Exercice 1 – Réparation automobile

Question 1
Créez un keyspace garage avec la stratégie de réplication SimpleStrategy et un facteur de réplication de 1 (faire help CREATE_KEYSPACE si besoin). Prenez ce keyspace comme keyspace courant avec la commande USE ;

cqlsh> CREATE KEYSPACE garage WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1 };

Question 2
Créez les tables Vehicule et Mecanicien (faire help CREATE_TABLE si besoin).

cqlsh> CREATE TABLE garage.Mecanicien(idmecano int PRIMARY KEY , nom text, prenom text, status text);
cqlsh:garage> CREATE TABLE garage.Vehicule(idvehicule int PRIMARY KEY , marque text, modele text, kilometrage int, mecano int);

Question 3
Copiez-collez dans le shell CQL, le fichier insertion_cassandra.txt fourni dans les ressources pour remplir la base.
✓

Question 4
Listez tous les véhicules (faire help SELECT si besoin)

idvehicule	kilometrage	marque	mecano	modele
5	10265	Renault	6	Megane
1	58600	Renault	3	clio
2	46871	Peugeot	2	206
4	125680	Citroen	2	Xsara
7	89845	Citroen	5	C3
6	68420	Peugeot	7	107
3	78954	Peugeot	1	607

Question 5
Listez toutes les marques de véhicules.

marque
Renault
Renault
Peugeot
Citroen
Citroen
Peugeot
Peugeot

Question 6
Donnez le nom du mécanicien numéro 3.

```
 nom     | prenom
---------+--------
 Turpin |    Guy
```

Question 7
Donnez les marques et les modèles de véhicule du mécanicien numéro 2. Pourquoi ceci affiche une erreur ?
Car les lignes sont stockées en fonction des clés de cluster, de base la clé primaire, avec notre requête on risque au pire de devoir parcourir tout le fichier.
Modifiez votre requête afin d'afficher le résultat attendu.

```
 marque  | modele
----------+--------
 Peugeot |    206
  Citroen |  Xsara
```

Question 8
Créez un index sur la colonne mecano de la table Vehicule (faire help CREATE_INDEX;) et refaite la requête de la question précédente (sans la clause ALLOW FILTERING).
cqlsh> CREATE INDEX ON vehicule (mecano) ;

Que remarquez-vous ?
La requête se lance sans erreur

Pourquoi ?
Car les valeurs de la colonne mecano sont maintenant également prisent en compte avec celles de la clé primaire lors du stockage.

Question 9
Donnez les véhicules avec un identifiant inférieur à 5. Que se passe-t-il si au lieu d'utiliser la clause ALLOW FILTERING, on comparait les id non pas par leur valeur mais par leur token (voir fonction TOKEN) ?

*cqlsh:garage> SELECT * FROM vehicule WHERE idvehicule < 5 ALLOW FILTERING ;*

idvehicule	*kilometrage*	*marque*	*mecano*	*modele*
1	*58600*	*Renault*	*3*	*clio*
2	*46871*	*Peugeot*	*2*	*206*
4	*125680*	*Citroen*	*2*	*Xsara*
3	*78954*	*Peugeot*	*1*	*607*

*cqlsh:garage> SELECT * FROM vehicule WHERE token(idvehicule) < 5;*

idvehicule	*kilometrage*	*marque*	*mecano*	*modele*
5	*10265*	*Renault*	*6*	*Megane*
1	*58600*	*Renault*	*3*	*clio*
2	*46871*	*Peugeot*	*2*	*206*
4	*125680*	*Citroen*	*2*	*Xsara*

Exercice 2 – Spark et Cassandra

Question 1
Écrire un programme spark qui copie la table mecanicien dans une table mecanicien_cpy.

```
import org.apache.spark._
import com.datastax.spark.connector._
import org.apache.spark.sql._
import com.datastax.spark.connector.SomeColumns

object Question1 extends App {
  val conf = new SparkConf().setAppName("Spark on Cassandra 1")
  .setMaster("local[*]").set("spark.cassandra.connection.host", "localhost")
  val sc = new SparkContext(conf)
  sc.setLogLevel("OFF")
  case class Mecanicien(idmecano: Int, nom: String, prenom: String, status:
String)
  val rdd = sc.cassandraTable[Mecanicien]("garage", "mecanicien")
  rdd.saveAsCassandraTable("garage", "mecanicien_cpy")
  sc.stop()
}
```

Question 2
Écrire un programme spark qui permet de fusionner les deux tables mecanicien et vehicule en une table reparation. On considérera qu'une réparation est identifiée par la clé primaire du véhicule concerné.

```
import org.apache.spark._
import com.datastax.spark.connector._
import org.apache.spark.sql._
import org.apache.spark.SparkContext
import Question1.Mecanicien
import java.util.ArrayList
import scala.collection.JavaConversions._

object Question2 extends App {
  val conf = new SparkConf().setAppName("Spark on Cassandra 2")
    .setMaster("local[*]").set("spark.cassandra.connection.host", "localhost")
  val sc = new SparkContext(conf)
  sc.setLogLevel("OFF")
  case class Vehicule(idvehicule: Int, kilometrage: Int, marque: String, mecano:
Int, modele: String)
  case class Reparation(idvehicule: Int, kilometrage: Int, marque: String,
modele: String, mecano: Int, nommecano: String, prenommecano: String,
statusmecano: String)
  val rddmecano = sc.cassandraTable[Mecanicien]("garage", "mecanicien")
  val rddvehicule = sc.cassandraTable[Vehicule]("garage", "vehicule")
  val reps = new ArrayList[Reparation]()
  for (v <- rddvehicule.collect()) {
    for (m <- rddmecano.filter(_.idmecano.equals(v.mecano)).collect()) {
      reps.add(new Reparation(v.idvehicule, v.kilometrage, v.marque, v.modele,
m.idmecano, m.nom, m.prenom, m.status))
    }
  }
  val rdd = sc.parallelize(reps)
  rdd.saveAsCassandraTable("garage", "reparation")
  sc.stop()
}
```

Question 3
Écrire un programme Spark qui modifie la table mécanicien en ajoutant une colonne vehicules qui liste l'ensemble des véhicules affectés au mécanicien.

```
import org.apache.spark._
import com.datastax.spark.connector._
import org.apache.spark.sql._
import Question1.Mecanicien
import Question2.Vehicule
import java.util.ArrayList
import scala.collection.JavaConversions._

object Question3 extends App {
  val conf = new SparkConf().setAppName("Spark on Cassandra 2")
    .setMaster("local[*]").set("spark.cassandra.connection.host", "localhost")
  val sc = new SparkContext(conf)
  val spark = SparkSession.builder().config("spark.cassandra.connection.host",
"localhost").getOrCreate()
  sc.setLogLevel("OFF")
  case class MecanicienPlus(idmecano: Int, nom: String, prenom: String, status:
String, vehicules: List[Int])
  val rddM = sc.cassandraTable[Mecanicien]("garage", "mecanicien")
  val rddV = sc.cassandraTable[Vehicule]("garage", "vehicule")
  val mec = new ArrayList[MecanicienPlus]()
  val ves = new ArrayList[Int]()
  for (m <- rddM.collect()) {
    for (v <- rddV.filter(_.mecano.equals(m.idmecano))) {
      ves.add(v.idvehicule)
    }
    mec.add(new MecanicienPlus(m.idmecano, m.nom, m.prenom, m.status,
ves.toList))
    ves.clear()
  }
  val rdd = sc.parallelize(mec)
  spark.sql("ALTER TABLE garage.mecanicien ADD vehicules list<text>;")
  rdd.saveToCassandra("garage", "mecanicien")
  sc.stop()
  spark.stop()
}
```