LST:Analytique des données

Module :  [***Ingénierie des Données.***](https://classroom.google.com/c/NzYxNjIzNzAzMTA1)

**RAPPORT :**

***Mini-projet N° 2  Analyse de Données Météorologiques***

**Réalisé Par : Encadré Par :**

**DOUAE EL MASALAM PR : Hassan ZILI**

**HAJAR EL FELLAK**

 AnnéeUniversitaire2024–2025 **Analyse et Conception du Data Warehouse Météorologique**

**1. Analyse des sources de données**

**1.1 Fichiers de dimensions**

**dim\_station**

* **Description**: Informations sur les stations météorologiques
* **Clé primaire**: code\_station
* **Attributs**: nom\_station, latitude, longitude, altitude, type\_station, date\_mise\_en\_service, est\_active
* **Observations**:
  + 30 stations météorologiques (FR001 à FR030)
  + Types variés: urbaine, aéroportuaire, montagne, littorale, etc.
  + Dates de mise en service s'étalant de 1872 à 1984
  + Une station inactive (FR030 Metz-Frescaty)

**dim\_localisation**

* **Description**: Informations géographiques et administratives des lieux de mesure
* **Clé primaire**: Position (implicite, à créer comme id\_localisation)
* **Attributs**: pays, region, departement, ville, code\_postal, type\_zone
* **Observations**:
  + Toutes les localisations sont en France
  + 13 régions représentées
  + Types de zones: urbain, littoral, montagne, rural, haute-montagne

**dim\_type\_mesure**

* **Description**: Types de mesures météorologiques effectuées
* **Clé primaire**: code\_mesure
* **Attributs**: nom\_mesure, unite, description, precision\_mesure
* **Observations**:
  + 9 types de mesures (TEMP\_MIN, TEMP\_MAX, TEMP\_MOY, PRECIP, HUMIDITE, PRESSION, VENT\_VIT, VENT\_DIR, VENT\_RAFALE)
  + Unités variées (°C, mm, %, hPa, km/h, °)
  + Différentes précisions selon les mesures

**dim\_temps**

* **Description**: Dimension temporelle avec hiérarchies et attributs
* **Clé primaire**: date\_complete
* **Attributs**: annee, mois, jour, jour\_semaine, nom\_jour, nom\_mois, trimestre, saison, est\_weekend, est\_jour\_ferie
* **Observations**:
  + Dates échantillonnées principalement en 2023
  + Inclut des jours fériés et des weekends
  + Couvre les quatre saisons

**1.2 Fichier de faits**

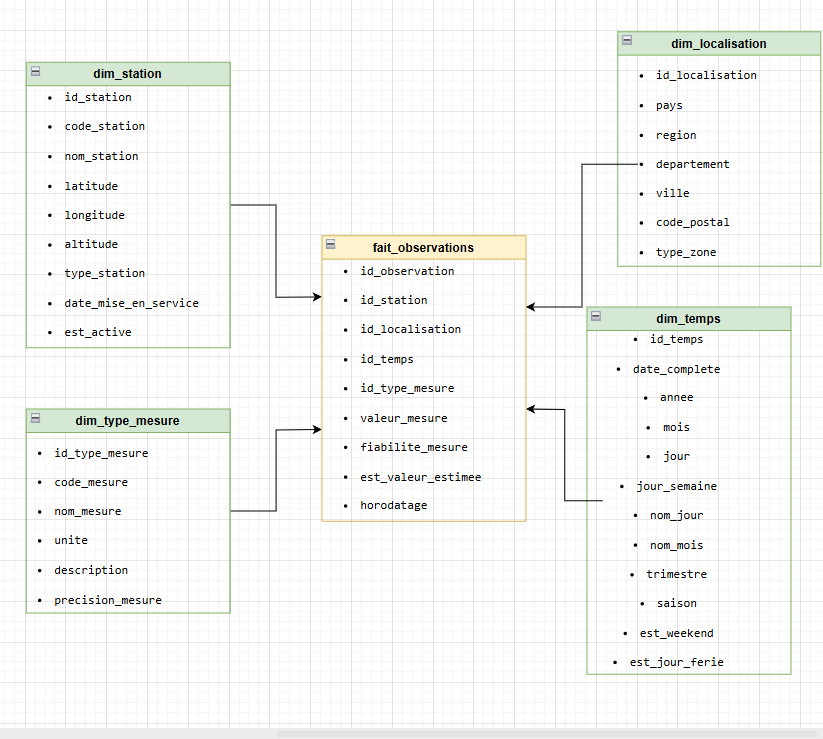
**fait\_observations**

* **Description**: Mesures météorologiques effectuées
* **Clés étrangères**: id\_station, id\_localisation, date\_complete, code\_mesure
* **Mesures**: valeur\_mesure, fiabilite\_mesure, est\_valeur\_estimee
* **Observations**:
  + Mesures prises à différentes heures de la journée (horodatage)
  + Fiabilité des mesures généralement entre 85% et 98%
  + Peu de valeurs estimées dans l'échantillon
  + Données pour plusieurs jours clés (01/01/2023, 01/05/2023, 14/07/2023, etc.)

**2. Conception du modèle dimensionnel**

**2.1 Schéma en étoile proposé**

Le modèle dimensionnel suivant est proposé:



1. **Table de faits**: fait\_observations
   * **Clés étrangères**: id\_station, id\_localisation, date\_complete, code\_mesure
   * **Mesures**: valeur\_mesure, fiabilite\_mesure, est\_valeur\_estimee, horodatage
2. **Tables de dimensions**:
   * dim\_station (clé: code\_station)
   * dim\_localisation (clé: id\_localisation)
   * dim\_temps (clé: date\_complete)
   * dim\_type\_mesure (clé: code\_mesure)

**2.2 Relations et contraintes d'intégrité**

* Chaque observation dans fait\_observations est associée à exactement:
  + Une station météorologique
  + Une localisation géographique
  + Une date
  + Un type de mesure
* Contraintes d'intégrité:
  + Les valeurs de fiabilité des mesures sont comprises entre 0 et 100
  + Les dates dans la table de faits doivent exister dans la dimension temps
  + Les stations référencées doivent exister dans la dimension station

**3. Conclusion :**

Le modèle dimensionnel proposé permet de:

* Analyser les données météorologiques selon diverses dimensions (géographique, temporelle, type de mesure)
* Effectuer des agrégations à différents niveaux (par région, par saison, par type de station)
* Suivre l'évolution des mesures dans le temps et l'espace

# 4. Installation et Configuration d'Apache NiFi

## 4.1 Prérequis et environnement technique

### 4.1.1 Configuration système requise

* **Système d'exploitation** : Windows 10/11, Linux Ubuntu 20.04+ ou macOS
* **Mémoire RAM** : Minimum 4 Go (8 Go recommandés pour les performances optimales)
* **Espace disque** : 2 Go d'espace libre minimum
* **Java** : OpenJDK 8 ou Oracle JDK 8+ (jusqu'à Java 11 supporté)
* **Processeur** : Architecture x64

### 4.1.2 Vérification de l'environnement Java

Avant l'installation de NiFi, il est essentiel de vérifier la présence et la version de Java :

java -version

javac -version

echo $JAVA\_HOME # Linux/macOS

echo %JAVA\_HOME% # Windows

Si Java n'est pas installé, télécharger et installer OpenJDK depuis <https://adoptopenjdk.net/>.

## 4.2 Installation d'Apache NiFi

### 4.2.1 Téléchargement

1. Accéder au site officiel Apache NiFi : <https://nifi.apache.org/download.html>
2. Télécharger la version stable la plus récente (actuellement 1.23.2)
3. Choisir le format adapté :
   * **Windows** : nifi-1.23.2-bin.zip

### 4.2.2 Extraction et installation

**Windows :**

# Extraire le fichier ZIP dans C:\nifihajar

# Créer une variable d'environnement NIFI\_HOME pointant vers C:\nifihajar

## 4.3 Configuration initiale de NiFi

### 4.3.1 Configuration de la mémoire JVM

Modifier le fichier $NIFI\_HOME/conf/bootstrap.conf pour ajuster l'allocation mémoire selon les ressources disponibles :

# Mémoire heap Java (ajuster selon la RAM disponible)

java.arg.2=-Xms2g

java.arg.3=-Xmx4g

# Mémoire pour les métadonnées de classes

java.arg.13=-XX:MaxMetaspaceSize=512m

### 4.3.2 Configuration des ports et sécurité

Éditer le fichier $NIFI\_HOME/conf/nifi.properties :

# Port d'écoute de l'interface web

nifi.web.http.port=8443

nifi.web.http.host=localhost

### 4.3.3 Configuration de la base de données PostgreSQL

Dans le même fichier nifi.properties, configurer la connexion à PostgreSQL :

# Configuration de la base de données pour l'état de NiFi

nifi.database.directory=./database\_repository

nifi.h2.url.append=;LOCK\_TIMEOUT=25000;WRITE\_DELAY=0;AUTO\_SERVER=FALSE

## 4.4 Démarrage et vérification de NiFi

### 4.4.1 Démarrage du service

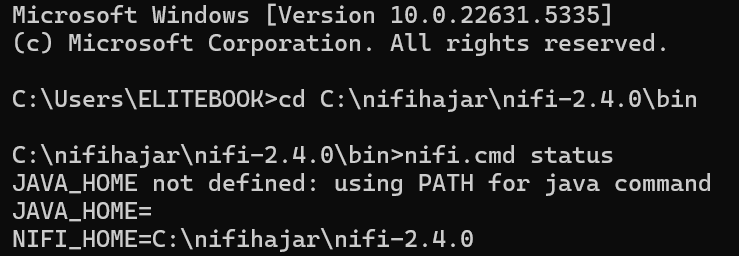
**Windows :**

cd %NIFI\_HOME%

bin\nifi.bat start

# Vérifier le statut

bin\nifi.bat status



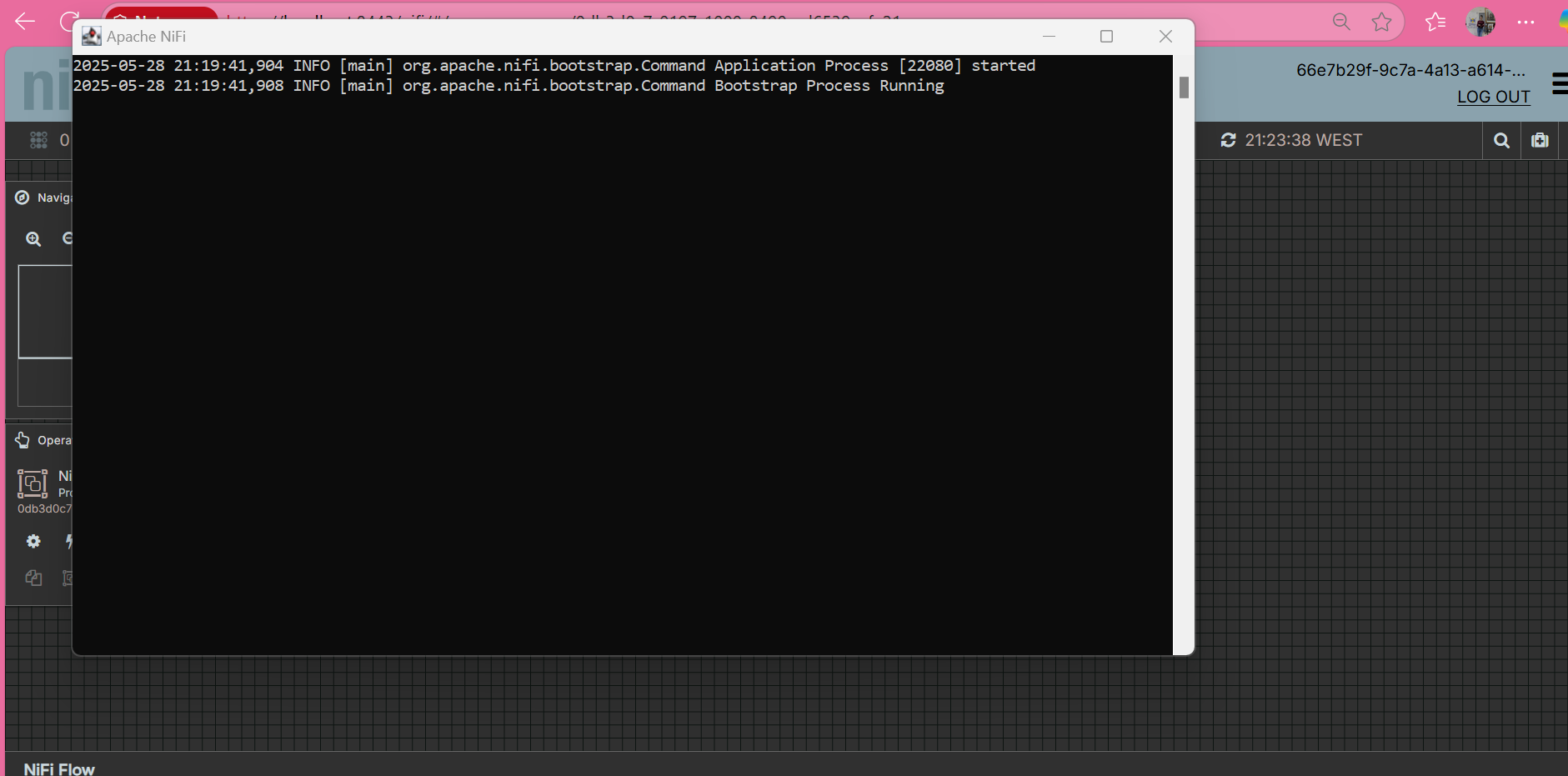
### 4.4.2 Vérification du démarrage

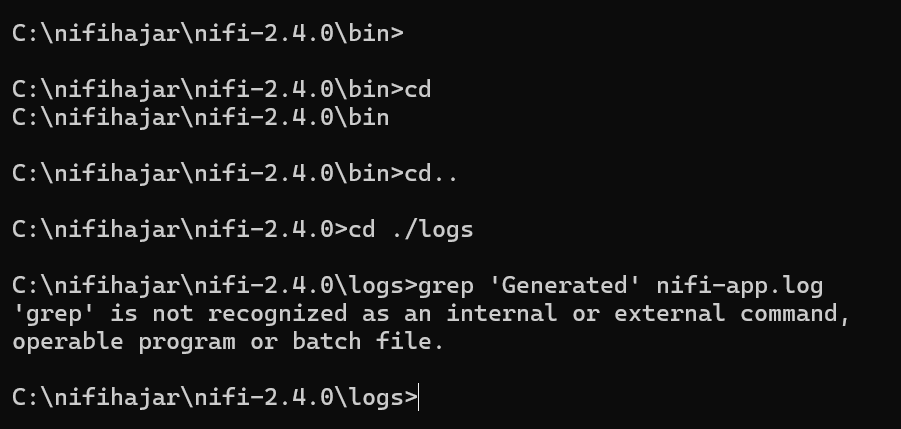
1. **Logs de démarrage** : Surveiller le fichier logs/nifi-app.log pour détecter d'éventuelles erreurs
2. **Test de connectivité** : Accéder à l'interface web via <http://localhost:8443/nifi>

### 4.5 Connexion à l'interface NiFi et première utilisation

#### 4.5.1 Connexion initiale

Lors du premier démarrage, Apache NiFi génère un mot de passe temporaire pour l’utilisateur admin. Ce mot de passe est enregistré dans le fichier de log nifi-app.log.





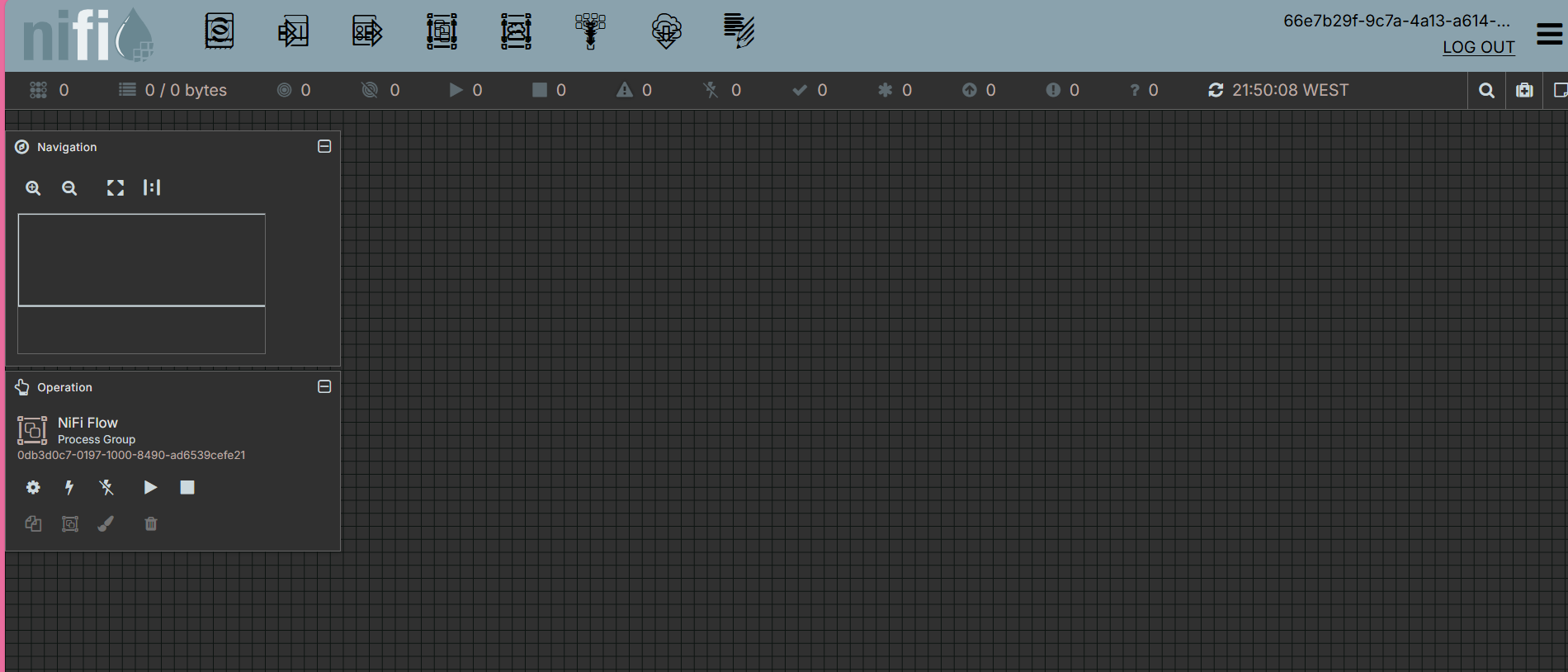
Étapes pour récupérer le mot de passe :

1. Ouvrir le fichier :
   * « C:\nifihajar\nifi-2.4.0\logs\nifi-app.log »
2. Renseigner :

* **Nom d’utilisateur** : 66e7b29f-9c7a-4a13-a614-a14525cc6ab5
* **Mot de passe** : 28criUf1IoytvDb/HhAm6CiTdlIOvo8c

## 4.5.2 Exploration de l'interface utilisateur

Après connexion réussie à l'interface NiFi via <http://localhost:8443/nifi>, l'environnement de travail se décompose en quatre zones fonctionnelles distinctes :



**Zone de composants** L'interface présente une palette organisée en catégories principales :

* **Processors** : Plus de 350 composants spécialisés pour l'extraction, la transformation et le chargement de données
* **Input Ports** : Points d'entrée standardisés pour recevoir des flux de données externes
* **Output Ports** : Points de sortie permettant l'export vers d'autres systèmes
* **Process Groups** : Conteneurs logiques facilitant l'organisation et la réutilisation des flux complexes
* **Remote Process Groups** : Composants dédiés à la connexion avec d'autres instances NiFi distantes
* **Funnels** : Éléments de convergence permettant la fusion de plusieurs flux de données
* **Templates** : Flux préconfigurés et réutilisables pour des cas d'usage standards
* **Labels** : Annotations textuelles pour la documentation des flux

**Canevas de conception** La zone de travail principale offre un environnement graphique intuitif où :

* Les composants sont positionnés par glisser-déposer depuis la palette
* Les connexions entre processeurs s'établissent visuellement
* La navigation s'effectue par zoom et déplacement sur des flux complexes

**Barre de contrôle** Les fonctionnalités de gestion globale comprennent :

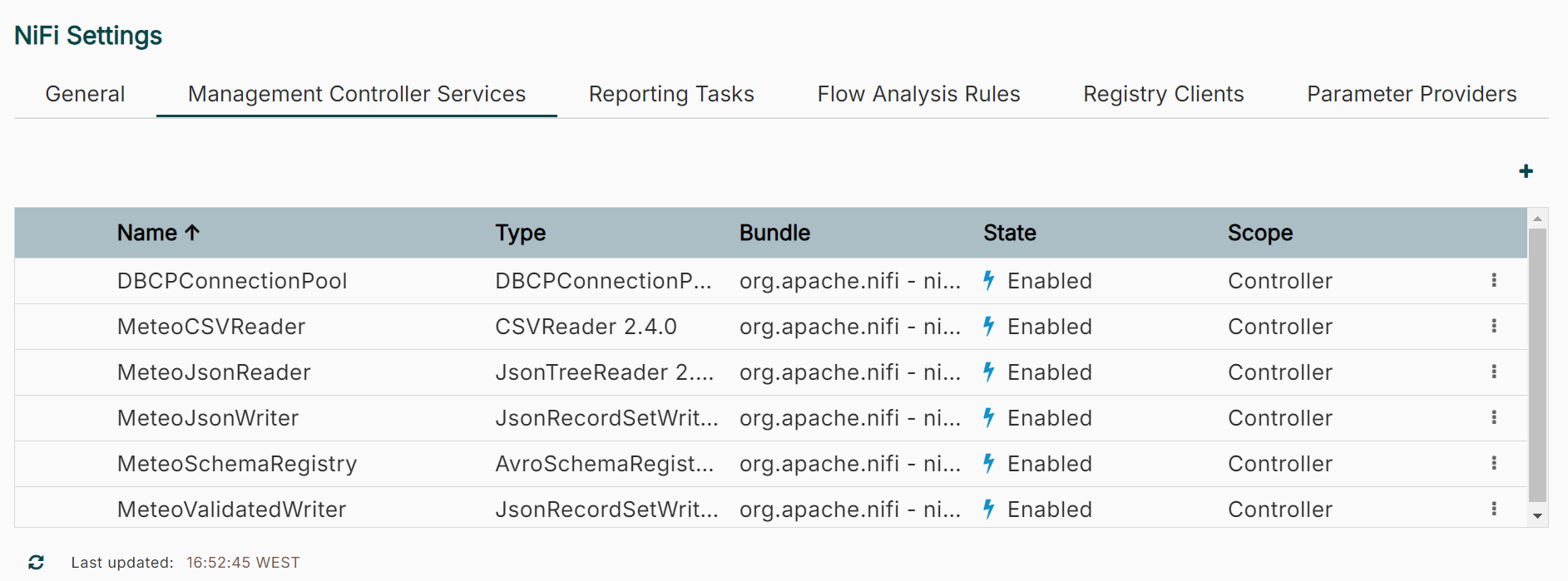
* **Flow Control** : Démarrage/arrêt des flux individuels ou globaux
* **Settings** : Configuration des paramètres généraux de l'instance NiFi
* **Controller Services** : Gestion centralisée des services partagés
* **Data Provenance** : Traçabilité complète des données traitées
* **Summary** : Tableau de bord des performances et statistiques

**Panneau de propriétés** Cet espace contextuel affiche dynamiquement :

* La configuration détaillée du composant sélectionné
* La documentation intégrée avec exemples d'usage
* Les relations et connexions du processeur
* L'historique des modifications apportées

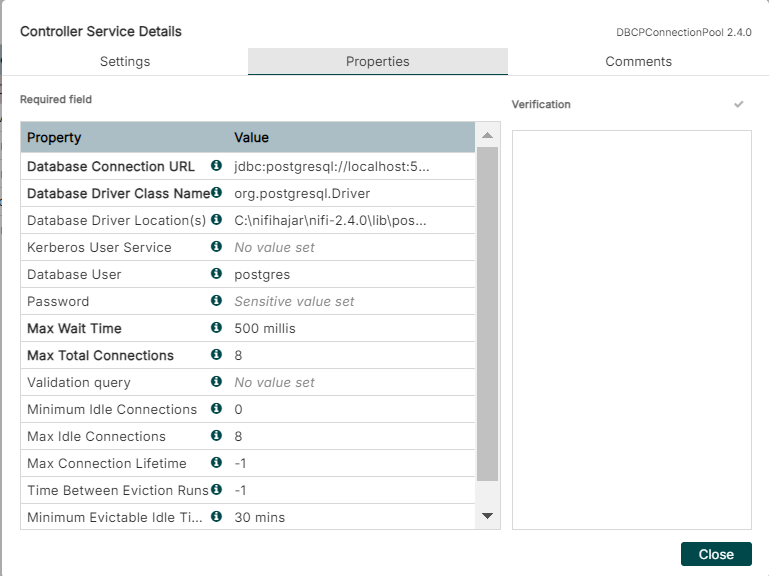
## 4.6 Configuration des Controller Services

Les Controller Services constituent l'épine dorsale de l'architecture NiFi, fournissant des services partagés réutilisables par l'ensemble des processeurs du flux.



## 4.6.1 Service de connexion à PostgreSQL

Le service **DBCPConnectionPool** a été configuré pour établir la connexion avec la base de données PostgreSQL du data warehouse météorologique. Ce service gère un pool de connexions optimisé avec 8 connexions maximales et utilise le driver PostgreSQL JDBC pour assurer une connectivité stable vers la base meteo\_dw.



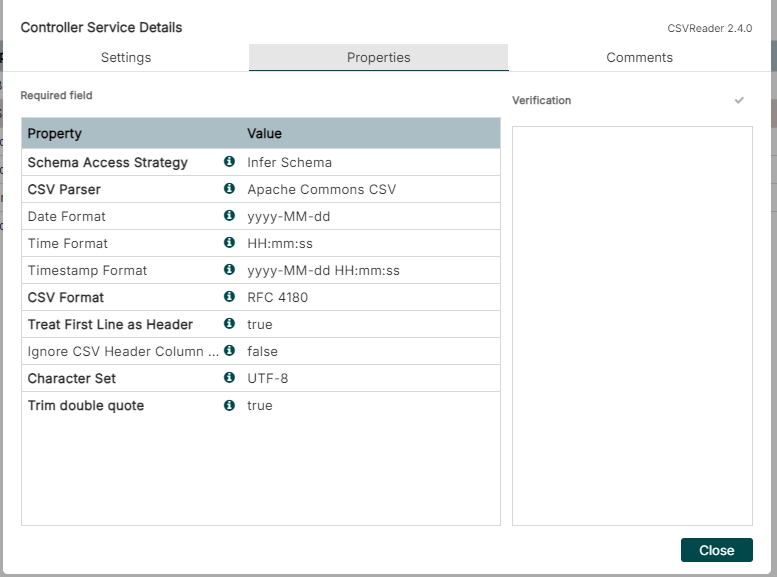
## 4.6.2 Services de lecture et d'écriture des données

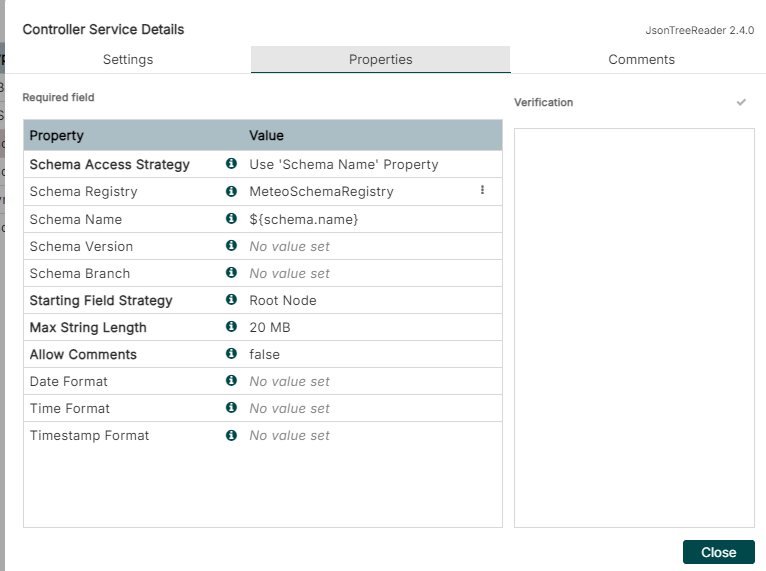
Trois services principaux ont été mis en place pour la gestion des formats de données :

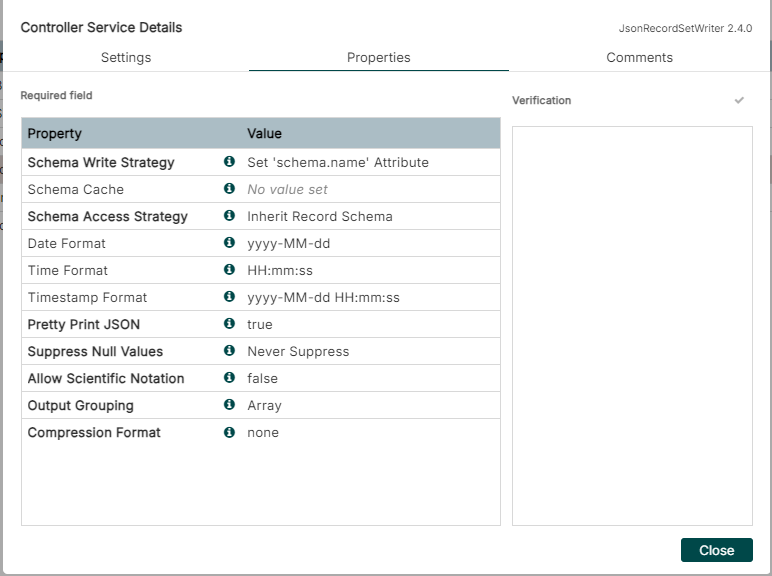
**CSVReader** : Ce service standardise la lecture des fichiers CSV météorologiques en traitant automatiquement les en-têtes, les délimiteurs et les caractères d'échappement. Il garantit une interprétation cohérente des données sources.

**CSVRecordSetWriter** : Configuré pour l'écriture des données transformées au format CSV avec encodage UTF-8 et gestion des caractères spéciaux présents dans les données météorologiques françaises.

**JsonRecordSetWriter** : Utilisé pour les transformations intermédiaires nécessitant un format JSON, notamment lors des enrichissements de données et des validations complexes.

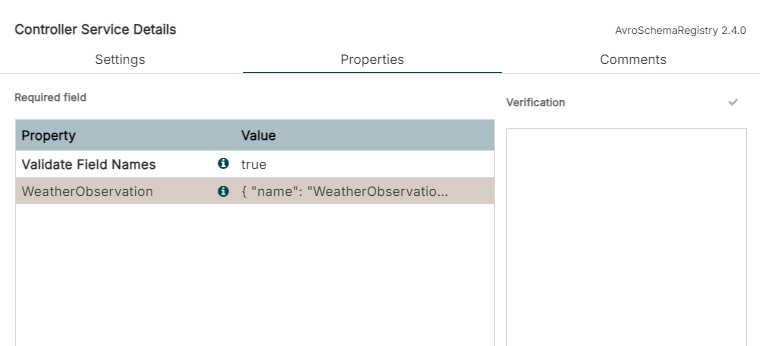




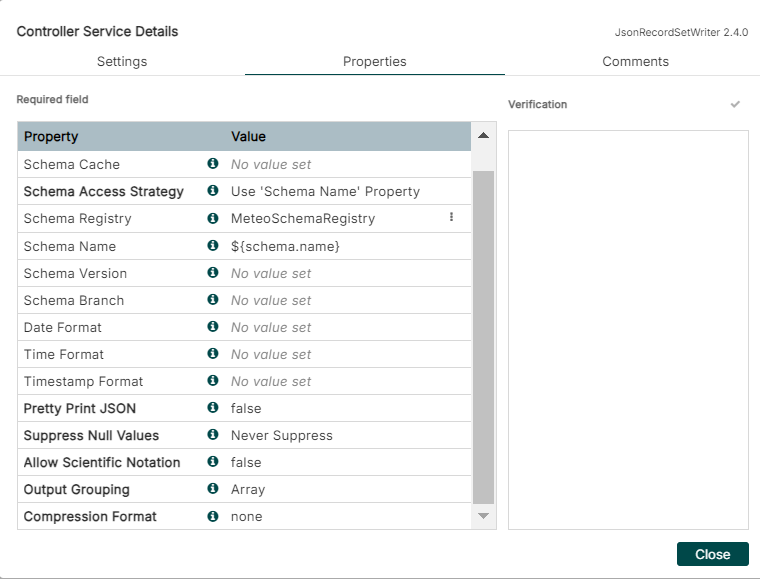


## 4.6.3 Services de validation et de schéma

**MeteoSchemaRegistry** : Service de type AvroSchemaRegistry qui centralise la gestion des schémas de données pour les quatre dimensions principales (stations, mesures, localisation, temps) et assure la cohérence structurelle des données traitées.



**Services de validation** : La validation des données est intégrée aux services de lecture/écriture avec des contrôles spécifiques aux mesures météorologiques : plages de températures (-50°C à +60°C), humidité (0-100%), précipitations non négatives, et coordonnées géographiques valides. Ces règles garantissent l'intégrité des données avant insertion dans le data warehouse.

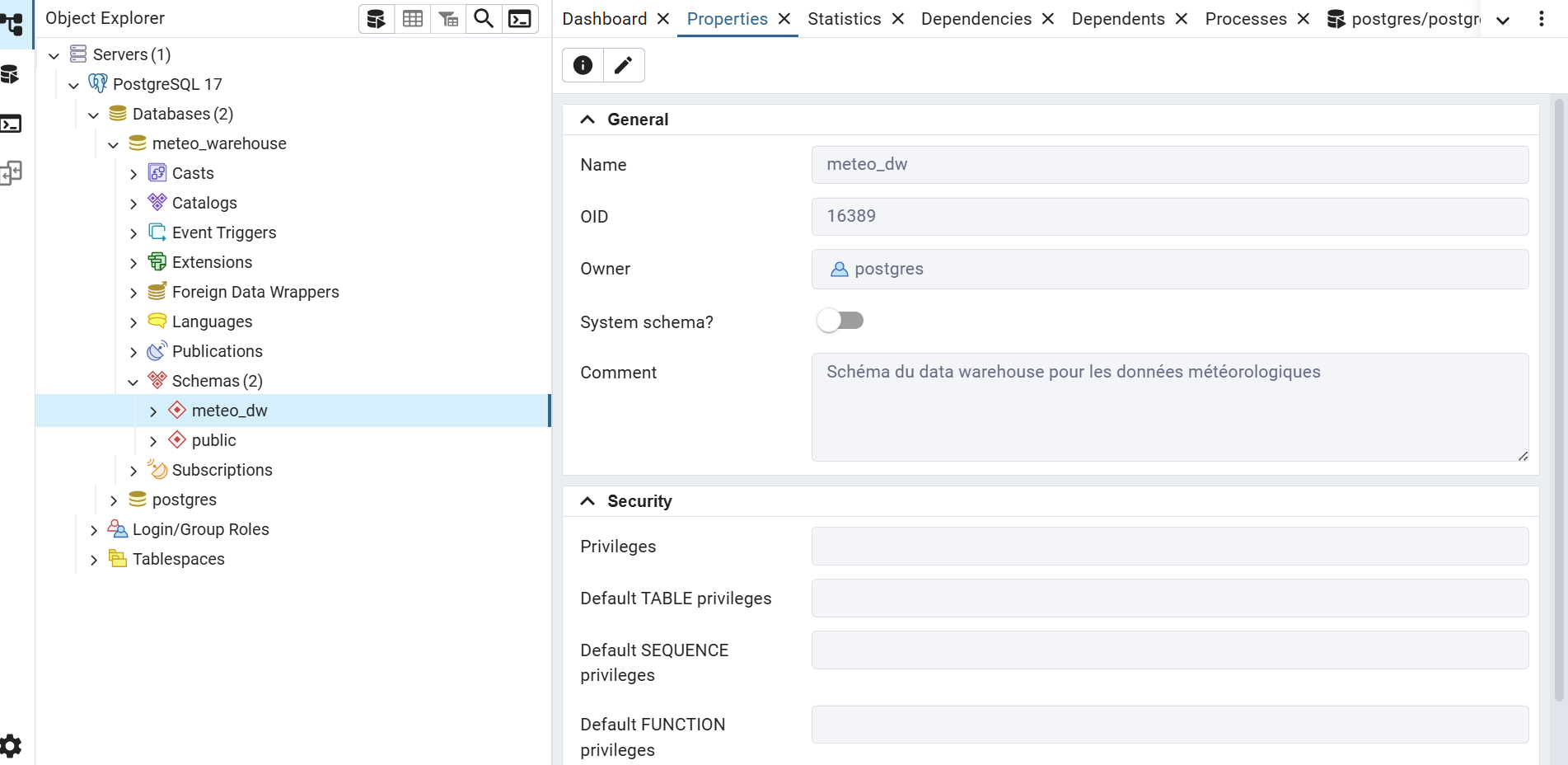


# 5. Phase 3 : Implémentation du Data Warehouse

## 5.1 Création du schéma dans PostgreSQL

### 5.1.1 Schéma dédié aux données météorologiques

Un schéma spécifique meteo\_dw a été créé dans PostgreSQL pour organiser et isoler les données du data warehouse météorologique. Ce schéma contient l'ensemble des tables dimensionnelles et de faits identifiées lors de la phase d'analyse.



### 5.1.2 Tables dimensionnelles implémentées

**Table dim\_station :**

sql

CREATE TABLE meteo\_dw.dim\_station (

code\_station VARCHAR(10) PRIMARY KEY,

nom\_station VARCHAR(100) NOT NULL,

latitude DECIMAL(10,6),

longitude DECIMAL(10,6),

altitude INTEGER,

type\_station VARCHAR(50),

date\_mise\_en\_service DATE,

est\_active BOOLEAN DEFAULT TRUE

);

**Table dim\_localisation**

sql

CREATE TABLE meteo\_dw.dim\_localisation (

id\_localisation SERIAL PRIMARY KEY,

pays VARCHAR(50) DEFAULT 'France',

region VARCHAR(100),

departement VARCHAR(100),

ville VARCHAR(100),

code\_postal VARCHAR(10),

type\_zone VARCHAR(50)

);

**Table dim\_type\_mesure**

sql

CREATE TABLE meteo\_dw.dim\_type\_mesure (

code\_mesure VARCHAR(20) PRIMARY KEY,

nom\_mesure VARCHAR(100) NOT NULL,

unite VARCHAR(20),

description TEXT,

precision\_mesure DECIMAL(5,2)

);

**Table dim\_temps**

sql

CREATE TABLE meteo\_dw.dim\_temps (

date\_complete DATE PRIMARY KEY,

annee INTEGER NOT NULL,

mois INTEGER NOT NULL,

jour INTEGER NOT NULL,

jour\_semaine INTEGER NOT NULL,

nom\_jour VARCHAR(20),

nom\_mois VARCHAR(20),

trimestre INTEGER,

saison VARCHAR(20),

est\_weekend BOOLEAN,

est\_jour\_ferie BOOLEAN

);

**Table de faits fait\_observations**

Sql

CREATE TABLE meteo\_dw.fait\_observations (

id\_observation SERIAL PRIMARY KEY,

code\_station VARCHAR(10) REFERENCES meteo\_dw.dim\_station(code\_station),

id\_localisation INTEGER REFERENCES meteo\_dw.dim\_localisation(id\_localisation),

date\_complete DATE REFERENCES meteo\_dw.dim\_temps(date\_complete),

code\_mesure VARCHAR(20) REFERENCES meteo\_dw.dim\_type\_mesure(code\_mesure),

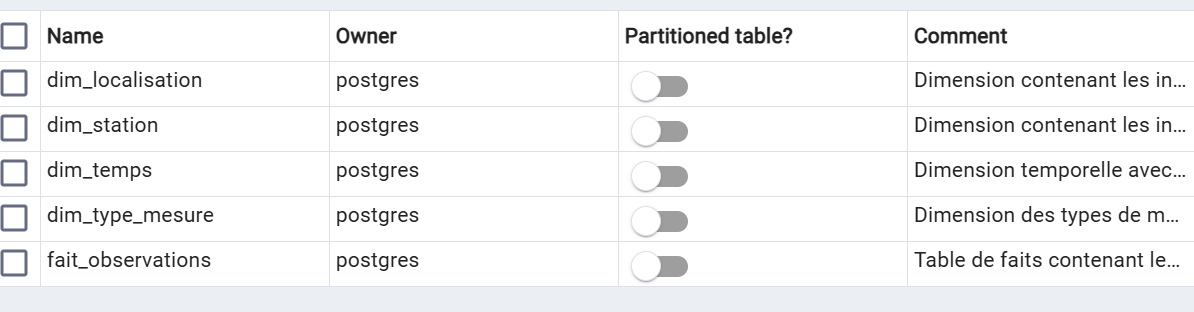
valeur\_mesure DECIMAL(10,2),

fiabilite\_mesure DECIMAL(5,2) CHECK (fiabilite\_mesure BETWEEN 0 AND 100),

est\_valeur\_estimee BOOLEAN DEFAULT FALSE,

horodatage TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);



## 5.2 Génération de la dimension temps

### Script de génération automatique

Un script PostgreSQL a été développé pour générer automatiquement la dimension temps avec toutes les hiérarchies nécessaires. Ce script peuple la table dim\_temps pour une période donnée en calculant automatiquement tous les attributs temporels.

sql

*-- Fonction de génération de la dimension temps*

CREATE OR REPLACE FUNCTION generer\_dimension\_temps(

date\_debut DATE,

date\_fin DATE

) RETURNS VOID AS $$

DECLARE

date\_courante DATE := date\_debut;

jour\_ferie BOOLEAN;

BEGIN

WHILE date\_courante <= date\_fin LOOP

*-- Détermination des jours fériés (logique simplifiée)*

jour\_ferie := (

(EXTRACT(MONTH FROM date\_courante) = 1 AND EXTRACT(DAY FROM date\_courante) = 1) OR *-- Nouvel An*

(EXTRACT(MONTH FROM date\_courante) = 5 AND EXTRACT(DAY FROM date\_courante) = 1) OR *-- Fête du Travail*

(EXTRACT(MONTH FROM date\_courante) = 7 AND EXTRACT(DAY FROM date\_courante) = 14) OR *-- Fête Nationale*

(EXTRACT(MONTH FROM date\_courante) = 12 AND EXTRACT(DAY FROM date\_courante) = 25) *-- Noël*

);

INSERT INTO meteo\_dw.dim\_temps VALUES (

date\_courante,

EXTRACT(YEAR FROM date\_courante),

EXTRACT(MONTH FROM date\_courante),

EXTRACT(DAY FROM date\_courante),

EXTRACT(DOW FROM date\_courante),

TO\_CHAR(date\_courante, 'Day'),

TO\_CHAR(date\_courante, 'Month'),

EXTRACT(QUARTER FROM date\_courante),

CASE

WHEN EXTRACT(MONTH FROM date\_courante) IN (12, 1, 2) THEN 'Hiver'

WHEN EXTRACT(MONTH FROM date\_courante) IN (3, 4, 5) THEN 'Printemps'

WHEN EXTRACT(MONTH FROM date\_courante) IN (6, 7, 8) THEN 'Été'

ELSE 'Automne'

END,

EXTRACT(DOW FROM date\_courante) IN (0, 6),

jour\_ferie

);

date\_courante := date\_courante + INTERVAL '1 day';

END LOOP;

END;

$$ LANGUAGE plpgsql;

