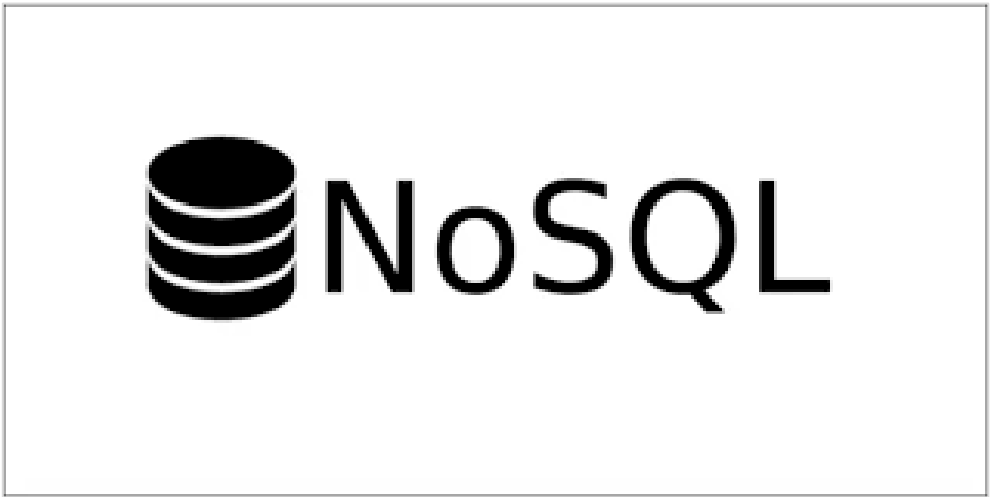
# P-DB-I165




Tiago Rodrigues Sousa – MID2
A02
32 périodes
Mathieu Meylan

# Table des matières

# SPÉCIFICATIONS

## Titre

Projet I165 MongoDB

## Description

Dans le cadre du projet I165, il a été demandé de faire plusieurs choses :

- Importation d'un db
- Gestion des rôles et des utilisateurs
- Des requêtes de sélection
- Des requêtes d'agrégation
- Des requêtes textuelles
- Exportation de la db

Pour ce faire nous avons suivi le module I165.

## Matériel et logiciels à disposition

- Un PC ETML
- Environnement docker avec container MongoDB (serveur et mongosh)
- Interface de gestion MongoDB compass + VS Code avec l'extension MongoDB
- Accès à Internet
- Docker auquel nous avons reçu un docker compose avec un serveur MongoDB et Mongo Compass

## Prérequis

Module I165

# RÉALISATION

## Restaurer une archive

Une archive d'un dump a été fournie et elle s'appelle db_mflix.gz. Il a été demandé de restaurer la base de données. Voici la commande :

```
docker exec -i mongo mongorestore --uri=mongodb://root:admin@localhost:27017
--authenticationDatabase=admin --gzip --archive=/backupdb/db_mflix.gz
```

Pour rappel, notre base de données MongoDB est stockée sur un container docker. Ce qui signifie que pour importer la base de données, il a fallu créer un endroit sur l'ordinateur que le container puisse voir.

***docker*** : notre base de données est stockée sur un container docker, ce bout de code indique que nous allons exécuter une commande sur docker

***exec*** : cela indique que nous allons exécuter une commande sur un container

***-i mongo*** : permet d'indiquer sur quel container la commande sera appliquée (ici, se sera sur le container mongo)

***mongorestore*** : indique que nous allons utiliser le programme « *mongorestore* » qui permet de restaurer la base de données qui a été dump (avec la commande mongodump)

***--uri*** : indique dans quel serveur nous voulons restaurer la base de données et les options de connexions

***mongodb://root:admin@localhost:27017*** :

- mongodb = le protocole
- root = nom d'utilisateur
- admin = mot de passe
- localhost (127.0.0.1) = adresse IP
- 27017 = port (attention si vous utilisez docker, a bien vérifié quel port est attaché au container)

***authenticationDatabase=admin*** : cette ligne de commande indique dans quel base de données il faut aller chercher les informations de connexion (ici, c'est admin)

***--gzip*** : cette option indique que la base de données que nous lui donnons est compressée (on peut le voir grâce au format .gz)

***--archive*** : cette option indique que la base de données que nous lui donnons a été archivée lors du dump, cette à dire que cela a donné un seul fichier (comme lorsqu'on compresse une arborescence de fichier)

***/backupdb/db_mflix.gz*** : ici nous lui indiquons le fichier (attention, il faut que le container puisse voir le fichier donc il faut au préalable partager un endroit entre le container et la machine host)

PS : il faut faire attention au nom des élément dans la commande, si quelque chose comporte des espaces ou des guillemets, à faire attention à entourer le mot de guillemets simple ou double.

## Faire un backup de la base de données

```
docker exec -i mongo mongodump
--uri="mongodb://root:admin@localhost:27017"
--authenticationDatabase="admin"
--db=db_mflix
--gzip --out=/backupdb/dump-db_mflix-08-03-2024
```

***docker*** : notre base de données est stockée sur un container docker, ce bout de code indique que nous allons exécuter une commande sur docker

***exec*** : cela indique que nous allons exécuter une commande sur un container

***-i mongo*** : permet d'indiquer sur quel container la commande sera appliquée (ici, se sera sur le container mongo)

***mongodump*** : indique que nous allons utiliser le programme « *mongodump* » qui permet de faire une sauvegarde complète de la base de données (en BSON)

***--uri*** : indique dans quel serveur nous voulons restaurer la base de données et les options de connexions

***mongodb://root:admin@localhost:27017*** :

- mongodb = le protocole
- root = nom d'utilisateur
- admin = mot de passe
- localhost (127.0.0.1) = adresse IP
- 27017 = port (attention si vous utilisez docker, a bien vérifié quel port est attacher au container)

***authenticationDatabase=admin*** : cette ligne de commande indique dans quel base de données il faut aller chercher les informations de connexion (ici, c'est admin)

***--gzip*** : cette option indique que nous allons compresser le dump en .gz

***--out/backupdb/mongo-dump-db_mflix-08-03-2024*** : cette option indique l'emplacement d'où va être le backup. Elle sera composée d'un dossier de la db dans laquelle plusieurs fichiers représentant chacun une collection. Attention, notre base de données est stockée sur un container donc le fichier va se retrouver là où est le volume partagé

Dans cette commande nous utilisons la méthode de compression gzip pour réduire un maximum la taille du fichier. Pour la restaurer, il suffit de suivre les étapes de « Restaurer le backup de la base de données » ci-dessous.

## Restaurer le backup de la base de données

```
docker exec -i mongo mongorestore
--uri="mongodb://root:admin@localhost:27017"
--authenticationDatabase="admin"
--db=db_mflix
--gzip /backupdb/dump-db_mflix-08-03-2024/db_mflix
```

Pour rappel, notre base de données MongoDB est stockée sur un container docker. Ce qui signifie que pour importer la base de données, il a fallu créer un endroit sur l'ordinateur que le container puisse voir.

***docker*** : notre base de données est stockée sur un container docker, ce bout de code indique que nous allons exécuter une commande sur docker

***exec*** : cela indique que nous allons exécuter une commande sur un container

***-i mongo*** : permet d'indiquer sur quel container la commande sera appliquée (ici, se sera sur le container mongo)

***mongorestore*** : indique que nous allons utiliser le programme « *mongorestore* » qui permet de restaurer la base de données qui a été dump (avec la commande mongodump)

***--uri*** : indique dans quel serveur nous voulons restaurer la base de données et les options de connexions

***mongodb://root:admin@localhost:27017*** :

- mongodb = le protocole
- root = nom d'utilisateur
- admin = mot de passe
- localhost (127.0.0.1) = adresse IP
- 27017 = port (attention si vous utilisez docker, a bien vérifié quel port est attacher au container)

***authenticationDatabase=admin*** : cette ligne de commande indique dans quel base de données il faut aller chercher les informations de connexion (ici, c'est admin)

***--gzip*** : cette option indique que la base de données que nous lui donnons est compressée (on peut le voir grâce au format .gz)

***/backupdb/dump-db_mflix-08-03-2024/db_mflix*** : ici nous lui indiquons le dossier (attention, il faut que le container puisse voir le fichier donc il faut au préalable partager un endroit entre le container et la machine host) où sont stockée les fichiers compressés des collections

## Index

Un index de type texte composé du champ sur « *title* » avec « *full plot* », un index de type unique sur l'adresse mail de l'utilisateur et un index sur le nom d'utilisateur dans un commentaire peuvent être intéressant.

Généralement, un utilisateur va chercher dans une barre de recherche, des mot clés ou directement le titre dans la barre de recherche. En mettant un index de type texte, on optimise le temps de recherche et le fait d'être composé permet de rechercher des films avec la même syntaxe.

Un mail est unique et il peut être intéressant de mettre un index de type unique car l'email va souvent être utilisé pour se connecter. Cela améliorait le temps de connexion d'un utilisateur. Même s'il y a peu de documents et que donc cela ne fait pas une grande différence, il faut prévoir dans le cas où il y a un grand nombre d'utilisateur.

Un index sur le nom d'utilisateur dans un commentaire peut être intéressant pour améliorer la rapidité à voir un profil lorsque les gens veulent en savoir plus sur la personne qui commente.

| **Collection** | **Indexe(s)** | **Requêtes** | **Avant (en ms)** | **Après (en ms)** |
|---|---|---|---|---|
| movies | Texte composé title, full plot | `use("db_mflix");`<br>`db.movies.find({ $text: { $search: "\"time travel\"" } }, { _id: 0, title: 1, fullplot: 1 });` | 34 | 14 |
| users | Unique email | `use("db_mflix");`<br>`db.users.find({email: "mark_addy@gameofthron.es"});` | 0 | 0 |
| comments | Normal name | `use("db_mflix");`<br>`db.comments.find({name: "John Bishop"}) ;` | 10 | 1 |

# CONCLUSION

## Bilan des fonctionnalités demandées

Toutes les demandes du cahier des charges ont été faites sauf 2 requêtes d'agrégation où il y a un problème. Cela concerne la requêtes 9 et 12 qui demandent de chercher les genres le plus populaire par rapport à quelque chose. Le problème étant qu'il n'affiche qu'un seul genre s'il y a plusieurs qui sont à égalité. Je ne sais pas techniquement comment faire. Je ne sais également pas le temps nécessaire que cela me prendrait à résoudre ce problème.

## Bilan personnel

Ce projet m'a permis d'approfondir mes connaissances acquises dans le module MongoDB. Il m'a fait découvrir des fonctions poussées lors des requêtes et m'a permis de bien comprendre comment utiliser MongoDB et dans quelle situation utiliser les outils fournis (export, import, restore, --archive, …).

Si le projet était à refaire, je n'aurais pas choisi Notion pour faire ma planification. J'ai eu du mal à l'utiliser et à pouvoir mettre des informations plus précises. Également, je ne sais pour quelle raison, je n'arrive pas à partager ma planification sans devoir inviter les personnes (alors que d'autre arrive juste via un lien).
Je pense également que j'aurais changé ma manière d'aborder les requêtes. Je fonce tête baissée à essayer des choses au lieu de prendre du temps pour réfléchir à comment faire.

Le projet c'est bien passé un général et je suis satisfait personnellement du travail fournit.

Tiago Rodrigues Sousa