TP - Gestion des données dans des bases NoSQL orientées Documents ou Colonnes

Introduction

Ce rapport portant sur la gestion des données dans les bases de données NoSQL. Le travail est divisé en deux parties : la première sur MongoDB (base orientée documents) et la seconde sur Apache Cassandra (base orientée colonnes).

1 Partie 1 – Base orientée documents : MongoDB

Commande d'importation

```
mongoimport --db tp --collection restaurants --drop --file restaurants.
    json
```

Question 1 : Comparaison SQL / MongoDB

— En SQL : tables relationnelles avec des colonnes fixes.
— En MongoDB : documents imbriqués dans une structure JSON souple.

```
{
  "address": {
    "building": "1007",
    "coord": [-73.856077, 40.848447],
    "street": "Morris Park Ave",
    "zipcode": "10462"
  },
  "borough": "Bronx",
  "cuisine": "Bakery",
  "grades": [
    {
      "date": {"$date": "2014-03-03T00:00:00Z"},
      "grade": "A",
      "score": 2
    }
  ],
  "name": "Morris Park Bake Shop",
  "restaurant_id": "30075445"
}
```

Question 2

```
db.restaurants.find()
```

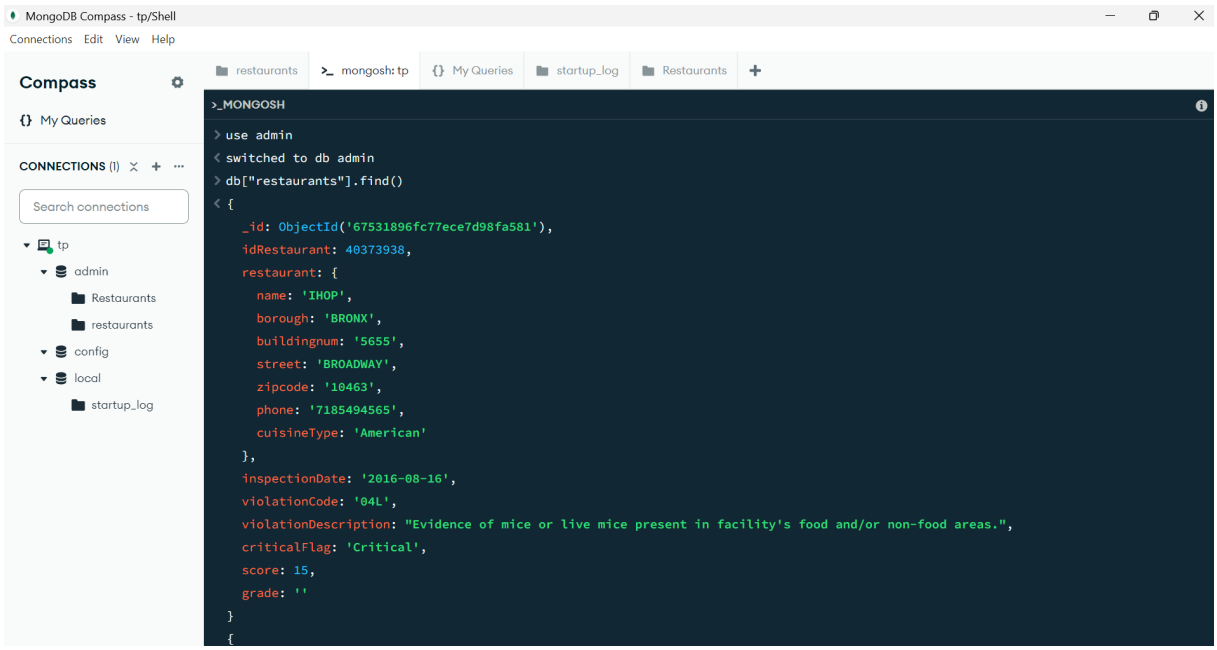

FIGURE 1 – Visualisation MongoDB shell

Question 3

```
db.restaurants.find().sort({ borough: 1, name: 1 })
```

Question 4

```
db.restaurants.aggregate([
  { $match: { name: "Morris Park Bake Shop" } },
  { $unwind: "$grades" },
  { $project: { _id: 0, name: 1, grade: "$grades.grade", score: "
     $grades.score", date: "$grades.date" } }
])
```

Question 5

```
db.restaurants.aggregate([
  { $match: { name: "Morris Park Bake Shop" } },
  { $unwind: "$grades" },
  {
    $group: {
      _id: "$name",
      moyenne_score: { $avg: "$grades.score" }
    }
```

```
  }
])
```

Question 6

```
db.restaurants.createIndex({ "address.coord": "2dsphere" })

db.restaurants.find({
  "address.coord": {
    $geoWithin: {
      $centerSphere: [[-73.93414657, 40.82302903], 1 / 6378.1]
    }
  }
})
```

Question 7

```
db.restaurants.find({ "grades.score": { $lt: 3 } })
```

Question 8

```
db.restaurants.insertOne({
  name: "Kebab d goulinant",
  address: {
    street: "Rue de Paris",
    zipcode: "94400",
    building: "",
    coord: []
  },
  borough: "Perdu",
  cuisine: "Turquie",
  phone: "+33123456789",
  grades: []
})
```

FIGURE 2 – Visualisation MongoDB shell

Question 9

```
db.restaurants.updateOne(
  { restaurant_id: "30099999" },
  {
    $addToSet: {
      grades: {
        date: ISODate("2014-10-01T00:00:00Z"),
        grade: "A",
        score: 11
      }
    }
  }
)
```

2 Partie 2 – Base orientée colonnes : Apache Cassandra

Étape 1 : Installation de Cassandra avec Docker

```
sudo docker pull cassandra
sudo docker run --name cassandra-node -d -p 9042:9042 cassandra
sudo docker ps
```

```
sudo docker exec -it cassandra-node cqlsh
```

Vérification de l'installation

```
SELECT release_version FROM system.local;
```

Résultat attendu : un numéro de version (ex : 4.0.1). Par défaut, les données sont stockées dans `/var/lib/cassandra` du conteneur Docker.

Configurer un volume persistant

```
docker volume create cassandra_data
docker run --name cassandra-node \
  -v cassandra_data:/var/lib/cassandra \
  -d -p 9042:9042 cassandra
```

Étape 2 – Exercice 1 : Métro parisien

Création du Keyspace et des tables

```
CREATE KEYSPACE IF NOT EXISTS Metros
WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor': 3
    };

CREATE TABLE lines(
  color text, name text, number text, route_name text,
  PRIMARY KEY(color)
);

CREATE TYPE line(line text, position int);

CREATE TABLE stops(
  description text,
  latitude float,
  lines set<frozen<line>>,
  longitude float,
  name text,
  PRIMARY KEY(description)
);
```

Vérification de la création des tables

```
DESCRIBE TABLE lines;
DESCRIBE TABLE stops;
```

Insertion de données JSON

```
INSERT INTO lines JSON '{
  "color": "#F58F53",
  "name": "Tramway 3A",
  "number": "3A",
  "route_name": "PONT GARIGLIANO - HOP G.POMPIDOU <-> PORTE DE VINCENNES
     "
}';

INSERT INTO stops JSON '{
  "description": "Jean Jaur s (23 boulevard) - 92012",
  "latitude": 48.84228,
  "lines": [{"line": "ligne-10", "position": 2}],
  "longitude": 2.238863,
  "name": "Boulogne-Jean-Jaur s"
}';
```

Recherche HTML via Python

Utiliser la librairie `cassandra-driver` pour exécuter les requêtes et générer des tableaux HTML. Créer des fonctions réutilisables pour éviter la redondance de code.

Exercice 2 – Base Restaurants avec fichiers CSV

Création du Keyspace `resto_NY`

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

Création des tables Restaurant et Inspection

```
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);
```

À chaque inspection

Chaque inspection crée une nouvelle entrée dans la table Inspection avec un identifiant de restaurant et une date.

Vérification des tables

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

Import des données depuis CSV

```
docker cp restaurants.csv cassandra-node:/restaurants.csv
docker cp restaurants_inspections.csv cassandra-node:/
   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=',';
```

Vérification présence des fichiers

```
docker exec -it cassandra-node ls /
```

Traitement des fichiers

Ils peuvent être supprimés pour libérer de l'espace ou déplacés dans un volume persistant.

Vérification du contenu

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

Requêtes CQL classiques

```
SELECT * FROM Restaurant;
SELECT name FROM Restaurant;
SELECT name, borough FROM Restaurant WHERE id = 41569764;
SELECT inspectiondate, grade FROM Inspection WHERE idRestaurant =
   41569764;
SELECT name FROM Restaurant WHERE CuisineType = 'French';
-- Erreur possible si pas d'index :
```

```
SELECT name FROM Restaurant WHERE borough = 'BROOKLYN';
SELECT grade, score FROM Inspection WHERE idRestaurant = 41569764 AND
    score >= 10;
-- Si pas d'index sur score, cette requ te peut   chouer   :
-- SELECT grade FROM Inspection WHERE score > 30;
```

Étape 4 – Cluster multi-nœuds Cassandra

```
sudo docker network create cassandra-cluster
sudo docker run --name cassandra-neoudParis --network cassandra-cluster
    -d cassandra
sudo docker run --name cassandra-neoudLyon --network cassandra-cluster -
    d cassandra
```

Importer la base créée : utiliser un volume persistant partagé ou synchroniser manuellement les répertoires `/var/lib/cassandra`.

Annexe A : Installation Docker si non disponible dans les dépôts

```
sudo apt install apt-transport-https ca-certificates curl software-
    properties-common -y
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
  sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io -y
```