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

Benoit Favre
Aix-Marseille Université, Polytech – 4a BDA

généré le 31 janvier 2024

1 Introduction

Note importante : si vous travaillez sur les machines de l'université, vous êtes soumis.e à un quota de disque. Si vous avez des fichiers corrompus ou vides à la suite d'un téléchargement ou d'une décompression, c'est qu'il est temps de faire un peu de ménage. Vous pouvez utiliser l'espace disque dans le répertoire suivant `/qamu/home/<login>`. Il faut explicitement aller dans ce répertoire avec `cd` pour qu'il soit automonté.

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 environnent virtuel, par exemple avec "`python3 -mvenv env`" puis "`source env/bin/activate`", puis installez les modules avec pip : "`pip install pymongo`" ou "`conda install pymongo`" avec conda. Voir la doc 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 7e art. Cette entreprise a une base de données de films, d'acteurs, etc., conservée depuis longtemps dans une base relationnelle poussiéreuse. En temps 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 [1].

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

1. `https://pageperso.lis-lab.fr/benoit.favre/bda/`

— 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)

Aujourd'hui, le rôle de la base de données relationnelle poussiéreuse sera joué par SQLite. Ça n'est pas une base poussiéreuse, bien au contraire, mais les cas d'utilisation pour lesquels elle a été conçue ne permettent pas de passer à l'échelle au niveau planétaire.

Installation Normalement, sqlite3 est déjà installé sur sur les machines de TP, et si vous voulez l'installer sur votre ordinateur, vous trouverez plein d'informations sur le web [2].

Insertion des données Créez une base de données SQLite locale et insérez-y les données IMDB (commencez par le plus petit des datasets). Faites attention à ajouter les contraintes sur les clés primaires et clés étrangères. Vous devez implémenter cette partie sous la forme d'un script Python qui utilise l'API SQLite3 de Python [3] pour lire les CSV et insérer les données.

Quelques requêtes sur la base Écrire 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) ? Notez que les films ne sont pas ordonnés par date de sortie, et donc 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 oeuvre une ou plusieurs jointure sur l'ensemble des tables de la base.

Performances Lorsque l'on utilise une base suffisamment grande, les requêtes avec plusieurs jointures peuvent durer longtemps. Chronométrez la durée d'exécution des requêtes avec le module Python `time`. Toutes les requêtes bénéficient-elles des index ? Créez des index dans SQLite pour accélérer le traitement des requêtes effectuées précédemment. Notez le temps qu'il faut pour les traiter en fonction des index que vous créez, et mesurez la différence de taille de la base en fonction des index créés.

2. Par exemple `https://www.devdungeon.com/content/sqlite3-tutorial`
3. `https://docs.python.org/3/library/sqlite3.html`

3 MongoDB (à faire en 4 séances)

La superstar du jour c'est MongoDB. C'est la base que vous avez choisie pour implémenter votre plan de domination du monde. Il va falloir copier les données de la base SQLite vers la base MongoDB.

Installation Téléchargez MongoDB[4] et dézippez l'archive dans votre répertoire personnel. Il y a plusieurs exécutables dans le répertoire bin :

— mongo : le shell client qui permet d'interagir avec la base en ligne de commande
— mongod : le serveur qui héberge des données
— mongos : le point d'accès lorsque l'on a plusieurs serveurs

Créez un répertoire "mongo-db" et lancez le serveur "`./mongod --dbpath ./mongo-db/`". Par défaut, il attend sur le port 27017. Lisez un peu ce qu'il raconte dans ses logs et lancez un client dans un autre terminal : "`./mongo mongodb://127.0.0.1:27017`".

Pour tester, depuis le client, vous pouvez ajouter une valeur à la base (ici, la collection s'appelle collection, ce qui n'est pas très original) :

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

Insertion de données plates Pour la suite, vous exploiterez le client Python. Vous allez commencer par reproduire à l'identique la base SQL dans MongoDB, chaque table devenant une collection. Alimentez la base MongoDB à l'aide de requêtes SQL sur la base SQLite. Vous ne devez pas utiliser les fichiers CSV.

Recréez à l'aide de requêtes MongoDB et de scripts Python les requêtes ci-avant. Qu'observez-vous en termes de performances, et de répartition du code entre l'application et la base de données ?

Insertion de données structurées Vous le savez, la vrai puissance des bases comme MongoDB réside dans la possibilité d'insérer et de requêter des données structurées. Dans cette partie vous allez imaginer un scénario : vous voulez proposer à vos utilisateurs un site web comme celui d'IMDB qui affiche pour chaque film toutes les informations qui le concernent : liste des acteurs et autres personnels qui y ont participé, titre dans toutes les langues, épisodes pour les séries, etc.

Créez une nouvelle collection MongoDB avec pour chaque film un objet JSON contenant toutes les informations citées ci-dessus. Vous devez le faire à partir de requêtes MongoDB. Comparez le temps qu'il faut pour récupérer un film de cette manière par rapport à des requêtes simulant des jointures entre les tables.

Résistance aux pannes Lancez deux instances de mongodb configurées comme un "Replicate Set"[5]. Vérifiez que vous pouvez en tuer l'une des deux sans perte de service. Une extension sympathique est de vous arranger avec les autres élèves sur le même réseau local pour co-construire un système distribué MongoDB.

Synchronisation bidirectionnelle Lors d'une période de transition, on souhaite synchroniser les deux bases de données, SQLite et MongoDB, car des clients n'ont pas encore fait la mise à jour et veulent pouvoir accéder à l'ancienne base. À chaque insertion ou mise à jour de l'une des deux bases, on veut que la seconde soit mise à jour. On ne nécessitera pas que les principes ACID soient garantis car MongoDB ne les garantis de toute façon pas.

4. `https://www.mongodb.com/try/download/community`, version courante, plateforme Debian10 (ou selon), package tgz. Une autre option 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.

5. `https://docs.mongodb.com/manual/sharding/`

Vous utiliserez uniquement les collections qui correspondent aux tables de la base SQL.

Pour synchroniser les deux bases, vous utiliserez des triggers, c'est à dire des fonctions déclenchées lors d'un événement.

Dans le langage de SQLite, il n'y a pas de fonction permettant directement d'alimenter une base externe depuis un trigger. Il faut donc construire une extension Python qui se chargera à chaque modification, de l'impacter dans la base MongoDB. Vous utiliserez `sqlite3_update_hook` [6], qui permet d'appeler du code Python à chaque modification de la base grâce à la librairie ctypes. Notez qu'il faut configurer le chemin d'accès à la librairie libsqlite3 [7].

Côté MongoDB, vous utiliserez un ChangeStream [8] pour récupérer les modifications et les propager à SQLite.

Optionnel : présentation sous la forme d'un site web Présentez les informations sous la forme d'un site web, à l'aide d'un framework comme Bottle [9] (facile) ou Django [10] (plus technique).

Vous pouvez par exemple récupérer le poster d'un films à partir de son identifiant IMDB avec l'API proposée par `http://www.omdbapi.com`.

4 Conclusion

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

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