Projet MBDS

ZHAO YUE
LUO YAJUAN
ZHANG QUAN
CHAYMAE FAZAZI-IDRISSI
Hoda AIT BAALI

Introduction :

Afin de valider l'écosystème BIG DATA Hadoop et la stratégie de construction de lacs de données, on doit réaliser l'architecture comme l'image ci-dessous, qui consiste à s'appuyer les tables externes pour accéder aux données de sources hétérogènes (Oracle NoSQL, Hadoop HDFS, Oracle SQL).

L'Accès aux données pour les Data Visualization et Data Analysis with R se fera via le langage SQL interrogeant des tables externes et internes.

Voici est notre plan global concernant la partie de Data lac :

Pour gérer les quatre tables dont on a besoin dans la partie de l'analyse de données,

- Créer les tables externes Immatriculations et Catalogue par HDFS
- Créer la table externe Clients avec Oracle NoSQL
- Créer « Marketing » en table interne SQL
- Créer des tables externes HIVE pointant vers les tables physiques Oracle Nosql ou des fichiers physiques HADOOP et puis des tables externes Oracle SQL pointant vers les tables externes HIVE correspondante.

Pendant le projet de cette partie, la répartition des tâches est claire et on a des rôles complémentaires : ZHANG et LUO a créé la table externe Immatriculations avec HDFS et la partie de HIVE correspondante et la table interne SQL. FAZAZI-IDRISSI ,AIT BAALI et YUE a créer la table Catalogue et la table externe Clients avec Oracle NoSQL.

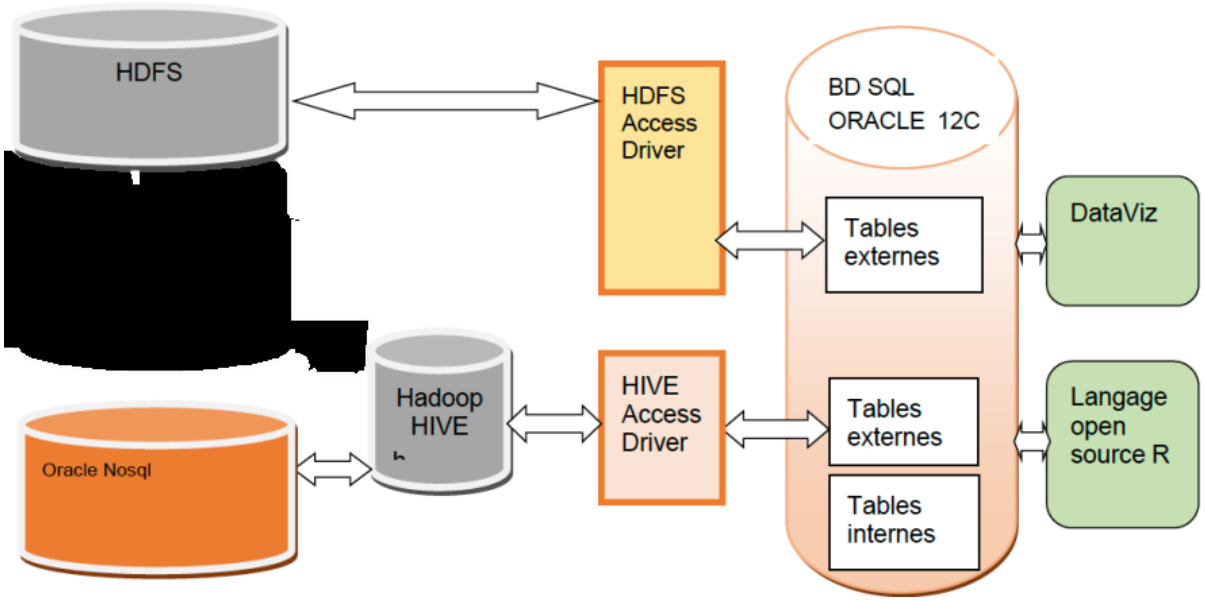

Image1 Architecture de data lac

ORACLE Nosql :

Cette dernière partie, comprend deux scripts :

--ConcessionnaireBase.java

--Script.txt

--OracleNoSQLExtTable.sql(yue_yajuan_chaymae_quan_hoda\HIVE).

Cette classe fournit les fonctions nécessaires pour gérer la table Client:

- La fonction void executeDDL(String statement) reçoit en paramètre
- une commande ddl et l'applique dans la base nosql.
- La displayResult affiche l'état de l'exécution de la commande
- la fonction createTableClient permet de créer une table client
- Insère une nouvelle ligne dans la table CLIENT
- Charge tous les clients du fichier client.

Le deuxième fichier comprend les étapes de manipulations de ORACLE NoSQL DATABASE : programation JAVA avec l''API KV (KEY/VALUE)

Le dernier fichier de script crée la table externe CLIENT_ONS_EXT dans le HIVE qui utilisera la table client, précédemment crée et charger. Il crée ensuite la table externe Oracle Sql correspondant à la table externe du HIVE.

Comme pour la partie HDFS, la connexion utilise les identifiants et paramètres vu en cours.

HDFS :

Cette partie comprend un script :

--hdfsTabEx.sql.

Ce script contient toutes les manipulations à réaliser :

- La création des fichiers dans Hadoop HDFS,
- La création de la table externe dans la base Hive,
- La création de la table externe dans la base Oracle SQL.

Sur notre sujet, on a mis deux tables dans la partie de HDFS, une est créé sur la VM de ZHANG, l'autre sur la VM de AIT BAALI, puis on a fusionné toutes les tables dans le ORACLE 12 sur la VM de ZHAO, on a créé les deux tables externes correspondantes dans Oracle 12 qui peuvent localiser ses fichiers de données par les pointeurs de HIVE.
Donc tout d'abord, il faut placer le fichier sur HDFS, pour cela pensez à vérifier le chemin d'accès au fichier à importer sur le script.
Afin de se connecter au HIVE, nous avons gardé les mêmes configurations que celles utiliser en cours.

Oracle SQL :

Cette partie contient un script :

-- OracleScript.sql.

--load_Marketing.ctl

Le script contient les créations de tablespaces et des tables. Le tablespace est utilisé pour gérés les indexes (mais après on a changé la structure de la table, on a pas utilisé finalement la tablespace qu'on a créé précédemment), sur la primary Key.
cette primary key est un id, qui n'existe pas dans les fichiers de données.
Cependant il a été nécessaire de l'ajouter car aucun des champs, même immatriculation, n'est pas unique. Cet id s'incrémentera à chaque insertion de ligne.
Au niveau des tables, des contraintes ont été appliqués, principalement des contraintes check respectant les plages de données énoncées dans le sujet.

Pour importer les données via sqlloader, on a réussi à importer les données de la table marketing mais pas catalogue, donc après plusieurs essaies d'échec on met la table catalogue comme une table externe chargé par HDFS, on a analysé toutes les possibilités mais il affiche toujours comme ci-dessous :

```
[ZHAO@bigdatalite ~]$ sqlldr userid = ZHAOBZ2021@ORCL/ZHAOBZ202101
control=/home/ZHAO/Projet/Data/load_Catalogue.ctl

SQL*Loader: Release 12.1.0.2.0 - Production on Thu Mar 25 19:28:38 2021
Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights reserved.

Path used:      Conventional
Commit point reached - logical record count 270

Table CATALOGUE:
  0 Rows successfully loaded.

Check the log file:
  load_Catalogue.log
for more information about the load.
```

En même temps, il se génère un fichier *catalogue.bad*, on peut voir toutes les données sont prises comme les bad données. On se dit qu'il n'a donc pas d'erreur dans notre fichier controleur (.ctl pour charger le fichier de donées), ça peut-être à cause du contraint qu'on a ajouté quand on crée la table, mais même on enlève tous les contraints dans la table catalogue, il fonctionne toujours pas(surtout on a fait la même chose concernant sqlloader dans le projet tune, et on a réussi)
Puisque on a réussi à importer les données de marketing, on a bien compris comment importer les données par sqlloader.

Data analyse sous R :

Le nettoyage des bases de donnée s'est fait lorsque nous avons importé les 4 tables depuis oracle. Le nettoyage s'est fait en SQL car nous considérons que les conflits de format entraînent une perte de données quand nous utilisons 'constraint' pour filter les données lorsque nous créons les tables. Afin d'obtenir un résultat direct et précis, nous utilisons catalogue_id pour l'apprentissage de notre modèle. Ç'est parce qu'il y n'a que 270 labels par rapport au data set, qui a plus que 100000 lignes. Malheureusement, nous ne pouvons pas obtenir un résultat idéal (chapitre 3). Cela nous proposons une solution globale. Nous séparons les caractères de catalogue et choisissons 5 caractères le plus importants pour influencer le résultat de la prédiction. Après d'utiliser et comparer les différents algorithmes, nous obtenons un modèle final pour la prédiction de test set(chapitre 4 et 5). En plus, nous concluons et listons les problèmes comme sur-apprentissage(overfitting) et proposons les solutions sous R(chapitre 6).

Le détail veillez référer le fichier DataAnalyse.pdf