TD/TP Bases de données avancées : SQL or not SQL

Khalil Chouikri
Polytech Marseille

Généré le 24 février 2025

1 Introduction

Ce travail est à réaliser en **Python 3**. Vous utiliserez les bibliothèques client des bases de données `MongoDB` et `SQLite3` depuis Python.

Installation des modules client

Créez un environnement virtuel, par exemple avec :

```
python3 -mvenv env
source env/bin/activate
```

Puis installez les modules avec `pip` :

```
pip install pymongo
```

ou avec `conda` :

```
conda install pymongo
```

Voir la documentation sur les environnements virtuels pour plus de détails. À l'université, vous pouvez utiliser **Anaconda** qui a déjà tout d'installé.

Objectifs

Imaginez que vous travaillez pour une entreprise dont la spécialité est d'informer le public sur le 7^{e} art. Cette entreprise a une base de données de films, d'acteurs, etc., conservée depuis longtemps dans une base relationnelle poussiéreuse. En tant que nouveau.elle CTO (*Chief Technical Officer*), vous avez l'ambition de conquérir le monde et d'offrir un service à l'échelle de la planète. Clairement, la base relationnelle ne tiendra pas le coup. Vous allez donc mettre en place une base **NoSQL** flambant neuve et gérer les petits désagréments liés à la transition entre les deux bases.

Données

Les données exploitées pour ce travail sont tirées d'`IMDB`, un site qui recense des films et épisodes de séries avec un tas d'informations associées comme la liste des acteurs. La partie publique de ces données est disponible ici : https://datasets.imdbws.com/.

Pour préparer ce TP, ces données ont été téléchargées, mises en forme par table, et filtrées selon plusieurs tailles de jeux de données, d'un ensemble jouet avec peu de films prenant une 10aine de Mo, à la totalité des données occupant environ 6Go. Le filtrage a été effectué par le nombre de votes et la nature des films, puis la base a été restreinte aux personnes ayant une connexion avec les films sélectionnés. Les différentes versions sont disponibles ici : https://pageperso.lis-lab.fr/benoit.favre/bda/.

Le jeu de données

Le jeu de données contient les tables suivantes :

— **movies** : la liste primaire des films
— **persons** : la liste primaire des personnes
— **characters** : le nom des personnages joués par les acteurs
— **directors** : les réalisateurs
— **episodes** : l'association entre épisodes, saisons et séries (seulement dans les plus gros datasets qui contiennent aussi des séries)
— **genres** : les genres des films
— **knownformovies** : les films pour lesquels une personne est connue
— **principals** : la liste des acteurs, réalisateurs, scénaristes principaux pour chaque film
— **professions** : les différentes professions de chaque personne
— **ratings** : la note moyenne et le nombre de votes de chaque film
— **titles** : les titres des films dans toutes les langues
— **writers** : les scénaristes

Les informations contenues dans ces tables peuvent être recoupées par jointure. On imagine que votre entreprise exploite ces informations depuis un logiciel client qui se connecte à la base pour récupérer ce qui l'intéresse.

2 SQLite (à faire en 2 séances)

SQLite jouera aujourd'hui le rôle de la base de données relationnelle poussiéreuse. Bien que SQLite ne soit pas une base poussiéreuse, mais plutôt moderne, ses cas d'utilisation initiaux ne permettent pas une mise à l'échelle planétaire.

2.1 Installation

Normalement, sqlite3 est déjà installé sur les machines de TP. Si vous souhaitez l'installer sur votre ordinateur personnel, vous trouverez de nombreuses ressources sur le web [1].

1. https://www.devdungeon.com/content/sqlite3-tutorial

2.2 Insertion des données

Créez une base de données SQLite locale et insérez-y les données IMDB (commencez par le plus petit des datasets). Portez une attention particulière aux points suivants :

— Ajout des contraintes sur les clés primaires

— Gestion des clés étrangères

Cette partie doit être implémentée sous forme d'un script Python utilisant l'API SQLite3 de Python [2] pour :

— Lire les fichiers CSV

— Insérer les données dans la base

2.3 Requêtes sur la base

Écrivez des requêtes SQL pour retrouver les informations suivantes :

1. Dans quels films a joué Jean Reno ?
2. Quels sont les trois meilleurs films d'horreur des années 2000 au sens de la note moyenne par les utilisateurs ?
3. Quels sont les scénaristes qui ont écrit des films jamais diffusés en Espagne (région ES dans la table titles) ?
4. Quels acteurs ont joué le plus de rôles différents dans un même film ?
5. Quelles personnes ont vu leur carrière propulsée par Avatar (quel que soit leur métier), et alors qu'elles n'étaient pas connues avant (ayant seulement participé à des films avec moins de 200000 votes avant avatar), elles sont apparues par la suite dans un film à succès (nombre de votes $>$ 200000) ?
   — Note : les films ne sont pas ordonnés par date de sortie
   — On utilisera l'année de sortie du film (strictement avant / après celle d'Avatar)

Vous pouvez utiliser Python et afficher les résultats dans le terminal. Pour bien vous familiariser avec la base, imaginez quelques autres requêtes mettant en œuvre une ou plusieurs jointures sur l'ensemble des tables de la base.

2.4 Performances

Lorsque l'on utilise une base suffisamment grande, les requêtes avec plusieurs jointures peuvent prendre beaucoup de temps. Pour analyser les performances :

— Chronométrez la durée d'exécution des requêtes avec le module Python `time`

— Évaluez si toutes les requêtes bénéficient des index

— Créez des index dans SQLite pour accélérer le traitement des requêtes précédentes

— Notez :
   — Le temps de traitement en fonction des index créés
   — La différence de taille de la base selon les index créés

2. https://docs.python.org/3/library/sqlite3.html

3 MongoDB (à faire en 4 séances)

MongoDB sera la base choisie pour implémenter votre plan de domination du monde. Il sera nécessaire de migrer les données de la base SQLite vers MongoDB.

3.1 Installation

Téléchargez MongoDB[3] et dézippez l'archive dans votre répertoire personnel. Le répertoire `bin` contient plusieurs exécutables :

— `mongo` : le shell client pour interagir avec la base en ligne de commande

— `mongod` : le serveur hébergeant les données

— `mongos` : le point d'accès pour les configurations multi-serveurs

Pour démarrer :

1. Créez un répertoire `"mongo-db"`
2. Lancez le serveur : `./mongod -dbpath ./mongo-db/`
3. Le serveur écoute par défaut sur le port 27017
4. Consultez les logs du serveur
5. Dans un autre terminal, lancez le client : `./mongo mongodb://127.0.0.1:27017`

Pour tester, depuis le client, vous pouvez ajouter une valeur à la base :

```
> db.collection.insert({a:3})
WriteResult({ "nInserted" : 1 })
> db.collection.find()
{ "_id" : ObjectId("622b8742370c3c81da1f89b5"), "a" : 3 }
```

3.2 Insertion de données plates

Pour cette partie, vous utiliserez le client Python. Procédez comme suit :

— Reproduisez la base SQL dans MongoDB, chaque table devenant une collection

— Alimentez la base MongoDB via des requêtes SQL sur la base SQLite

— N'utilisez pas les fichiers CSV directement

Recréez les requêtes précédentes avec MongoDB et des scripts Python. Analysez :

— Les performances

— La répartition du code entre l'application et la base de données

3.3 Insertion de données structurées

MongoDB excelle dans la gestion des données structurées. Imaginons le scénario suivant : vous souhaitez créer un site web similaire à IMDB affichant pour chaque film toutes ses informations :

3. https://www.mongodb.com/try/download/community, version courante, plateforme Debian10 (ou selon), package tgz. Une alternative est d'utiliser une BD gratuite sur MongoDB Atlas, mais elle sera limitée en taille. Après avoir créé un compte, vous aurez accès à une URL MongoDB à laquelle vous pouvez connecter le client.

— Liste des acteurs et autres personnels
— Titres dans toutes les langues
— Épisodes pour les séries
— Autres métadonnées pertinentes

Tâches à réaliser :

1. Créez une nouvelle collection MongoDB
2. Pour chaque film, créez un objet JSON contenant toutes les informations citées
3. Utilisez uniquement des requêtes MongoDB
4. Comparez les performances :
   — Temps de récupération avec la nouvelle structure
   — Temps avec des requêtes simulant des jointures entre tables

3.4 Résistance aux pannes

Configurez un système tolérant aux pannes :

1. Lancez deux instances MongoDB configurées en "Replicate Set" [4]
2. Testez la tolérance aux pannes en arrêtant une instance
3. Extension possible : collaborez avec d'autres étudiants sur le réseau local pour construire un système MongoDB distribué

3.5 Synchronisation bidirectionnelle

Pendant la période de transition, on souhaite synchroniser les deux bases de données :

— Objectif : maintenir SQLite et MongoDB synchronisés pour les clients n'ayant pas migré
— Exigences :
   — Synchronisation des insertions et mises à jour entre les deux bases
   — Les principes ACID ne sont pas requis (MongoDB ne les garantit pas)

3.6 Contraintes et mise en œuvre

Vous devrez respecter les contraintes suivantes :

— Utiliser uniquement les collections correspondant aux tables de la base SQL

3.7 Mécanisme de synchronisation

La synchronisation entre les bases sera basée sur des triggers (fonctions déclenchées lors d'événements).

4. https://docs.mongodb.com/manual/sharding/

3.7.1 Côté SQLite

Le langage SQLite ne permet pas nativement d'alimenter une base externe depuis un trigger. Pour contourner cette limitation :

- Construire une extension Python pour gérer les modifications
- Utiliser `sqlite3_update_hook`[5] pour :
  - Appeler du code Python à chaque modification de la base
  - Utiliser la librairie `ctypes`
- Configuration requise : définir le chemin d'accès à la librairie `libsqlite3`[6]

3.7.2 Côté MongoDB

Pour la propagation des modifications :

- Utiliser un ChangeStream[7]
- Objectif : récupérer et propager les modifications vers SQLite

3.8 Extension optionnelle : Interface Web

Vous pouvez présenter les informations via une interface web en utilisant l'un des frameworks suivants :

- Bottle[8] (recommandé pour les débutants)
- Django[9] (pour une approche plus technique)

Fonctionnalité supplémentaire : Vous pouvez enrichir l'interface en récupérant les posters des films via leur identifiant IMDB grâce à l'API fournie par http://www.omdbapi.com.

4 Conclusion

Félicitations, vous êtes maintenant prêt.e à conquérir le monde à l'aide de vos compétences sur les bases de données !

5. https://stackoverflow.com/questions/16872700/sqlite-data-change-notification-callbacks-in-pytho
6. Par exemple : `dll = CDLL('libsqlite3.so')`
7. https://docs.mongodb.com/manual/changeStreams/, https://www.mongodb.com/developer/quickstart/python-change-streams/
8. https://bottlepy.org
9. https://www.djangoproject.com/