Saé

Présentation

Le but de la Saé est de développer une application en langage Python exploitant une base de données regroupant des informations sur les restaurants de New York. Cette application permet de localiser les 3 restaurants les plus proches de l'utilisateur.
Pour cela, l'application prend en entrée des variables qui seront placées au début du programme :

- les coordonnées de l'utilisateur
  Par exemple : `latitude_client, longitude_client = 40.870, -73.751`
- le type de cuisine ou une chaîne vide si aucune préférence n'est précisée
  Exemple : `type_cuisine = 'Mediterranean'` ou `''`

Les coordonnées de la collection restaurants (et celles de l'utilisateur) sont données en degrés décimaux (decimal degrees) dans l'ordre: longitude,latitude.
Les coordonnées de l'utilisateur sont dans les intervalles : latitude Sud Nord : [40.50 , 41.20] et longitude Ouest Est : [-74.26, -73.20]

Une interface graphique n'est pas demandée et les résultats seront affichées en console.
Exemple d'affichage :

```
(base) C:\IUT\SD\SAe>python SAe.py
Connexion reussie a MongoDB
Affichage a partir de restaurants                                                :
distance : 3.6089962918630416 km name : Jp S Waterside Restaurant, cuisine : Seafood
distance : 3.68097278765676 km name : City Island Lobster House, cuisine : Seafood
distance : 3.7436714752169276 km name : Sugar And Spice, cuisine : American
time0.49535369999999995

(base) C:\IUT\SD\SAe>python SAe.py
Connexion reussie a MongoDB
Affichage a partir de rhistory
distance : 3.6089962918630416 km name : Jp S Waterside Restaurant, cuisine : Seafood
distance : 3.68097278765676 km name : City Island Lobster House, cuisine : Seafood
distance : 3.7436714752169276 km name : Sugar And Spice, cuisine : American
time0.007154300000000002
```

Le premier affichage utilise les données de MongoDB alors que le second a utilisé les données du cache. Le temps d'exécution est fourni à titre de comparaison et permet de vérifier l'efficacité du cache.

Les délivrables de la SAé sont :

- 2 fichiers Python : conversion.py, sae.py
- un document de présentation de votre projet

Etape1 : le catalogue de données

Les données des restaurants forment un grand catalogue. Les catalogues de données sont gérés efficacement dans une base de données orientée documents de type MongoDB. Malheureusement, les données sont actuellement dans une base relationnelle disséminées dans 3 tables :

- sql_main contient pour chaque restaurant : restaurant_id, nom (name), type_de_cuisine (cuisine), quartier (borough)
- sql_geo contient pour chaque restaurant : restaurant_id, le document adresse au format jsonb
- sql_feedback contient pour chaque restaurant : restaurant_id, les notes (grades) attribuées par les utilisateurs au format jsonb

Votre point de départ est le fichier insert_queries.sql qui permet de créer et remplir les 3 tables sql des restaurants dans PostgreSQL. Vous devrez regrouper les données relationnelles d'un restaurant dans un même document au format json puis l'importer dans MongoDB. L'étape 1 est donc une étape de migration d'une base relationnelle (PostgreSQL) vers une base NoSQL (MongoDB).

Le catalogue est volumineux. Ainsi, vous développerez un programme de migration dans un fichier Python appelé conversion.py. Vous fournirez ce fichier dans le rendu du projet.

Exemple d'une migration SQL vers NoSQL pour un restaurant :
Les données pour le restaurant_id 50018995 apparaissent dans les tables relationnelles :

```
sae=# select * from sql_main where restaurant_id = 50018995;
 restaurant_id |     name     | cuisine | borough
---------------+--------------+---------+----------
      50018995 | Cold Press D | Other   | Brooklyn
```

```
sae=# select * from sql_geo where restaurant_id = 50018995;
 restaurant_id |                                                   address

--------------+------------------------------------------------------------------------------------------------------------
-------------------------------
      50018995 | {"coord": {"type": "Point", "coordinates": [-73.9691347, 40.6389857]}, "street": "Cortelyou Rd", "zipco
de": "11218", "building": "921"}
```

```
sae=# select * from sql_feedback where restaurant_id = 50018995;
 restaurant_id | grades
---------------+--------
      50018995 | []
```

Et votre programme conversion.py créera un document unique qui sera enfin importé dans MongoDB. Par exemple :

```
{"address": {"building": "921", "coord":{"type":"Point", "coordinates" : [-73.9691347, 40.6389857]},
"street": "Cortelyou Rd", "zipcode": "11218"}, "borough": "Brooklyn", "cuisine": "Other", "grades": [],
"name": "Cold Press D", "restaurant_id": "50018995"}
```

On rappelle que l'ordre des champs dans un document n'a pas d'importance.

Dans MongoDB, la collection des documents s'appellera restaurants.

Remarques : dans votre programme conversion.py, vous pouvez travailler à partir du fichier insert_queries.sql ou à partir d'une base PostgreSQL dans laquelle vous aurez importé le fichier insert_queries.sql.

Etape 2 : l'exploitation des documents

Dans cette étape, vous coderez une première version du programme principal dans un fichier nommé sae.py. Celui-ci effectue les actions suivantes :

- se connecte à MongoDB
- parcourt la collection restaurants de la base MongoDB
- calcule la distance entre les coordonnées utilisateurs et les coordonnées de chaque restaurant
- vérifie également si un type de cuisine particulier est demandé. Si c'est le cas, les réponses retournées ne concernent que le type demandé
- liste les restaurants les plus proches dans l'ordre croissant puis affiche : la distance d'éloignement (en km), le nom et le type de cuisine pour chaque restaurant
- affiche le temps de traitement en s

Le programme traitera les cas d'erreur :

- base MongoDB non accessible (erreur d'adresse, de port). Le programme affiche un message d'erreur.
- les coordonnées sont en dehors des intervalles latitude Sud Nord : [40.50 , 41.20] et longitude Ouest Est : [-74.26, -73.20]. Le programme affiche un message d'erreur et ne parcourt pas la base de données
- une erreur concerne le type de cuisine mal orthographié dans la base alors le programme retourne les 3 restaurants les plus proches sans préférence de cuisine

Etape 3 : la mise en cache des documents

Les performances peuvent être améliorées à l'aide d'un cache. Un cache est traditionnellement sauvegardé dans une base clé:valeur. Nous utiliserons PostgreSQL pour stoker les données dans une table PostgreSQL appelée cache.

La table « cache » conserve au maximum les 20 dernières « questions, réponses » de la base MongoDB.

Vous définirez un format de sauvegarde efficace pour la table cache.
Vous modifierez également le code du fichier sae.py pour :

- interroger d'abord le cache et vérifier si la donnée recherchée existe. Une tolérance de +/-0,001 sur latitude ou longitude sera acceptée. Si la donnée existe (on parle de cache hit), le programme renvoie la valeur correspondant de cache
- après interrogation de cache et si la donnée n'existe pas dans cache (on parle de cache miss), votre programme se connecte à MongoDB et effectue les mêmes tâches qu'à l'étape 2.
- Avant d'afficher le temps d'exécution, votre programme mettra à jour le cache : il vérifiera que la table ne contient pas plus de 19 lignes avant d'insérer la dernière réponse. Si la table contient 20 lignes alors elle sera entièrement nettoyée et la dernière réponse sera la nouvelle première ligne.

Etape 4 : présentation

Vous rédigerez un document présentant le travail effectué : introduction, la base relationnelle et les données initiales, les deux bases NoSQL et le formatage des données des documents sous MongoDB et du cache sous PostgreSQL, l'application conversion.py, l'application finale sae.py, les performances du cache, une conclusion

Le document rédigé sera utilisé pour présenter vos résultats pendant 15 minutes au maximum pour un groupe de deux personnes (ou 10 minutes max pour une personne seule) sur la dernière séance. La présentation sera suivie de 10 minutes de questions réponses.

Planning et évaluation

21/11/25 4h30
05/12/25 4h30
19/12/25 4h30
09/01/26 4h30
Une demi journée de 3h est réservée pour les présentations des groupes.

Chaque groupe dispose de 12h pour coder les deux programmes demandés et rédiger une présentation.

Le travail demandé sera effectué en autonomie pour chaque groupe.

Il s'appuie sur :

- les connaissances et outils étudiés dans le cours et en TD
- les drivers psycopg2 et pymongo utilisés dans les Tds pour communiquer avec PostgreSQL et MongoDB à partir d'un programme Python

L'évaluation repose sur :

- la qualité des deux programmes Python
- la conformité des deux programmes Python au cahier des charges
- la qualité du document écrit
- la clarté de la présentation orale
- la capacité à répondre aux questions