Rapport de Projet Data Warehouse – École de Danse

1. Introduction

Ce projet s'inscrit dans le cadre de la formation DP-600 et vise à concevoir et implémenter une architecture DataWarehouse complète pour une école de danse. L'objectif principal est d'analyser les données d'inscriptions, de tarification et de performance commerciale de l'établissement.

2. Objectifs du Projet

- Concevoir une architecture de données de type Lakehouse avec une zone de staging et une zone de Data Warehouse.
- Permettre une analyse multidimensionnelle des inscriptions aux cours de danse.
- Implémenter plusieurs types de gestion de l'historique (Slowly Changing Dimensions – SCD) pour suivre l'évolution des données.
- Fournir des rapports analytiques interactifs avec Power BI.

3. Données Sources

Les données proviennent de plusieurs fichiers CSV représentant différents aspects de l'activité de l'école de danse :

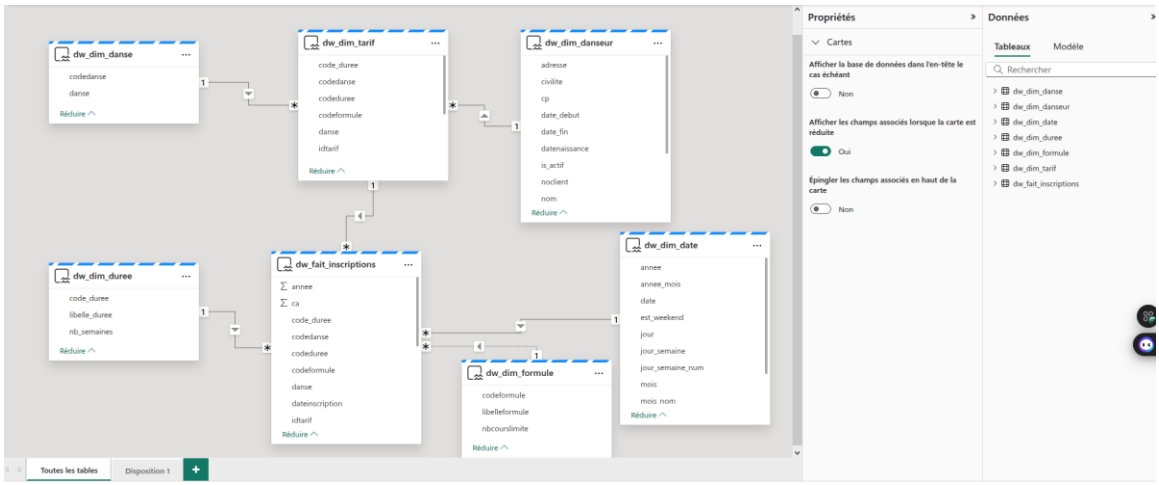
- `danseurs.csv`: Informations personnelles des clients.
- `inscriptions.csv`: Historique des inscriptions aux cours.
- `tarifs.csv`: Tarification des cours.
- `formule.csv`: Informations sur les formules proposées.
- `duree.csv`: Durée des sessions de formation.
- `danse.csv`: Type de danse proposée.

Chaque fichier est chargé dans une table staging avec nettoyage des noms de colonnes pour respecter les bonnes pratiques d'intégration de données.

4. Architecture du Data Warehouse

4.1 Modèle Conceptuel de Données (MCD)

Le modèle dimensionnel adopte une **architecture en étoile** classique avec une table de faits centrale et 6 dimensions principales :



4.2 Description des Dimensions

4.2.1 DIM_DATE (Dimension Temporelle)

Type : Dimension générée automatiquement

```
# Génération de la dimension temporelle
df_dates = spark.sql(f"""
    SELECT explode(sequence(to_date('{start_date}'),
to_date('{end_date}'), interval 1 day)) as date
""")
```

Attributs clés :

- `date` : Date complète
- `jour`, `mois`, `annee` : Composants temporels
- `trimestre` : Période trimestrielle
- `jour_semaine`, `mois_nom` : Libellés textuels
- `est_weekend` : Indicateur booléen

4.2.2 DIM_DANSEUR (Dimension Client)

Type : SCD Type 2 (historisation complète) **Justification :** Les informations clients évoluent (déménagement, préférences marketing)

Attributs trackés :

- `adresse`, `cp`, `ville` : Données géographiques
- `pub` : Consentement marketing

4.2.3 DIM_FORMULE (Dimension Produit)

Type : SCD Type 1 (remplacement) **Justification :** Les formules sont définitives, pas d'historisation nécessaire

4.2.4 DIM_TARIF (Dimension enrichie)

Type : Dimension composite enrichie par jointures

```
df_tarifs_enrichis = df_tarifs \
    .join(df_formules, "codeformule", "left") \
    .join(df_durees, df_tarifs["codeduree"] == df_durees["code_duree"],
"left") \
    .join(df_danses, "codedanse", "left")
```

4.3 Table de Faits

FAIT_INSCRIPTIONS

Granularité : Une ligne par inscription **Mesures calculées :**

- `ca` : Chiffre d'affaires (montant de l'inscription)
- `nbseances` : Nombre de séances calculé selon la formule
- `tarifparseance` : Tarif unitaire par séance

Logique métier implémentée :

```
# Détermination du type de formule
df = df.withColumn(
    "typeformule",
    when(col("libelleformule").rlike("(?i)carnet"), "Carnet")
    .when(col("libelleformule").rlike("(?i)volonte"), "Hebdo")
    .otherwise("Hebdo")
)

# Calcul du nombre de séances
df = df.withColumn(
    "nbseances",
    when(col("typeformule") == "Carnet", col("nbcourslimite"))
    .when(col("libelleformule").rlike("(?i)volonte"),
col("nb_semaines") * 5)
    .otherwise(col("nb_semaines") * col("nbcourslimite"))
```

5. Traitement des données (ETL)

a. Zone de staging

Les fichiers CSV sont chargés dans Spark avec encodage et séparation personnalisée. Nettoyage des colonnes et sauvegarde dans des tables temporaires : `stg_danseurs`, `stg_inscriptions`, etc.

b. Construction des dimensions

Chaque table staging est transformée puis chargée dans la zone Gold en tant que dimension. Les types SCD sont appliqués comme suit :

- `dw_dim_formule` : SCD 1 (remplacement)
- `dw_dim_danseur` : SCD 2 (historisation complète)

c. Construction de la table de faits

La table `dw_fait_inscriptions` est enrichie par jointure avec `dw_dim_tarif` puis calcul :

- du nombre de séances (`nbseances`)
- du chiffre d'affaires (`ca`)
- du tarif par séance (`tarifparseance`)

d. Génération de la dimension temps

Via PySpark, nous générons dynamiquement un calendrier de dates, enrichi de colonnes : mois, semaine, trimestre, weekend, etc.

6. Visualisation Power BI

Nous avons connecté notre entrepôt de données à Power BI pour créer des visualisations dynamiques permettant d'explorer :

- Le chiffre d'affaires mensuel par danse ou formule
- La répartition des inscriptions par durée et type de formule
- La fidélisation des clients via les historiques SCD
- Les pics d'inscriptions par période (via `dw_dim_date`)

7. Dépôt Git

Ce dépôt constitue le référentiel principal du code utilisé dans Microsoft Fabric, y compris les transformations appliquées aux données sources, la génération des dimensions et de la table de faits, ainsi que la logique métier (SCD, calculs, enrichissements).

Dépôt public :
https://github.com/ada-wq/dw-fabric-danse

8. Conclusion

Ce projet nous a permis de mettre en pratique les principes fondamentaux du Data Warehousing, de la modélisation en étoile à l'application des SCD. L'intégration des données hétérogènes, la création des dimensions enrichies, et l'exploitation dans Power BI offrent un outil puissant d'analyse décisionnelle pour l'école de danse.