Rapport TP

Prise en main de MongoDB - Partie 2

Addou Aicha Amira

Bombay Marc

3ème Année Ingénieur - INFO - Apprentissage

Importation de BDD

Les données utilisées font partie de la base de données provenant du fichier `films.json` donné dans le Moodle.

Elles ont été importées dans l'image Docker de MongoDB par la commande

`docker run --it mongoimport --db lesfilms --collection films films.json --jsonArray`

Requêtes

1. Vérification de la bonne importation

```
> db.films.find({}).count()
< 278
```

- Les données sont bien présentes, on a 278 films.

2. Structure d'un document

Pour identifier la structure du document nous avons utilisé la commande

`db.films.findOne()`

Il est composé comme suit:

- `_id`: Utilisé comme identifiant du document pour le film
- `title`: Titre du film
- `year`: Année de parution
- `genre`: Genre du film
- `summary`: Résumé (synopsis)
- `country`: Pays du film
- `director`: Objet contenant:
  - `_id`: Identifiant
  - `last_name`: Nom de famille
  - `first_name`: Prénom
  - `birth_date`: Année de naissance
- `actors`: Tableau, avec des objets du même type que director, sans _id
- `grades`: Tableau des différentes notes reçues:

- `note`: Note numérique
- `grade`: Note en lettre

3. Afficher la liste des films d'action

```
> db.films.find({genre:"Action"},{title:1})
< {
    _id: 'movie:24',
    title: 'Kill Bill : Volume 1'
  }
  {
    _id: 'movie:98',
    title: 'Gladiator'
  }
  {
    _id: 'movie:180',
    title: 'Minority Report'
  }
  {
    _id: 'movie:218',
    title: 'Terminator'
  }
  {
    _id: 'movie:272',
    title: 'Batman Begins'
  }
  {
    _id: 'movie:280',
    title: 'Terminator 2: Judgment Day'
  }
```

- Condition simple pour retrouver tous les documents où **`genre = "Action"`**
- Ajout d'un filtre pour ne renvoyer que le titre pour ne pas avoir trop d'informations inutiles.

4. Afficher le nombre de films d'action

Il suffit d'ajouter **`.count()`** pour cela.

```
> db.films.find({genre:"Action"},{title:1}).count()
< 36
```

-

5. Films d'action produits en France

On ajoute à la requête précédente une condition sur le champ `country` afin de filtrer les films d'action produits en France.

```
> db.films.find({genre:"Action",country:"FR"},{title:1}).count()
< 3
> db.films.find({genre:"Action",country:"FR"},{title:1})
< {
    _id: 'movie:4034',
    title: "L'Homme de Rio"
  }
  {
    _id: 'movie:9322',
    title: 'Nikita'
  }
  {
    _id: 'movie:25253',
    title: 'Les tontons flingueurs'
  }
```

6. Films d'action produits en France en 1963

On affine la recherche en ajoutant une condition supplémentaire sur l'année de production.

```
> db.films.find({genre:"Action",country:"FR",year:1963},{title:1})
< {
    _id: 'movie:25253',
    title: 'Les tontons flingueurs'
  }
> db.films.find({genre:"Action",country:"FR",year:1963},{title:1}).count()
< 1
```

7-8. Masquage d'attribut

La projection permet d'afficher uniquement certains attributs des documents, tout en masquant d'autres. Ici on rend visibles **titre** et **country** est visible. On a masqué également l'attribut "_id" qui par défaut est toujours visible.

```
> db.films.find(
    { genre: "Action", country:"FR" },
    { title: 1, country: 1, _id: 0 }
  )
< {
    title: "L'Homme de Rio",
    country: 'FR'
  }
  {
    title: 'Nikita',
    country: 'FR'
  }
  {
    title: 'Les tontons flingueurs',
    country: 'FR'
  }
```

9. Titres + grades des films d'action français, sans _id

Utilisation du filtre pour n'avoir que les deux champs title et grades. On combine filtre et projection.

```
> db.films.find(
    { genre: "Action", country: "FR" },
    { title: 1, grades: 1, _id: 0 }
  )
< {
    title: "L'Homme de Rio",
    grades: [
      {
        note: 4,
        grade: 'D'
      },
      {
        note: 30,
        grade: 'E'
      },
      {
        note: 34,
        grade: 'E'
      },
      {
        note: 28,
        grade: 'F'
      }
    ]
  }
```

10. Films d'action français avec une note > 10 (certains < 10 apparaissent)

On note que l'on écrira **`grades.note`** pour accéder à l'attribut **`note`** de **`grades`**.
La condition **`{ $gt: 10 }`** ici nous donne tous les films où **au moins une note est supérieure (strictement) à 10**, d'où l'apparition ici du film *L'Homme de Rio* même s'il a reçu une note à 4.

```
> db.films.find(
    { genre: "Action", country: "FR", "grades.note": { $gt: 10 } },
    { title: 1, grades: 1, _id: 0 }
  )
< {
    title: "L'Homme de Rio",
    grades: [
      {
        note: 4,
        grade: 'D'
      },
      {
        note: 30,
        grade: 'E'
      },
      {
        note: 34,
        grade: 'E'
      },
      {
        note: 28,
        grade: 'F'
      }
    ]
  }
```

11. Films d'action réalisés en France n'ayant que des notes supérieures à 10

Notre première idée était d'utiliser l'opérateur **`$all <condition>`**, mais celui-ci sert uniquement à vérifier la présence de valeurs précises dans un tableau et ne permet pas de tester une condition sur chaque élément.
Pour vérifier que toutes les notes sont strictement supérieures à 10, on utilise plutôt la combinaison **`$elemMatch`** avec **`$not`**, ce qui garantit qu'aucun élément ne contient une note inférieure ou égale à 10.

```
> db.films.find({
    genre: "Action",
    country: "FR",
    grades: { $not: { $elemMatch: { note: { $lte: 10 } } } }},
    {title:1}
  )
< {
    _id: 'movie:9322',
    title: 'Nikita'
  }
  {
    _id: 'movie:25253',
    title: 'Les tontons flingueurs'
  }
```

12. Afficher les différents genres

Peut se faire via **`distinct(<champ>)`** comme suit.

```
> db.films.distinct("genre")
< [
    'Action',           'Adventure',
    'Aventure',         'Comedy',
    'Comédie',          'Crime',
    'Drama',            'Drame',
    'Fantastique',      'Fantasy',
    'Guerre',           'Histoire',
    'Horreur',          'Musique',
    'Mystery',          'Mystère',
    'Romance',          'Science Fiction',
    'Science-Fiction',  'Thriller',
    'War',              'Western'
  ]
```

13. Afficher les différentes valeurs de grades attribués

La question est ambiguë : on peut vouloir soit les différents grades alphabétiques (grades.grade), soit toutes les paires note/grade du tableau grades. Nous avons donc exécuté les deux versions :

- **`db.films.distinct("grades.grade")`** pour les grades possibles,
- **`db.films.distinct("grades")`** pour les combinaisons complètes.

```
> db.films.distinct("grades")
< [
    { note: 1, grade: 'A' },  { note: 1, grade: 'B' },  { note: 1, grade: 'C' },
    { note: 1, grade: 'D' },  { note: 1, grade: 'F' },  { note: 2, grade: 'A' },
    { note: 2, grade: 'C' },  { note: 2, grade: 'D' },  { note: 2, grade: 'F' },
    { note: 3, grade: 'B' },  { note: 3, grade: 'C' },  { note: 3, grade: 'D' },
    { note: 3, grade: 'E' },  { note: 3, grade: 'F' },  { note: 4, grade: 'A' },
    { note: 4, grade: 'B' },  { note: 4, grade: 'C' },  { note: 4, grade: 'D' },
    { note: 4, grade: 'E' },  { note: 4, grade: 'F' },  { note: 5, grade: 'A' },
    { note: 5, grade: 'B' },  { note: 5, grade: 'C' },  { note: 5, grade: 'D' },
```

```
> db.films.distinct("grades.grade")
< [ 'A', 'B', 'C', 'D', 'E', 'F' ]
```

14. Afficher tous les films avec au moins un artiste d'une liste donnée

On n'a retrouvé l'attribut **`_id`** que dans **`director`**.
Cependant, aucun des directeurs de la base n'avait au moins un des identifiants précisés dans la liste d'origine (**`["artist:4","artist:18","artist:11"]`**)

Ci-dessous la liste des identifiants présents en base, triés dans l'ordre alphanumérique.
On remarque par exemple l'absence de **`"artist:11"`** qui aurait dû être entre **`10930`** et **`11147`**

```
> db.films.distinct("director._id")

< [
    'artist:1',     'artist:10001',   'artist:1031204', 'artist:1032',
    'artist:10491', 'artist:10702',   'artist:108',     'artist:1090',
    'artist:10921', 'artist:10930',   'artist:11147',   'artist:11195',
    'artist:11401', 'artist:1150',    'artist:11528',   'artist:1175620',
    'artist:11770', 'artist:1177758', 'artist:11983',   'artist:12114',
    'artist:1223',  'artist:12238',   'artist:1224',    'artist:122423',
    'artist:12329', 'artist:1243',    'artist:12989',   'artist:13284',
    'artist:13294', 'artist:136168',  'artist:137427',  'artist:138',
    'artist:13848', 'artist:144817',  'artist:149265',  'artist:15127',
    'artist:15344', 'artist:1614',    'artist:16294',   'artist:1650',
    'artist:16938', 'artist:16960',   'artist:1726824', 'artist:17496',
    'artist:1769',  'artist:1776',    'artist:18175',   'artist:18563',
    'artist:190',   'artist:19077',   'artist:19665',   'artist:19800',
    'artist:201',   'artist:20561',   'artist:2087',    'artist:21669',
    'artist:21684', 'artist:2176',    'artist:21769',   'artist:2226',
    'artist:223',   'artist:224',     'artist:231431',  'artist:2352',
    'artist:240',   'artist:24882',   'artist:2559',    'artist:2636',
    'artist:2690',  'artist:2710',    'artist:28615',   'artist:291263',
    'artist:30711', 'artist:30715',   'artist:3146',    'artist:32134',
    'artist:34613', 'artist:3556',    'artist:35927',   'artist:37626',
    'artist:3776',  'artist:3831',    'artist:39',      'artist:39104',
    'artist:3950',  'artist:3974',    'artist:39996',   'artist:40',
    'artist:4109',  'artist:42',      'artist:4385',    'artist:4387',
    'artist:4415',  'artist:488',     'artist:5026',    'artist:510',
    'artist:5231',  'artist:525',     'artist:54291',   'artist:54472',
    ... 33 more items
  ]
```

Donc pour tester notre requête on a utilisé les identifiants suivants:

```
> db.films.find({
    "director._id": { $in: ["artist:40", "artist:1", "artist:510"] }},
    {title:1, director:1}
  )

< {
    _id: 'movie:11',
    title: 'La Guerre des étoiles',
    director: {
      _id: 'artist:1',
      last_name: 'Lucas',
      first_name: 'George',
      birth_date: 1944
    }
  }
  {
    _id: 'movie:75',
    title: 'Mars Attacks!',
    director: {
      _id: 'artist:510',
      last_name: 'Burton',
      first_name: 'Tim',
      birth_date: 1958
    }
  }
  {
    _id: 'movie:838',
    title: 'American Graffiti',
    director: {
      _id: 'artist:1',
      last_name: 'Lucas',
      first_name: 'George',
      birth_date: 1944
    }
```

15. Films qui n'ont pas de résumé

Dans le TP précédent, nous avions utilisé l'opérateur **`$exists`** pour détecter l'absence d'un champ dans les documents.
Ici, cette requête ne retourne aucun film, ce qui signifie que tous les documents possèdent bien un champ summary.
En revanche, en testant les résumés vides (**`summary: ""`**), nous obtenons des résultats, ce qui montre que certains films ont un champ présent mais vide.

```
> db.films.find({ summary: { $exists: false } })

<
> db.films.find({ summary: "" })
< {
    _id: 'movie:181812',
    title: 'Star Wars, épisode IX',
    year: 2019,
    genre: 'Science-Fiction',
    summary: '',
    country: 'GB',
    director: {
      _id: 'artist:15344',
      last_name: 'Abrams',
      first_name: 'J.J.',
      birth_date: 1966
```

```
> db.films.find({ summary: "" }).count()
< 1
> db.films.find({ summary: { $exists: false } }).count()

< 0
```

16. Films tournés avec Leonardo DiCaprio en 1997

La requête utilise **`$elemMatch`** suivie des deux correspondances recherchées

```
> db.films.find({
    year: 1997,
    actors: {
      $elemMatch: { first_name: "Leonardo", last_name: "DiCaprio" }
    }
  },
  {title:1}
  )
< {
    _id: 'movie:597',
    title: 'Titanic'
  }
```

17. Films avec Leonardo DiCaprio ou réalisés en 1997

On utilise ici l'opérateur **`$or`** :

```
> db.films.find({
    $or: [
      { year: 1997 },
      {
        actors: {$elemMatch: { first_name: "Leonardo", last_name: "DiCaprio" }}
      }
    ]
  },{title:1,_id:0})

< {
    title: 'Jackie Brown'
  }
  {
    title: 'Le monde perdu : Jurassic Park'
  }
  {
    title: 'Starship Troopers'
  }
  {
    title: 'Titanic'
  }
  {
    title: 'Volte/Face'
  }
  {
    title: 'On connaît la chanson'
  }
  {
    title: 'Inception'
  }
  {
    title: 'Django Unchained'
  }
  {
    title: 'The Revenant'
  }
```