Systèmes de Gestion de Documents
Projet

BARRANCO Rémy
DUSOLEIL Yoan

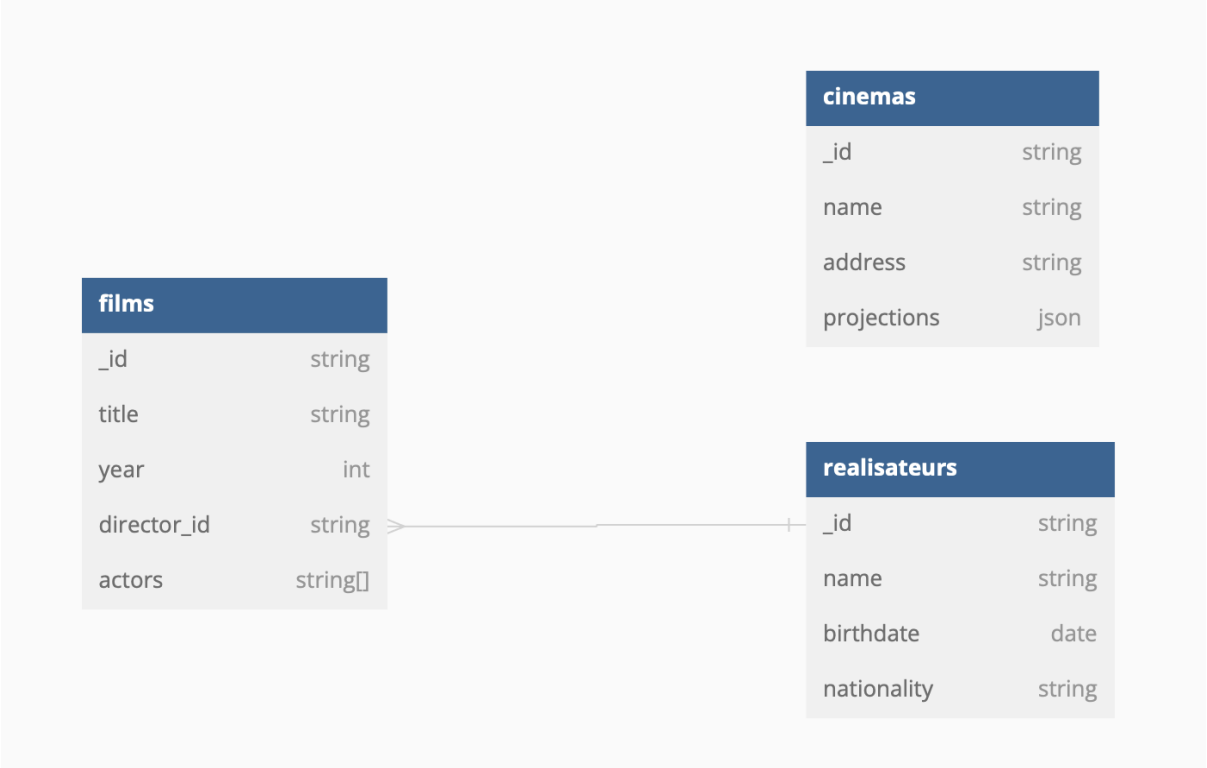
Introduction

Ce projet se concentre sur l'utilisation de la base de données NoSQL MongoDB pour modéliser, insérer et manipuler des données. L'objectif est d'acquérir une expertise dans cette technologie, qui offre une flexibilité et une capacité à gérer de grands ensembles de données.

Création des collections

La structure de notre base de données est définie à travers trois collections principales : films, réalisateurs, cinémas. Chaque collection est conçue pour stocker des informations cruciales et interdépendantes, permettant ainsi une intégration et une manipulation efficace des données. Par exemple, la collection films inclut non seulement des détails tels que le titre et l'année de sortie mais aussi des références aux réalisateurs et une liste des acteurs, facilitant ainsi les requêtes complexes et les analyses.
Les schémas des collections ont été rigoureusement définis en utilisant **$jsonSchema** pour s'assurer que les données insérées respectent les contraintes imposées, telles que les types de données et les champs obligatoires. Cela aide à maintenir l'intégrité des données tout au long du cycle de vie de l'application.

Le choix d'éviter un schéma polymorphique strict est délibéré pour assurer la clarté des structures de données et la facilité de maintenance. Chaque collection est conçue avec des schémas bien définis pour éviter les ambiguïtés et les complexités de validation des données polymorphes.



Insertion des données

1. **Insertion de réalisateurs :**

- On insère une liste de réalisateurs célèbres avec des champs pour leur nom, date de naissance (utilisant `new Date()` pour créer des objets date), et nationalité.
- Exemple : Christopher Nolan est inséré avec la date "1970-07-30" et la nationalité "British".

2. **Insertion de films (`films`) :**

- On crée une liste de films en définissant leur titre, année, réalisateur et acteurs.
- Le champ réalisateur pour chaque film est déterminé en trouvant l'ID correspondant dans la collection `realisateurs`. Pour cela, on utilise `findOne` qui récupère l'ID du réalisateur spécifié.
- Par exemple, pour le film "Inception", on attribue Christopher Nolan comme réalisateur en cherchant son ID dans la collection `realisateurs`.

3. **Insertion de cinémas (`cinemas`) :**

- On insère une liste de cinémas avec leur nom et adresse.
- Un champ `projections` est initialisé comme un tableau vide, destiné à stocker des informations sur les séances programmées.

4. **Ajout de projections aux cinémas :**

- On utilise une boucle pour générer des projections sur 30 jours pour des films aléatoires dans des cinémas aléatoires à des heures précises (12h, 15h, 18h, 21h).
- On récupère aléatoirement un film et un cinéma pour chaque projection en utilisant `$sample` pour sélectionner au hasard dans les collections `films` et `cinemas`.
- On crée une date pour la projection, calculée pour être sur les prochains 30 jours à l'une des quatre heures données.
- Chaque projection spécifie l'ID du film, la date et heure de projection, et un champ `occupancy` qui indique le nombre total de places et le nombre de places réservées (généré aléatoirement).

Requêtage Mongo

Récupération de tous les films d'un réalisateur :

La requête suivante nous permet de récupérer tous les films de Christopher Nolan. En effet, on peut supposer qu'un utilisateur souhaite rechercher les films d'un réalisateur spécifique.

```
// Récupérer les films de Christopher Nolan
db.films.aggregate([
    {
        $lookup: {
            from: "realisateurs",            // joindre avec la collection "realisateurs"
            localField: "director",          // utiliser le champ "director" de "films"
            foreignField: "_id",             // correspondre avec le champ "_id" de "realisateurs"
            as: "director_details"           // stocker le résultat sous le nom "director_details"
        }
    },
    { $unwind: "$director_details" },         // décomposer l'array "director_details"
    { $match: { "director_details.name": "Christopher Nolan" } } // Filtrer pour obtenir uniquement les films de Nolan
]);
```

(Voir résultats en Annexe 1)

Récupération de tous les films ayant un acteur spécifique :

Celle-ci nous donne tous les films ou Leonardo DiCaprio incarne un rôle. Pareil que la précédente mais cette fois pour acteur.

```
// Trouver tous les films avec Leonardo DiCaprio
db.films.find({ "actors": "Leonardo DiCaprio" });
```

Compter le nombre de films d'un réalisateur :

Cette requête nous permet de compter le nombre de films par réalisateur. Requête qui pourrait être utile par exemple sur la fiche d'un réalisateur.

```
// Compter le nombre de films par réalisateur
db.films.aggregate([
    {
        $group: {
            _id: "$director",                 // grouper par réalisateur
            count: { $sum: 1 }                // compter le nombre de films pour chaque réal
        }
    },
    {
        $lookup: {
            from: "realisateurs",            // joindre avec "realisateurs"
            localField: "_id",               // "_id" est l'ID du realisateur
            foreignField: "_id",             // Ccorrespondance sur "_id" dans "realisateurs"
            as: "director_details"           // tésultat dans "director_details"
        }
    },
    { $unwind: "$director_details" },         // décomposer le tableau "director_details"
    { $project: { _id: 0, director_name: "$director_details.name", count: 1 } } // afficher uniquement les champs
nécessaires
]);
```

(Voir résultats en Annexe 2)

Année de sortie moyenne des films d'un réalisateur :

Calculer l'année de sortie moyenne des films par réalisateur. Cette requête pourrait donner une indication sur la période d'activité d'un réalisateur.

```
// Calculer l'année moyenne des films par réalisateur
db.films.aggregate([
    {
        $group: {
            _id: "$director",                // grouper par réalisateur
            averageYear: { $avg: "$year" }    // calculer la moyenne des années de sortie des films
        }
    },
    {
        $lookup: {
            from: "realisateurs",
            localField: "_id",
            foreignField: "_id",
            as: "director_details"
        }
    },
    { $unwind: "$director_details" },
    { $project: { _id: 0, director_name: "$director_details.name", averageYear: 1 } }
]);
```

Ajout d'une critique à un film :

Dans cette requête nous ajoutons des revues et des notes à un film dans la collection *films*. Cette requête démontre parfaitement la flexibilité permise par MongoDB et les SGD car nous ajoutons directement un nouveau champ à notre collection puisque le champ *reviews* n'existait pas. De plus, le jsonSchema permet cette modification car il agit uniquement sur les champs renseignés lors de sa définition. Il peut cependant être mis à jour pour inclure de nouveaux champs (comme les avis par exemple).

```
// Ajouter des critiques au film Inception
db.films.updateOne(
    { "title": "Inception" },
    { $push: {
        "reviews": {
            $each: [
                { "rating": 5, "comment": "Absolutely fantastic!" },
                // ...
            ]
        }
    }}
);
```

Ajout d'un acteur au film Inception :

```
// ajout d'un acteur a inception
db.films.updateOne(
    { "title": "Inception" },
    { $addToSet: { "actors": "Ellen Page" } }
);
```

Calcul de statistiques sur les notes :

Calculer la note moyenne et la note maximale des films. Requête qui serait utile pour établir un classement.

```
db.films.aggregate([
    { $unwind: "$reviews" }, // décompose le tableau reviews pour chaque document de film, ce qui crée un document par critique.
    { $group: { // regroupe les résultats par ID de film.
        _id: "$_id",
        averageRating: { $avg: "$reviews.rating" }, // calcule la moyenne des notes pour chaque groupe (film).
        maxRating: { $max: "$reviews.rating" } // trouve la note maximale pour chaque groupe (film).
    }},
    { $lookup: { // jointure avec la collection films pour récupérer les détails du film.
        from: "films",
        localField: "_id",
        foreignField: "_id",
        as: "film_details"
    }},
    { $unwind: "$film_details" }, // décompose le tableau film_details pour accéder aux détails du film.
    { $project: { // détermine les champs à inclure dans le résultat final.
        _id: 0,
        title: "$film_details.title",
        averageRating: 1,
        maxRating: 1
    }}
]);
```

(Voir résultats en Annexe 3)

Films réalisés par un réalisateur né à une période :

Donne les films des réalisateurs nés avant 1970. Purement fantaisiste pas de réel application.

```
db.realisateurs.aggregate([
    { $match: { "birthdate": { $lt: ISODate("1970-01-01T00:00:00Z") } } }, //filtre pour trouver les réalisateurs nés
avant 1970.
    { $lookup: { // jointure avec la collection films
        from: "films",
        localField: "_id",
        foreignField: "director",
        as: "films"
    }},
    { $unwind: "$films" }, // Décompose le tableau films pour accéder à chaque film individuellement.
    { $project: { // sélectionne les champs à inclure dans le résultat final
        _id: 0,
        directorName: "$name",
        filmTitle: "$films.title",
        filmYear: "$films.year"
    }}
]);
```

Cinémas ayant plus de 20 projections :

```
db.cinemas.aggregate([
    { $project: {
        name: 1, // Inclut le nom du cinéma.
        numberOfFilms: { $size: "$projections" }
    }},
    { $match: { numberOfFilms: { $gt: 20 } } } // Filtre retourner que les cinémas ayant projeté plus de 20 films.
]);
```

Nous sommes également sûrs, grâce à la validation par jsonSchema, que toutes nos requêtes sont possibles et renverront un résultat cohérent. En effet, pour insérer correctement un objet dans une collection il doit obligatoirement respecter les conditions du schéma.

Python

Connexion DB

Le fichier Python *dbutils.py* fourni définit une classe Connection pour gérer la connexion à une base de données MongoDB via PyMongo. Voici une description concise des composants du fichier :

1. Importation du MongoClient : Le module pymongo est importé pour utiliser MongoClient, permettant de créer des connexions à MongoDB.
2. Paramètres de Connexion :
    - username et password : Informations d'authentification pour l'utilisateur.
    - auth_source : Base de données utilisée pour l'authentification (généralement admin).
    - host et port : Adresse et port du serveur MongoDB.
    - database : Nom de la base de données cible.
3. Classe Connection :
    - Le constructeur initialise une instance de MongoClient avec les paramètres spécifiés et établit une connexion à la base de données désignée, accessible via self.db.

Cette classe encapsule la logique de connexion, rendant la gestion de la base de données plus structurée.

Note moyenne

Le fichier *averageRating.py* contient une méthode calculate_average_ratings() utilisant une agrégation pour traiter les données des films dans la collection MongoDB.
L'agrégation décompose les critiques associées à chaque film, calcule la note moyenne de ces critiques, et regroupe les résultats par identifiant de film.

Film par réalisateur

Le script *getFilmsByDirector.py* utilise une fonction nommée get_films_by_director_name pour récupérer et afficher les films d'un réalisateur donné, basé sur son nom. Cette fonction procède en deux étapes principales :

1. Identification du réalisateur : La fonction recherche d'abord l'identité du réalisateur dans la collection réalisateurs en utilisant son nom. Si aucun réalisateur n'est trouvé correspondant au nom spécifié, la fonction renvoie une liste vide, indiquant l'absence de correspondance dans la base de données.
2. Récupération des films : Si un réalisateur est trouvé, la fonction procède à chercher tous les films associés à cet ID de réalisateur dans la collection films. Cela permet de récupérer tous les films dirigés par ce réalisateur spécifique.

Liste des réalisateurs par taux d'occupation des salles

Le script *realisateursByMaxOccupency.py* a pour objectif de récupérer depuis la collection *cinemas* la liste de tous les réalisateurs dont les films sont projetés et d'ordonner la liste en fonction du taux d'occupation moyen des salles par réalisateur.

```python
def realisateursbymaxoccupency():
    results = db.cinemas.aggregate([
        {"$unwind": "$projections"},                                    # On déplie toutes les projections
        {"$project": {"_id": 0, "projection": "$projections"}},         # On projette uniquement la projection

        # On lie les ObjectId des films aux projections pour pouvoir récupérer le réalisateur
        {"$lookup": {"from": "films", "localField": "projection.filmid", "foreignField": "_id", "as": "film"}},

        # On projette uniquement le réalisateur et les informations d'occupation des salles
        {"$project": {"_id": 0, "director": "$film.director", "occupancy": "$projection.occupancy"}},

        # On lie les ObjectId des réalisateurs aux projections
        {"$lookup": {"from": "realisateurs", "localField": "director", "foreignField": "_id", "as": "director"}},

        # On projette uniquement le nom du réalisateur et les informations d'occupation des salles
        {"$project": {"_id": 0, "name": "$director.name", "occupancy": 1}},

        # On groupe par réalisateur et on calcule le taux d'occupation moyen
        {"$group": {"_id": "$name", "occupation": {"$avg": {"$divide": ["$occupancy.reserved", "$occupancy.total"]}}}},

        # On projette le nom du réalisateur et le taux d'occupation en pourcentage
        {"$project": {"_id": 0, "name": "$_id", "occupation": {"$multiply": ["$occupation", 100]}}},
        {"$sort": {"occupation": -1}}                                   # On trie par taux d'occupation décroissant
    ])
```

Pour réaliser cette sélection, il est nécessaire de déplier le tableau *projections* composant les entrées dans la collection *cinemas*. Une fois le tableau déplié on peut donc projeter son contenu, lié les *objectId* des films aux entrées correspondantes dans la collection *films* afin d'en récupérer le réalisateur.

Une fois l'association réalisateur/projection faite il reste plus qu'à les regrouper avec pour id le nom du réalisateur et calculer la moyenne, lors du regroupement, du taux d'occupation de chaque salle.

Enfin on sélectionne le nom du réalisateur ainsi que le taux moyen en pourcentage et on tri le résultat dans l'ordre décroissant. (Voir résultats en Annexe 4)

L'intérêt de ce script est de pouvoir rapidement mettre en évidence les réalisateurs les plus populaires. Les projections ayant un attribut contenant la date, on peut très facilement au début de l'agrégation décider de sélectionner uniquement les films en salle entre une certaine période permettant de savoir à un instant T les réalisateurs les plus populaires idéal pour maintenir des statistiques sur les cinémas.

Obtenir la meilleure séance pour un film

Le script *getNextBestProjection.py* permet de récupérer les "meilleures séances" pour un film donné par cinéma. La meilleure séance étant bien évidemment celle avec le plus de places libres.

Un mécanisme map/reduce a été utilisé pour la réalisation de cette requête :

1. Nous récupérons l'ensemble des valeurs sur lesquelles nous souhaitons travailler (ici toutes les projections d'un films dans tous les cinémas)

```python
def get_available_places_by_cinema(title):
    # Récupérer toutes les projections d'un film donné
    places_by_cinema = db.cinemas.aggregate([
        {"$unwind": "$projections"},
        {"$project": {"_id": 0, "name": "$name", "projection": "$projections"}},
        {"$match": {"projection.filmid": db.films.find_one({"title": title})["_id"]}},
        {"$project": {"_id": 0, "name": 1, "datetime": "$projection.datetime", "availablePlaces": {"$subtract":
["$projection.occupancy.total", "$projection.occupancy.reserved"]}}},
    ])

    return places_by_cinema
```

2. Enfin nous procédons à l'étape "map" qui consiste à associer toutes les projections d'un même cinéma à ce cinéma qui servira de clé pour les retrouver

```python
def map(values):
    result = {} # Accumulateur du map
    for value in values:
        if value["name"] not in result:
            result[value["name"]] = [] # Si une clé n'est pas dans l'accumulateur, on l'ajoute
        # Ajout du tuple (places disponibles, date) pour la clé cinema
        result[value["name"]].append((value["availablePlaces"], value["datetime"]))
    return result
```

3. L'étape "reduce" parcours les clés et applique une fonction permettant de trouver la séance avec le max de places disponibles

```python
def reduce(values):
    result = {} # Accumulateur du reduce
    for cinema, projections in values.items():
        # On réduit chaque valeur pour une clé en prenant la projection avec le plus de places disponibles
        best_projection = max(projections, key=lambda x: x[0])
        result[cinema] = best_projection
    return result
```

Ainsi nous obtenons pour chaque cinéma la meilleure projection du film précisé lors de la sélection initiale de notre jeu de données.

Nous avons choisi d'appliquer le mécanisme map/reduce dans la partie pyMongo de ce projet car les fonction mapReduce de mongosh ont été dépréciées en faveur des fonctions d'agrégat depuis la version 5.0 de MongoDB et il nous semblait plus pertinent des les appliquer dans la partie contenant les scripts de data analyse plutôt que dans la partie de requêtage "simple" de la base de données afin de représenter au mieux un cas d'usage réel de MongoDB avec pyMongo. (Voir résultats en Annexe 5)

Conclusion

Nous avons pu au travers de ce projet mettre en évidence l'intérêt des systèmes de gestion de documents : la flexibilité offerte par le format de stockage JSON permet ainsi de stocker des données de nature différente et/ou d'ajouter aisément de nouveaux attributs aux documents déjà existants (ajout de commentaires aux films par exemple) tout en assurant l'intégrité de la donnée en rendant nécessaire certains champs grâce à la validation par jsonSchema.

De plus, nous avons pu découvrir la facilité d'application des fonctions d'agrégation et des outils tels que map-reduce pour appliquer des calculs à des grands volumes de documents.

Enfin, nous avons pu mettre en oeuvre un accès à MongoDB depuis des scripts python avec pyMongo permettant d'abstraire les accès aux documents et/ou d'appliquer à l'aide d'un paradigme map-reduce ou des fonctions d'agrégat des calculs directement sur la donnée afin d'en ressortir des indicateurs de statistiques pour l'utilisateur final.

Annexes

Annexe 1 : Résultats récupération de tous les films d'un réalisateur


```
[
    {
        "_id": {"$oid": "663404719bfd4b076b520078"},
        "actors": ["Leonardo DiCaprio", "Joseph Gordon-Levitt"],
        "director": {"$oid": "663404709bfd4b076b52006f"},
        "director_details": {
            "_id": {"$oid": "663404709bfd4b076b52006f"},
            "name": "Christopher Nolan",
            "birthdate": {"$date": "1970-07-30T00:00:00.000Z"},
            "nationality": "British"
        },
        "reviews": [
            {"rating": 5, "comment": "Absolutely fantastic!"},
            {"rating": 4, "comment": "Really good, but a bit confusing."},
            {"rating": 5, "comment": "Masterpiece."},
            {"rating": 4, "comment": "Great visuals, solid plot."},
            {"rating": 5, "comment": "Nolan did it again!"},
            {"rating": 3, "comment": "Good but not great."},
            {"rating": 4, "comment": "Loved the complexity."},
            {"rating": 5, "comment": "Perfect, as expected."},
            {"rating": 4, "comment": "Engaging and thrilling."}
        ],
        "title": "Inception",
        "year": 2010
    },
    {
        "_id": {"$oid": "663404719bfd4b076b520079"},
        "actors": ["Matthew McConaughey", "Anne Hathaway"],
        "director": {"$oid": "663404709bfd4b076b52006f"},
        "director_details": {
            "_id": {"$oid": "663404709bfd4b076b52006f"},
            "name": "Christopher Nolan",
            "birthdate": {"$date": "1970-07-30T00:00:00.000Z"},
            "nationality": "British"
        },
        "title": "Interstellar",
        "year": 2014
    },
    {
        "_id": {"$oid": "663404719bfd4b076b52007a"},
        "actors": ["Christian Bale", "Heath Ledger"],
        "director": {"$oid": "663404709bfd4b076b52006f"},
        "director_details": {
            "_id": {"$oid": "663404709bfd4b076b52006f"},
            "name": "Christopher Nolan",
            "birthdate": {"$date": "1970-07-30T00:00:00.000Z"},
            "nationality": "British"
        },
        "title": "The Dark Knight",
        "year": 2008
    },
    {
        "_id": {"$oid": "663404719bfd4b076b52007b"},
        "actors": ["Christian Bale", "Tom Hardy"],
        "director": {"$oid": "663404709bfd4b076b52006f"},
        "director_details": {
            "_id": {"$oid": "663404709bfd4b076b52006f"},
            "name": "Christopher Nolan",
            "birthdate": {"$date": "1970-07-30T00:00:00.000Z"},
            "nationality": "British"
        },
        "title": "The Dark Knight Rises",
        "year": 2012
    }
]
```

Annexe 2 : Compter le nombre de films d'un réalisateur


```
[
    {"count": 3, "director_name": "Francis Ford Coppola"},
    {"count": 2, "director_name": "James Cameron"},
    {"count": 1, "director_name": "Frank Darabont"},
    {"count": 3, "director_name": "Ridley Scott"},
    {"count": 1, "director_name": "Robert Zemeckis"},
    {"count": 3, "director_name": "Lana Wachowski"},
    {"count": 4, "director_name": "David Fincher"},
    {"count": 4, "director_name": "Christopher Nolan"},
    {"count": 4, "director_name": "Quentin Tarantino"}
]
```


Annexe 3 : Calcul de statistiques sur les notes


```
[
    {"averageRating": 4.333333333333333, "maxRating": 5, "title": "Inception"},
    {"averageRating": 4.0, "maxRating": 5, "title": "The Dark Knight"},
    {"averageRating": 2.5, "maxRating": 4, "title": "Forrest Gump"}
]
```


Annexe 4 : Liste des réalisateurs par taux d'occupation des salles

```
Realisateurs par taux d'occupation maximal :
{'name': ['Christopher Nolan'], 'occupation': 31.156250000000004}
{'name': ['David Fincher'], 'occupation': 30.074999999999996}
{'name': ['Quentin Tarantino'], 'occupation': 25.21875}
{'name': ['Robert Zemeckis'], 'occupation': 24.4375}
{'name': ['James Cameron'], 'occupation': 23.8125}
{'name': ['Ridley Scott'], 'occupation': 23.791666666666668}
{'name': ['Francis Ford Coppola'], 'occupation': 21.541666666666668}
{'name': ['Frank Darabont'], 'occupation': 20.625}
```

Annexe 5 : Obtenir la meilleure séance pour un film

```
Cinéma Darcy : 2024-04-06 18:00:00 (193 places disponibles)
Cinéma Eldorado : 2024-04-09 18:00:00 (200 places disponibles)
```