mis à jour à chaque insertion)

23. Créer un index composé sur year et imdb.rating

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

- Ici, MongoDB crée un **index multi-colonnes** (appelé *compound index*).
- ordre croissant pour **year** et decroissant pour **`imdb.rating`**
- Il est utile pour optimiser des requêtes qui utilisent **les deux champs** dans cet ordre. Par exemple :

  ```
  db.movies.find({ year: 2005 }).sort({ "imdb.rating": -1 })
  ```