Cassandra - Travaux Pratiques[1]

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

Après avoir lancé votre machine Cassandra (avec docker), vous aurez besoin d'une interface cliente pour y accéder. Pour cela, nous utiliserons **DevCenter** de *DataStax*:
DevCenter : <https://academy.datastax.com/downloads/ops-center#devCenter>

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

Partie 1: Approche relationnelle

Nous allons étudier ici la création d'une base de données (appelée **Keyspace**), puis son interrogation.

Création de la base de données

```
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 sélectionner la base de données pour vos prochaines requêtes.

```
USE resto_NY;
```

L'équivalent existe dans une interface graphique bien entendu.

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.

---

[1] Référence : support de P.Rigaux

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 *ColumnFamily* :

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                              PORTS
  NAMES b1fa2c7c255d        poklet/cassandra:latest   "/bin/sh -c
  start"   6 minutes ago       Up 6 minutes        0.0.0.0:32787-
  >22/tcp,    0.0.0.0:32786->7000/tcp,    0.0.0.0:32785->7001/tcp,
  0.0.0.0:32784->7199/tcp,                0.0.0.0:32783->8012/tcp,
  0.0.0.0:32782->9042/tcp,                0.0.0.0:32781->9160/tcp,
  0.0.0.0:32780->61621/tcp   cassandra
  ```

  le container-ID est : *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 le changez il faut l'impacter dans l'instruction précédente.

Pour vérifier le contenu des tables:

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

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

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>).

Requêtes CQL simples

Pour la suite, 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 N° 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*).
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.
10. Grades des inspections dont l'identifiant est compris entre 40 000 000 et 40 000 100. Aide: Utiliser la fonction '*token()*'.
11. Compter le nombre de lignes retournées par la requête précédente.

    Aide: Utiliser '*COUNT(*)*'

CQL Avancé

1. Pour la requête ci-dessous faites en sorte qu'elle soit exécutable sans *ALLOW FILTERING*.

```
SELECT Name FROM Restaurant WHERE borough='BROOKLYN' ;
```

2. Utilisons les deux indexes sur *Restaurant* (*borough* et *cuisineType*). Trouvez tous les noms de restaurants français de Brooklyn.
3. Utiliser la commande *TRACING ON* avant la d'exécuter à nouveau la requête pour identifier quel index a été utilisé.
4. On veut les noms des restaurants ayant au moins un grade 'A' dans leurs inspections. Est-ce possible en CQL?