

# Guide réalisation ETL bout en bout sur Talend data intégration : Mise en place d'un Data Warehouse.

- La première partie de mini projet consiste à mettre en place un système d'extraction de données de suivi des dépenses via Talend studio. Cette première partie nous permettra d'extraire les données provenant d'un fichier Excel vers les différents fichiers plats.
- La deuxième partie de mini projet Talend studio consistera à mettre en place un système d'intégration de données de suivi des dépenses via Talend Data Intégration. Cette deuxième partie est la suite de la première partie qui permettra de charger les données dans les tables ODS (Operational Data Store) à partir des 3 fichiers plats créer auparavant.
- La troisième partie de mini projet Talend studio est la suite de la deuxième partie qui consistera également à mettre en place un système d'intégration de données de suivi des dépenses via Talend studio. Cette troisième partie permettra de charger les données dans les tables du Data Warehouse ou Entrepôt de Données à partir des données présentes dans les tables de l'Operational Data Store(ODS).

## Les données sources

Voici les fichiers Excel d'où proviennent les données pour la 1<sup>ère</sup> partie.

Ces données sont stockées dans un fichier Excel nommée **Saisie\_Depense.xlsx**

A	B	C
<b>Liste détaillée des catégories</b>		
	<b>Catégorie</b>	
	Dépenses Fixes	
	Dépenses Variables	
	Dépenses Exceptionnelles	

### Liste détaillée des sous-catégories

Sous-catégorie	Catégorie
Abonnement Logiciel	Dépenses Fixes
Abonnement Télécommunications	Dépenses Fixes
Abonnement Transport	Dépenses Fixes
Assurance Maison	Dépenses Fixes
Assurance Voiture	Dépenses Fixes
Autres Abonnements	Dépenses Fixes
Eau	Dépenses Fixes
Énergie (Électricité & Gaz)	Dépenses Fixes
Frais Bancaire	Dépenses Fixes
Impôts et Taxes	Dépenses Fixes
Loyer	Dépenses Fixes
Alimentation	Dépenses Variables
Carburant	Dépenses Variables
Coiffeur	Dépenses Variables
Garderie des Enfants	Dépenses Variables

### Liste détaillée des dépenses

Numéro dépense	Date dépense	Description dépense	Catégorie dépense	Sous-catégorie dépense	Montant dépense
D00001	01/01/2022	Achat Essence	Dépenses Variables	Carburant	32,41
D00002	02/01/2022	Alimentation	Dépenses Variables	Alimentation	20,37
D00003	06/01/2022	Abonnement Microsoft	Dépenses Fixes	Abonnement Logiciel	6,48
D00004	09/01/2022	Alimentation	Dépenses Variables	Alimentation	20,82
D00005	11/01/2022	Achat Essence	Dépenses Variables	Carburant	32,41
D00006	12/01/2022	Frais Electricité	Dépenses Fixes	Énergie (Électricité & Gaz)	52,08
D00007	12/01/2022	Frais bancaire	Dépenses Fixes	Frais Bancaire	0,58
D00008	13/01/2022	Soins et Hygiène	Dépenses Variables	Soins et Hygiène	12,93
D00009	13/01/2022	Alimentation	Dépenses Variables	Alimentation	0,97
D00010	16/01/2022	Alimentation	Dépenses Variables	Alimentation	26,19
D00011	16/01/2022	Alimentation	Dépenses Variables	Alimentation	7,44
D00012	16/01/2022	Loisirs	Dépenses Variables	Loisirs	4,54

## 1ère partie : Système d'extraction de données

Cette première étape du mini projet est répartie en 4 étapes :

- Découverte de Talend
- Création des groupes de contexte
- Création des différents schémas génériques et création des jobs d'extractions des données.
- Création du job principal(Orchestration)

### Découverte de Talend data intégration

**Travail à réaliser :**

- Ouvrir Talend Open Studio.
- On crée un nouveau projet nommé **idepense\_reporting** avec l'option Java.

La fenêtre de Talend est composée des vues suivantes :

- Barres d'outils et menus (en haut)
- Repository (en haut à gauche) : Ce référentiel contient tous les éléments techniques du projet
- Design Workspace (au centre) : Cet espace de modélisation permet de concevoir graphiquement les business model et les jobs.
- Palette (en haut à droite) : Cette palette graphique permet d'accéder aux différents composants
- Différentes vues (en bas au centre) :
  - Job : infos sur le job sélectionné
  - Component : configuration du composant sélectionné
  - Run job : exécution des jobs
  - Problems : erreurs
- Outline et Code Viewer (en bas à gauche) : Ces fenêtres fournissent un aperçu du code et du schéma du job ou du business model.

Création des groupes de contexte :

- On crée un nouveau groupe de contexte nommé **config\_Depense\_File**
- Initialisation des variables :
  - **depenseFileRepo** : récupère le dossier qui contient le fichier source « INPUT/ »
  - **depenseFileName** : récupère le nom de fichier source « SAISIE\_DEPENSE.xlsx »
  - **projetFolder** : récupère le nom du dossier du projet « c:/bi/depense/ »
  - **encodage** : récupère l'encodage des fichiers « UTF8 »
  - **fieldSeparator** : récupère les séparateurs de champs « | »
  - **dataFolder** : récupère les données du dossier « DATA/ »

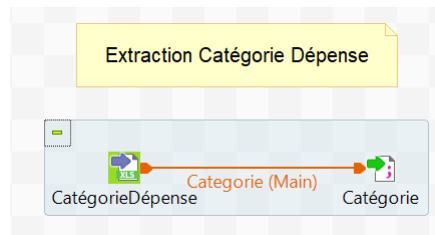
### Création des différents schémas génériques : Fichiers plats

Dans la métadonnée clic droit sur schémas générique puis créer un schéma générique

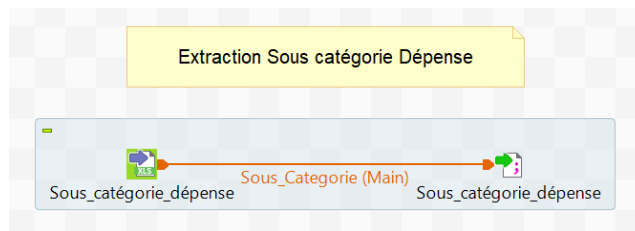
- CATEGORIE
  - DEPENSE
  - SOUS CATEGORIE
- 

# Création des jobs d'extraction des données (Lecture du fichier Excel et extraction des données vers les fichiers plats)

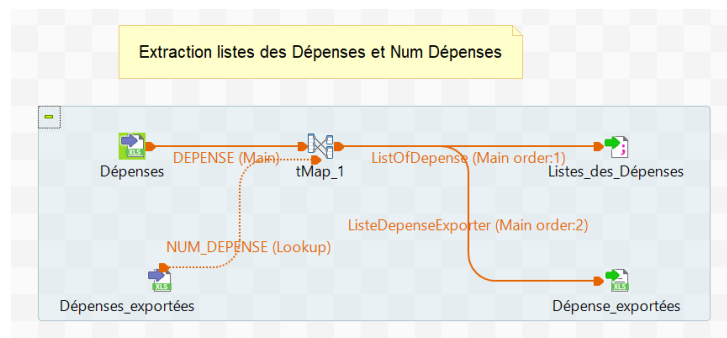
Un job nommé jExtractDepense : extraction catégorie dépense :



Un job nommé jExtractSousCategorieDepense : Extraction Sous-catégorie dépense



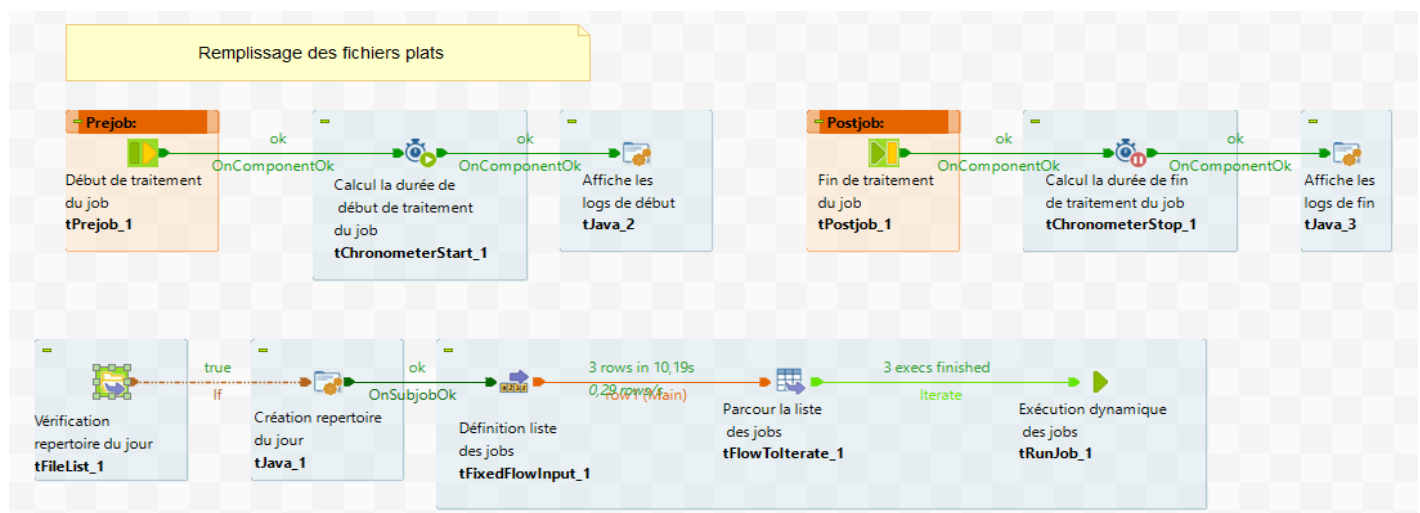
Un job nommé jExtractCategorieDepense : extraction des dépenses :



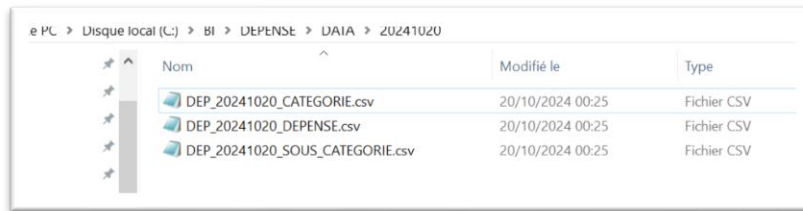
## Orchestration des jobs :

Un dernier job est nécessaire pour faire appel à tous les jobs créés auparavant, il sera nommé **jExtractData**. A son exécution, le job doit créer et charger les jobs extractions(jExtractDepense, jExtractSousCategorieDepense, jExtractCategorieDepense). Ci-dessous un aperçu de l'ordonnancement des job d'extraction :

Création du job principal(**jExtractData**) : remplissage des fichiers plats.



Ci-dessous les 3 fichiers horodatés générés :



Nom	Modifié le	Type
DEP_20241020_CATEGORIE.csv	20/10/2024 00:25	Fichier CSV
DEP_20241020_DEPENSE.csv	20/10/2024 00:25	Fichier CSV
DEP_20241020_SOUS_CATEGORIE.csv	20/10/2024 00:25	Fichier CSV

## Conclusion

Le système d'extraction mis en place, permet de récupérer les données provenant d'un fichier Excel et à partir de ce fichier générer les 3 fichiers plats et la génération ns avons tout simplement effectuer dans un répertoire du jour donc ns avons défini des répertoires horodatés.

## 2<sup>ème</sup> partie :Chargement des tables ODS (Operational Data Store) –Suivi des dépenses

La deuxième partie de mini projet Talend consiste à mettre en place un système d'intégration de données de suivi des dépenses via Talend Data Intégration. Cette deuxième partie permettra de charger les données dans les tables ODS à partir des 3 fichiers plats créer auparavant. Cette partie comprend 9 étapes :

- Création de la base de données via PostgreSQL
- Création des différents schémas(ODS, PARAM\_log)
- Création des tables ODS ET table des CONTEXTES
- Mise en place des différents processus d'intégration de données talend
- Création des groupes de contexte
- Création des métadonnées : Connexion à la base de données, Configuration du chargement implicite des variables de contexte
- Création des jobs ODS (Transformation et Chargement de la donnée)
- Création du job principal(Orchestration)

Création de la base de données "IDEPENSE\_INGESTION\_DB" qui permet d'héberger les données

```
-- Création de la base de données "IDEPENSE_INGESTION_DB"  
DROP DATABASE IF EXISTS "IDEPENSE_INGESTION_DB";  
CREATE DATABASE "IDEPENSE_INGESTION_DB";
```

Création du schéma DEPENSE\_ODS :

```
DROP SCHEMA IF EXISTS "DEPENSE_ODS";  
CREATE SCHEMA "DEPENSE_ODS";
```

Création du schéma PARAMS\_LOG :

```
DROP SCHEMA IF EXISTS "PARAMS_LOG";  
CREATE SCHEMA "PARAMS_LOG";
```

Création des tables ODS et contexte:

Table ODS\_CATEGORIE :

```
DROP TABLE IF EXISTS "ODS_CATEGORIE" ;  
CREATE TABLE "ODS_CATEGORIE"  
(  
  "LB_CATEGORIE" VARCHAR(100) NOT NULL,  
  "LB_NOM_FICHIER" VARCHAR(100) NOT NULL,  
  "DT_INSERTION" TIMESTAMP NOT NULL,  
  "LB_JOB_NAME" VARCHAR(100) NOT NULL,  
  CONSTRAINT "ODS_CATEGORIE_PKEY" PRIMARY KEY ("LB_CATEGORIE")  
);
```

## Table ODS\_SOUS\_CATEGORIE :

```
DROP TABLE IF EXISTS "ODS_SOUS_CATEGORIE" ;
CREATE TABLE "ODS_SOUS_CATEGORIE"
(
  "LB_SOUS_CATEGORIE" VARCHAR(100) NOT NULL,
  "LB_CATEGORIE"      VARCHAR(50)  NOT NULL,
  "LB_NOM_FICHIER"    VARCHAR(100) NOT NULL,
  "DT_INSERTION"      TIMESTAMP    NOT NULL,
  "LB_JOB_NAME"       VARCHAR(100) NOT NULL,
  CONSTRAINT "ODS_SOUS_CATEGORIE_PKEY" PRIMARY KEY ("LB_SOUS_CATEGORIE")
);
```

## Table ODS\_SOUS\_DESCRIPTION :

```
DROP TABLE IF EXISTS "ODS_DESCRIPTION" ;
CREATE TABLE "ODS_DESCRIPTION"
(
  "LB_DESCRIPTION_DEPENSE" VARCHAR(255) NOT NULL,
  "LB_NOM_FICHIER"        VARCHAR(100) NOT NULL,
  "DT_INSERTION"          TIMESTAMP    NOT NULL,
  "LB_JOB_NAME"           VARCHAR(100) NOT NULL,
  CONSTRAINT "ODS_DESCRIPTION_PKEY" PRIMARY KEY ("LB_DESCRIPTION_DEPENSE")
);
```

## Table ODS\_DEPENSE :

```
DROP TABLE IF EXISTS "ODS_DEPENSE" ;
CREATE TABLE "ODS_DEPENSE"
(
  "NUM_DEPENSE"          VARCHAR(50)  NOT NULL,
  "DT_DEPENSE"           TIMESTAMP    NOT NULL,
  "LB_DESCRIPTION_DEPENSE" VARCHAR(255) NOT NULL,
  "LB_SOUS_CATEGORIE"    VARCHAR(50)  NOT NULL,
  "MT_DEPENSE"           NUMERIC      NOT NULL,
  "LB_NOM_FICHIER"       VARCHAR(100) NOT NULL,
  "DT_INSERTION"         TIMESTAMP    NOT NULL,
  "LB_JOB_NAME"          VARCHAR(100) NOT NULL,
  CONSTRAINT "ODS_DEPENSE_PKEY" PRIMARY KEY ("NUM_DEPENSE")
);
```

## Table ODS\_REJET :

```
DROP TABLE IF EXISTS "ODS_REJET" ;
CREATE TABLE "ODS_REJET"
(
  "LB_CHEMIN_FICHIER" VARCHAR(255) NOT NULL,
  "LB_NOM_FICHIER"    VARCHAR(100) NOT NULL,
  "NUM_LIGNE_REJET"   INTEGER       NOT NULL,
  "LB_LIGNE_REJET"    VARCHAR(500)  NOT NULL,
  "LB_MESSAGE_REJET"  VARCHAR(100)  NOT NULL,
  "LB_NOM_FLUX"       VARCHAR(100)  NOT NULL,
  "DT_REJET"          TIMESTAMP     NOT NULL,
  "LB_JOB_NAME"       VARCHAR(100)  NOT NULL,
  CONSTRAINT "ODS_REJET_PKEY" PRIMARY KEY
("LB_CHEMIN_FICHIER", "LB_NOM_FICHIER", "NUM_LIGNE_REJET", "LB_LIGNE_REJET")
);
```

## Table CONTEXTE :

```
DROP TABLE IF EXISTS "CONTEXTE";
CREATE TABLE "CONTEXTE"
(
  key   VARCHAR(100) NOT NULL,
  value VARCHAR(255) ,
  CONSTRAINT "CONTEXTE_PKEY" PRIMARY KEY (key)
);
```

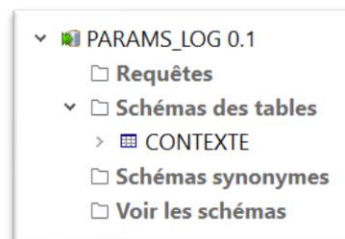
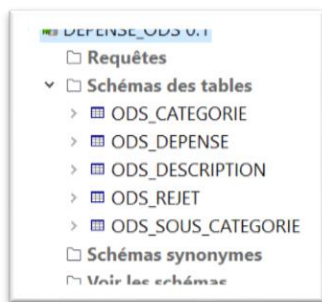
Mise en place des différents processus d'intégration de données via talend :

- Création des groupes de contexte :
  - On crée un nouveau groupe de contexte nommé **Connexion\_ODS** sur Talend
  - Initialisation des variables :
    - o **serverName** : récupère le nom de serveur
    - o **database**: récupère le nom de la BD « IDEPENSE\_INGESTION\_DB »
    - o **port** : récupère le port
    - o **password** : récupère le mot de passe
    - o **utilisateur** : récupère le nom utilisateur
    - o **additionalParam** : champ vide
    - o **schema\_ods** : récupère le schéma des « DEPENSE\_ODS »

Reproduire les mêmes informations pour le groupe de contexte **Connexion\_Params**

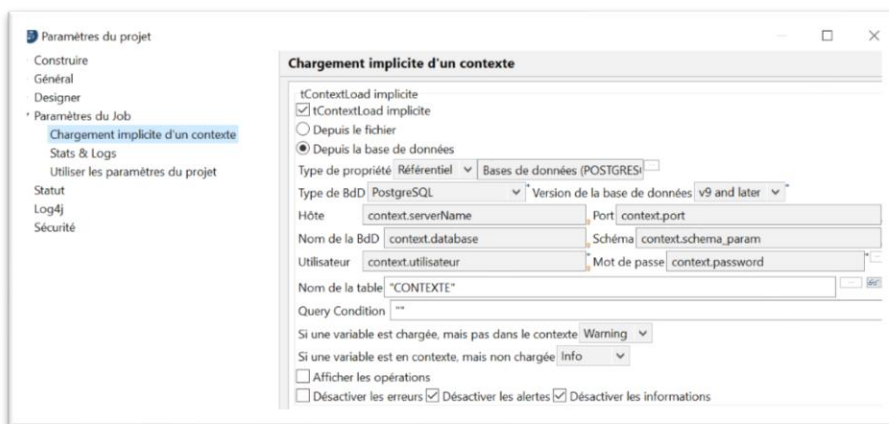
- o **serverName** : récupère le nom de serveur
- o **database**: récupère le nom de la BD « IDEPENSE\_INGESTION\_DB »
- o **port** : récupère le port
- o **password** : récupère le mot de passe
- o **utilisateur** : récupère le nom utilisateur
- o **additionalParam** : champ vide
- o **schema\_param**: récupère le schéma « PARAMS\_LOG »

Création des métadonnées : Connexion à la base de données et récupérer le schéma



Configuration du chargement implicite des variables de contexte :

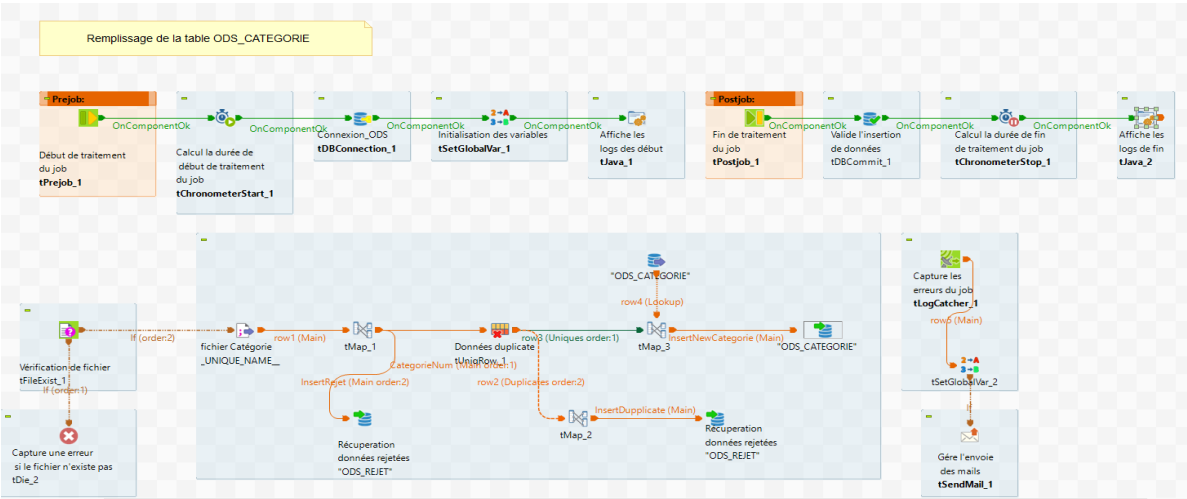
- o Ouvrir Talend-Fichier-modifier les propriétés du job-paramètre du job puis chargement implicite d'un contexte



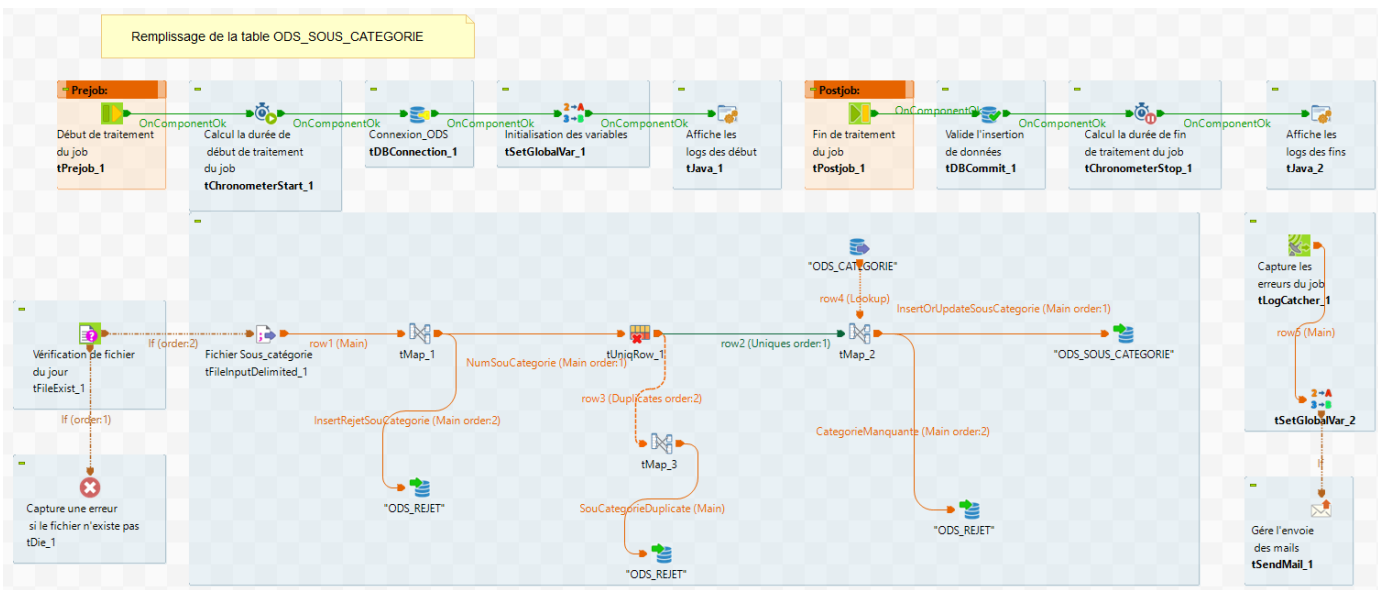
Création des jobs ODS (Transformation et Chargement de la donnée) :

- o ODS
  - o jOdsCategorie 0.1
  - o jOdsDepense 0.1
  - o jOdsDescription 0.1
  - o jOdsSousCategorie 0.1

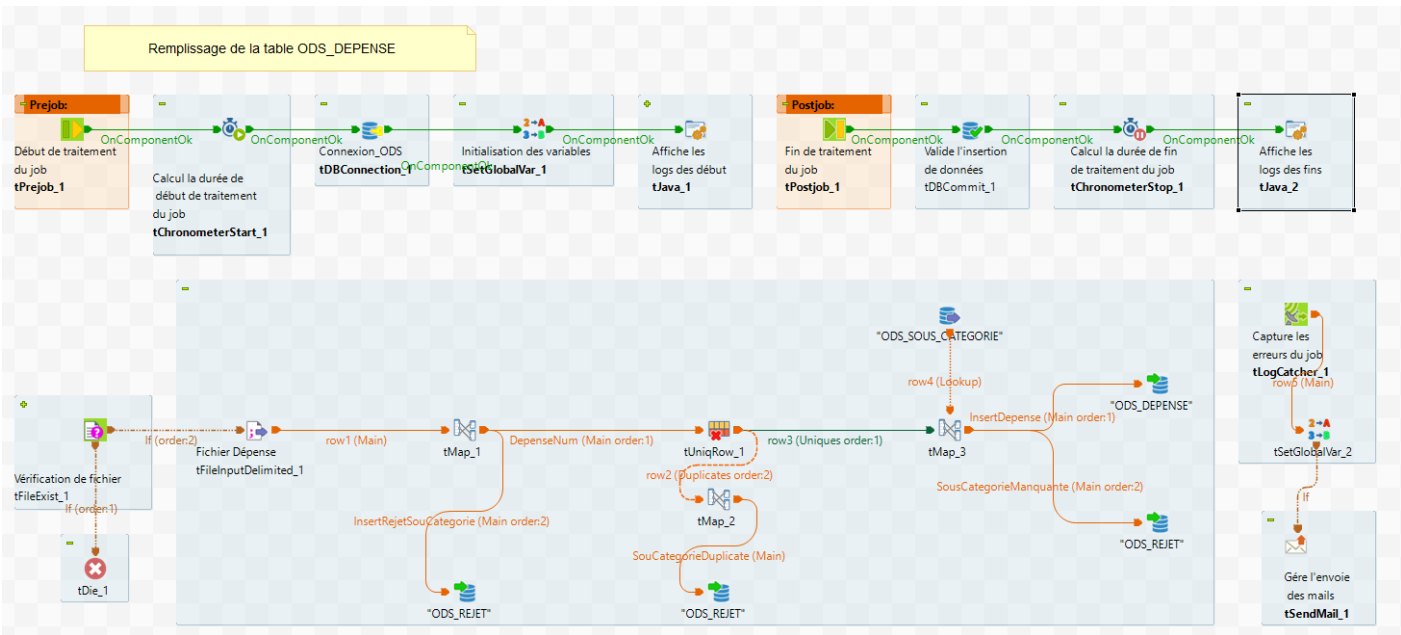
Le job nommé jOds\_Categorie: remplissage de la table ODS\_CATEGORIE



Le job nommé jOds\_Sous\_Categorie : remplissage de la table ODS\_SOUS\_CATEGORIE

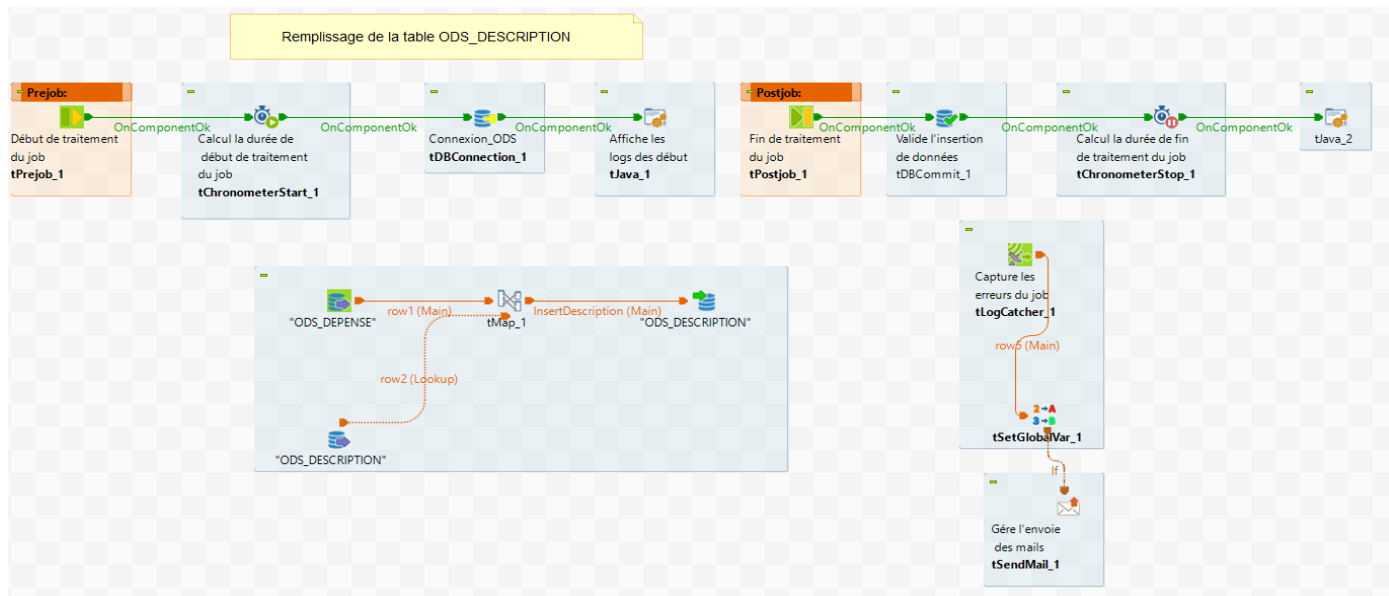


Le job nommé jOds\_Depense : remplissage de la table ODS\_DEPENSE



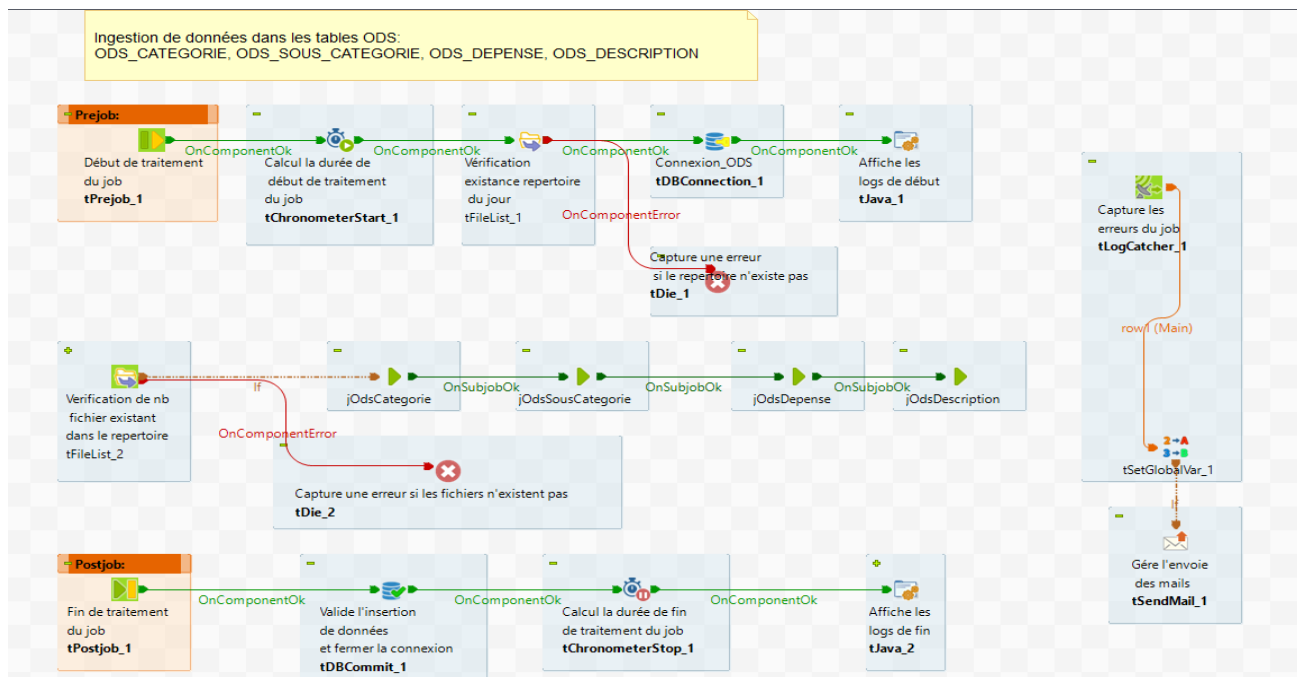
Le job nommé jOds\_Description : remplissage de la table ODS\_DESCRIPTION





Création du job principal(Orchestration ODS)

Le job nommé jChargeODS:



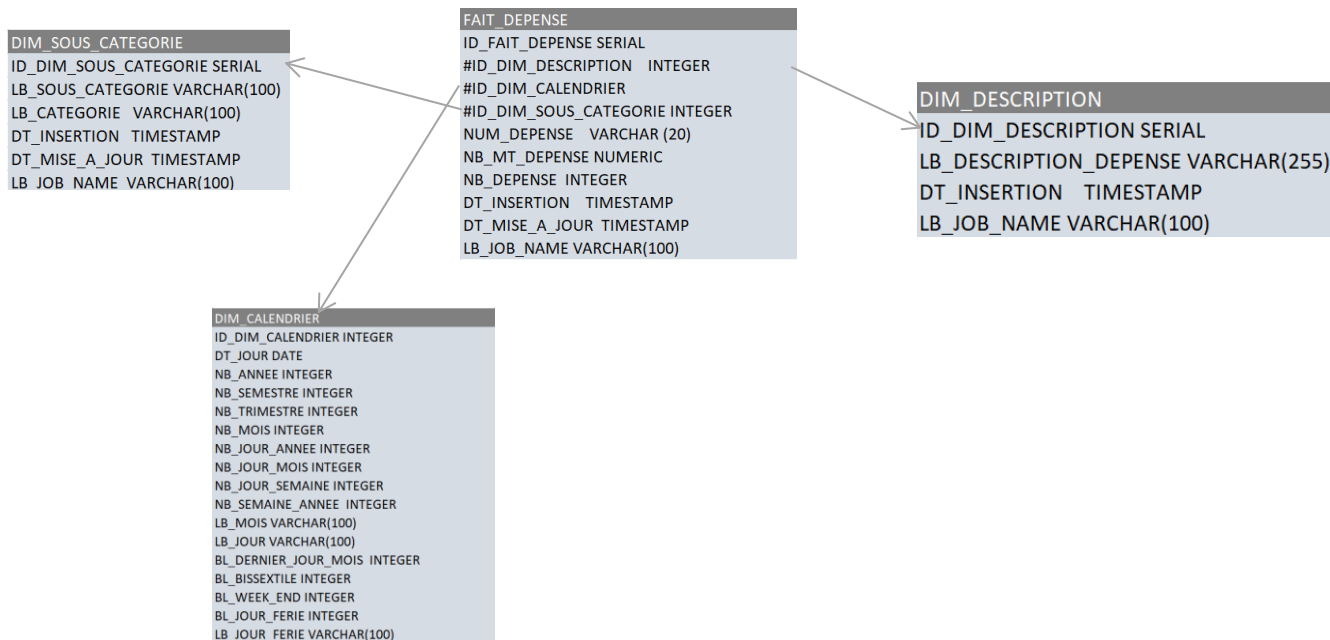
### 3<sup>ème</sup> partie : Chargement de Data Warehouse (Entrepôt de Données) –Suivi des dépenses .

La troisième partie de mini projet Talend est la suite de la première partie consiste à mettre en place un système d'intégration de données de suivi des dépenses via Talend Data studio. Cette troisième partie permettra de charger les données dans les tables du Data Warehouse ou Entrepôt de Données à partir des données présentes dans les tables de l'Operational Data Store(ODS).

### Schéma de l'entrepôt

Voici le schéma en étoile de l'entrepôt de données :





Une fois le schéma validé, il faut créer l'entrepôt sous PostgreSQL.

Cette partie est répartie en 7 étapes :

- Création des différents schémas(DEPENSE\_DWH)
- Création des tables des dimensions
- Mise en place des différents processus d'intégration de données via Talend Data intégration
- Création des groupes de contexte
- Création des métadonnées : Connexion à la base de données, et récupération des schémas
- Création des jobs DWH (Transformation et Chargement de la donnée)
- Création du job principal(Orchestration).

Création des différents schémas(DEPENSE\_DWH)

```

DROP SCHEMA IF EXISTS "DEPENSE_DWH" CASCADE;
CREATE SCHEMA "DEPENSE_DWH";

```

Création des tables des dimensions :

Table de dimension : DIM\_SOUS\_CATEGORIE

```

-- DROP TABLE IF EXISTS "DIM_SOUS_CATEGORIE" ;
CREATE TABLE "DIM_SOUS_CATEGORIE"
(
    "ID_DIM_SOUS_CATEGORIE" SERIAL NOT NULL,
    "LB_SOUS_CATEGORIE" VARCHAR(100) NOT NULL,
    "LB_CATEGORIE" VARCHAR(100) NOT NULL,
    "DT_INSERTION" TIMESTAMP NOT NULL,
    "LB_JOB_NAME" VARCHAR(100) NOT NULL,
    CONSTRAINT "DIM_SOUS_CATEGORIE_PKEY" PRIMARY KEY ("ID_DIM_SOUS_CATEGORIE")
);

```

Table de dimension : DIM\_DESCRIPTION

```

-- DROP TABLE IF EXISTS "DIM_DESCRIPTION" ;
CREATE TABLE "DIM_DESCRIPTION"
(
    "ID_DIM_DESCRIPTION" SERIAL NOT NULL,
    "LB_DESCRIPTION_DEPENSE" VARCHAR(255) NOT NULL,
    "DT_INSERTION" TIMESTAMP NOT NULL,
    "LB_JOB_NAME" VARCHAR(100) NOT NULL,
    CONSTRAINT "DIM_DESCRIPTION_PKEY" PRIMARY KEY ("ID_DIM_DESCRIPTION")
);

```

## Table de dimension : DIM\_CALENDRIER

```
-- DROP TABLE IF EXISTS "DIM_CALENDRIER";
CREATE TABLE "DIM_CALENDRIER"
(
    "ID_DIM_CALENDRIER"    INTEGER NOT NULL,
    "DT_JOUR"              DATE NOT NULL,
    "NB_ANNEE"             INTEGER NOT NULL,
    "NB_SEMESTRE"         INTEGER NOT NULL,
    "NB_TRIMESTRE"        INTEGER NOT NULL,
    "NB_MOIS"             INTEGER NOT NULL,
    "NB_JOUR_ANNEE"       INTEGER NOT NULL,
    "NB_JOUR_MOIS"        INTEGER NOT NULL,
    "NB_JOUR_SEMAINE"     INTEGER NOT NULL,
    "NB_SEMAINE_ANNEE"    INTEGER NOT NULL,
    "LB_MOIS"             VARCHAR(100) NOT NULL,
    "LB_JOUR"             VARCHAR(100) NOT NULL,
    "BL_DERNIER_JOUR_MOIS" INTEGER NOT NULL,
    "BL_BISSEXTILE"       INTEGER NOT NULL,
    "BL_WEEK_END"        INTEGER NOT NULL,
    "BL_JOUR_FERIE"       INTEGER NOT NULL,
    "LB_JOUR_FERIE"       VARCHAR(100),
    CONSTRAINT "DIM_CALENDRIER_PKEY" PRIMARY KEY ("ID_DIM_CALENDRIER")
);
```

## Table de faits : FAI\_DEPENSE

```
-- DROP TABLE IF EXISTS "FAIT_DEPENSE" ;
CREATE TABLE "FAIT_DEPENSE"
(
    "ID_FAIT_DEPENSE"      SERIAL NOT NULL,
    "ID_DIM_CALENDRIER"    INTEGER NOT NULL,
    "ID_DIM_DESCRIPTION"   INTEGER NOT NULL,
    "ID_DIM_SOUS_CATEGORIE" INTEGER NOT NULL,
    "NUM_DEPENSE"          VARCHAR NOT NULL,
    "NB_MT_DEPENSE"        NUMERIC NOT NULL,
    "NB_DEPENSE"          INTEGER NOT NULL,
    "DT_INSERTION"         TIMESTAMP NOT NULL,
    "DT_MISE_A_JOUR"       TIMESTAMP,
    "LB_JOB_NAME"          VARCHAR(100) NOT NULL,
    CONSTRAINT "FAIT_DEPENSE_PKEY" PRIMARY KEY ("ID_FAIT_DEPENSE"),
    CONSTRAINT "FAIT_DEPENSE_SOUS_CATEGORIE_FKEY" FOREIGN KEY ("ID_DIM_SOUS_CATEGORIE")
        REFERENCES "DIM_SOUS_CATEGORIE" ("ID_DIM_SOUS_CATEGORIE"),
    CONSTRAINT "FAIT_DEPENSE_CALENDRIER_FKEY" FOREIGN KEY ("ID_DIM_CALENDRIER")
        REFERENCES "DIM_CALENDRIER" ("ID_DIM_CALENDRIER"),
    CONSTRAINT "FAIT_DEPENSE_DESCRIPTION_FKEY" FOREIGN KEY ("ID_DIM_DESCRIPTION")
        REFERENCES "DIM_DESCRIPTION" ("ID_DIM_DESCRIPTION")
);
```

Les requêtes ont été exécuté avec succès sous postgresSQL.

Vérification du résultat de l'exécution des requêtes sous postgresSQL en lançant la requête :

```
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'DEPENSE_DWH';
```

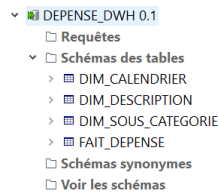
	table_name name	🔒
1	DIM_SOUS_CATEGORIE	
2	FAIT_DEPENSE	
3	DIM_CALENDRIER	
4	DIM_DESCRIPTION	

Mise en place des différents processus d'intégration de données via Talend studio

- Création des groupes de contexte :
  - On crée un nouveau groupe de contexte nommé **Connexion\_DWH** via Talend
  - Initialisation des variables :
    - **serverName** : récupère le nom de serveur
    - **database** : récupère le nom de la base de données
    - **port** : récupère le port
    - **password** : récupère le mot de passe

- **utilisateur** : récupère le nom utilisateur
- **additionalParam** : champ vide
- **schema\_dwh**: récupère le schéma des « Depense\_dwh »

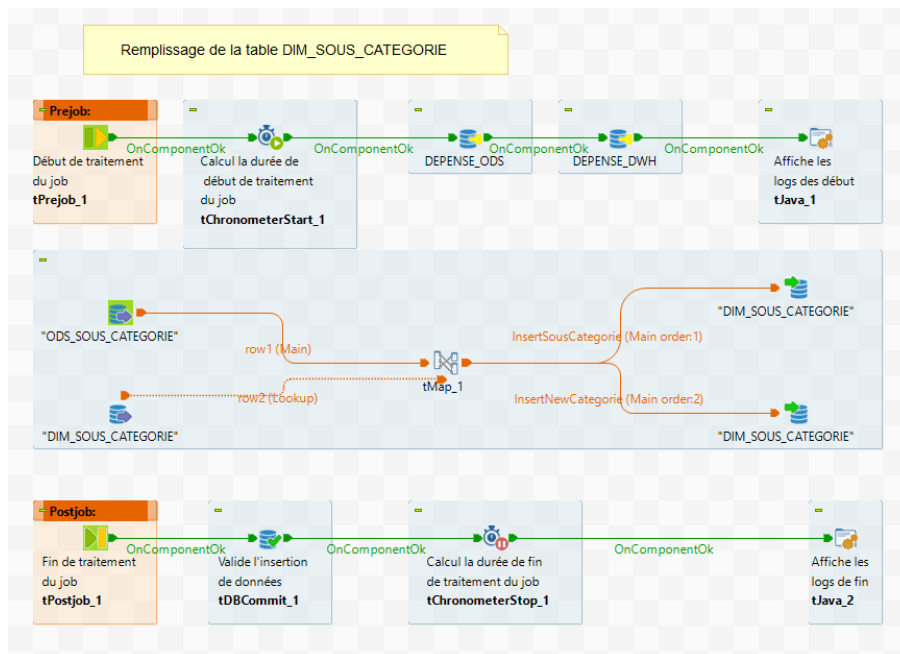
Création des métadonnées : Connexion à la base de données, et récupération des schémas



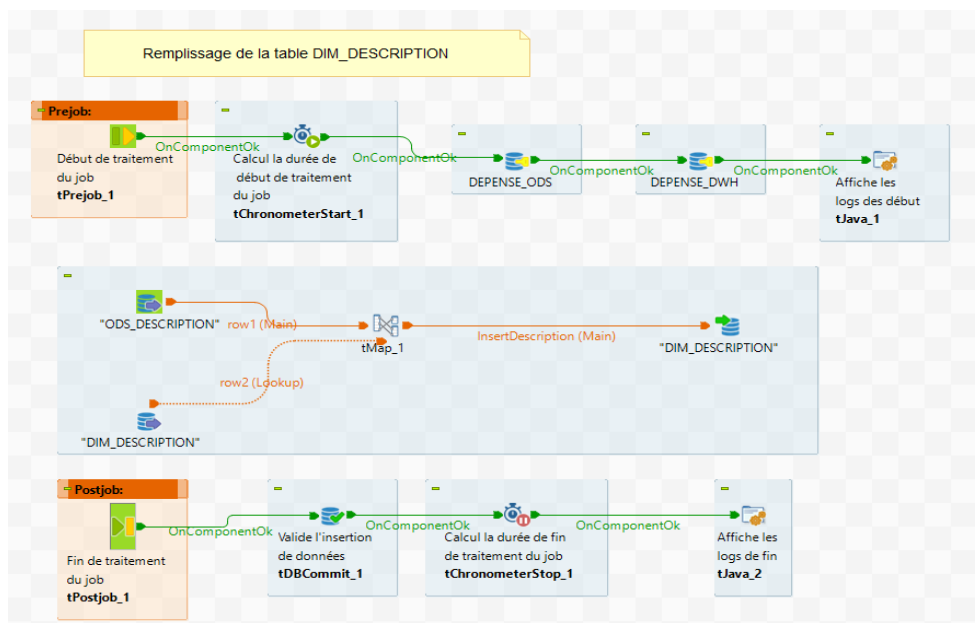
Création des jobs DWH (Transformation et Chargement de la donnée) :



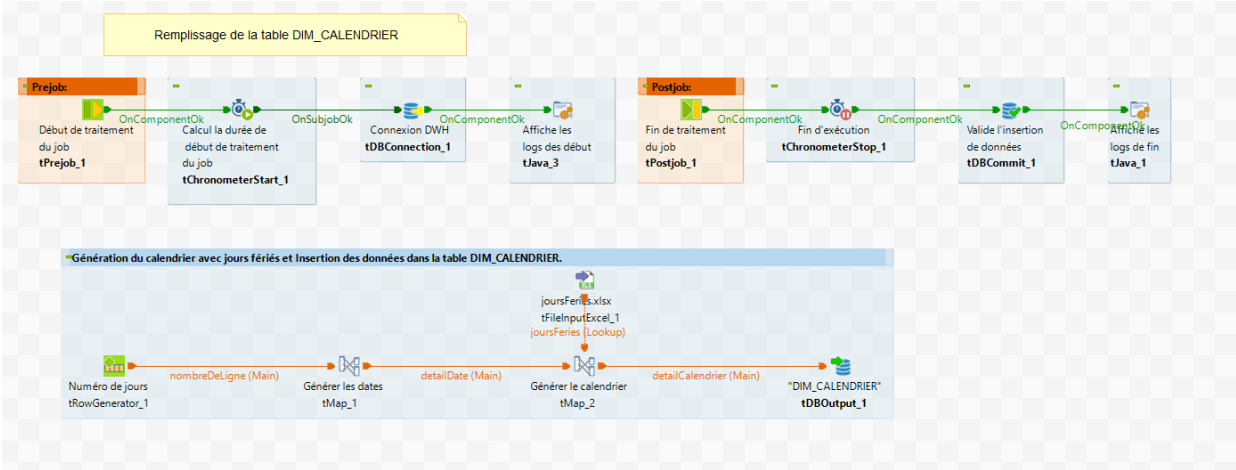
Le job nommé jDim\_Sous\_Categorie: Remplissage de la table DIM\_SOUS\_CATEGORIE



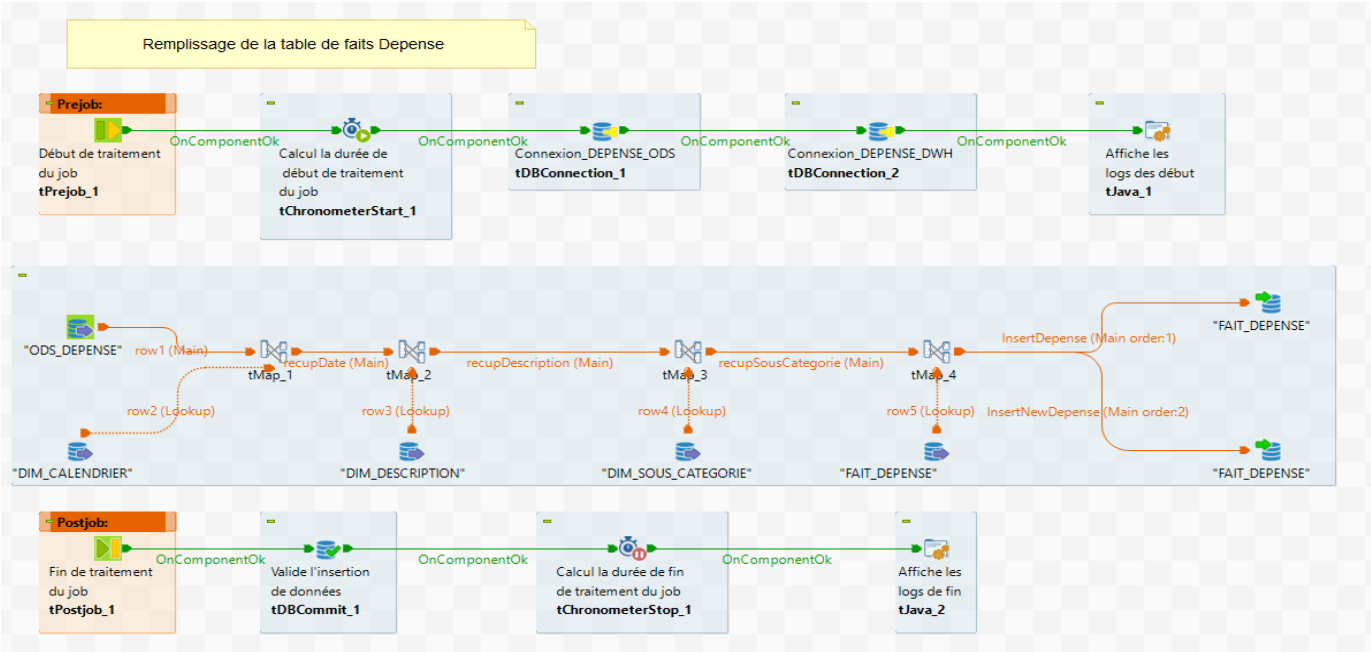
Le job nommé jDim\_Description : Remplissage de la table DIM\_DESCRIPTION



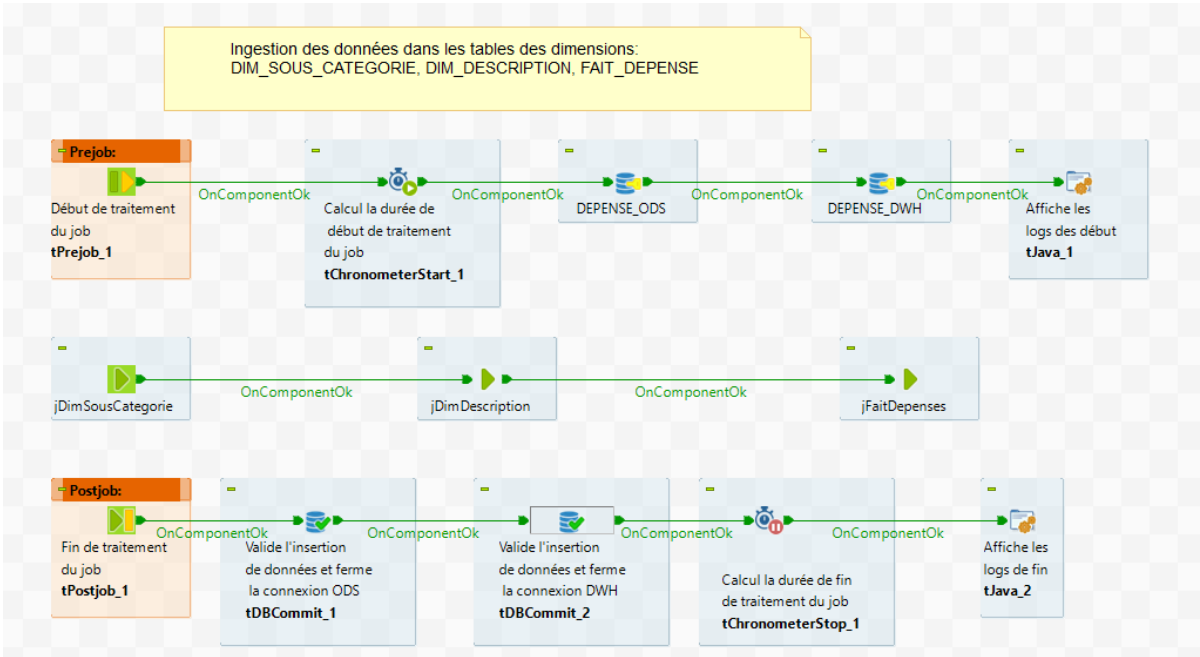
Le job nommé jGenerate\_Calendar: Remplissage de la table DIM\_CALENDRIER



Le job nommé jFait\_Depenses: Remplissage de la table FAIT\_DEPENSE



Création du job principal(Orchestration DWH) :



Orchestration (Alimentation BDD) :

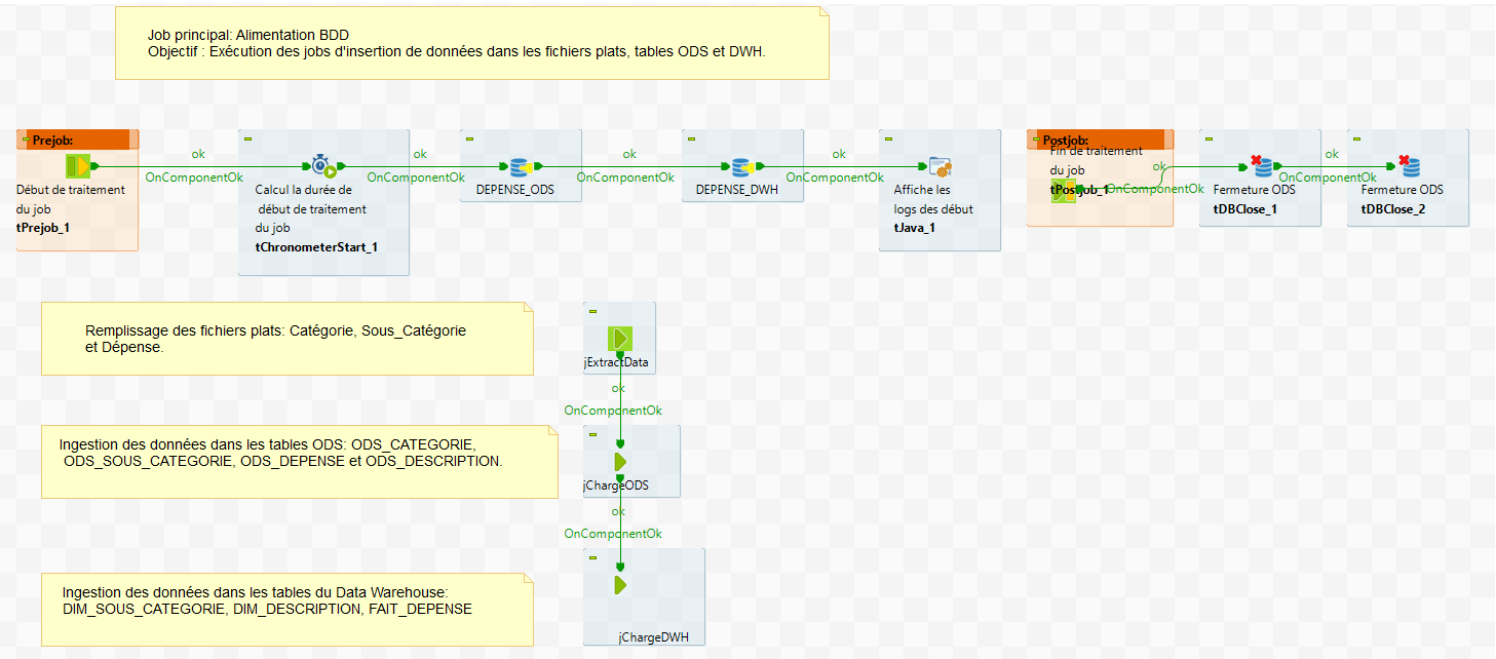
Un dernier job est nécessaire pour faire appel à tous les jobs créés auparavant, il sera nommé jAlimentation BDD. A son exécution, le job doit créer et charger les jobs :

jExtractData : qui extrait les données de fichier Excel et charge les fichier plats

jChargeODS : qui charge les tables ODS

jChargeDWH : qui charge les tables des dimensions du Data Warehouse

Ci-dessous un aperçu de l’ordonnancement du job Alimentation BDD:



Les Jobs ont été exécuté avec succès sous Talend studio. Vérification du résultat de l’exécution des jobs sous PostgreSQL en lançant les requêtes :

Remplissage de la table DIM SOUS CATEGORIE :

- Nombres de lignes de la table Dim\_Sous\_Categorie

```
SELECT count(*) FROM "DEPENSE_DWH"."DIM_SOUS_CATEGORIE";
```

	count	
	bigint	
1	24	

- Les cinq premières lignes de la table Dim\_Sous\_Categorie :

```
SELECT * FROM "DEPENSE_DWH"."DIM_SOUS_CATEGORIE" LIMIT 5;
```

ID_DIM_SOUS_CATEGORIE [PK] integer	LB_SOUS_CATEGORIE character varying (100)	LB_CATEGORIE character varying (100)	DT_INSERTION timestamp without time zone
7	Abonnement Logiciel	Dépenses Fixes	2024-06-04 18:16:34.353
8	Abonnement Télécommunications	Dépenses Fixes	2024-06-04 18:16:34.353
9	Abonnement Transport	Dépenses Fixes	2024-06-04 18:16:34.353
10	Assurance Maison	Dépenses Fixes	2024-06-04 18:16:34.353
11	Assurance Voiture	Dépenses Fixes	2024-06-04 18:16:34.353

Remplissage de la table DIM DESCRIPTION :

- Nombres de lignes de la table DIM\_DESCRIPTION

```
SELECT count(*) FROM "DEPENSE_DWH"."DIM_DESCRIPTION";
```

	count bigint
1	24

- Les cinq premières lignes de la table Dim\_Description

```
SELECT * FROM "DEPENSE_DWH"."DIM_DESCRIPTION" LIMIT 5;
```

	ID_DIM_DESCRIPTION [PK] integer	LB_DESCRIPTION_DEPENSE character varying (255)	DT_INSERTION timestamp without time zone	LB_JOB_NAME character varying (100)
1	1	Paieement Facture d'eau	2024-06-04 12:56:42.628	jDimDescription
2	2	Frais pharmacie	2024-06-04 12:56:42.629	jDimDescription
3	3	Paieement assurance voiture	2024-06-04 12:56:42.629	jDimDescription
4	4	Frais santé	2024-06-04 12:56:42.629	jDimDescription
5	5	Frais bancaire	2024-06-04 12:56:42.629	jDimDescription

## Remplissage de la table DIM\_CALENDAR :

- Nombres de lignes de la table DIM\_CALENDAR

```
SELECT count(*) FROM "DEPENSE_DWH"."DIM_CALENDRIER";
```

	count bigint
1	2192

- Les cinq premières lignes de la table DIM\_CALENDAR

```
SELECT * FROM "DEPENSE_DWH"."DIM_CALENDRIER" LIMIT 5;
```

ID_DIM_CALENDRIER [PK] integer	DT_JOUR date	NB_ANNEE integer	NB_SEMESTRE integer	NB_TRIMESTRE integer	NB_MOIS integer	NB_JOUR_ANNEE integer	NB_JOUR_MOIS integer	NB_JOUR_SEMAINE integer
20200101	2020-01-01	2020	1	1	1	1	1	3
20200102	2020-01-02	2020	1	1	1	2	2	4
20200103	2020-01-03	2020	1	1	1	3	3	5
20200104	2020-01-04	2020	1	1	1	4	4	6
20200105	2020-01-05	2020	1	1	1	5	5	7

## Remplissage de la table FAIT DEPENSE :

- Nombres de lignes de la table Fait\_Depense

```
SELECT count(*) FROM "DEPENSE_DWH"."FAIT_DEPENSE";
```

	count bigint
1	357

- Les cinq premières lignes de la table Fait\_Depense :

```
SELECT * FROM "DEPENSE_DWH"."FAIT_DEPENSE" LIMIT 5;
```

	ID_FAIT_DEPENSE [PK] integer	ID_DIM_CALENDRIER integer	ID_DIM_DESCRIPTION integer	ID_DIM_SOUS_CATEGORIE integer	NUM_DEPENSE character varying	NB_MT_DEPENSE numeric	NB_DEPENSE integer	DT_INSERTION timestamp without time zone
1	1	20220101	14	19	D00001	32.41	1	2024-06-04 22:07:57.505
2	2	20220102	18	18	D00002	20.37	1	2024-06-04 22:07:57.506
3	3	20220106	24	7	D00003	6.48	1	2024-06-04 22:07:57.506
4	4	20220109	18	18	D00004	20.82	1	2024-06-04 22:07:57.506
5	5	20220111	14	19	D00005	32.41	1	2024-06-04 22:07:57.506

## Planification du job jAlimentationBDD

Si les données sources évoluent en permanence. Les jobs doivent donc être planifiés régulièrement. Le lancement des jobs pourra se faire par exemple toutes les nuits pour prendre en compte les données modifiées pendant la journée. La planification des jobs peut se faire grâce au planificateur de tâches du système d'exploitation.

### Travail à réaliser :

- Exporter chaque job dans le répertoire C:/orion.
  - o On clic droit sur le job puis sur construire le job
  - o On sélectionne le type de construction : job standalone
  - o On coche l'option Extract the zip file
  - o Options par défaut puis finish
- On relance les jobs en cliquant sur les fichiers .bat.
- On vérifie que tout a bien fonctionné. Les jobs vont maintenant être planifiés grâce au planificateur de tâches de Windows.

### Travail à réaliser :

- Ouvrir le planificateur de tâches de Windows
  - o Accessoires / Outils système / Planificateur de tâches
- On clique sur créer une tâche... et l'on donne un nom : RunJobAlimentationBDD
- On crée un déclencheur (dans 15 min par exemple)
- On crée une action :
  - o Action : Démarrer un programme
  - o Programme :
    - o C:\BI\DEPENSE\BUILD\JAlimentation\JAlimentation\_run.bat
- Attendre que la tâche se lance.
- Vérifier que tout a bien fonctionné.

## Conclusion

Le système d'extraction mis en place, permet de récupérer les données provenant d'un fichier Excel et à partir de ce fichier, générer les 3 fichiers plats et la génération ns avons tout simplement effectuer dans un répertoire du jour donc ns avons définit des répertoires horodatés.

A partir de 3 fichiers horodatés générés on a pu alimenter l'ODS (Operational Data Store). On a également alimenter l'entrepôt de données à partir de différentes tables ODS et pour finir on a créé un job principal(jAlimentationBDD) qu'on a pu construit et utiliser le planificateur de tâches de Windows pour programmer l'exécution automatique du job jAlimentationBDD qui est un job principal afin d'alimenter le DWH tous le jour à l'heure programmer.

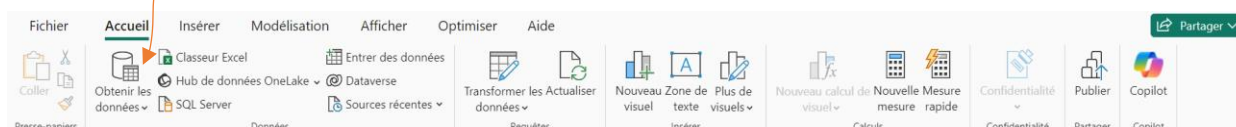
## Mise en place des tableaux de bords :



# Guide réalisation ETL pour ajout de données à partir de Data Warehouse (Source de données) dans Power BI Traitement pour toutes les tables afin d'avoir des données propres à analyser

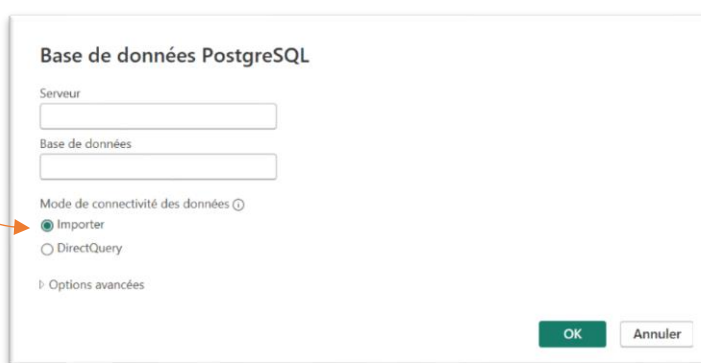
1. Ouvrir Power BI
2. Depuis le menu principal de la page accueil cliquer sur icone « **Obtenir les données** » puis « **Base de données PostgreSQL** »
3. Choisir le mode de connectivité de données(Importer) et les informations de la connexion à la base de données(DWH)
4. Sélectionner les tables concernées et cliquer sur “Transformer les données”
5. Modifier les titres des colonnes, nom des tables, Suppression des colonnes inutiles, renommage toutes les étapes appliquées.
6. On ferme et applique les modifications faites.

## ÉTAPE 2



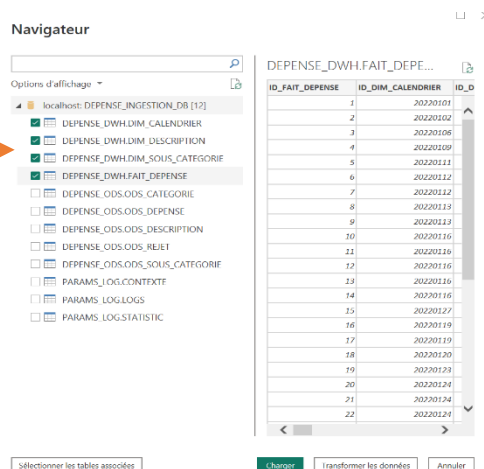
**Importer** : Import pour les datasets souvent stockés, consultés et optimisés avec VertiPaq

## ÉTAPE 3



Sélectionner les tables concernées et cliquer sur “Transformer les données”

## ÉTAPE 4



Les données seront ainsi importées dans Power Query

ÉTAPE 5

Modifier les titres des colonnes en supprimant tous les underscores

Requêtes [4]

×

✓

fx

= Source([Schema="DEPENSE\_DWH",Item="DIM\_SOUS\_CATEGORIE"])[Data]

	ID_DIM_SOUS_CATEGORIE	LB_SOUS_CATEGORIE	LB_CATEGORIE	DT_INSERTION	DT_MISE_A_JOUR
1		7 Abonnement Logiciel	Dépenses Fixes	04/06/2024 18:16:34	
2		8 Abonnement Télécommunications	Dépenses Fixes	04/06/2024 18:16:34	
3		9 Abonnement Transport	Dépenses Fixes	04/06/2024 18:16:34	
4		10 Assurance Maison	Dépenses Fixes	04/06/2024 18:16:34	
5		11 Assurance Voiture	Dépenses Fixes	04/06/2024 18:16:34	

ÉTAPE 6

Modifier les noms des tables

Requêtes [4]

×

✓

fx

= Source([Schema="DEPENSE\_DWH",Item="DIM\_SOUS\_CATEGORIE"])[Data]

	ID_DIM_SOUS_CATEGORIE	LB_SOUS_CATEGORIE	LB_CATEGORIE	DT_INSERTION	DT_MISE_A_JOUR
1		7 Abonnement Logiciel	Dépenses Fixes	04/06/2024 18:16:34	
2		8 Abonnement Télécommunications	Dépenses Fixes	04/06/2024 18:16:34	
3		9 Abonnement Transport	Dépenses Fixes	04/06/2024 18:16:34	

Paramètres d'une requête

PROPRIÉTÉS

Nom

DEPENSE\_DWH\_DIM\_SOUS\_CATEGORIE

Toutes les propriétés

ÉTAPE 7

Suppression des colonnes inutiles importées par power query

On renomme toutes les étapes appliquées

Requêtes [5]

×

✓

fx

Table.RenameColumns(DEPENSE\_DWH\_DIM\_SOUS\_CATEGORIE, [{"ID\_DIM\_SOUS\_CATEGORIE", "ID SOUS CATEGORIE"}],

SERTION	DT MISE A JOUR	LIGNE ACTIVE	JOB NAME	DEPENSE_DWH_DIM_SOUS_CATEGORIE
1	04/06/2024 18:16:34	null	1  DimSousCategorie	Table
2	04/06/2024 18:16:34	null	1  DimSousCategorie	Table
3	04/06/2024 18:16:34	null	1  DimSousCategorie	Table
4	04/06/2024 18:16:34	null	1  DimSousCategorie	Table
5	04/06/2024 18:16:34	null	1  DimSousCategorie	Table
6	04/06/2024 18:16:34	null	1  DimSousCategorie	Table
7	04/06/2024 18:16:34	null	1  DimSousCategorie	Table
8	04/06/2024 18:16:34	null	1  DimSousCategorie	Table
9	04/06/2024 18:16:34	null	1  DimSousCategorie	Table

Paramètres d'une requête

PROPRIÉTÉS

Nom

SOUS CATEGORIE

Toutes les propriétés

ÉTAPES APPLIQUÉES

Source

Navigation

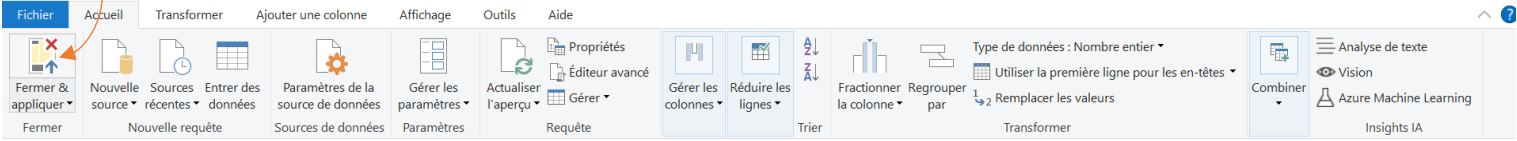
Renommages des colonnes

Suppression de colonnes inutili...

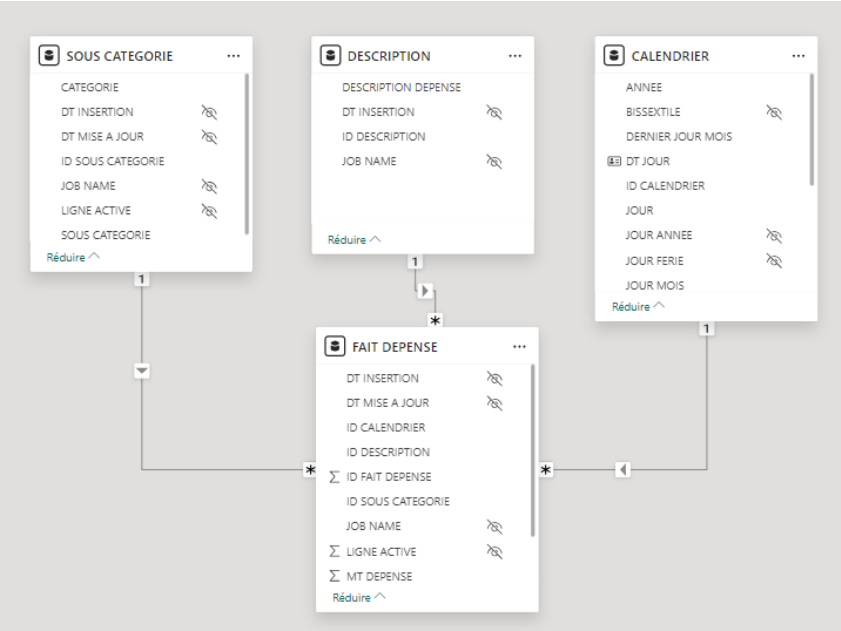
NB : on peut continuer les mêmes traitement pour toutes les autres tables

On ferme et applique les modifications faites

ÉTAPE 8



Guide technique : Les relations entre les tables et les types de liaisons



Les relations doivent être établies depuis chaque table vers la table de Faits(fact)

On désactive toutes les colonnes qui ne servent pas à l'analyse

### Type de relation entre les tables

Cardinalité

Direction du filtre croisé

Plusieurs à un (\*:1)

À sens unique

- Toutes les relations doivent être de **cardinalité plusieurs(\*) à un(1) : (\*:1)**
- **La direction de filtrage croisé** doit être **d'un seul sens** depuis la table de faits vers les différentes tables des dimensions

Toutes les tables des dimensions sont liées à la table de faits pour mettre en relation les valeurs et les calculs effectués selon la table de faits ce qui nous permet d'avoir un Dashboard dynamique et efficace

### Tableaux de bords

#### Dictionnaire de données :

Indicateur	Rapport	Axe d'analyse
Montant de dépenses	Montant total de dépenses par catégorie	catégorie
	Montant total de dépenses par sous-catégorie	sous-catégorie
	Montant total de dépenses par période(jour, mois, trimestre, semestre, années)	jour, mois, trimestre, semestre, années
Nombre de dépenses	Nombre total de dépenses par catégorie	catégorie
	Nombre total de dépenses par sous-catégorie	sous-catégorie
	Nombre total de dépenses par période(jour, mois, trimestre, semestre, années)	jour, mois, trimestre, semestre, années

# Catégories

Dépenses  
catégories



Année-Mois-Jour

- ✓ ☐ 2022  
^ ☐ 2023  
✓ ☐ Janvier  
✓ ☐ Février  
✓ ☐ Mars  
✓ ☐ Avril  
✓ ☐ Mai  
✓ ☐ Juin  
—

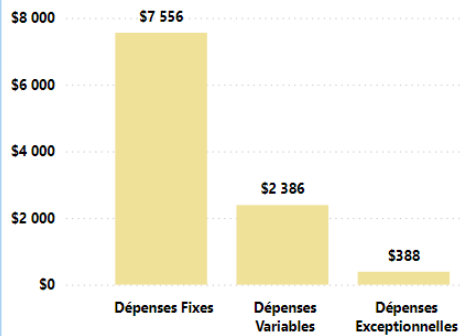
Montant total de  
dépenses

\$10 330

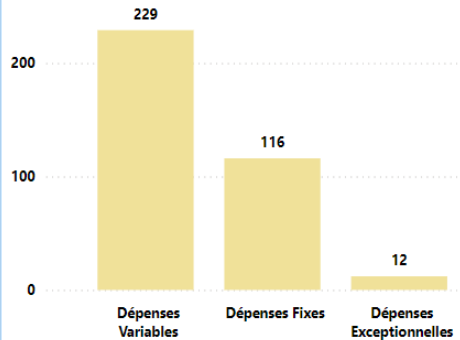
Catégories

3

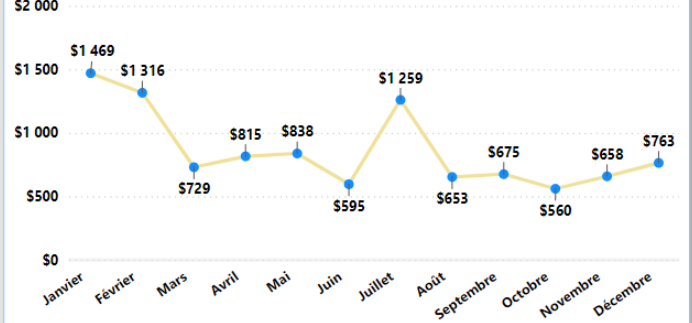
Dépenses/Catégories



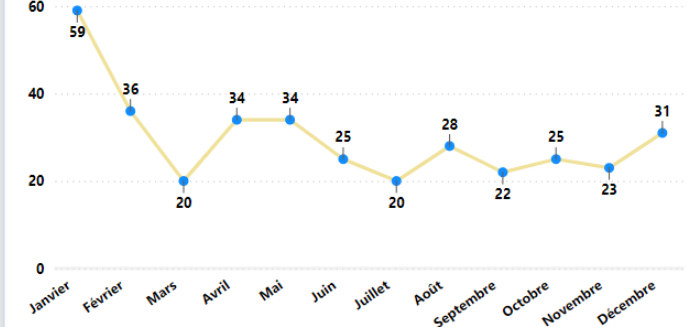
Nombre de dépenses/Catégories



Dépenses/période



Nombre de dépenses/période



# Sous-catégories

Dépenses  
sous-catégories



Année-Mois-Jour

- ✓ ☐ 2022  
^ ☐ 2023  
✓ ☐ Janvier  
✓ ☐ Février  
✓ ☐ Mars  
✓ ☐ Avril  
✓ ☐ Mai  
✓ ☐ Juin  
—

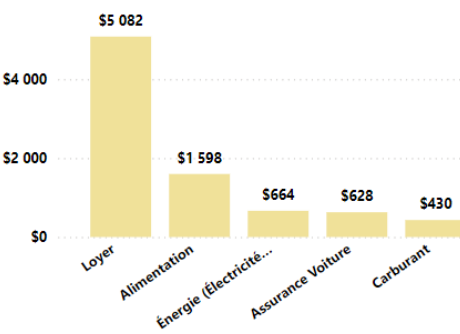
Montant total de  
dépenses

\$10 330

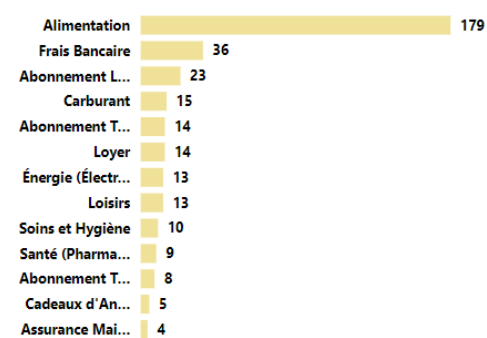
Sous-catégories

24

Top 5 Dépenses/Sous-catégorie



Nombre de dépenses/Sous-catégories



Dépenses/Sous-catégories

