# Rapport de TP1 : Prise en main de MongoDB (NoSQL)

**Nom/Prénom : GNOUG Omayma**

**Date : 27/11/2025**

**Sujet : Interrogation et manipulation d'une base de données documentaire.**

## 1. Introduction

L'objectif de ce TP est de se familiariser avec MongoDB, un système de gestion de bases de données NoSQL orienté documents. Nous avons appris à importer un jeu de données, à comprendre la structure JSON des documents et à effectuer des requêtes de recherche (filtrage, projection, opérateurs logiques) sur une collection de films.

## 2. Importation et vérification des données

Nous avons commencé par importer un fichier JSON contenant des données sur des films dans la base de données lesfilms.

**Commande d'import :**

Bash
```
./mongoimport --db lesfilms --collection films films.json --jsonArray
```

**Vérification (Exercices 1 & 2) :**

Pour nous assurer que l'importation s'est bien déroulée et comprendre le schéma des données (qui est flexible), nous avons compté les documents et affiché un premier élément.

- db.films.count() : Retourne le nombre total de documents.
- db.films.findOne() : Affiche la structure d'un film (champs disponibles comme *genre*, *country*, *year*, etc.).

**3. Requêtes de filtrage simple (Exercices 3 à 6)**

Nous avons effectué des recherches basées sur des critères d'égalité simple et combiné plusieurs critères pour affiner les résultats.

- **Filtrage par genre :** db.films.find({ genre: "Action" })
- **Comptage spécifique :** Utilisation de .count() sur le résultat du filtre.
- **Critères multiples :** Recherche des films d'action produits en France, puis affinage par année (1963).
    - *Exemple :* db.films.find({ genre: "Action", country: "France", year: 1963 })

**4. Projections et Affichage (Exercices 7 & 8)**

Pour optimiser la lecture et ne récupérer que les informations pertinentes, nous avons utilisé des projections (le deuxième argument de la méthode find).

- **Exclusion de l'ID :** { _id: 0 } pour masquer l'identifiant technique généré par MongoDB.
- **Sélection de champs :** { title: 1, grade: 1 } pour n'afficher que le titre et la note.

**5. Opérateurs de comparaison et tableaux (Exercices 9 & 10)**

Nous avons manipulé des opérateurs pour des requêtes plus complexes sur les valeurs numériques et les tableaux.

- **Supériorité ($gt) :** Sélection des films avec une note (rating) strictement supérieure à 10.
- **Validation sur un tableau (Double négation) :** Pour trouver les films dont **toutes** les notes sont supérieures à 10, nous avons utilisé une logique d'exclusion : "Ne contient aucun élément inférieur ou égal à 10".
    - *Commande :* ratings: { $not: { $elemMatch: { $lte: 10 } } }

**6. Analyse des valeurs distinctes (Exercices 11 & 12)**

Pour connaître l'étendue des données sans afficher tous les documents, nous avons utilisé la commande distinct.

- db.films.distinct("genre") : Liste tous les genres uniques présents dans la collection.
- db.films.distinct("grade") : Liste toutes les notes attribuées.

**7. Opérateurs logiques et d'existence (Exercices 13 à 16)**

Enfin, nous avons exploré des opérateurs permettant des recherches flexibles :

- **Inclusion (`$in`) :** Recherche de films contenant au moins un artiste parmi une liste donnée (["artist:4", "artist:18"...]).
- **Existence (`$exists`) :** Identification des documents mal formés ou incomplets (ex: films sans résumé).
- **Opérateur OU (`$or`) :** Recherche combinant deux conditions indépendantes (Films avec "Leonardo DiCaprio" **OU** sortis en 1997).

### 8. Conclusion

Ce TP a permis de valider les compétences fondamentales d'interrogation sous MongoDB. Nous maîtrisons désormais la syntaxe de find(), l'utilisation des projections pour formater la sortie, ainsi que les opérateurs essentiels ($gt, $in, $or, $exists, $elemMatch) pour extraire des informations précises d'une base de données non relationnelle.