ARCHITECTURE DÉCISIONNELLE(DataMart)

Préparé par :

SEHAKI Sofiane

Merbah Yanis

GHERSBRAHAM Anis

Création d'un Datamart pour les Trajets de Taxi à New York

Introduction

Notre projet a pour objectif de mettre en place un système automatisé pour collecter, stocker et analyser les données relatives aux trajets de taxi à New York à partir de janvier 2023. Pour cela, nous utiliserons Minio comme outil de Data Lake pour stocker les données brutes. De plus, PostgreSQL sera utilisé à la fois comme base de données relationnelle pour le stockage initial et comme base de données OLAP pour le Datamart. Cette approche intégrée nous permettra de traiter efficacement les données, offrant ainsi des perspectives précieuses pour l'analyse et la prise de décision.

1. Extraction de Données

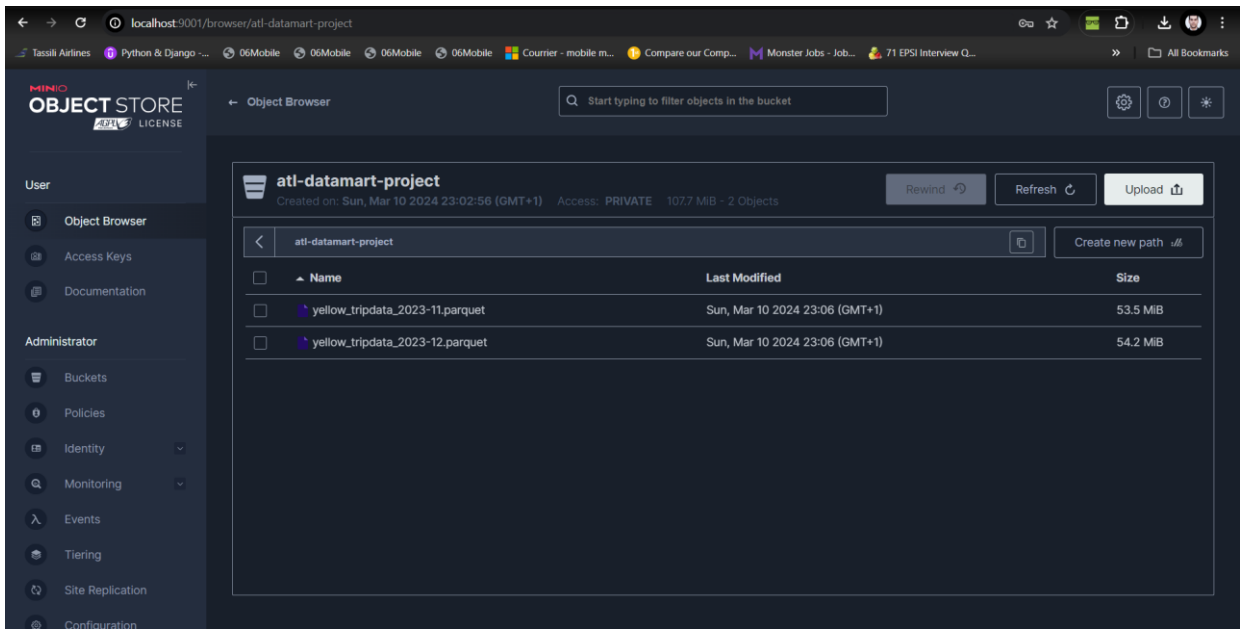
```
import os
import sys
from urllib.request import urlretrieve
from minio import Minio
from minio.error import S3Error
def download_data():
    """
    Télécharge les fichiers de données depuis les URLs spécifiées et les enregistre dans
le dossier 'data/raw'.
    """
    urls = [
        'https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2023-
11.parquet',
        'https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2023-
12.parquet',
    ]
    save_dir = "../../data/raw"
    os.makedirs(save_dir, exist_ok=True)

    for url in urls:
        file_name = os.path.basename(url)
        save_path = os.path.join(save_dir, file_name)
        print(f"Téléchargement de {file_name}...")

        try:
            urlretrieve(url, save_path)
            print(f"{file_name} téléchargé avec succès et enregistré dans {save_path}.")
        except Exception as e:
            print(f"Échec du téléchargement de {file_name}. Erreur : {e}")

def upload_to_minio():
    """
    Téléverse les fichiers Parquet depuis le dossier 'data/raw' vers un bucket Minio.
    """
    minio_client = Minio(
        "localhost:9000",
        access_key="minio",
        secret_key="minio123",
        secure=False
    )
    bucket_name = "atl-datamart-project"
    if not minio_client.bucket_exists(bucket_name):
        minio_client.make_bucket(bucket_name)
        print(f"Bucket {atl-datamart-project} créé.")
    else:
        print(f"Bucket {atl-datamart-project} existe déjà.")
    data_dir = "../../data/raw"
    for file_name in os.listdir(data_dir):
        if file_name.endswith(".parquet"):
            file_path = os.path.join(data_dir, file_name)
            print(f"Téléversement de {file_name} dans {bucket_name}...")

            try:
                with open(file_path, "rb") as file_data:
                    file_stat = os.stat(file_path)
                    minio_client.put_object(
                        bucket_name,
                        file_name,
                        file_data,
                        file_stat.st_size
                    )
                print(f"{file_name} téléversé avec succès dans {bucket_name}.")
            except S3Error as e:
                print(f"Erreur Minio lors de l'accès au bucket {bucket_name} : {e}")
            except Exception as e:
                print(f"Erreur inattendue : {e}")
def main():
    download_data()
    upload_to_minio()
if __name__ == '__main__':
    sys.exit(main())
```



2. Chargement des Données dans PostgreSQL

- Un script Python se charge de récupérer les données stockées dans Minio et de les charger dans PostgreSQL pour stockage initial.

```
from minio import Minio
from minio.error import S3Error
import pandas as pd
from io import BytesIO
from sqlalchemy import create_engine
import pyarrow.parquet as pq
# Configuration Minio
```

```
minio_client = Minio(
       "localhost:9000",
       access_key="minio",
       secret_key="minio123",
       secure=False
   )

# Configuration PostgreSQL
db_params = {
'host': 'localhost',
'port': 5432,
'user': 'postgres',
'password': 'admin',
'database': 'TaxiYolow'
}
# Nom du seau (bucket) Minio et liste des objets à télécharger
bucket_name = "datamartbukuts"
object_names = ["yellow_tripdata_2023-06.parquet", "yellow_tripdata_2023-07.parquet",
"yellow_tripdata_2023-08.parquet", "yellow_tripdata_2023-09.parquet"]
# Fonction pour télécharger un objet depuis Minio
def download_object(bucket, object_name):
try:
response = minio_client.get_object(bucket, object_name)
return response.read()
except S3Error as e:
print(f"Erreur lors du téléchargement de l'objet {object_name}: {e}")
return None
# Boucle pour télécharger chaque objet depuis Minio
for object_name in object_names:
data_bytes = download_object(bucket_name, object_name)
# Charger les données avec PyArrow
if data_bytes:
table = pq.read_table(BytesIO(data_bytes))
df = table.to_pandas()
# Créer une connexion à la base de données PostgreSQL
engine = create_engine(f'postgresql+psycopg2://{db_params["user"]}:
{db_params["password"]}@{db_params["host"]}:
{db_params["port"]}/{db_params["database"]}')
# Insérer le DataFrame dans la base de données
try:
table_name = object_name.replace('.parquet', '').replace('.', '_')
df.to_sql(table_name, engine, if_exists="replace", index=False)
print(f"Données du fichier {object_name} chargées dans la base de données avec succès.")
except Exception as e:
print(f"Erreur lors de l'insertion des données du fichier {object_name} dans la base de
données : {e}")
finally:
engine.dispose() # Fermer la connexion
```

3. Modèle en Flocon et Datamart

- Concevez un modèle en flocon pour structurer les données dans le Datamart.
- Stockez les résultats dans une base de données OLAP dédiée à PostgreSQL pour accroître la performance des requêtes.

```
-- Tables de dimension
CREATE TABLE dim_temps (
temps_id SERIAL PRIMARY KEY,
date_heure_depart timestamp without time zone
);
CREATE TABLE dim_location (
location_id SERIAL PRIMARY KEY,
PULocationID bigint,
DOLocationID bigint
);
CREATE TABLE dim_payment (
payment_id SERIAL PRIMARY KEY,
payment_type bigint
);
-- Table de fait
CREATE TABLE faits_trajet_taxi (
trajet_id SERIAL PRIMARY KEY,
temps_id INT REFERENCES dim_temps(temps_id),
location_id INT REFERENCES dim_location(location_id),
payment_id INT REFERENCES dim_payment(payment_id),
passenger_count double precision,
trip_distance double precision,
fare_amount double precision,
extra double precision,
mta_tax double precision,
tip_amount double precision,
tolls_amount double precision,
improvement_surcharge double precision,
total_amount double precision,
congestion_surcharge double precision,
airport_fee double precision
);
```

3.1 Partitionnement des données :

Définissez une stratégie de partitionnement. Pour cet exemple, nous allons partitionner la table de faits par mois.

```
CREATE TABLE faits_trajet_taxi_partitioned (
trajet_id SERIAL PRIMARY KEY,
temps_id INT REFERENCES dim_temps(temps_id),
location_id INT REFERENCES dim_location(location_id),
payment_id INT REFERENCES dim_payment(payment_id),
passenger_count double precision,
trip_distance double precision,
fare_amount double precision,
extra double precision,
mta_tax double precision,
tip_amount double precision,
tolls_amount double precision,
```

```
improvement_surcharge double precision,
total_amount double precision,
congestion_surcharge double precision,
airport_fee double precision
) PARTITION BY RANGE (temps_id);
```

3.2 Transfert de Données entre Bases de Données

```
CREATE EXTENSION dblink;
-- Connectez-vous à la base de données source
SELECT dblink_connect('taxiyolow_connection','host=localhost dbname=TaxiYolow
user=postgres password=idir');
-- Insérez les données dans la table de faits du datamart depuis la table source dans
l'autre base de données
INSERT INTO "Datamart.faits_trajet_taxi_partitioned" (
temps_id, location_id, payment_id,
passenger_count, trip_distance, fare_amount,
extra, mta_tax, tip_amount,
tolls_amount, improvement_surcharge, total_amount,
congestion_surcharge, airport_fee
)
SELECT
temps_id, location_id, payment_id,
passenger_count, trip_distance, fare_amount,
extra, mta_tax, tip_amount,
tolls_amount, improvement_surcharge, total_amount,
congestion_surcharge, airport_fee
FROM
dblink('taxiyolow_connection','SELECT temps_id, location_id, payment_id,
passenger_count, trip_distance, fare_amount, extra, mta_tax, tip_amount, tolls_amount,
improvement_surcharge, total_amount, congestion_surcharge, airport_fee FROM
yellow_tripdata_2023_01') AS data_source(temps_id INT, location_id INT, payment_id INT,
passenger_count DOUBLE PRECISION, trip_distance DOUBLE PRECISION, fare_amount DOUBLE
PRECISION, extra DOUBLE PRECISION, mta_tax DOUBLE PRECISION, tip_amount DOUBLE
PRECISION, tolls_amount DOUBLE PRECISION, improvement_surcharge DOUBLE PRECISION,
total_amount DOUBLE PRECISION, congestion_surcharge DOUBLE PRECISION, airport_fee DOUBLE
PRECISION);
-- Déconnectez de la base de données source
SELECT dblink_disconnect('taxiyolow_connection');
```

Ce code SQL réalise le transfert de données entre deux bases de données. Il utilise l'extension dblink pour se connecter à une base de données source, extrait des données spécifiques de la table `yellow_tripdata`, puis insère ces données dans une table locale appelée "Datamart.faits_trajet_taxi_partitioned". Enfin, la connexion à la base de données source est fermée. En résumé, le code facilite le mouvement de données entre bases de données distantes dans un scénario d'intégration de données.

4. Visualisation et Analyse des Données

Pour l'exploration de nos données à l'aide de Power BI, un outil de visualisation. Avec des graphiques et des tableaux interactifs, pour rendre les données plus compréhensibles et accessibles

Notre tableau de bord Power BI offre une vision holistique et dynamique des données liées aux trajets de taxi. Il intègre judicieusement des segments, des tableaux, des graphiques à barres empilées et des graphiques en courbe pour fournir une analyse approfondie des différents aspects des trajets de taxi.

Segments :

Les segments divisent les données en catégories significatives, permettant une vue rapide et interactive des informations. Ils facilitent le filtrage dynamique pour se concentrer sur des sous-ensembles spécifiques des données.

Tableau :

Le tableau présente de manière tabulaire des données brutes, offrant une référence rapide aux détails spécifiques des trajets de taxi. Il permet une exploration détaillée pour une prise de décision informée.

Graphique à Barres Empilées :

Le graphique à barres empilées illustre le nombre de passagers pour chaque mois, fournissant une vue claire de la répartition du nombre de passagers tout au

long de la période analysée. Chaque barre représente un mois, et les segments empilés montrent la répartition du nombre de passagers pour chaque trajet.

Graphique en Courbe :

Le graphique en courbe représente la somme de la congestion_surcharge pour chaque mois, permettant d'identifier les tendances et les fluctuations dans les frais de congestion sur la période. Cette visualisation temporelle aide à comprendre les variations mensuelles des frais de congestion.

Conclusion

Ces tps offre une solution complète, de l'extraction des données brutes à partir de sources externes jusqu'à la création d'un Datamart OLAP avec des visualisations interactives dans Tableau ou Power BI. La combinaison de Minio pour le Data Lake, PostgreSQL pour le stockage initial, et PostgreSQL pour le Datamart offre une solution robuste pour l'analyse de données de trajets de taxi à New York.