



Rapport du projet TPT : Lacs de Données et les tables externes

Fait par:

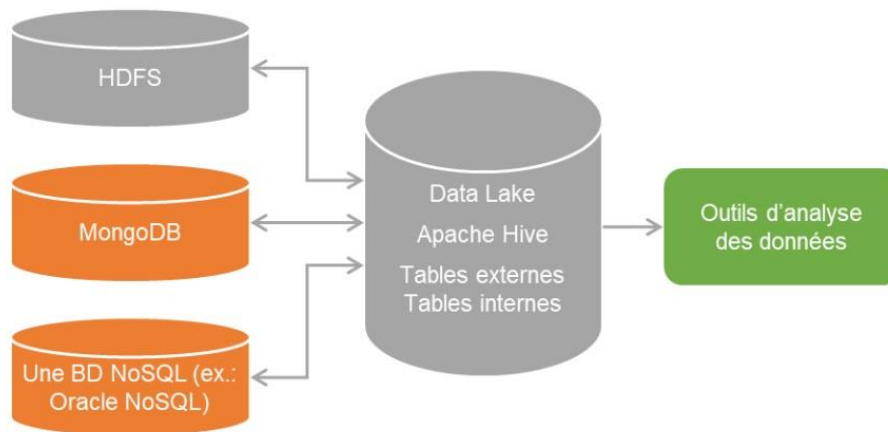
- Ferol Tatang Fomekon
- Fares

Table des matières

| | |
|---|----|
| Introduction..... | 2 |
| Chapitre 1..... | 3 |
| I. Le descriptif du projet..... | 3 |
| II. L'architecture du projet avec la ventilation des données par base ou système | 3 |
| III. Le descriptif de votre démarche pour l'adaptation du fichier CO2.csv et son intégration dans la table catalogue incluant les commandes / programmes que vous avez exécutés et leur résultat / output 3 | |
| Chapitre 2 : Scripts pour l'architecture | 5 |
| I. Le script de création de documents et de chargement de données dans MongoDB, le script de création des tables externes MongoDB dans Hive et enfin le script de création des tables ex ternes correspondantes dans Oracle NoSQL..... | 5 |
| II. Le script de création de tables dans Oracle NoSQL | 7 |
| III. Le script de création des tables externes correspondantes Hadoop HDFS dans Hive..... | 7 |
| IV. Le programme java de chargement des données dans la base Oracle NoSQL..... | 8 |
| V. Le script de chargement des données dans Hadoop HDFS | 9 |
| VI. Code source du programme Hadoop MapReduce utilisé pour l'adaptation du fichier CO2.csv et son intégration dans la table Catalogue..... | 9 |
| Conclusion | 10 |
| REFERENCE..... | 10 |

Introduction

Ce projet consiste à aider un concessionnaire automobile à mieux cibler les véhicules susceptibles d'intéresser ses clients. Pour cela, nous avons accès à son catalogue de véhicules, son fichier clients de l'année en cours, ainsi qu'à toutes les informations sur les immatriculations effectuées cette année. Notre objectif est de proposer un outil qui permettra d'évaluer en temps réel le type de véhicule le plus susceptible d'intéresser les clients qui se présentent dans la concession et d'envoyer une documentation précise sur le véhicule le plus adéquat pour des clients sélectionnés par son service marketing. Mais dans ce rapport, nous allons nous intéresser à la construction de l'architecture de DATA LAKE mixte autour de Hive comme le présente la figure 1 (Architecture du projet) avec pour objectif de traiter le fichier co2.csv grâce aux rdd (resilient Distributed dataset) et d'intégrer les colonnes intéressantes dans le fichier catalogue.



Architecture du projet

Chapitre 1

I. Le descriptif du projet

Le projet consiste à construire un Data Lake en utilisant des outils de traitement de données tels que MongoDB, Hive, Oracle NoSQL, HDFS et Hadoop. Les données seront réparties entre différentes bases de données et pourront être manipulées via HiveSQL. Le Concessionnaire demande ensuite d'adapter un fichier CO2.csv pour intégrer des informations manquantes dans la table Catalogue du Concessionnaire. Pour cela, il faudra écrire un programme MapReduce avec Hadoop ou Spark. Le but est d'améliorer la qualité des modèles de Machine Learning en disposant de données plus complètes.

II. L'architecture du projet avec la ventilation des données par base ou système

Voici l'architecture du projet avec la ventilation des données par base ou système :

ORACLE NoSQL : contient les données « catalogue »

MongoDB : contient les données « Marketing »

Hadoop HDFS : contient des données « Immatriculation , CO2 , et Clients_31 » sous forme de fichiers

Hive : contient une table interne avec les données du fichier " Clients_32", ainsi que des tables externes et internes pour accéder aux autres sources de données dans les différentes bases. Les données doivent être manipulées avec SQL.

Data lake : construit avec l'aide de Spark et des drivers pour se connecter aux différentes bases et systèmes de traitement. Stocke et analyse les données disponibles dans les différentes sources.

Fichier CO2.csv : utilisé pour adapter les informations complémentaires dans la table Catalogue du Concessionnaire grâce à un programme map/reduce avec Spark.

III. Le descriptif de votre démarche pour l'adaptation du fichier CO2.csv et son intégration dans la table catalogue incluant les commandes / programmes que vous avez exécutés et leur résultat / output

La démarche pour adapter le fichier CO2.csv et l'intégrer dans la table catalogue est la suivante :

Chargement du fichier CO2.csv dans HDFS en utilisant la commande `hadoop fs -put /vagrant/TPT/CO2.csv`.

Exploration et nettoyage du fichier CO2.csv pour supprimer les valeurs manquantes et erronées.

```
: print(df_Bm.value_counts())
print(df_Bm.mode())
print(df_Bm.isna().sum())
```

```
bonus_malus
8753€          185
-€             100
-6000€         53
7890€          27
8460€          17
873€           14
7340€          14
7073€          13
763€           7
680€           7
dtype: int64
bonus_malus
0      8753€
bonus_malus    0
dtype: int64
```

Remplacer les valeurs -€ de la colonne 'bonus_malus' par le mode de cette colonne

```
: df_Bm["bonus_malus"] = df_Bm["bonus_malus"].replace("-€", df_Bm['bonus_malus'].mode()[0])
```

Intéressons nous à la colonne cout_energie

```
: df_Cout = df_separated.select(concat(regex_replace(trim("Cout_energie"), "[^0-9-]", ""), lit("€")).alias("Cout_energie"))
```

```
: df_Cout = df_Cout.toPandas()
```

```
: dfFormater.isna().sum()
```

```
: _C0          0
Marque        0
Modele        0
Rejets CO2 g/km  0
bonus_malus    0
Cout_energie    0
dtype: int64
```

Le code est a récupérer dans le noteook

Création d'une table externe dans Hive :

```
drop table CO2_hive_ext;
CREATE EXTERNAL TABLE CO2_hive_ext(
  MarqueModele string,
  BonusMalus string,
  RejetsCO2gkm string,
  Coutenerie string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 'hdfs:/carbone'
TBLPROPERTIES ("skip.header.line.count"="1");
```

Table externe CO2 dans hive

```

drop table catalogue_hive_ext;
CREATE EXTERNAL TABLE catalogue_hive_ext (
marque string,
nom string,
puissance int,
longueur string,
nbPlaces integer,
nbPortes int,
couleur string,
occasion string,
prix int,
id int)
STORED BY 'oracle.kv.hadoop.hive.table.TableStorageHandler'
TBLPROPERTIES (
"oracle.kv.kvstore" = "kvstore",
"oracle.kv.hosts" = "localhost:5000",
"oracle.kv.hadoop.hosts" = "localhost/127.0.0.1",
"oracle.kv.tableName" = "catalogue");

-- vérifie
select * from catalogue_hive_ext limit 5;

```

Table externe Catalogue dans hive

Création de la connexion dans le jupyter notebook avec hive et récupérer la tables catalogues enfin de faire la jointure.

Chapitre 2 : Scripts pour l'architecture

- I. Le script de création de documents et de chargement de données dans MongoDB, le script de création des tables externes MongoDB dans Hive et enfin le script de création des tables ex ternes correspondantes dans Oracle NoSQL.

Script pour créer des documents et charger des données dans MongoDB :

Nous avons aux préalables convertis le documents Immatriculation.csv en json ensuite nous avons exécuter ce code suivant ;

```

===== ACTIVITES DANS MongoDB=====
## Charger Marketing dans MongoDB
> use tpt
switched to db tpt
> db.createCollection("Marketing");
-- show collections;

-- Approche : via ``mongoimport`` cvs
-- Les données sont dans les fichiers :
-- Marketing.csv.csv

db.tpt.remove({});

mongoimport --db tpt --collection Marketing --file /vagrant/TPT/Marketing.csv --type csv --ignoreBlanks --headerline

```

Pour créer la table externe MongoDB dans Hive, nous avons utilisé le script suivant :

```
+++++
+++++création de table externe HIVE pour une table MongoDB+++++
+++++

drop table Marketing_hive_mongo_ext;
CREATE EXTERNAL TABLE Marketing_hive_mongo_ext (
  id INT,
  age INT,
  sexe string,
  taux FLOAT,
  situationFamilliale string,
  nbEnfantsAcharge INT,
  2eme_voiture string
)
STORED BY 'com.mongodb.hadoop.hive.MongoStorageHandler'
WITH SERDEPROPERTIES('mongo.columns.mapping'='{
  "id": "_id",
  "situationFamilliale": "situationFamilliale",
  "nbEnfantsAcharge": "nbEnfantsAcharge",
  "2eme_voiture" : "2eme_voiture"}')
TBLPROPERTIES ('mongo.uri'='mongodb://localhost:27017/tpt.Marketing');

SELECT * FROM Marketing_hive_mongo_ext LIMIT 5;
```

Script de créations des tables externes correspondantes à Oracle Nosql dans Hive :

```
+++++
+++++création de table externe HIVE pour une table oracle NoSQL+++++
+++++

drop table catalogue_hive_ext;
CREATE EXTERNAL TABLE catalogue_hive_ext (
  marque string,
  nom string,
  puissance int,
  longueur string,
  nbPlaces integer,
  nbPortes int,
  couleur string,
  occasion string,
  prix int,
  id int)
STORED BY 'oracle.kv.hadoop.hive.table.TableStorageHandler'
TBLPROPERTIES (
  "oracle.kv.kvstore" = "kvstore",
  "oracle.kv.hosts" = "localhost:5000",
  "oracle.kv.hadoop.hosts" = "localhost/127.0.0.1",
  "oracle.kv.tableName" = "catalogue");

-- vérifie
select * from catalogue_hive_ext limit 5;
```

II. Le script de création de tables dans Oracle NoSQL

Script de création de tables dans Oracle Nosql :

```
## Charger catalogue dans oracleNosql

-- creation de la table catalogue
execute 'drop table catalogue'
execute 'Create table catalogue (
marque string,
nom string,
puissance integer,
longueur string,
nbPlaces integer,
nbPortes integer,
couleur string,
occasion string,
prix integer,
id NUMBER GENERATED BY DEFAULT AS IDENTITY, PRIMARY KEY (id))'

-- chargement des données
put table -name catalogue -file /vagrant/TPT/catalogue.json

-- vérifier
execute 'select * from catalogue'
```

III. Le script de création des tables externes correspondantes Hadoop HDFS dans Hive

Script de création de tables externes correspondantes Hadoop hdfs dans Hive :

```
+++++
+++++ création de tables externes HIVE pour une table hadoop+++++
+++++

-- Table externe `Clients_31`
--aller dans hadoop :   hadoop fs -mkdir /clients
                        hadoop fs -mv Clients_31.csv/clients

drop table Client_31_hive_ext;
CREATE EXTERNAL TABLE Client_31_hive_ext(
age FLOAT,
sexe string,
taux FLOAT,
situationFamiliiale string,
nbEnfantsAcharge string,
2eme_voiture BOOLEAN,
immatriculation string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 'hdfs:/clients'
TBLPROPERTIES ("skip.header.line.count"="1");

-- vérifier
select * from Client_31_hive_ext limit 5;
```




Table externe Hive pour
Client_31


```
-- TABLE externe `CO2`
--aller dans hadoop :   hadoop fs -mkdir /carbone
                        hadoop fs -mv CO2.csv /carbone

drop table CO2_hive_ext;
CREATE EXTERNAL TABLE CO2_hive_ext(
  MarqueModele string,
  BonusMalus string,
  RejetsCO2gkm string,
  Coutenerie string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 'hdfs:/carbone'
TBLPROPERTIES ("skip.header.line.count"="1");

--vérifier
select * from CO2_hive_ext limit 5;
```

Table externe Hive pour CO2

```
drop table Immatriculations_hive_ext;
CREATE EXTERNAL TABLE Immatriculations_hive_ext(
  Immatriculation string,
  Marque string,
  Nom string,
  Puissance FLOAT,
  Longueur string,
  NbPlaces FLOAT,
  NbPortes FLOAT,
  Couleur string,
  Occasion BOOLEAN,
  Prix FLOAT
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 'hdfs:/Immatriculations'
TBLPROPERTIES ("skip.header.line.count"="1");

select * from Immatriculations_hive_ext limit 5;
```

Table externe Hive pour
Immatriculation

IV. Le programme java de chargement des données dans la base Oracle NoSQL.

Connection

```
nohup java -Xmx64m -Xms64m -jar $KVHOME/lib/kvstore.jar kvlite -secure-config disable -root $KVROOT &

java -jar $KVHOME/lib/kvstore.jar runadmin -port 5000 -host localhost

--Vérifier si la base NoSQL est en cours d'exécution
$ java -jar $KVHOME/lib/kvstore.jar ping -host localhost -port 5000
-- reponse : KVStore ping succeeded in 34 ms.

-- sinon redemarrer
$ java -jar $KVHOME/lib/kvstore.jar start -root $KVROOT -host localhost -port 5000

kv->connect store -name kvstore
```

Chargement des données :

```
-- chargement des données
put table -name catalogue -file /vagrant/TPT/catalogue.json

-- vérifier
execute 'select * from catalogue'
```

V. Le script de chargement des données dans Hadoop HDFS

```
===== ACTIVITES DANS HADOOP HDFS =====  
# Charger Immatriculation dans HADOOP  
hadoop fs -put /vagrant/TPT/Immatriculations.csv  
  
# Charger CO2 dans HADOOP  
hadoop fs -put /vagrant/TPT/CO2.csv  
  
#Charger Clients_31 dans HADOOP  
hadoop fs -put /vagrant/TPT/Clients_31.csv
```

VI. Code source du programme Hadoop MapReduce utilisé pour l'adaptation du fichier CO2.csv et son intégration dans la table Catalogue.

Appliquer la méthode `map()` pour séparer la première colonne en deux

```
: rdd_separated = rdd_CO.map(lambda x: (x[0], x[1].split(" ")[0], " ".join(x[1].split(" ")[1:], *x[2:])))
```

Conclusion

Nous avons réussi à remplir les objectifs de notre rapport, qui consistaient à mettre en place un data lake sous Hive, à traiter les données du fichier `co2.csv` à l'aide de RDD et à fusionner certaines colonnes de ce fichier avec le fichier de catalogue. Nous pouvons conclure que ces différentes tâches ont été menées à bien.





REFERENCE

[Stack Overflow - Where Developers Learn, Share, & Build Careers](#)

[ChatGPT - Advanced AI Chatbot by OpenAI \(chat-gpt.org\)](#)

[SergioSim/hadoop_hive_spark \(github.com\)](#)

NOTES IMPORTANTES

-  Je joins :
-  Ce rapport
-  Mon code : TPT_VersionFinal.md
-  Le lien vers mon driver avec les vidéos :

https://drive.google.com/drive/folders/1eryRWSglSa5vgkQV_DILufUXbd6tA3uT?usp=share_link