TP3: Hive et Oozie

Exercice 1 :

Récupérez la liste des élus municipaux en opendata (si pas déjà fait lors du dernier TP) sur la VM dans un dossier TP3:

wget https://s3.eu-west-3.amazonaws.com/hadoop-telecom/TP2/elus_mun2014.zip

Dézippez le avec unzip
Utilisez head pour voir le header (premiere ligne contenant les noms de colonne)
Créez une database tp3
Créez une table **externe** dans hive vers /user/cloudera/TP3/elus_mun avec toute ces colonnes (tout au format string):

```
use tp3;
create external table elus_mun (
nomliste string,
nompsn string ,
prepsn string ,
sexe string,
naissance string,
libcsp string,
nat string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
   "separatorChar" = ",",
   "quoteChar" = "\""
)
location '/user/cloudera/TP3/elus_mun'
tblproperties("skip.header.line.count"="1")
```

On utilise OpenCSVSerde pour pouvoir définir un format de CSV avec un séparateur et un caractère de quote. (Dans les données, des virgules peuvent apparaître dans les champs. la valeur est alors entourée par des quotes. Exemple: prenom: "Pierre-Charles, Vaka")

La propriété:
tblproperties("skip.header.line.count"="1")
nous permet de ne pas prendre en compte la première ligne du fichier CSV qui contient les noms de colonnes.

On va faire quelques transformations sur les données. Créez une table comme suit:

```
CREATE TABLE tp3.elus_municipaux (
liste STRING,
nom STRING,
prenom STRING,
naissance STRING,
profession STRING,
nationalite STRING
)
PARTITIONED BY (sexe STRING)
row format delimited
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
location '/user/cloudera/TP3/elus_municipaux';
```

Utilisez un insert overwrite pour peupler cette table depuis la table tp3.elus_mun, mais en ne gardant que les enregistrements uniques (mot clé distinct) et avec le champ sexe non null et non vide.
https://cwiki.apache.org/confluence/display/Hive/LanguageManual%2BDML

Attention l'ordre des champs de votre select doit respecter celui de la table. La clé de partition apparaît en dernier. Elle doit apparaître dans le select.

Avant d'exécuter cette requête, exécutez celle ci:
set hive.exec.dynamic.partition.mode=nonstrict;

Cela a pour effet d'autoriser la création de partitions dynamiques (basée sur la valeur d'une colonne existante) alors qu'aucune partition statique n'existe.

Solution en fin de TP

Vérifiez que cette nouvelle table possède 525146 lignes.
Constatez sur HDFS le partitionnement dynamique avec hadoop fs -ls

Si vous le souhaitez, vous pouvez également créer des buckets sur un champ que vous souhaitez, et constatez à nouveau la transformation sur HDFS.

En utilisant cette nouvelle table (même questions que le dernier exo du dernier TP):

Afficher les 10 premières lignes
Compter le nombre d'ingénieur
Afficher le top 5 des prénoms les plus utilisés parmis les femmes et le nombre d'occurrence associé.

Solution en fin de TP

Utilisez le mot-clé “explain extended” pour visualiser l’effet d’une partition:
hive -e "explain extended select * from tp3.elus_municipaux where sexe = 'F'"
hive -e "explain extended select * from tp3.elus_municipaux where nom like '%arg%'"

C’est un simple fetch, mais vous pouvez déjà visualiser la différence du nombre de lignes lues (numRows) lorsqu’il y a une partition.

Exercice 2 :

Oozie permet de définir des séquences de tâches dans Hadoop, de les piloter et les scheduler. Nous allons voir comment utiliser oozie pour piloter une des actions.

Pour tester le gestionnaire de workflow oozie nous allons lancer une action simple avec HIVE : extraire de HIVE un rapport comprenant les 100 prénoms les plus courants et les stocker dans HDFS.

Au préalable nous devons créer un script HIVE (avec l’extension .hql) qui réalise cette tâche :
INSERT OVERWRITE DIRECTORY 'TP3/oozie/output'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
select prenom, count(*) as cnt from tp3.elus_municipaux group by prenom ORDER BY cnt DESC limit 100;

Le script doit être stocké dans HDFS, par exemple dans TP3/oozie.

La création du workflow peut se faire via l’interface graphique ou via la ligne de commande. Vie l’interface graphique, un workflow est configurable via des actions drag & drop. L’outil séduit car il est graphique. On préfère pour un usage d’entreprise développer les workflows oozie via des lignes de commande. C’est une solution plus longue à mettre en place, mais recommandée. Cela permet de mieux gérer les versions de oozie, le déploiement, les packages associés et de passer outre les limitations de hue. Passer par la ligne de commande et coder des workflows oozie comporte néanmoins des risques. Les workflow

oozie sont des fichiers xml, il est facile de faire des erreurs. Ils s'avèrent coûteux à développer en terme de temps, et compliqués à déboguer à la première approche.

Récupérer les fichiers workflow.xml et job.properties du repository Github.
Mettez sur HDFS les fichiers workflow.xml et oozieScript.hql:
/user/raj_ops/TP3/oozie/workflow.xml
/user/raj_ops/TP3/oozie/oozieScript.hql

lancez la commande oozie suivante: oozie job --config job.properties -run
Visualisez l'état des jobs avec la commande: oozie jobs

Note: Vous pouvez suspendre et reprendre un job
oozie job {id} -suspend
oozie job {id} -resume

Si vous souhaitez avoir accès à l'UI d'Oozie pour les logs, voici la procédure pour installer next.js:

Arrêter Oozie Server
wget http://archive.cloudera.com/gplextras/misc/ext-2.2.zip
sudo cp ext-2.2.zip /usr/hdp/current/oozie-client/libext/
sudo chown oozie:hadoop /usr/hdp/current/oozie-client/libext/ext-2.2.zip
sudo -u oozie /usr/hdp/current/oozie-server/bin/oozie-setup.sh prepare-war
Démarrer Oozie Server
Allez sur l'Ui (liens dans quick links)

Observez ensuite le résultat sur HDFS.

Exercice 3 : UDF python HIVE

Hive permet l'utilisation des udf.
Créez un fichier transform.py sur votre machine comme suit :

```
import sys
for line in sys.stdin:
        line = line.strip()
        sys.stderr.write(line)
        prenom, nom = line.split('\t')
        l_name = nom.lower()
        print ('\t'.join([prenom, str(l_name)]))
```

Dans ce fichier nous allons récupérer les champs liste et nom puis les afficher avec nom en minuscule.

Dans Hive, tapez :

```
add FILE ./transform.py;
select transform (prenom, nom) using 'python transform.py' as (prenom, nom) from tp3.elus_municipaux;
```

Si vous le souhaitez, vous pouvez également écrire l'UDF en Java: Créez un projet Maven et ajouter la dépendance suivante:

```
<dependency>
 <groupId>org.apache.hive</groupId>
 <artifactId>hive-exec</artifactId>
 <version>1.2.1</version>
 <scope>provided</scope>
</dependency>
```

Créez ensuite un classe qui hérite de UDF:

```
import org.apache.hadoop.hive.ql.exec.UDF;

public class Transform extends UDF {
  public String evaluate(String name) {
     //code here
  }
}
```

Ajouter ensuite le jar via la hive cli, et déclarer la fonction temporaire:

```
add jar hdfs:/user/raj_ops/TP3/transform.jar;
CREATE TEMPORARY FUNCTION tolowertest as 'com.demo.app.Transform';
```

Note: l'UDF Java prend un paramètre en entrée.

De la même manière, grâce à une fonction python ou Java, affichez les personnes de moins de 60 ans étant retraitées.

Solutions

Exercice 1:

Peupler la table avec partitionnement dynamique:

```
insert overwrite table tp3.elus_municipaux partition (sexe)
select distinct nomliste,nompsn ,prepsn ,naissance,libcsp,nat,sexe from tp3.elus_mun where
sexe is not null and sexe !=  '';
```

Solutions requêtes

```
select * from tp3.elus_municipaux limit 10;

select count(*) from tp3.elus_municipaux where profession like ”%Ing%nieur%”;

select prenom, count(*) as cnt from tp3.elus_municipaux where sexe=”F” group by prenom
ORDER BY cnt DESC limit 5;
```

Exercice 3:

Script Python

```
import sys
import datetime

now = datetime.datetime.now()
nb_days_60_years = 60 * 365

for line in sys.stdin:
        line = line.strip()
        nom, prenom, naissance, profession = line.split('\t')
        if "retrait" not in profession:
        continue
        age_in_timespan = now - datetime.datetime.strptime(naissance, '%Y-%m-%d')
        if age_in_timespan.days > nb_days_60_years:
        continue
        print ('\t'.join([nom, prenom, naissance, profession]))
```

```
select transform (nom, prenom, naissance, profession) using 'python transform2.py' as (nom,
prenom, naissance, profession) from tp3.elus_municipaux;
```