Rapport de Projet: Data Warehouse avec Delta Lake

Adil H.
Université Paris-Dauphine

2 Janvier 2026

Résumé

Ce rapport présente la conception et l'implémentation d'un data warehouse moderne utilisant l'architecture Lakehouse avec Delta Lake. Le pipeline ETL intègre des données depuis PostgreSQL vers un système de stockage en trois couches (Bronze, Silver, Gold) en utilisant Apache Spark et PySpark. Le projet démontre les principes d'ingénierie de données, le traitement distribué et la génération de rapports business.

Table des matières

1 Introduction

1.1 Contexte du Projet

Ce projet s'inscrit dans le cadre du cours d'Ingénierie des Données et vise à construire un data warehouse complet en utilisant des technologies modernes du Big Data.

1.2 Objectifs

— Implémenter un pipeline ETL complet
— Démontrer l'architecture Lakehouse
— Utiliser Delta Lake pour le stockage transactionnel
— Générer des rapports business automatisés

2 Architecture Technique

2.1 Stack Technologique

Composant	**Technologie**
Traitement	Apache Spark 3.5.0
Stockage	Delta Lake 3.0.0
Langage	Python 3.10 / PySpark
Base Source	PostgreSQL 15
OS	Windows 10

TABLE 1 – Stack technologique utilisée

2.2 Architecture Lakehouse

L'architecture implémentée suit le modèle **Medallion (Bronze-Silver-Gold)** :

Bronze Layer (Raw Data) : Données brutes ingérées directement depuis PostgreSQL avec ajout de métadonnées d'ingestion.

Silver Layer (Cleaned Data) : Données nettoyées, standardisées et enrichies avec des validations et transformations business.

Gold Layer (Business Data) : Données agrégées et optimisées pour les requêtes analytiques et les rapports business.

2.2.1 Couche Bronze

— Données brutes ingérées depuis PostgreSQL
— Format : Delta Lake
— Métadonnées ajoutées : timestamp, source
— Tables : clients, produits, ventes

2.2.2 Couche Silver

— Nettoyage et standardisation
— Validation des données
— Calcul de métriques (marges, etc.)

— Correction d'encodage

2.2.3 Couche Gold

— Agrégations business
— Métriques KPI
— Tables : ventes_quotidiennes
— Prête pour reporting

3 Implémentation

3.1 Structure des Scripts

Listing 1 – Structure des fichiers

```
TP_DataWarehouse/
        05_bronze_ingestion.py        # Ingestion PostgreSQL      Bronze
        06_verify_bronze.py           # V rification donn es
        07_silver_transformation.py # Nettoyage Silver
        08_gold_aggregation.py        # Agr gation Gold
        09_generer_rapport.py         # Reporting final
```

3.2 Configuration Spark pour Windows

Listing 2 – Configuration Spark

```
builder = SparkSession.builder \
    .appName("Bronze-Ingestion") \
    .master("local[*]") \
    .config("spark.sql.extensions",
            "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog",
            "org.apache.spark.sql.delta.catalog.DeltaCatalog")
```

3.3 Ingestion depuis PostgreSQL

Listing 3 – Lecture JDBC

```
df = spark.read \
    .format("jdbc") \
    .option("url", "jdbc:postgresql://localhost:5432/retailpro_dwh") \
    .option("dbtable", "clients_source") \
    .option("user", "postgres") \
    .option("password", "*****") \
    .option("driver", "org.postgresql.Driver") \
    .load()
```

4 Résultats

4.1 Données Traitées

Couche	Tables	Lignes	Taille
Bronze	3	9	15 KB
Silver	3	9	18 KB
Gold	1	1	5 KB

TABLE 2 – Volume de données traitées

4.2 Métriques Calculées

Métrique	Valeur
Chiffre d'Affaires Total	1 109,96 €
Nombre de Ventes	3
Panier Moyen	369,99 €
Marge Moyenne	75,29%

TABLE 3 – Métriques business générées

5 Problèmes Rencontrés et Solutions

5.1 Problème 1 : Configuration Delta Lake

Problème : Erreur "DeltaSparkSessionExtension required"
Solution : Ajouter les configurations obligatoires :

```
.config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension")
.config("spark.sql.catalog.spark_catalog",
        "org.apache.spark.sql.delta.catalog.DeltaCatalog")
```

5.2 Problème 2 : Chemins Windows

Problème : "No FileSystem for scheme 'C'"
Solution : Utiliser le préfixe `file:///` :

```
BRONZE_PATH = "file:///C:/lakehouse/bronze"
```

5.3 Problème 3 : Driver PostgreSQL

Problème : ClassNotFoundException pour JDBC
Solution : Ajouter le driver dans Spark :

```
.config("spark.jars", "file:///C:/TP_DataWarehouse/drivers/postgresql-42.7
```

6 Conclusion

6.1 Bilan Technique

Le projet a démontré avec succès :
— Pipeline ETL fonctionnel
— Architecture Lakehouse implémentée
— Intégration PostgreSQL-Spark réussie
— Reporting automatisé

6.2 Perspectives d'Amélioration

— Ajouter plus de tables Gold (CLV, tendances)
— Implémenter des data quality checks
— Automatiser avec Airflow
— Ajouter des visualisations Power BI

6.3 Compétences Développées

— Architecture de données moderne
— PySpark et traitement distribué
— Delta Lake et stockage transactionnel
— Ingénierie de données complète

7 Annexes

7.1 Commande d'Exécution Complète

```
# Activer l'environnement virtuel
venv\Scripts\activate

# Executer le pipeline
python 05_bronze_ingestion.py
python 06_verify_bronze.py
python 07_silver_transformation.py
python 08_gold_aggregation.py
python 09_generer_rapport.py
```

7.2 Structure des Données

Listing 4 – Schéma Bronze

```
clients: client_id, nom, prenom, email, ville, segment,
         created_at, updated_at, ingestion_timestamp,
         source_system, source_table

produits: produit_id, nom_produit, categorie, prix_unitaire,
          cout_achat, created_at, updated_at, ingestion_timestamp,
```

```
            source_system, source_table

ventes: vente_id, client_id, produit_id, date_vente, quantite,
        montant_total, created_at, ingestion_timestamp,
        source_system, source_table
```

7.3 Visualisation des Résultats

Diagramme d'Architecture : Un diagramme montrant le flux de données de PostgreSQL vers les couches Bronze, Silver et Gold serait recommandé pour illustrer l'architecture.

Table Gold : La table `ventes_quotidiennes` contient une ligne avec les agrégations du jour : date, nombre de ventes (3), CA total (1109.96), panier moyen (369.99).

Rapport Généré : Le script `09_generer_rapport.py` produit un rapport console avec statistiques globales, top jours, et analyse détaillée.

PROJET RÉUSSI
Toutes les fonctionnalités implémentées et validées
Pipeline opérationnel et prêt pour production