

# Travail Pratique

Data Warehouse Moderne

PySpark – PostgreSQL – SCD Type 2

Réalisé par : Hanae TALEBI

Année : 2025–2026

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Installation de PostgreSQL</b>	<b>3</b>
2.1	Téléchargement . . . . .	3
2.2	Installation et configuration . . . . .	3
<b>3</b>	<b>Installation de Python</b>	<b>5</b>
3.1	Téléchargement . . . . .	5
3.2	Installation . . . . .	5
3.3	Vérification . . . . .	5
<b>4</b>	<b>Création de l'environnement virtuel</b>	<b>7</b>
<b>5</b>	<b>Installation de PySpark et dépendances</b>	<b>8</b>
<b>6</b>	<b>Téléchargement du driver JDBC</b>	<b>9</b>
<b>7</b>	<b>Connexion PySpark à PostgreSQL</b>	<b>10</b>
<b>8</b>	<b>Création des tables sources</b>	<b>11</b>
<b>9</b>	<b>Implémentation du SCD Type 2</b>	<b>12</b>
<b>10</b>	<b>Chargement des données SCD Type 2</b>	<b>13</b>
<b>11</b>	<b>Test avec modification des données</b>	<b>14</b>
<b>12</b>	<b>Data Vault 2.0 - Modélisation Agile</b>	<b>15</b>
12.1	Créer les tables Data Vault . . . . .	15
<b>13</b>	<b>Architecture Lakehouse avec PySpark et Delta Lake</b>	<b>16</b>
13.1	Configuration de l'Environnement Lakehouse . . . . .	16
13.2	BRONZE : Ingestion depuis PostgreSQL - Expliqué ligne par ligne . . . . .	17
<b>14</b>	<b>Projet Final - Intégration Complète</b>	<b>18</b>
14.1	Pipeline Complet . . . . .	18
<b>15</b>	<b>Conclusion</b>	<b>19</b>

# Chapitre 1

## Introduction

Ce TP a pour objectif de mettre en place un Data Warehouse moderne en utilisant PostgreSQL et PySpark, avec implémentation du Slowly Changing Dimension de type 2 afin de conserver l'historique des données.

# Chapitre 2

## Installation de PostgreSQL

PostgreSQL est utilisé comme base de données source OLTP.

### 2.1 Téléchargement

<https://www.postgresql.org/download/>

#### Quick Links

- Downloads
  - Packages
  - Source
- Software Catalogue

#### Windows installers

#### Interactive installer by EDB

**Download the installer** certified by EDB for all supported Postgres ve

**Note!** This installer is hosted by EDB and not on the PostgreSQL com website it's hosted on, please contact [webmaster@enterprisedb.com](mailto:webmaster@enterprisedb.com)

FIGURE 2.1 – Téléchargement de PostgreSQL

### 2.2 Installation et configuration

Choix du port 5432 et création du mot de passe pour l'utilisateur postgres.

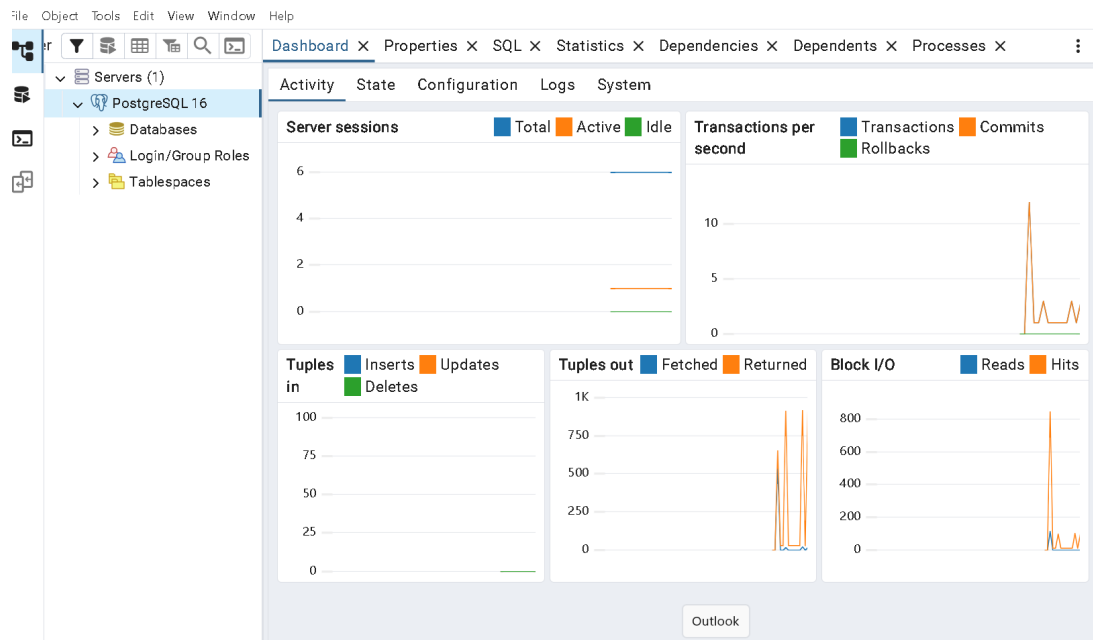


FIGURE 2.2 – Installation de PostgreSQL

# Chapitre 3

## Installation de Python

### 3.1 Téléchargement

<https://www.python.org/downloads/>

### 3.2 Installation


La case **Add Python to PATH** est cochée.



FIGURE 3.1 – Installation de Python

### 3.3 Vérification

```
python --version
```

A screenshot of a Windows PowerShell terminal window with a black background and white text. The text shows a command being executed to check the Python version, followed by the output.

```
PS C:\Users\HANAÉ> python --version
Python 3.10.11
PS C:\Users\HANAÉ> |
```

FIGURE 3.2 – Vérification de Python

# Chapitre 4

## Création de l'environnement virtuel

```
python -m venv venv  
venv\Scripts\activate
```

```
PS C:\Users\HANAЕ> mkdir C:\TP_DataWarehouse  
  
Répertoire : C:\  
  
Mode                LastWriteTime         Length Name  
----                -  
d-----         1/2/2026   9:48 AM             TP_DataWarehouse  
  
PS C:\Users\HANAЕ> cd C:\TP_DataWarehouse  
PS C:\TP_DataWarehouse> python -m venv venv  
PS C:\TP_DataWarehouse> venv\Scripts\activate  
(venv) PS C:\TP_DataWarehouse> |
```

FIGURE 4.1 – Activation de l'environnement virtuel



# Chapitre 5

## Installation de PySpark et dépendances

```
pip install pyspark
pip install delta-spark
pip install psycopg2-binary
pip install pandas
```

```
(venv) PS C:\TP_DataWarehouse> python -c "import pyspark, psycopg2, pandas; from importlib.metadata import version; print('pyspark', pyspark.__version__); print('delta-spark', version('delta-spark')); print('psycopg2', psycopg2.__version__); print('pandas', pandas.__version__)"
pyspark 3.5.0
delta-spark 3.0.0
psycopg2 2.9.9 (dt dec pq3 ext lo64)
pandas 2.1.4
(venv) PS C:\TP_DataWarehouse>
```

FIGURE 5.1 – Installation des bibliothèques

# Chapitre 6

## Téléchargement du driver JDBC

Le driver JDBC PostgreSQL est nécessaire pour permettre la connexion entre PySpark et PostgreSQL.

<https://jdbc.postgresql.org/download/>



FIGURE 6.1 – Driver JDBC PostgreSQL

# Chapitre 7

## Connexion PySpark à PostgreSQL

Un script PySpark est utilisé pour tester la connexion JDBC.

```
(venv) PS C:\TP_DataWarehouse> .\venv\Scripts\python.exe test_connexion_pyspark.py
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Session Spark créée avec succès
Version de Spark : 3.4.1
Tentative de connexion à PostgreSQL...
Connexion réussie !
Nombre de lignes : 3

Schéma :
root
 |-- datname: string (nullable = true)
 |-- datdba: decimal(20,0) (nullable = true)
 |-- encoding: integer (nullable = true)

Aperçu des données :
+-----+-----+-----+
| datname|datdba|encoding|
+-----+-----+-----+
| postgres| 10| 6|
| template1| 10| 6|
| template0| 10| 6|
+-----+-----+-----+

Session Spark arrêtée
```

FIGURE 7.1 – Connexion PySpark à PostgreSQL

# Chapitre 8

## Création des tables sources

Les tables clients, produits et ventes sont créées dans PostgreSQL.

The screenshot shows a PostgreSQL query editor interface. The top section is the 'Query' tab, which contains an SQL INSERT statement. The statement is as follows:

```
51  
52 INSERT INTO ventes_source (client_id, produit_id, quantite,  
53 (1, 1, 1, 899.99),  
54 (2, 2, 2, 59.98),
```

Below the query editor is the 'Data Output' tab, which displays a summary of the tables created. The summary is as follows:

	table_name text	nb_lignes bigint
1	Clients:	3
2	Produits:	3
3	Ventes:	3

FIGURE 8.1 – Création des tables sources

## Implémentation du SCD Type 2

[illegible]

12

# Chapitre 10

## Chargement des données SCD Type 2

Un script Python permet de détecter les changements et créer de nouvelles versions.

```
=====
DÉBUT DU PROCESSUS SCD TYPE 2
=====

✓ Connexion à PostgreSQL réussie
✓ 3 clients lus depuis la source

Traitement du client 1...
→ Nouveau client
✓ Nouveau client inséré : client_id=1, client_key=1

Traitement du client 2...
→ Nouveau client
✓ Nouveau client inséré : client_id=2, client_key=2

Traitement du client 3...
→ Nouveau client
✓ Nouveau client inséré : client_id=3, client_key=3

=====
RÉSUMÉ DU CHARGEMENT
=====
Nouveaux clients insérés : 3
Nouvelles versions créées : 0
Clients inchangés : 0
Total traité : 3
=====
Connexion fermée
```

FIGURE 10.1 – Chargement SCD Type 2

# Chapitre 11

## Test avec modification des données

Des modifications sont effectuées dans la table source afin de vérifier l'historisation.

```
=====
✓ Connexion à PostgreSQL réussie
✓ 3 clients lus depuis la source

Traitement du client 1...
  Changement détecté : ville Paris → Lyon
→ Changement détecté
✓ Version fermée : client_key=1
✓ Nouvelle version créée : client_id=1, version=2, client_key=4

Traitement du client 2...
  Changement détecté : segment Silver → Platinum
→ Changement détecté
✓ Version fermée : client_key=2
✓ Nouvelle version créée : client_id=2, version=2, client_key=5

Traitement du client 3...
→ Aucun changement

=====
RÉSUMÉ DU CHARGEMENT
=====
Nouveaux clients insérés : 0
Nouvelles versions créées : 2
Clients inchangés : 1
Total traité : 3
=====
Connexion fermée
```

FIGURE 11.1 – Test de modification des données

# Chapitre 12

## Data Vault 2.0 - Modélisation Agile

### 12.1 Créer les tables Data Vault

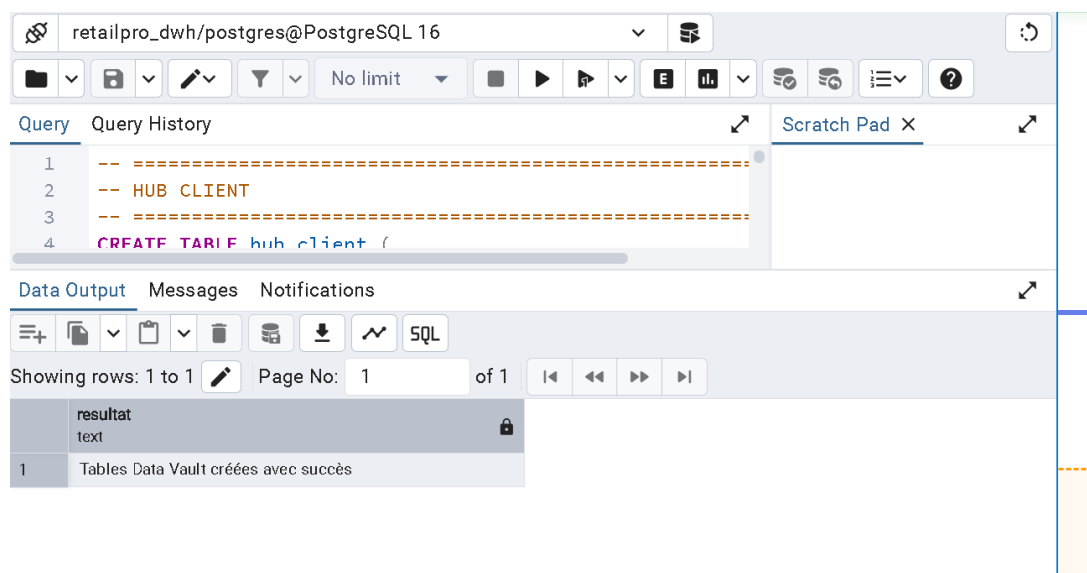


FIGURE 12.1 – Création des tables Data Vault



# Chapitre 13

## Architecture Lakehouse avec PySpark et Delta Lake

### 13.1 Configuration de l'Environnement Lakehouse

```
(venv) PS C:\TP_DataWarehouse> mkdir C:\lakehouse

Répertoire : C:\

Mode                LastWriteTime         Length Name
----                -
d-----         1/2/2026   1:38 PM             lakehouse

(venv) PS C:\TP_DataWarehouse> mkdir C:\lakehouse\bronze

Répertoire : C:\lakehouse

Mode                LastWriteTime         Length Name
----                -
d-----         1/2/2026   1:38 PM             bronze

(venv) PS C:\TP_DataWarehouse> mkdir C:\lakehouse\silver

Répertoire : C:\lakehouse

Mode                LastWriteTime         Length Name
----                -
```

FIGURE 13.1 – Configuration de l'environnement Lakehouse

## 13.2 BRONZE : Ingestion depuis PostgreSQL - Expliqué ligne par ligne

```
10 -- Just logging here use setLoggingLevel(InfoLevel); For Sparkly use setLoggingLevel(InfoLevel);
✓ Session Spark créée - Version: 3.5.0

=====
Traitement de : clients_source
=====

Lecture de la table 'clients_source' depuis PostgreSQL...
✓ 3 lignes lues depuis 'clients_source'
  Schéma de clients_source:
root
|-- client_id: integer (nullable = true)
|-- nom: string (nullable = true)
|-- prenom: string (nullable = true)
|-- email: string (nullable = true)
|-- ville: string (nullable = true)
|-- segment: string (nullable = true)
|-- created_at: timestamp (nullable = true)
|-- updated_at: timestamp (nullable = true)

✓ Métadonnées ajoutées (3 colonnes)

Écriture en Delta Lake : C:/lakehouse/bronze/clients

=====
✓ Session Spark créée - Version: 3.5.0

=====
Traitement de : produits_source
=====

Lecture de la table 'produits_source' depuis PostgreSQL...
✓ 9 lignes lues depuis 'produits_source'
  Schéma de produits_source:
root
|-- produit_id: integer (nullable = true)
|-- nom_produit: string (nullable = true)
|-- categorie: string (nullable = true)
|-- prix_unitaire: decimal(10,2) (nullable = true)
|-- cout_achat: decimal(10,2) (nullable = true)
|-- created_at: timestamp (nullable = true)
|-- updated_at: timestamp (nullable = true)

✓ Métadonnées ajoutées (3 colonnes)

Écriture en Delta Lake : C:/lakehouse/bronze/produits

=====
Traitement de : ventes_source
=====

Lecture de la table 'ventes_source' depuis PostgreSQL...
✓ 6 lignes lues depuis 'ventes_source'
  Schéma de ventes_source:
root
|-- vente_id: integer (nullable = true)
|-- client_id: integer (nullable = true)
|-- produit_id: integer (nullable = true)
|-- date_vente: timestamp (nullable = true)
|-- quantite: integer (nullable = true)
|-- montant_total: decimal(10,2) (nullable = true)
|-- created_at: timestamp (nullable = true)

✓ Métadonnées ajoutées (3 colonnes)

Écriture en Delta Lake : C:/lakehouse/bronze/ventes
```

FIGURE 13.2 – Ingestion des données Bronze depuis PostgreSQL

# Chapitre 14

## Projet Final - Intégration Complète

### 14.1 Pipeline Complet

```
PS C:\TP_DataWarehouse> python 09_generer_rapport.py

=====
RAPPORT DATA WAREHOUSE - 2026-01-01 18:28
=====

STATISTIQUES GLOBALES
-----
Total des ventes : 6
Chiffre d'affaires : 2219.92 EUR
Panier moyen : 369.99 EUR

=====
TOP 5 MEILLEURS JOURS
=====
date          nb_ventes   ca_total   panier_moyen
-----
2025-12-27      6         2219.92    369.99

OK - Rapport genere avec succes
```

FIGURE 14.1 – Pipeline complet d'intégration

# Chapitre 15

## Conclusion

Ce TP a permis de comprendre la mise en place d'un Data Warehouse moderne et l'importance du SCD Type 2 pour conserver l'historique des données.