REPUBLIQUE DU SENEGAL

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR, DE LA RECHERCHE ET DE L'INNOVATION

UNIVERSITE GASTON BERGER DE SAINT LOUIS

INSTITUT POLYTECHNIQUE DE SAINT LOUIS

SECTION INFORMATIQUE ET TELECOMMUNICATION

COURS : BIG DATA

NIVEAU : ING3 INFO-TELECOM

UNIVERSITE GASTON BERGER

L'excellence au service du développement

INSTITUT POLYTECHNIQUE DE SAINT-LOUIS

Projet : Ingestion de données dans Big Data

Présenté par :

AMADOU ALIOU SOW

Sous la direction de :

Dr. Djibril MBOUP

ANNEE ACADEMIQUE 2023/2024

Plan :

INTRODUCTION

Dans un contexte où les données sont essentielles pour la prise de décision, les outils de Big Data comme Apache Sqoop et Apache Hive sont indispensables pour gérer et analyser de grandes quantités d'informations. Ce rapport présente l'utilisation de ces deux outils pour importer et traiter des données d'une base de données relationnelle.

Le projet utilise la base de données e-commerce retail_db, qui contient des informations sur les ventes avec six tables principales : Departments, Categories, Products, Order Items, Orders, et Customers. L'objectif est d'importer ces données dans Hadoop avec Sqoop et de les analyser avec Hive pour extraire des insights pertinents.

Apache Sqoop facilite l'échange de données entre les bases de données relationnelles et Hadoop, tandis qu'Apache Hive permet l'analyse des données stockées dans Hadoop via une interface SQL. Le projet se divise en deux parties : l'ingestion des données avec Sqoop et le traitement avec Hive. L'objectif est de démontrer l'intégration et l'analyse efficaces de grandes quantités de données à l'aide de ces outils.

Prérequis :

Pour faire le TP, nous avons installé dans notre machine locale qui contient les prérequis suivants

- **Apache Sqoop**
- **Apache Hive**
- **MariaDB**

I. PART I : Ingestion des données avec Apache Sqoop

Description :

Dans ce tutoriel, nous allons voir comment charger les données d'une base de données externe dans le Big data avec Apache Sqoop (https://sqoop.apache.org/).

Prérequis :

Pour faire le TP, nous avons installé dans notre machine locale MySQL comme SGBD relationnel et Un éditeur de base de données

Puis, nous avons téléchargé les scripts SQL de la base de données retail_db.sql sur ce lien drive ci-dessous :
https://drive.google.com/file/d/1CHwWhfJn4edCuAOHiWr6iyT4wJ-zPNbU/view?usp=share_link

Retail DB est une base de données qui contient des données de ventes d'une entreprise e-commerce. Cette base de données comporte 6 tables :

- Departments
- Categories
- Products
- Order Items
- Orders
- Customers

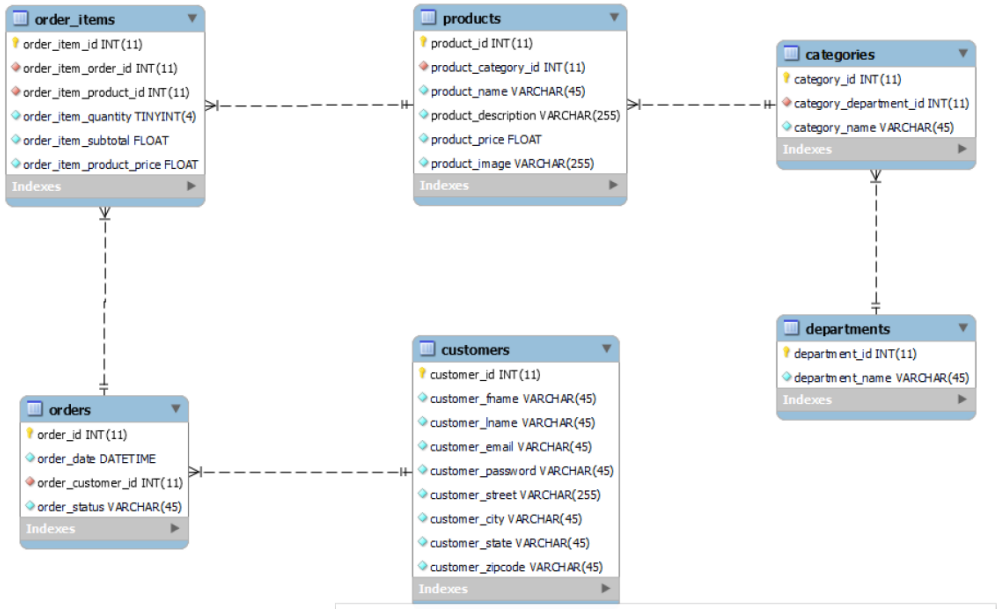

Figure : Diagramme de classe de retail_db.

Préparation de la Base de Données

- Démarrer MySQL en mode console

```
C:\Users\HP\hadoopVagrant>mysql -u root -p
```

- Création d'un compte utilisateur admin retail_dba

```
mysql> CREATE USER 'retail_dba'@'localhost' IDENTIFIED BY 'hadoop';
Query OK, 0 rows affected (0.21 sec)
```

- Créeons la base de données retail_db

```
mysql> CREATE database retail_db;
Query OK, 1 row affected (0.06 sec)
```

- Ajouter les droits d'utilisateur sur la base de données retail.db

```
mysql> GRANT ALL PRIVILEGES ON retail_db.* TO 'retail_dba'@'localhost';
Query OK, 0 rows affected (0.22 sec)
```

```
mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

- Se connecter en tant que retail_dba:

```
mysql> mysql -u retail_dba -phadoop
```

- Chargement

```
mysql> source C:/Users/HP/hadoopVagrant/hadoop/retail.sql;
```

- Afficher les tables

```
mysql> USE retail_db;
Database changed
```

```
mysql> show tables;
+--------------------+
| Tables_in_retail_db |
+--------------------+
| categories         |
| customers          |
| departments        |
| order_items        |
| orders             |
| products           |
+--------------------+
6 rows in set (0.11 sec)
```

- Démarrer Vagrant VM :

```
HP@DESKTOP-2AS MINGW64 ~/hadoopVagrant (main)
$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'SopeKhadim/hadoopVM' version '2.0' is up to date...
```

- Se connecter :

```
HP@DESKTOP-2AS MINGW64 ~/hadoopVagrant (main)
$ vagrant ssh
Last login: Sun Jul 28 17:16:39 2024 from 10.0.2.2
```

- Démarrer Hadoop et vérifier la connectivité

```
[vagrant@10 ~]$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as vagrant in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [10.0.2.15]
10.0.2.15: Warning: Permanently added '10.0.2.15' (ECDSA) to the list of known hosts.
Starting resourcemanager
Starting nodemanagers
[vagrant@10 ~]$ jps
5298 NameNode
5573 SecondaryNameNode
5879 ResourceManager
5400 DataNode
6137 Jps
5983 NodeManager
[vagrant@10 ~]$
```

Après avoir vérifié si notre machine virtuelle et notre machine locale sont dans le même réseau. Nous configurons Apache Sqoop pour facilite l'échange de données entre les bases de données relationnelles et Hadoop, et Apache Hive pour l'analyse des données stockées dans Hadoop via une interface SQL où nous exécutons les requêtes sql demandés

II. PART II: Data Processing avec Apache Hive

Description :

Dans ce projet, nous allons voir le traitement et la transformation des données dans le Big Data avec Apache Hive.

- Démarrer Vagrant VM :

```
HP@DESKTOP-2AS MINGW64 ~/hadoopVagrant (main)
$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'SopeKhadim/hadoopVM' version '2.0' is up to date...
```

- Se connecter :

```
HP@DESKTOP-2AS MINGW64 ~/hadoopVagrant (main)
$ vagrant ssh
Last login: Sun Jul 28 17:16:39 2024 from 10.0.2.2
```

- Démarrer Hadoop et vérifier la connectivité

```
[vagrant@10 ~]$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as vagrant in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [10.0.2.15]
10.0.2.15: Warning: Permanently added '10.0.2.15' (ECDSA) to the list of known hosts.
Starting resourcemanager
Starting nodemanagers
[vagrant@10 ~]$ jps
5298 NameNode
5573 SecondaryNameNode
5879 ResourceManager
5400 DataNode
6137 Jps
5983 NodeManager
[vagrant@10 ~]$
```

Après vérification dans Hive nous constatons que les tables ne sont pas créées après leur importation, nous avons exécuté les scripts de *Data Definition Language* (DDL) pour créer le schéma des tables dans Hive.
Nous parvenons à Afficher les tables dans hive :

```
hive> show tables;
OK
categories
customers
departments
order_items
orders
Time taken: 0.104 seconds, Fetched: 5 row(s)
```

III. Exercice

Nous allons donner les requêtes sql pour répondre aux questions, les données stockées sur cette base de données sont très volumineux donc il est très difficile de faire une capture sur chaque résultat. Nous allons fournir alors la requête correspondante à chaque question.

Voici les requêtes SQL pour répondre à chaque question :

1. **Trouver le nombre total de commandes passées par chaque client au cours de l'année 2014. Le statut de la commande doit être COMPLET, le format `order_date` est au format `unix_timestamp`.**

```
SELECT
    c.customer_id,
    CONCAT(c.customer_fname, ' ', c.customer_lname) AS customer_name,
    COUNT(o.order_id) AS total_orders
FROM
```

```
    orders o
JOIN
    customers c ON o.order_customer_id = c.customer_id
WHERE
    o.order_status = 'COMPLET'
    AND FROM_UNIXTIME(o.order_date, '%Y') = '2014'
GROUP BY
    c.customer_id, c.customer_fname, c.customer_lname;
```

2. **Afficher le nom et le prénom des clients qui n'ont passé aucune commande, triés par `customer_lname` puis `customer_fname`.**

```
SELECT
    c.customer_fname,
    c.customer_lname
FROM
    customers c
LEFT JOIN
    orders o ON c.customer_id = o.order_customer_id
WHERE
    o.order_id IS NULL
ORDER BY
    c.customer_lname, c.customer_fname;
```

3. **Afficher les détails des top 5 clients par revenu pour chaque mois. Vous devez obtenir tous les détails du client ainsi que le mois et les revenus par mois. Les données doivent être triées par mois dans l'ordre croissant et les revenus par mois dans l'ordre décroissant.**

```
WITH MonthlyRevenue AS (
    SELECT
        c.customer_id,
        CONCAT(c.customer_fname, ' ', c.customer_lname) AS
customer_name,
        FROM_UNIXTIME(o.order_date, '%Y-%m') AS month,
        SUM(oi.order_item_subtotal) AS total_revenue
    FROM
        orders o
    JOIN
        customers c ON o.order_customer_id = c.customer_id
    JOIN
        order_items oi ON o.order_id = oi.order_item_order_id
    WHERE
        o.order_status IN ('COMPLET', 'CLOSED')
    GROUP BY
        c.customer_id, customer_name, month
)
SELECT
    *
FROM
    (SELECT
         *,
         ROW_NUMBER() OVER (PARTITION BY month ORDER BY total_revenue
DESC) AS rank
     FROM
         MonthlyRevenue
    ) ranked
WHERE
    rank <= 5
```

```
ORDER BY
    month, total_revenue DESC;
```

4. **Trouver toutes les commandes terminées ou fermées (completed ou closed), puis calculez le revenu total pour chaque jour pour chaque département. La sortie doit afficher : `order_date`, `department_name` et `order_revenue`.**

```
SELECT
    FROM_UNIXTIME(o.order_date, '%Y-%m-%d') AS order_date,
    d.department_name,
    SUM(oi.order_item_subtotal) AS order_revenue
FROM
    orders o
JOIN
    order_items oi ON o.order_id = oi.order_item_order_id
JOIN
    products p ON oi.order_item_product_id = p.product_id
JOIN
    categories c ON p.product_category_id = c.category_id
JOIN
    departments d ON c.category_department_id = d.department_id
WHERE
    o.order_status IN ('COMPLET', 'CLOSED')
GROUP BY
    order_date, d.department_name;
```

5. **Trouver le rank de chaque catégorie par revenu obtenue dans chaque département à partir de toutes les transactions. Affichez les résultats par `department_name` et classez-les par ordre croissant.**

```
WITH CategoryRevenue AS (
    SELECT
        d.department_name,
        c.category_name,
        SUM(oi.order_item_subtotal) AS total_revenue
    FROM
        orders o
    JOIN
        order_items oi ON o.order_id = oi.order_item_order_id
    JOIN
        products p ON oi.order_item_product_id = p.product_id
    JOIN
        categories c ON p.product_category_id = c.category_id
    JOIN
        departments d ON c.category_department_id = d.department_id
    WHERE
        o.order_status IN ('COMPLET', 'CLOSED')
    GROUP BY
        d.department_name, c.category_name
)
SELECT
    department_name,
    category_name,
    total_revenue,
    RANK() OVER (PARTITION BY department_name ORDER BY total_revenue
DESC) AS rank
FROM
    CategoryRevenue
ORDER BY
```

```
    department_name, rank;
```

6. **Afficher le pourcentage de chaque catégorie par revenu dans chaque département. Afficher les résultats par `department_name` et pourcentage par ordre décroissant.**

```
WITH CategoryRevenue AS (
    SELECT
        d.department_name,
        c.category_name,
        SUM(oi.order_item_subtotal) AS total_revenue
    FROM
        orders o
    JOIN
        order_items oi ON o.order_id = oi.order_item_order_id
    JOIN
        products p ON oi.order_item_product_id = p.product_id
    JOIN
        categories c ON p.product_category_id = c.category_id
    JOIN
        departments d ON c.category_department_id = d.department_id
    WHERE
        o.order_status IN ('COMPLET', 'CLOSED')
    GROUP BY
        d.department_name, c.category_name
),
DepartmentRevenue AS (
    SELECT
        department_name,
        SUM(total_revenue) AS dept_revenue
    FROM
        CategoryRevenue
    GROUP BY
        department_name
)
SELECT
    cr.department_name,
    cr.category_name,
    cr.total_revenue,
    (cr.total_revenue / dr.dept_revenue) * 100 AS revenue_percentage
FROM
    CategoryRevenue cr
JOIN
    DepartmentRevenue dr ON cr.department_name = dr.department_name
ORDER BY
    cr.department_name, revenue_percentage DESC;
```

7. **Afficher tous les clients qui ont passé une commande d'un montant supérieur à 200 $.**

```
SELECT DISTINCT
    c.customer_id,
    CONCAT(c.customer_fname, ' ', c.customer_lname) AS customer_name
FROM
    orders o
JOIN
    order_items oi ON o.order_id = oi.order_item_order_id
JOIN
    customers c ON o.order_customer_id = c.customer_id
```

```
WHERE
    oi.order_item_subtotal > 200;
```

8. **Afficher les clients de la table `customers` dont les noms `customer_fname` commencent par "Rich".**

```
SELECT
    customer_id,
    customer_fname,
    customer_lname
FROM
    customers
WHERE
    customer_fname LIKE 'Rich%';
```

9. **Fournir le nombre total de clients dans chaque état (`state`) dont le prénom commence par « M ».**

```
SELECT
    customer_state,
    COUNT(*) AS total_customers
FROM
    customers
WHERE
    customer_fname LIKE 'M%'
GROUP BY
    customer_state;
```

10. **Trouver le produit le plus cher dans chaque catégorie.**

```
SELECT
    c.category_name,
    p.product_name,
    MAX(p.product_price) AS max_price
FROM
    products p
JOIN
    categories c ON p.product_category_id = c.category_id
GROUP BY
    c.category_name, p.product_name;
```

11. **Trouvez les 10 meilleurs produits qui ont généré les revenus les plus élevés.**

```
WITH ProductRevenue AS (
    SELECT
        p.product_name,
        SUM(oi.order_item_subtotal) AS total_revenue
    FROM
        order_items oi
    JOIN
        products p ON oi.order_item_product_id = p.product_id
    GROUP BY
        p.product_name
)
SELECT
    product_name,
    total_revenue
FROM
```

```
    ProductRevenue
ORDER BY
    total_revenue DESC
LIMIT 10;
```

CONCLUSION

Ce projet démontre l'intégration et le traitement des données dans un environnement Big Data en utilisant Apache Sqoop et Apache Hive. Nous avons importé les données de la base e-commerce `retail_db` vers Hadoop avec Sqoop, et effectué des analyses approfondies avec Hive.

La première étape a permis d'importer efficacement les données en format Parquet, facilitant leur stockage et leur accès dans Hadoop. La deuxième étape a mis en évidence les capacités d'Apache Hive pour l'analyse des données via des requêtes SQL, fournissant des insights sur les commandes, les revenus et les clients.

En conclusion, l'utilisation combinée de Sqoop et Hive montre comment surmonter les défis de gestion des grandes quantités de données et comment tirer parti des outils Big Data pour obtenir des informations exploitables. La maîtrise de ces outils est cruciale pour optimiser les opérations et les stratégies d'entreprise.

Ces requêtes vous permettront d'extraire les informations pertinentes de la base de données en utilisant Apache Hive pour répondre à chaque question posée.