TP N° 6 : Bases de données NoSQL « Cassandra »

Les exercices qui suivent sont à effectuer sur machine, avec Cassandra.

- ✓ Installation de Cassandra sous Windows
- ✓ Si vous êtes à l'aise en ligne de commande, vous pourrez également accéder à Cassandra avec `cqlsh` qui se lance avec la commande suivante :

```
sudo docker exec -it mon-cassandra cqlsh
# ceci suppose que mon-cassandra est le nom de votre container
# Option -it pour disposer d'un terminal interactif persistant
# cqlsh pour lancer cette commande au démarrage
```

Le sujet des travaux pratiques est la mise en place d'une base de données représentant des restaurants, et des inspections de ces restaurants. Un échantillon de données est disponible:

- La base Restaurants au format csv (archive ZIP)

Note

Avant de vous lancer dans le travail proprement dit, vous êtes invités fortement à prendre le temps d'ouvrir cette archive zip et d'en examiner le contenu (au moins les en-têtes, pour avoir une première idée de la structure des données initiales).

Bien entendu, on supppose qu'à terme cette base contiendra tous les restaurants du monde, et toutes les inspections, ce qui justifie d'utiliser un système apte à gérer de grosses volumétries.

Partie 1: Approche relationnelle

Nous allons étudier ici la création d'une base de données (appelée **Keyspace**), puis son interrogation. Cette première phase du TP consiste à créer la base comme si elle était relationnelle, et à effectuer des requêtes simples. Une fois les limites atteintes, nous utiliserons les spécificités de Cassandra pour aller plus loin.

Création de la base de données

Avant d'interroger la base de données, il nous la créer. Pour commencer :

```
CREATE KEYSPACE IF NOT EXISTS resto_NY
   WITH REPLICATION = { 'class' : 'SimpleStrategy',
'replication_factor': 1};
```

Nous créons ainsi une base de données *resto_NY* pour laquelle le facteur de réplication est mis à 1, ce qui suffit dans un cadre centralisé.
Sous `csqlsh`, vous pouvez maintenant sélectionner la base de données pour vos prochaines requêtes.

```
USE resto_NY;
```

Bien entendu vous pouvez exécuter ces commandes via DbVisualizer.

Tables

Nous pouvons maintenant créer les tables (*Column Family* pour Cassandra) *Restaurant* et *Inspection* à partir du schéma suivant :

```
 CREATE TABLE Restaurant (
   id INT, Name VARCHAR, borough VARCHAR, BuildingNum VARCHAR, Street
VARCHAR,
   ZipCode INT, Phone text, CuisineType VARCHAR,
   PRIMARY KEY ( id )
 ) ;

 CREATE INDEX fk_Restaurant_cuisine ON Restaurant ( CuisineType ) ;

 CREATE TABLE Inspection (
   idRestaurant INT, InspectionDate date, ViolationCode VARCHAR,
   ViolationDescription VARCHAR, CriticalFlag VARCHAR, Score INT,
GRADE VARCHAR,
   PRIMARY KEY ( idRestaurant, InspectionDate )
 ) ;

CREATE INDEX fk_Inspection_Restaurant ON Inspection ( Grade ) ;
```

Nous pouvons remarquer que chaque inspection est liée à un restaurant via l'identifiant de ce dernier.

Pour vérifier si les tables ont bien été créées (sous `cqlsh`).

```
DESC Restaurant;
DESC Inspection;
```

Nous pouvons voir le schéma des deux tables mais également des informations relatives au stockage dans la base *Cassandra*.

Import des données

Maintenant, nous pouvons importer les fichiers CSV pour remplir les *Column Family* :

1. Décompresser le fichier "restaurants.zip" (il contient le fichier "restaurants.csv" et "restaurants_inspections.csv")

   Note

   En mode console, sur le répertoire de téléchargement du fichier *restaurants.zip*, il suffit de mettre la commande :

   ```
   unzip restaurants.zip
   ```

2. Importer un fichier CSV :

   - Dans votre console (machine locale, pas docker), copier les fichiers sous « **Docker** » (container "Cassandra")
     - `docker cp path-to-file/restaurants.csv docker-container-ID:/`
     - `docker cp path-to-file/restaurants_inspections.csv docker-container-ID:/`

     Note

     Le chemin « *path-to-file* » correspond à l'endroit où a été décompressé le fichier restaurants.zip
     le docker-container-ID peut être récupéré grâce à la commande

     ```
     docker ps
     ```

```
CONTAINER ID        IMAGE
COMMAND              CREATED             STATUS
NAMES
b1fa2c7c255d        poklet/cassandra:latest
"/bin/sh -c start"   6 minutes ago       Up 6
minutes
```

Ici, le container-ID est donc *b1fa2c7c255d*

3. Dans la console *cqlsh*, importer les fichiers "**restaurants.csv**" et "**restaurants_inspections.csv**"

```
use resto_NY ;
COPY Restaurant (id, name, borough, buildingnum,
street,
                    zipcode, phone, cuisinetype)
   FROM '/restaurants.csv' WITH DELIMITER=',';
COPY Inspection (idrestaurant, inspectiondate,
violationcode,
                    violationdescription, criticalflag,
score, grade)
   FROM '/restaurants_inspections.csv' WITH
DELIMITER=',';
```

Note

Les fichiers sont copiés à la racine du container. Si vous changez le dossier de stockage, il faut bien sûr l'indiquer dans l'instruction précédente.

Vous pouvez vérifier l'existence des fichiers dans le container avec :

```
ls /*.csv
```

Pour vérifier le contenu des tables:

```
SELECT count(*) FROM Restaurant;
SELECT count(*) FROM Inspection;
```

Ce qui devrait vous indiquer environ 25 000 restaurants et 150 000 inspections. Nous sommes prêts!

Interrogation

Les requêtes qui suivent sont à exprimer avec **CQL** (pour *Cassandra Query Language*) qui est fortement inspirée de SQL. Vous trouverez la syntaxe complète ici :
<https://cassandra.apache.org/doc/latest/cql/dml.html#select>).
Notez que certaines requêtes seront refusées par CQL. Essayez de comprendre pourquoi, et trouvez un contournement. Il y en a essentiellement deux: créer un index ou utiliser `ALLOW FILTERING`.

Requêtes CQL simples

Pour la suite des exercices, exprimer en *CQL* les requêtes suivantes :

1. Liste de tous les restaurants
2. Liste des noms de restaurants
3. Nom et quartier (*borough*) du restaurant dont l'id est 41569764
4. Dates et grades des inspections de ce restaurant
5. Noms des restaurants de cuisine Française (*French*)

6. Noms des restaurants situés dans *BROOKLYN* (attribut *borough*; si vous recevez une erreur en retour notez-la bien)
7. Grades et scores donnés pour une inspection pour le restaurant n° 41569764 avec un score d'au moins 10
8. Grades (non nuls) des inspections dont le score est supérieur à 30
9. Nombre de lignes retournées par la requête précédente