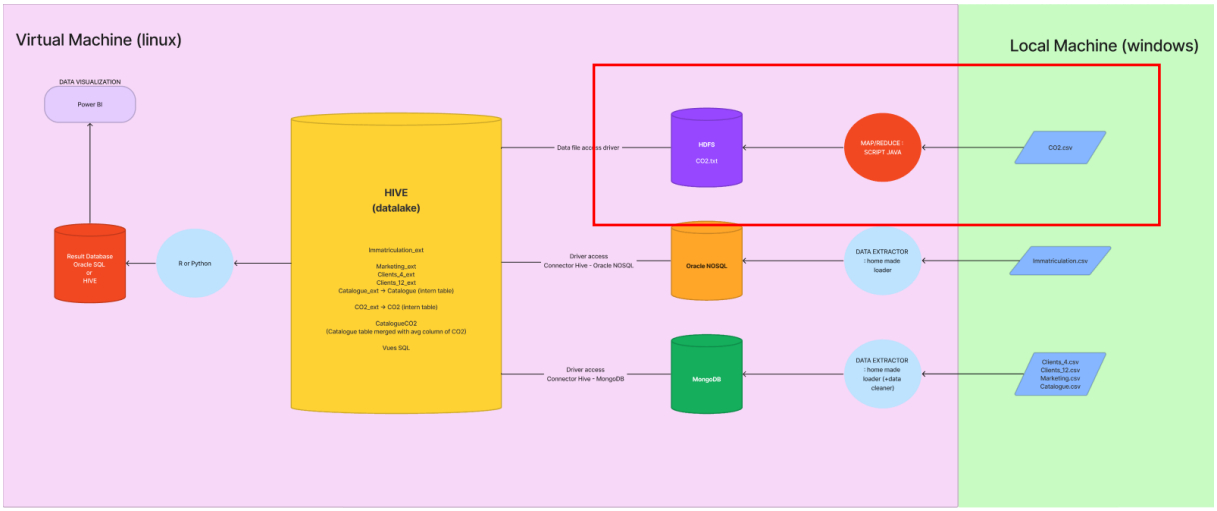
Rapport HDFS

Git repo (https://github.com/MaxencePachot/HDFS_PROJETGR4_CDPP)

Le fichier CO2.csv téléchargé au début présentait beaucoup d'incohérences, nous avons donc dû modifier ce dernier afin d'avoir le document le plus complet et le plus facile à exploiter possible.



MAP / REDUCE / MAIN

MAP

Elle va être la partie « nettoyage » du code. En outre, nous avons modifier tous les caractères spéciaux, tous les colonnes vides ou illogiques, afin de pouvoir séparer toutes les données.

Les colonnes à séparer sont les suivantes : Marque/Modèle, Bonus/Malus, Rejets CO2 g/km, Coût en énergie.

On commence donc par enlever toutes les valeurs incohérentes du fichier tels que les espaces insécables ou les marques/modèles possédant des virgules en leur sein (car nous avons choisi la virgule comme séparateur de donnée). À la suite de cela, nous pouvons enfin séparer les différentes colonnes via la fonction « split ».

Ensuite vient la partie de modification des colonnes :

Pour Marque/Modèle :

- On conserve seulement la marque et non le modèle.
- On gère les cas où les marques présentes dans le fichier CO2.csv n'apparaissent pas dans le fichier Catalogue.csv en passant à la marque suivante.

Pour Bonus/Malus :

- On gère les cas particuliers
- On enlève les espaces, les symboles ainsi que les valeurs parasites
- On assigne une valeur nulle aux valeurs manquantes

Pour Rejet CO2 :

- On gère les cas particuliers

Pour Coût en énergie :

- On gère les cas particuliers

La dernière étape du MAP est de définir nos couples clefs/valeurs de sortie qui seront ensuite couples clefs/valeurs d'entrée de la fonction REDUCE.

REDUCE

Elle va s'occuper de rassembler toutes les données.

La première étape est de parcourir toutes les valeurs associées à chaque clef. Ces valeurs seront ensuite récupérées, affichées dans la console, séparées dans un tableau, associées à chaque place du tableau crée à l'étape précédente.

On va ensuite s'occuper des calculs demandés par le client : les moyennes. On les affiche ensuite logiquement afin de voir nos résultats.

Une particularité de ce map/reduce est que nous devons nous occuper des marques non présentes dans CO2.csv mais bien présentes dans le Catalogue.csv.

Nous ajoutons donc une ligne dans l'excel CO2, et les valeurs de ses propriétés seront une moyenne des moyennes de toutes les autres marques présentes dans ce même fichier.

MAIN

C'est le « superviseur » du code. Il va s'occuper entre d'autres de définir les classes driver, map et reduce, de définir les types des clefs/valeurs du programme, et de vérifier si le programme se lance correctement.

EXECUTION ET SORTIE DU PROGRAMME

EXECUTION

Voici les différentes commandes que nous avons utilisées pour exécuter le programme Hadoop.

(Tous les cmds sont à ouvrir à cette adresse : C:\vagrant-projects\OracleDatabase\21.3.0 qui est le path que nous avons utilisé tout au long du projet)

(A chaque serveur démarré changer de cmd)

-- Connexion à hdfs :

```
start-dfs.sh
```

```
start-yarn.sh
```

-- On créer un dossier dans hdfs

```
hadoop fs -mkdir CO2input
```

-- On ajoute une ligne dans le csv qui va nous permettre d'implémenter la moyenne des marques n'étant pas dans CO2.csv mais dans Catalogue.csv

460,ZDACIA,0,0,0
461,ZDAIHATSU,0,0,0
462,ZFIAT,0,0,0
463,ZFORD,0,0,0
464,ZHONDA,0,0,0
465,ZLANCIA,0,0,0
466,ZSAAB,0,0,0
467,ZSEAT,0,0,0

-- On envoie CO2.csv de la machine locale vers le stockage de la VM (en supprimant au cas où une CO2.csv existerait)

```
rm CO2.csv
```

```
cp ../../vagrant/ProjetTPABigData/hdfs/CO2.csv ~
```

-- On place CO2.csv dans ce dossier (en le supprimant avant au cas où il existerait)

```
hadoop fs -rm -r CO2input/CO2.csv
```

```
hadoop fs -put CO2.csv CO2input
```

-- Compiler le programme java en jar

```
cd ../../vagrant/ProjetTPABigData/hdfs/CO2/hadoop_intellij_project_gradle/
./gradlew
./gradlew clean
./gradlew jar
```

-- retourner au répertoire courant de la VM

```
cd ~
```

-- Supprimer le dossier CO2output

```
hadoop fs -rm -r /CO2output
```

-- On execute le .JAR (programme map_reduce)

```
hadoop jar
../../vagrant/ProjetTPABigData/hdfs/CO2/hadoop_intellij_project_gradle/build/libs/projetBigData-1.0.0.jar org.mbds.hadoop.projetBigData.CO2 CO2input /CO2output
```

-- On vérifie les résultats

```
hadoop fs -cat /CO2output/*
```

-- On créer un dossier dans lequel on va mettre le fichier trié (CO2) (en le supprimant avant au cas où il existerait)

```
rm -r fileSortedCO2Final
```

```
mkdir -p fileSortedCO2Final
```

-- On copie le fichier de hdfs vers le répertoire local

```
hadoop fs -getmerge /CO2output/* fileSortedCO2Final/CO2Sorted.txt
```

-- On créer un dossier dans hdfs (on le supprime avant au cas où)

```
hadoop fs -rm -r hdfsFileSortedCO2Final
```

```
hadoop fs -mkdir hdfsFileSortedCO2Final
```

-- On renvoie le txt dans hdfs

```
hadoop fs -put fileSortedCO2Final/CO2Sorted.txt hdfsFileSortedCO2Final
```

-- On vérifie

```
hadoop fs -cat hdfsFileSortedCO2Final/CO2Sorted.txt
```

-- Arrêter hdfs :

```
stop-yarn.sh
```

```
stop-dfs.sh
```

SORTIE

```
[vagrant@oracle-21c-vagrant ~]$ hadoop fs -cat hdfsFileSortedCO2Final/CO2Sorted.txt
AUDI;-2400;26;191
BMW;-631;39;80
HYUNDAI;-4000;8;151
JAGUAR;-6000;0;271
KIA;-4000;10;157
MERCEDES;7790;187;749
MINI;-3000;21;126
NISSAN;5802;160;681
PEUGEOT;-3000;15;144
RENAULT;-6000;0;206
SKODA;-666;27;98
VOLKSWAGEN;-1714;23;96
VOLVO;0;42;72
ZDACIA;-1370;42;232
ZDAIHATSU;-1370;42;232
ZFIAT;-1370;42;232
ZFORD;-1370;42;232
ZHONDA;-1370;42;232
ZLANCIA;-1370;42;232
ZSAAB;-1370;42;232
ZSEAT;-1370;42;232
```

Sortie de l'exécution du programme Hadoop