Rapport TP3 : Data Lake et Data Warehouse

Objectif du TP

Ce TP vise à construire une chaîne d'intégration de données en temps réel à partir de Kafka, avec stockage dans un Data Lake et un Data Warehouse. Il inclut également des mécanismes d'orchestration, de sécurité et de gouvernance.

---

1. Structure du Data Lake

- **Emplacement** : data_lake/all_transactions/
- **Partitionnement** : par date YYYY-MM-DD
- **Format** : fichiers JSON, un message par ligne (mode append)
- **Exemple de chemin** : data_lake/all_transactions/2025-05-14/batch.json
- **Ajout de nouveaux topics** : dynamique via fichier kafka_config.json

Pourquoi append ? Les streams Kafka étant infinis, le mode append est adapté. Chaque exécution ajoute les messages dans un fichier journalisé par date.

---

2. Structure du Data Warehouse (MySQL)

Structure de la base datawarehouse et table all_transactions peuplée automatiquement depuis Kafka

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| datawarehouse      |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.09 sec)

mysql> USE datawarehouse;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+------------------------+
| Tables_in_datawarehouse |
+------------------------+
| all_transactions       |
+------------------------+
1 row in set (0.00 sec)
```

```
mysql> DESCRIBE all_transactions;
+---------------------------+--------------+------+-----+---------+-------+
| Field                     | Type         | Null | Key | Default | Extra |
+---------------------------+--------------+------+-----+---------+-------+
| transaction_id            | varchar(100) | NO   | PRI | NULL    |       |
| timestamp                 | varchar(50)  | YES  |     | NULL    |       |
| user_id                   | varchar(100) | YES  |     | NULL    |       |
| user_name                 | varchar(100) | YES  |     | NULL    |       |
| product_id                | varchar(100) | YES  |     | NULL    |       |
| amount                    | double       | YES  |     | NULL    |       |
| currency                  | varchar(10)  | YES  |     | NULL    |       |
| transaction_type          | varchar(50)  | YES  |     | NULL    |       |
| status                    | varchar(50)  | YES  |     | NULL    |       |
| city                      | varchar(100) | YES  |     | NULL    |       |
| country                   | varchar(100) | YES  |     | NULL    |       |
| payment_method            | varchar(50)  | YES  |     | NULL    |       |
| product_category          | varchar(100) | YES  |     | NULL    |       |
| quantity                  | int          | YES  |     | NULL    |       |
| shipping_street           | varchar(255) | YES  |     | NULL    |       |
| shipping_zip              | varchar(20)  | YES  |     | NULL    |       |
| shipping_city             | varchar(100) | YES  |     | NULL    |       |
| shipping_country          | varchar(100) | YES  |     | NULL    |       |
| device_os                 | varchar(50)  | YES  |     | NULL    |       |
| device_browser            | varchar(50)  | YES  |     | NULL    |       |
| device_ip                 | varchar(100) | YES  |     | NULL    |       |
| customer_rating           | int          | YES  |     | NULL    |       |
| discount_code             | varchar(50)  | YES  |     | NULL    |       |
| tax_amount                | double       | YES  |     | NULL    |       |
| thread                    | int          | YES  |     | NULL    |       |
| message_number            | int          | YES  |     | NULL    |       |
| timestamp_of_reception_log | varchar(100) | YES  |     | NULL    |       |
+---------------------------+--------------+------+-----+---------+-------+
27 rows in set (0.09 sec)
```

```
mysql> SELECT * FROM all_transactions LIMIT 5;
| transaction_id                       | timestamp                  | user_id | user_name | product_id | amount | currency | transaction_type | status    | city  | country | payment_method | product_category | quantity | ship
ing_street | shipping_zip | shipping_city | shipping_country | device_os | device_browser | device_ip   | customer_rating | discount_code | tax_amount | thread | message_number | timestamp_of_reception_log |
| fa270481-ff75-4e43-bc22-b8c0f606f1e8 | 2025-05-14T11:26:01.076909 | U001    | John Doe  | P123       | 299.99 | USD      | purchase         | completed | Paris | France  | credit_card    | electronics      |        1 | 123
ain St     | 75000        | Paris         | France           | Windows   | Chrome         | 192.168.1.1 |               5 | SPRING10      |         20 |      1 |              1 | 2025-05-14T11:26:01.076920 |
| TX12345                              | 2025-04-30T10:19:53.709739 | U001    | John Doe  | P123       | 299.99 | USD      | purchase         | completed | Paris | France  | credit_card    | electronics      |        1 | 123
ain St     | 75000        | Paris         | France           | Windows   | Chrome         | 192.168.1.1 |               5 | SPRING10      |         20 |      1 |              1 | 2025-04-30T10:19:53.709750 |
2 rows in set (0.00 sec)
```

Table principale : all_transactions

- Clé primaire : transaction_id
- Champs stockés : tous les attributs JSON dénormalisés

Autres tables

- **user_permissions** : pour gérer les droits d'accès aux dossiers du Data Lake

CREATE TABLE user_permissions (

username VARCHAR(100),

datalake_path VARCHAR(255),

permission_level ENUM('read', 'write', 'admin'),

PRIMARY KEY (username, datalake_path)

);

---

3. Orchestration avec Apache Beam

- Script : orchestration/dataflow_scheduler.py
- Bibliothèques : apache-beam, schedule
- Fréquence : toutes les 10 minutes
- Fonction : simulateur de tâche planifiée

schedule.every(10).minutes.do(run_job)

---

4. Gouvernance & Sécurité

Permissions d'accès

- Table user_permissions pour contrôler qui accède à quel dossier du Data Lake

Suppression des données historiques

- Script utils/cleanup_datalake.py
- Supprime les sous-dossiers plus vieux que 7 jours

5. Ajout de nouveaux feeds Kafka

- **Fichier de config** : config/kafka_config.json

- Les consumers lisent tous les topics listés dynamiquement.
- Réutilisation des scripts kafka_to_datalake.py et kafka_to_mysql.py sans duplication de code.

---

6. Monitoring

- Chaque script affiche un log en temps réel ("Transaction insérée", "Message écrit...")
- En cas d'échec (ex: MySQL down), les erreurs sont logguées grâce à des try/except

---