MongoDB

Nicoleta Preda

October 20, 2023

Plan

Documents structurés

- **Le document** est l'unité de stockage de données dans les bases NoSQL.
- Le **concept de document** est similaire au **concept de fichier** dans les systèmes d'exploitation.
- À différence d'un fichier, les données d'un document doivent être structurées d'après un format prédéfini (e.g., JSON, XML).

Popularité et importance:

- **Le document (JSON)** est le moyen standard d'échange entre des applications à distance,
- et ceci indépendamment du stockage des données à l'application source ou cible.

De manière abstraite, énumérons ce que un document devrait être capable à stocker:

1. une valeur atomique (un entier, une chaîne de caractères),
2. une paire clé-valeur,
3. un tableau (une liste) de valeurs,
4. un ensemble de paires clé-valeur,
5. une structure imbriquée représentant une composition des structures précédentes.

Structure d'une base de données MondoDB

Collection = ensemble de documents, généralement d'une structure similaire.

Document = un objet JSON

- un objet JSON est une structure imbriqué qui peut être représentée comme un arbre.
- stockage interne au format BSON (Binary JSON)
- taille maximale autorisée de 16 Mo.
- chaque document a un identifient unique, typiquement un champ nommé **_id** à la racine.
- Restrictions sur les champs des documents:
  Le champ de premier niveau **_id** est réservé pour une clé primaire.
  Caractères interdits pour les noms de champs: $ et . (point).
  $ est réservé pour les opérateurs de requête.
  Le point (.) est utilisé lors de l'accès aux champs imbriqués.
  L'ordre des champs est préservé à l'exception des champs **_id**, toujours déplacé au début.
  Les noms des champs doivent être uniques.

Clés Primaires

Caractéristiques des identifiants:

- **uniques** au sein **d'une collection** de documents.
- **immuables** (ne peuvent pas être modifiés une fois attribués).
- Peuvent être **de n'importe quel type sauf tableau JSON**.
- Typiquement, **ObjectId** - type BSON spécial de 12 octets (l'option par défaut).
  rapide à générer, ordonné, basé sur un horodatage, un identifiant de machine, un identifiant de processus, et un compteur local au processus.

Plan

Conception de la base de données

Aucun schéma explicite n'est fourni, ni attendu ni imposé.

Cependant...

- Il est nécessaire de structuré les documents au sein d'une collection d'après le même schéma pour **faciliter les recherches**.
- Les besoins **de représenter les données sans (trop de) redondances**, poussent les concepteurs à réfléchir à une structuration, et donc à un schéma, pour les documents.

Comment restructurer cette BD dans une collection de documents JSON?

Films

idF	title	director	country	year	genre
A	Life of Brian	4	UK	1979	comedie
B	Good Bye, Lenin!	3	DE	2003	comedie
C	La La Land	6	USA	2016	musical
D	The Favourite	7	UK	2018	drama

Persons

idP	name	birthdate
1	Daniel Brühl	1978-06-16
2	Katrin Sass	1956-10-23
3	Wolfgang Becker	1954-06-22
4	Terry Jones	1942-02-01
5	Emma Stone	1988-11-06
6	Damien Chazelle	1985-01-19
7	Yorgos Lanthimos	1973-04-27
8	Olivia Coleman	1974-01-30
9	Rachel Weisz	1970-03-07
10	Ryan Gosling	1980-11-02

Casting

idF	**idP**	**role**
B	1	Alexander Kerner
B	2	Christiane
A	4	Mandy Cohen
A	4	Colin
A	4	Simon the Holy Man
C	5	Mia
C	10	Sebastian Wilder
D	5	Abigail
D	8	Queen Anne
D	9	Sarah Churchill

- Les clés primaires sont en gras.
- Les clés étrangère sont : Casting.idF $\subseteq$ Film.idF, Casting.idA $\subseteq$ Person.idP, Film.director $\subseteq$ Person.idP

Document JSON pour le film The Favorite

```
{
  "_id": ObjectId("..."),
  "title": "The Favourite",
  "director": {
    "name": "Yorgos Lanthimos",
    "birthdate": new Date("1973-04-27T00:00:00Z")
  },
  "country": "UK",
  "year": 2018,
  "genre": "drama",
  "cast": [
    {
      "name": "Emma Stone",
      "birthdate": new Date("1988-11-06T00:00:00Z"),
      "role": "Abigail"
    },
    {
      "name": "Olivia Coleman",
      "birthdate": new Date("1974-01-30T00:00:00Z"),
      "role": "Queen Anne"
    },
    {
      "name": "Rachel Weisz",
      "birthdate": new Date("1970-03-07T00:00:00Z"),
      "role": "Sarah Churchill"
    }
  ]
}
```

- Le champ **_id** n'as pas besoin d'être donnée.
- Si **_id** n'est pas fourni, MongoDB va le créer et il va l'initialiser avec un identifiant unique dans la collection.
- Nous pourrions réutiliser les identifiants existants de la base de données relationnelle pour les documents MongoDB, par exemple, en assignant manuellement **"_id": "D"** au lieu de générer de nouveaux IDs.
- 2018 n'apparaît pas entre des guillemets. Il est considéré de type integer et pas string.
- Les dates utilisent le format ISO 8601 (YYYY-MM-DDTHH:mm:ss.sssZ) et sont stockées en tant qu'ISODate en MongoDB, compatibles avec les objets Date de JavaScript, facilitant ainsi les comparaisons dans les requêtes.

Document JSON pour le film La La Land

```
{
  "title": "La La Land",
  "director": {
    "name": "Damien Chazelle",
    "birthdate": ISODate("1985-01-19T00:00:00Z")
  },
  "country": "USA",
  "year": 2016,
  "genre": "musical",
  "cast": [
    {
      "name": "Emma Stone",
      "birthdate": ISODate("1988-11-06T00:00:00Z"),
      "role": "Mia"
    },
    {
      "name": "Ryan Gosling",
      "birthdate": ISODate("1980-11-02T00:00:00Z"),
      "role": "Sebastian Wilder"
    }
  ]
}
```

Concept: documents imbriqués ou embarqués (structure hiérarchique)

```
"_id": ObjectId("..."),
"title": "La La Land",
"director": {
    "name": "Damien Chazelle",
    "birthdate": ISODate("1985-01-19T00:00:00Z")
    },
"country": "USA",
"year": 2016,
"genre": "musical",
"cast": [
        {
         "name": "Emma Stone",
         "birthdate": ISODate("1988-11-06T00:00:00Z"),
         "role": "Mia"
        },
       {
         "name": "Ryan Gosling",
         "birthdate": ISODate("1980-11-02T00:00:00Z"),
         "role": "Sebastian Wilder"
        }
 ]
```

- **Top-Level Document:** le document principal
- **Nested Documents:** les objets "director" et "cast" et tous les objets non nommés dans le tableau du "cast"
- **Arrays of Documents:** les objets non nommés dans le tableau du "cast"

Stratégies de Référencement dans MongoDB

- **Documents Embarqués :** Pas de référence, au lieu d'une référence, copier le document voulu dans le document parent. Commun en NoSQL, optimise les performances de lecture mais peut entraîner des duplications de données et des incohérences.
- **Références Manuelles :** Sauvegarde manuelle de l'_id d'un document dans un autre. Nécessite une deuxième requête pour récupérer le document référencé.
- **DBRefs :** Standard MongoDB pour stocker des références, inclut l'_id, le nom de la collection, et parfois le nom de la base de données.

Plan

Langage de Requête dans MongoDB

- **Fondé sur JavaScript :** Les commandes peuvent être des requêtes simples ou des scripts entiers.
- **Curseurs pour les Résultats de Requêtes :** Les requêtes de lecture retournent un curseur pour parcourir les documents.
- **Commandes par Collection :** Chaque commande s'exécute sur une seule collection.

Types d'Opérations

- **Opérations CRUD :** Création, Lecture, Mise à jour, Suppression.
- Opérations avancées pour l'analyse des données (MapReduce, pipelines, regroupements).

Opérations CRUD

- **Aperçu :**
  - Les opérations CRUD se réfèrent à la création, la lecture, la mise à jour, et la suppression de documents dans une base de données MongoDB.
- **Insertion :** `db.collection.insert()`
  - Insère un nouveau document dans une collection.
- **Mise à jour :** `db.collection.update()`
  - Modifie un ou plusieurs documents existants ou insère un nouveau document si celui-ci n'existe pas.
- **Suppression :** `db.collection.remove()`
  - Supprime un ou plusieurs documents existants.
- **Recherche :** `db.collection.find()`
  - Trouve des documents en fonction des conditions de filtrage.
  - La projection et/ou le tri peuvent également être appliqués.

Création d'une Collection

- **Création Implicite :**
  - Une collection est automatiquement créée lors de l'insertion du premier document.
  - Commande utilisée: `db.collection.insert()`
- **Création Explicite :**
  - Créer une collection manuellement sans insérer de documents.
  - Commande utilisée: `db.createCollection("nom")`
- **Note :**
  - `"nom"` représente le nom que vous souhaitez attribuer à la collection.
  - La création explicite est utile si vous voulez définir des options spécifiques lors de la création de la collection (comme la taille maximale, etc.).

MongoDB: opération d'insertion

- **Rôle:** Ajouter de nouveaux documents à une collection.
- **Commande :** db.collection.insert()
- **Syntaxe :**
  - `db.collection.insert(documentOrArray, options)`
  - `documentOrArray` : Un document ou un tableau de documents à insérer.
  - `options` : Facultatif. Un document spécifiant les options d'écriture.
- **Retour :** Un document contenant le statut de l'opération.
- **Note :** Si aucun identifiant (_id) n'est spécifié, MongoDB en attribue un automatiquement.

Example (L'insertion d'un document dans la collection acteurs)

```
db.actors.insert({
    "name": "Emma Stone",
    "birthdate": new Date("1988-11-06T00:00:00Z")
 })
```

Après l'insertion, dans la collection acteurs de la BD MongoDB

```
{
 "_id": ObjectId("507f1f77..."),
 "name": "Emma Stone",
 "birthdate": new Date("1988-11-06T00:00:00Z")
}
```

Insertion d'un tableau de documents

Example (L'insertion de deux documents dans la collection acteurs)

```
db.actors.insert([
  {
    "name": "Emma Stone",
    "birthdate": new Date("1988-11-06T00:00:00Z")
  },
  {
    "name": "Olivia Coleman",
    "birthdate": new Date("1974-01-30T00:00:00Z")
  }
])
```

Après l'insertion, dans la collection acteurs de la BD MongoDB

```
{
 "_id": ObjectId("507f1f77..."),
 "name": "Emma Stone",
 "birthdate": new Date("1988-11-06T00:00:00Z")
}
```

```
{
 "_id": ObjectId("507f1f88..."),
 "name": "Olivia Colman",
 "birthdate": new Date("1974-01-30T00:00:00Z")
}
```

Opération de recherche find dans MongoDB

- La "sélection" et la "projection" sont deux concepts clés des opérations de recherche.
- Elles permettent de filtrer et de formater les résultats des requêtes.

Sélection

- Défini les conditions pour filtrer les documents d'une collection.
- Les résultats sont des documents dans leurs intégralité.
- Les conditions peuvent concerner des attributs de niveau racine, de sous-documents, ou des éléments dans des tableaux.

Exemple: Sélection des films de la réalisatrice Greta Gerwig

```
db.films.find({ "director.name": "Greta Gerwig" })
```

Projection

- Détermine les parties spécifiques des documents à récupérer.
- Façonne l'ensemble des résultats.
- Permet de limiter les données retournées aux champs spécifiques.

Exemple: Sélection des films de la réalisatrice Greta Gerwig, puis projection pour extraire seulement le titre et le réalisateur du film.

```
db.movies.find({ "director.name": "Greta Gerwig" },
               {"title": 1,"director": 1})
```

Sélection

Example (Les films, documents top-level de la collection films, produits en 2018?)

```
db.films.find({ "year": 2018})
```

Example (Les films, où "Emma Stone" a joué?)

```
db.films.find({ "cast.name": "Emma Stone" })
```