R5C06 Exploitation de bases de données S5

-

Compte-rendu TP3 – Construction d’un magasin de données avec Talend

Aminata OUMOU RASSOUL NGOM

Groupe 13

S5 ~ Parcours C

BUT Informatique

Tables des matières

Introduction

Ce TP a pour objectif de mettre en œuvre la création et l'alimentation d'un magasin de données à partir d'un entrepôt existant, en utilisant l'outil Talend Open Studio. Contrairement à l'entrepôt de données, le magasin de données est orienté vers l'analyse métier et repose sur une granularité plus fine et une structure adaptée à l'analyse décisionnelle.

Dans ce travail, la base DW13 joue le rôle de source de données, tandis que la base DM13 correspond au magasin de données cible. L'objectif principal est d'extraire les informations pertinentes de l'entrepôt, de les transformer selon les règles définies (jointures, regroupements, calculs, filtrage des doublons) puis de les charger dans les différentes tables dimensionnelles du magasin.

Trois dimensions sont ainsi construites : MAGASINS, PRODUITS et TEMPS, chacune nécessitant un traitement spécifique, notamment la gestion des jointures, des conversions de types et des regroupements. Une attention particulière est portée à la cohérence des données, au respect des clés primaires et étrangères, ainsi qu'à la gestion des valeurs manquantes ou incohérentes.

L'ensemble du processus est réalisé à l'aide de jobs Talend, permettant d'automatiser l'extraction, la transformation et le chargement des données. Les résultats sont ensuite vérifiés à l'aide de requêtes SQL afin de s'assurer de la conformité des volumes attendus pour chaque dimension du magasin de données.

Ce compte rendu présente de manière progressive les différentes étapes de réalisation du TP, les choix effectués lors de la conception des jobs, ainsi que les vérifications effectuées pour valider le bon fonctionnement de l'alimentation du magasin de données.

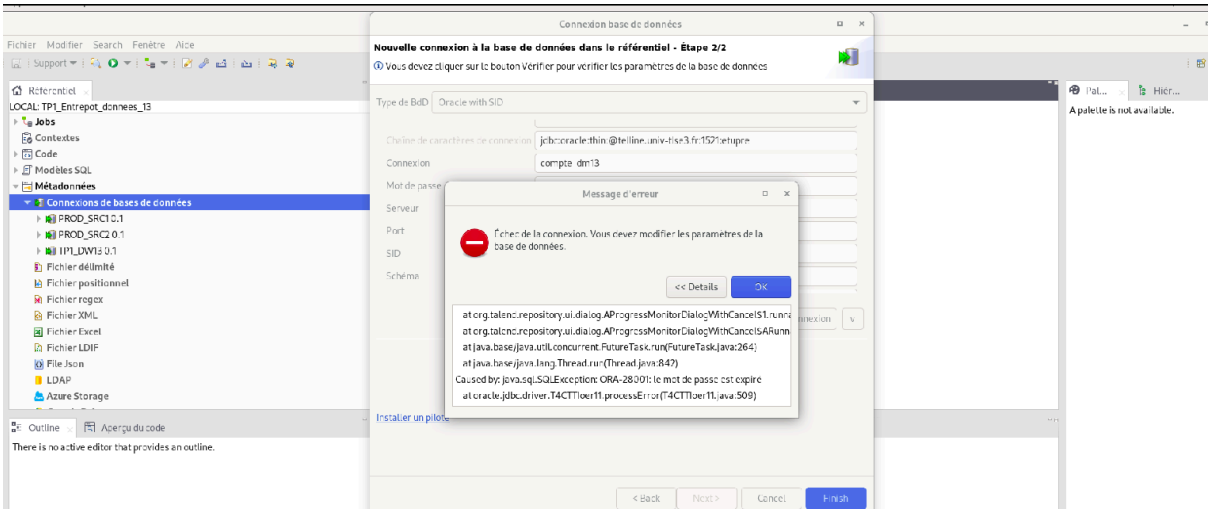
Création et alimentation du magasin de données

Avant de pouvoir procéder à l'alimentation des dimensions du magasin de données, il a été nécessaire de mettre en place l'environnement technique permettant d'accueillir les tables du magasin. Initialement, le magasin de données devait être créé et exploité dans un compte dédié de type *compte_dmXX*. Cependant, lors de la tentative de connexion à ce compte via Talend, une erreur d'authentification est apparue indiquant que le mot de passe était expiré, rendant toute connexion impossible depuis l'outil.

Malgré la création effective du compte côté base de données, il n'était donc pas possible d'y accéder pour y déployer les tables et exécuter les jobs d'alimentation. Afin de ne pas bloquer l'avancement du TP et après vérification de la faisabilité technique, une solution alternative a été retenue.

En effet, après analyse et recherches, il est apparu qu'il était tout à fait possible de créer un magasin de données directement au sein de l'entrepôt existant, à condition de respecter une organisation claire des tables et une nomenclature explicite. Cette approche est d'ailleurs couramment utilisée dans des contextes pédagogiques ou lorsque les droits d'accès sont limités.

Le choix a donc été fait de créer les tables du magasin de données directement dans le schéma **COMPTE_DW13**, tout en conservant une séparation logique claire entre l'entrepôt et le magasin à travers le préfixe **DM13_**. Cette solution permet de poursuivre l'ensemble du TP sans perte fonctionnelle, tout en respectant les objectifs pédagogiques demandés.



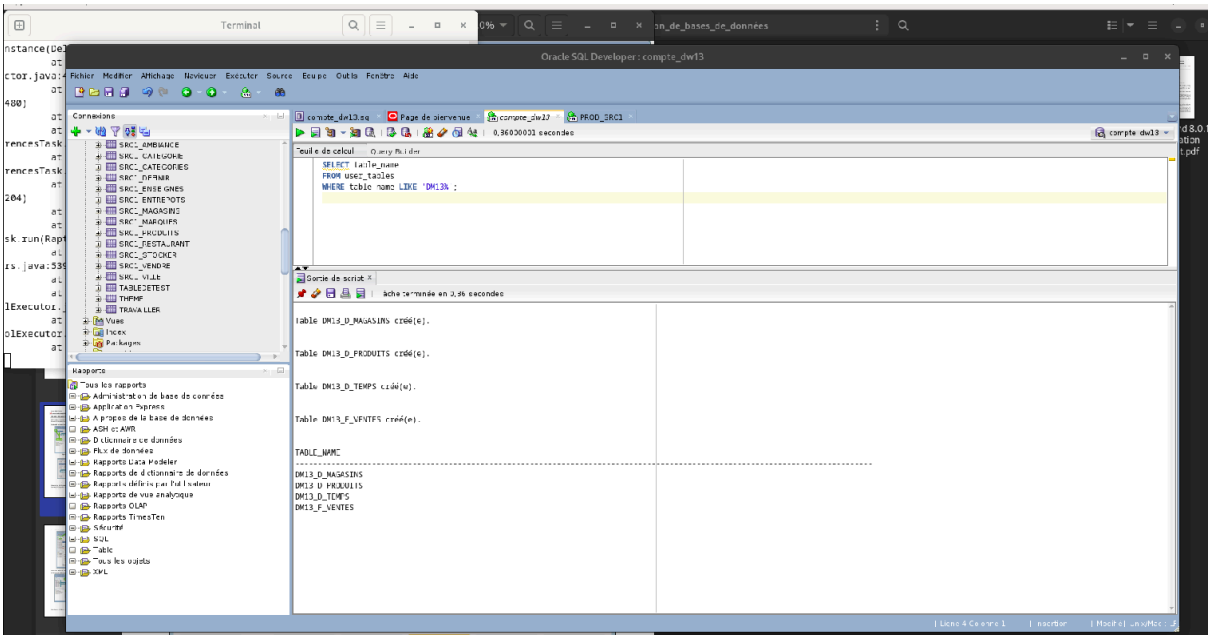
message d'erreur lors de la connexion au compte DM (mot de passe expiré)

Création des tables du magasin de données

Une fois la décision prise de créer le magasin de données directement dans le schéma **COMPTE_DW13**, les différentes tables du magasin ont été mises en place à l'aide de scripts SQL exécutés dans SQL Developer. Cette étape a permis de créer l'ensemble des structures nécessaires au stockage des données décisionnelles, à savoir les tables **DM13_D_MAGASINS**, **DM13_D_PRODUITS**, **DM13_D_TEMPS** et **DM13_F_VENTES**.

La création de ces tables a été réalisée en respectant scrupuleusement le dictionnaire de données fourni dans l'énoncé du TP. Les types, longueurs de champs et clés primaires ont été définis conformément au modèle dimensionnel attendu. Les relations futures entre la table de faits et les tables de dimensions ont également été prises en compte afin d'assurer la cohérence globale du schéma en étoile.

Une fois les scripts exécutés, une vérification a été effectuée dans SQL Developer afin de confirmer la bonne création des tables dans le schéma **COMPTE_DW13**. Cette vérification a permis de s'assurer que l'ensemble des tables du magasin était bien présent et prêt à être alimenté.



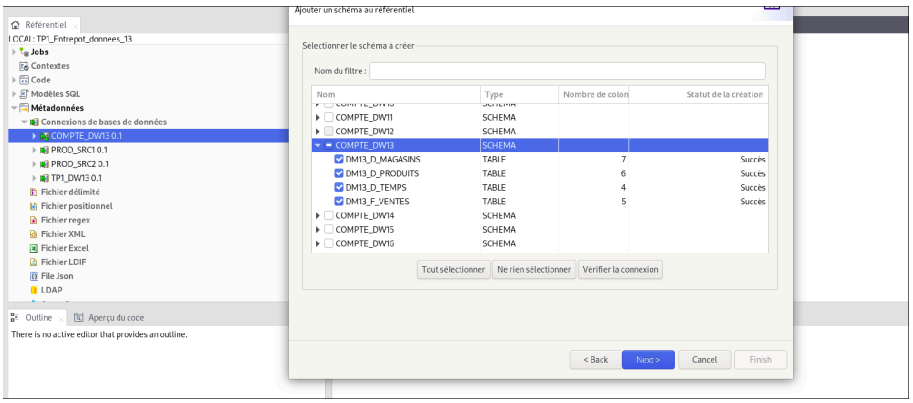
affichage des tables DM13 dans SQL Developer

Récupération des métadonnées dans Talend

Après la création physique des tables, l'étape suivante a consisté à importer ces structures dans Talend afin de pouvoir les utiliser dans les différents jobs d'alimentation. Une connexion à la base **COMPTE_DW13** a donc été configurée dans le référentiel Talend.

Une fois la connexion validée, l'assistant de récupération de schéma a été utilisé pour importer les tables du magasin de données. Les tables **DM13_D_MAGASINS**, **DM13_D_PRODUITS**, **DM13_D_TEMPS** et **DM13_F_VENTES** ont ainsi été ajoutées au référentiel du projet.

Cette étape est essentielle car elle permet à Talend de connaître précisément la structure des tables cibles. Les schémas récupérés sont ensuite réutilisés directement dans les composants tDBOutput, ce qui garantit la cohérence des types de données et limite les erreurs lors des insertions.



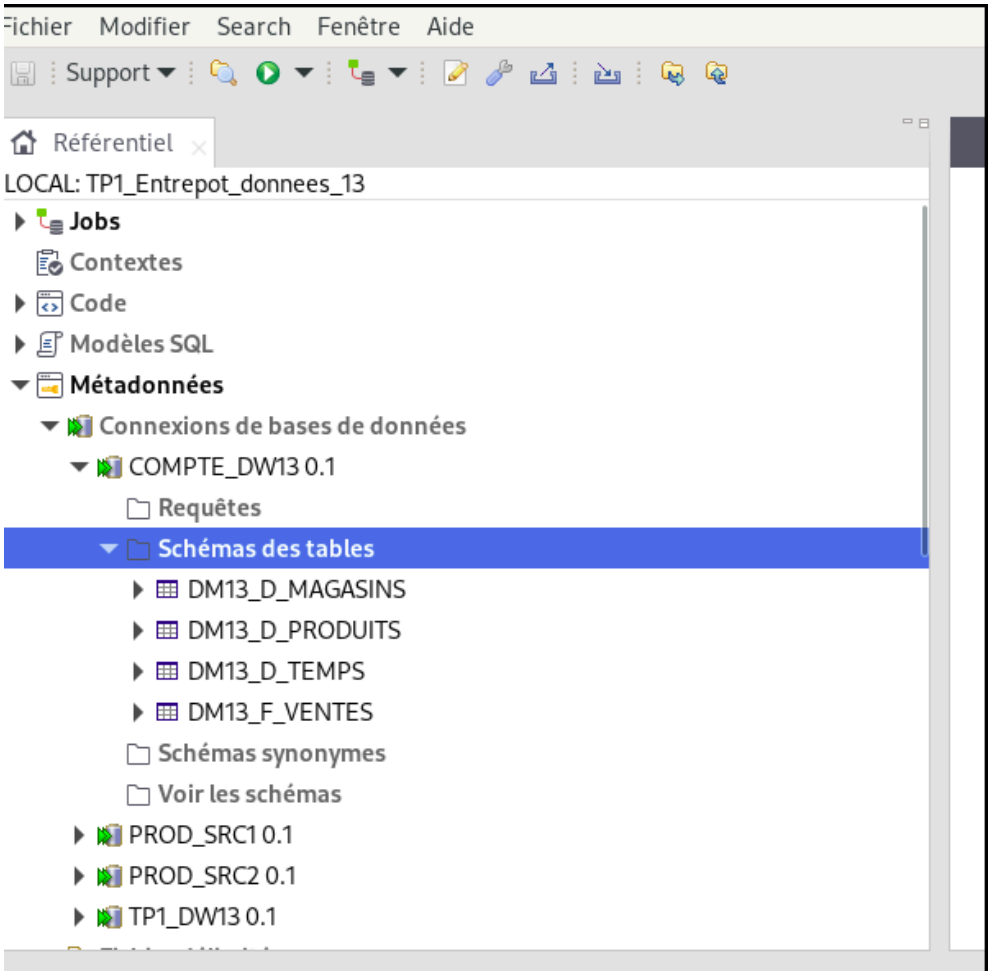
récupération des schémas DM13 dans le référentiel Talend

Organisation du projet Talend

Une fois les métadonnées disponibles, le projet Talend a été organisé de manière claire et structurée. Un job distinct a été créé pour chaque dimension du magasin de données afin de faciliter la lecture, la maintenance et les tests.

Ainsi, un job est dédié à la dimension temps, un autre à la dimension magasins et un troisième à la dimension produits. Cette organisation permet de séparer logiquement les traitements et d'exécuter chaque alimentation de manière indépendante si nécessaire.

Même si le magasin de données est hébergé dans le même schéma que l'entrepôt, la séparation logique est assurée par la nomenclature utilisée et par la structuration des jobs. Cette approche garantit une bonne lisibilité du projet et respecte les principes classiques de modélisation décisionnelle.



arborescence des jobs Talend

1. Alimentation de la dimension TEMPS

1.1 Objectif

L'objectif de cette étape est d'alimenter la dimension DM13_D_TEMPS du magasin de données.
Contrairement à l'entrepôt de données, où les ventes sont enregistrées au jour le jour, le magasin de données adopte une granularité mensuelle.
L'objectif est donc d'extraire les dates de vente depuis l'entrepôt, puis d'en déduire les informations temporelles nécessaires à l'analyse décisionnelle : mois, libellé du mois, trimestre et année.

Les données sources proviennent de la table DW13_VENTES, et plus précisément de la colonne DATE_VT, à partir de laquelle sont calculés les différents attributs temporels.

1.2 Construction du job Talend

Pour réaliser cette alimentation, un job spécifique nommé Job_D_TEMPS a été créé dans Talend Open Studio.

Ce job repose sur une chaîne classique d'ETL composée de trois composants principaux :

- tDBInput : lecture des données depuis l'entrepôt DW13
- tMap : transformation et formatage des données temporelles
- tDBOutput : insertion des données dans la table DM13_D_TEMPS

1.3 Paramétrage du tDBInput

Le composant tDBInput est configuré pour se connecter à la base DW13 et lire la table DW13_VENTES.

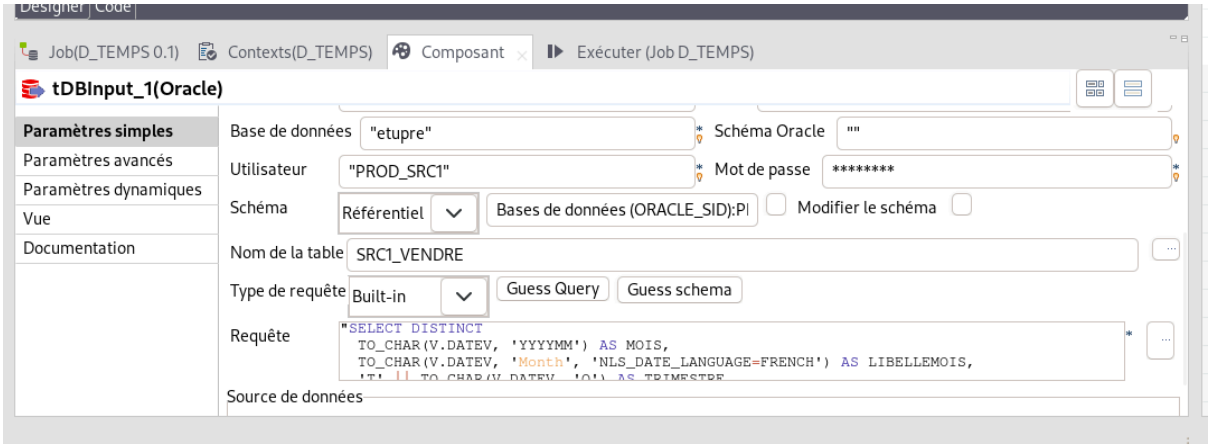
Une requête SQL est utilisée afin d'extraire uniquement les informations nécessaires à la dimension temps.
Les fonctions SQL permettent de calculer directement les attributs temporels :

- le mois au format *MM/YYYY*
- le libellé du mois en français
- le trimestre
- l'année

La requête utilisée est de la forme :

```
SELECT DISTINCT
    TO_CHAR(DATE_VT, 'MM/YYYY') AS MOIS,
    TO_CHAR(DATE_VT, 'Month', 'NLS_DATE_LANGUAGE=FRENCH') AS
LIBELLEMOIS,
    TO_CHAR(DATE_VT, 'Q/YYYY') AS TRIMESTRE,
    TO_CHAR(DATE_VT, 'YYYY') AS ANNEE
FROM DW13_VENTES
```

Cette requête permet d'éviter les doublons grâce au `DISTINCT` et garantit que chaque mois n'apparaît qu'une seule fois dans la dimension.



Configuration du tDBInput avec la requête SQL

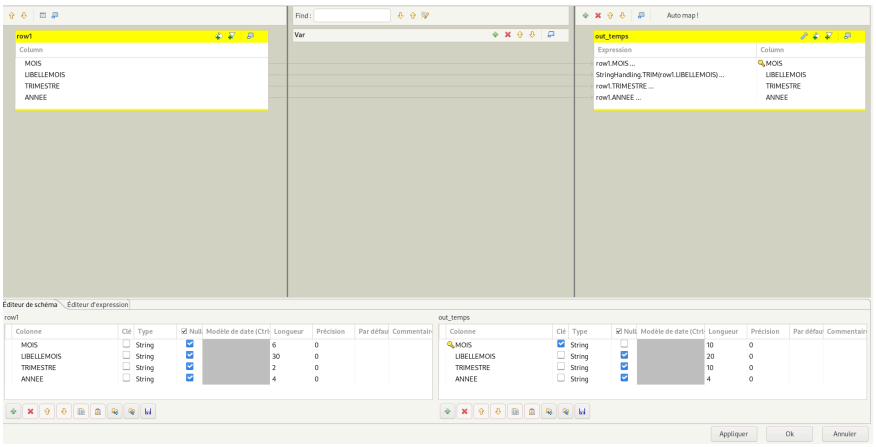
1.4 Mapping dans le tMap

Le composant tMap permet de faire la correspondance entre les colonnes issues du tDBInput et celles de la table cible DM13_D_TEMPS.

Les correspondances sont les suivantes :

- `MOIS` → MOIS
- `LIBELLEMOIS` → LIBELLEMOIS
- `TRIMESTRE` → TRIMESTRE
- `ANNEE` → ANNEE

Aucune transformation complexe n'est nécessaire, mis à part un `TRIM()` appliqué sur le libellé du mois afin d'éviter les espaces générés par la fonction `TO_CHAR`.



Vue du tMap avec le mapping des colonnes

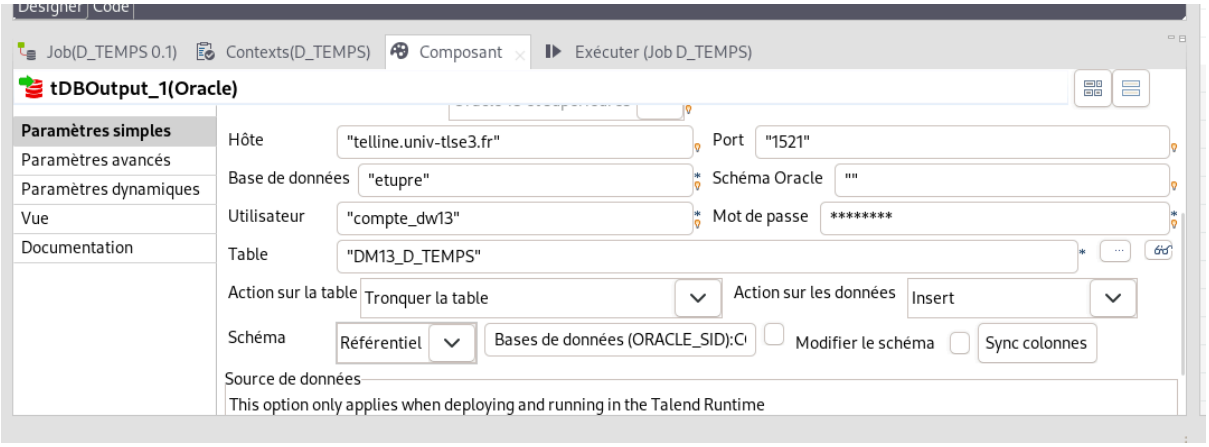
1.5 Paramétrage du tDBOutput

Le composant tDBOutput est configuré pour insérer les données dans la table DM13_D_TEMPS du magasin de données.

Les paramètres principaux sont :

- Table cible : DM13_D_TEMPS
- Action sur la table : Truncate table
- Action sur les données : Insert
- Schéma : basé sur le référentiel Talend

Le mode *Truncate + Insert* permet de garantir que la table est vidée avant chaque rechargement, évitant ainsi les doublons.



Paramétrage du tDBOutput

2. Alimentation de la dimension MAGASINS

2.1 Objectif

L'objectif de cette étape est d'alimenter la dimension DM13_D_MAGASINS du magasin de données.
Cette dimension permet de décrire les points de vente à travers différentes informations telles que le code du magasin, son nom, son enseigne, sa ville ainsi que son département et sa région.

Contrairement à l'entrepôt de données, le magasin de données présente une granularité plus fine et impose une structuration plus rigoureuse des données. Il est donc nécessaire d'éliminer les doublons, de contrôler les types et de s'assurer de la cohérence des informations géographiques avant l'insertion dans la table cible.

Les données utilisées pour cette alimentation proviennent de l'entrepôt DW13, principalement de la table des magasins, complétée par les informations géographiques déjà intégrées lors des travaux précédents.

2.2 Mise en place du job d'alimentation

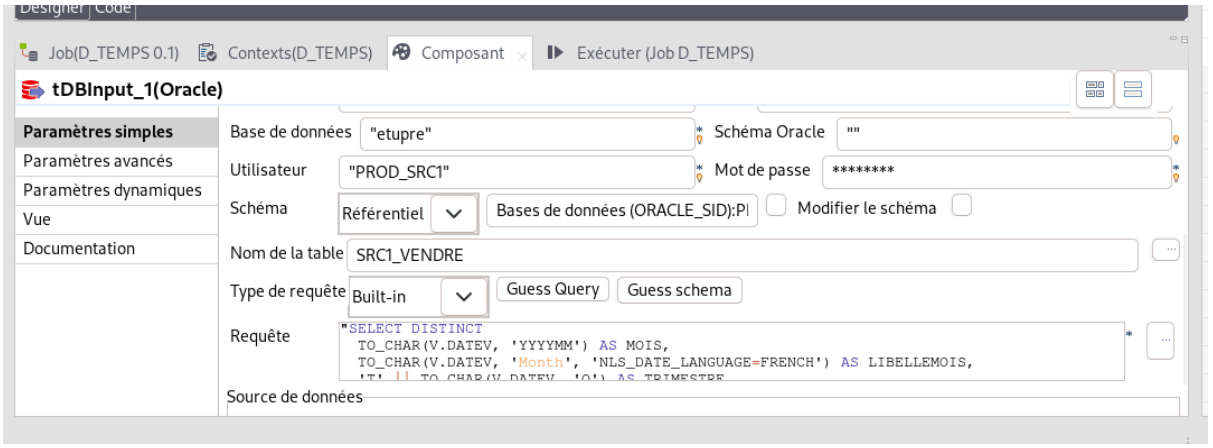
Pour réaliser cette alimentation, un job Talend nommé Job_D_MAGASINS a été créé.
Ce job reprend l'architecture classique d'un flux ETL, composée d'un composant d'extraction, d'un composant de transformation et d'un composant de chargement.

Le composant tDBInput est connecté à la base DW13. Afin d'éviter les doublons et de simplifier le traitement, une vue spécifique appelée V_MAGASINS_UNIQUES a été utilisée. Cette vue permet de ne conserver qu'une seule occurrence par magasin et d'assurer la cohérence des données extraites.

La requête associée à ce composant permet notamment de récupérer :

- *le code du magasin,*
- *la raison sociale,*
- *la ville,*
- *le code postal,*
- *les informations nécessaires à la construction du département.*

Cette étape garantit que seules les données utiles au magasin de données sont transmises au reste du traitement.



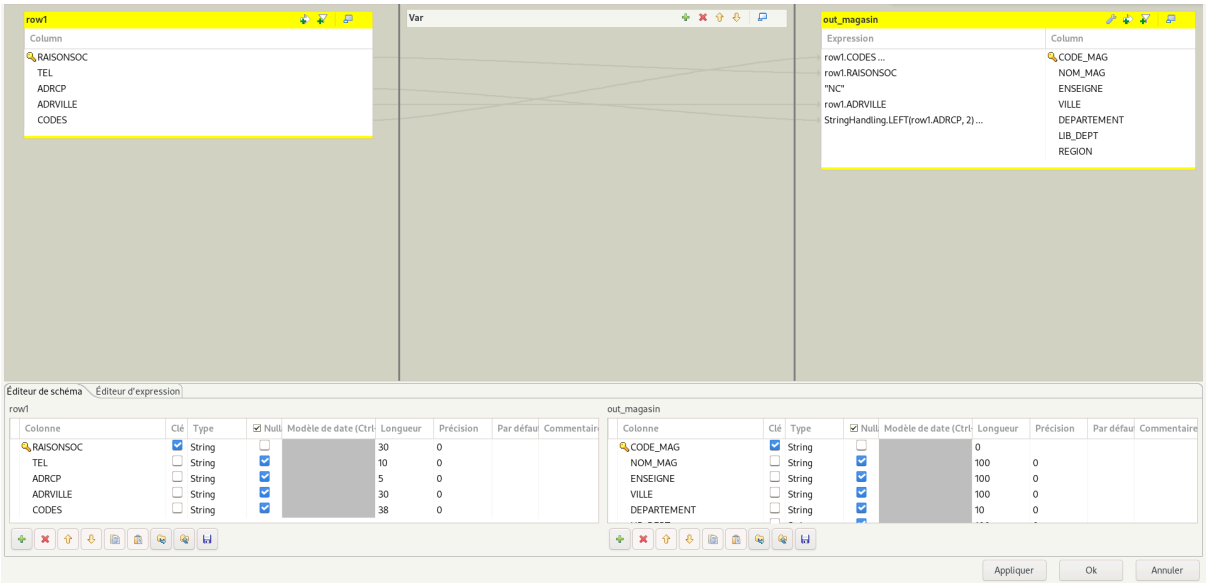
Configuration tDBInput

2.3 Transformation des données avec tMap

Les données extraites sont ensuite traitées dans un composant tMap, qui joue un rôle central dans ce job.

Le tMap permet de réaliser les correspondances entre les colonnes sources et les colonnes de la table DM13_D_MAGASINS, tout en appliquant certaines transformations nécessaires.
Le code du magasin est directement utilisé comme clé primaire.
Le nom du magasin est récupéré depuis la raison sociale.
L'enseigne est fixée à une valeur par défaut lorsque l'information n'est pas disponible.
La ville est transmise telle quelle.
Le département est calculé à partir du code postal à l'aide d'une extraction des deux premiers caractères, ce qui permet d'uniformiser les valeurs et d'éviter les erreurs observées précédemment dans l'entrepôt.

Cette étape permet également d'éviter les problèmes de typage rencontrés lors des premières tentatives, notamment entre chaînes de caractères et valeurs numériques.



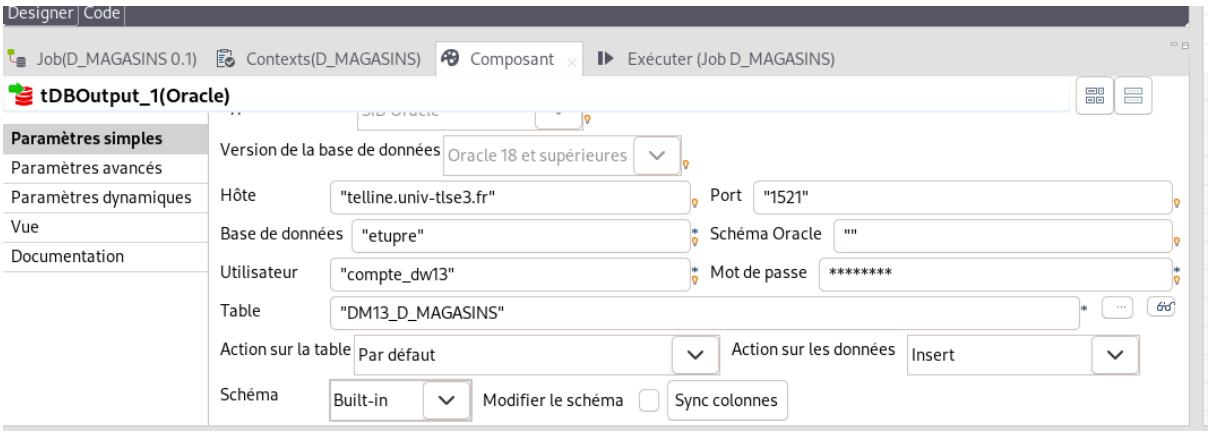
Configuration tMap

2.4 Chargement dans la table DM13_D_MAGASINS

Une fois les données transformées, elles sont insérées dans la table DM13_D_MAGASINS à l'aide d'un composant tDBOutput connecté au compte du magasin de données.

Le mode d'insertion utilisé est un simple INSERT, la structure de la table ayant déjà été créée au préalable. Le schéma est récupéré depuis le référentiel Talend afin d'assurer la cohérence entre les types attendus et les données insérées.

Cette étape finalise le processus d'alimentation de la dimension magasins.



Configuration tDBOutput

3. Alimentation de la dimension PRODUITS

3.1 Objectif

Cette étape a pour objectif d'alimenter la dimension DM13_D_PRODUITS du magasin de données.
Contrairement aux dimensions TEMPS et MAGASINS, la dimension PRODUITS présente une complexité plus importante, car elle regroupe des informations issues de plusieurs sources et nécessite plusieurs transformations avant insertion.

L'objectif est de constituer une dimension produit complète, contenant :

- *l'identifiant du produit,*
- *son libellé,*
- *sa catégorie,*
- *sa classe (catégorie parente),*

- *sa marque,*
- *ainsi que le groupe auquel appartient cette marque.*

Ces informations sont indispensables pour permettre par la suite des analyses par type de produit, famille ou marque dans le magasin de données.

3.2 Construction du job Talend

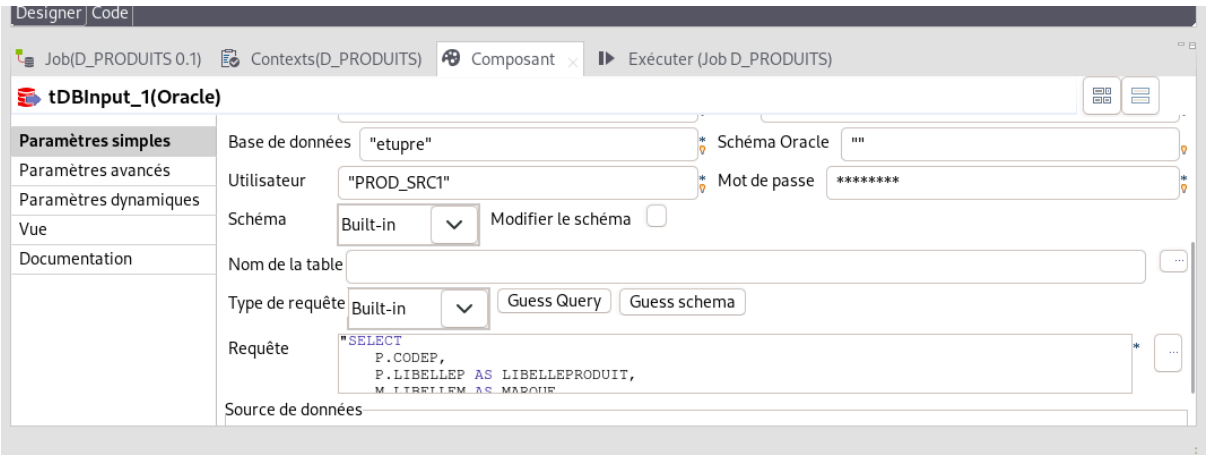
Pour cette alimentation, un job spécifique nommé Job_D_PRODUITS a été créé dans Talend Open Studio.
Comme pour les autres dimensions, le job repose sur une architecture classique en trois étapes : extraction, transformation et chargement.

Les données sources proviennent principalement des tables de l'entrepôt DW13, notamment :

- *la table des produits,*
- *la table des catégories,*
- *la table des marques.*

Certaines informations n'étant pas directement disponibles sous une forme exploitable, des transformations sont nécessaires afin d'obtenir une structure conforme au modèle du magasin de données.

Le composant tDBInput est utilisé pour interroger la base DW13 et récupérer les informations utiles. La requête permet d'extraire le code du produit, son libellé, sa catégorie ainsi que sa marque, en effectuant les jointures nécessaires avec les tables de référence.



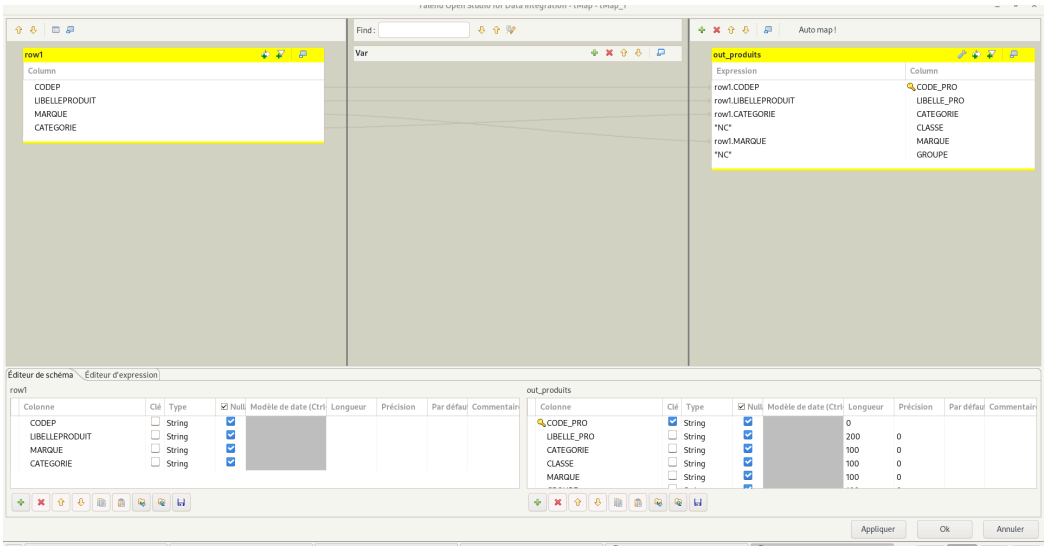
Paramétrage du tDBInput pour la dimension PRODUITS

3.3 Transformation des données dans le tMap

Le composant tMap joue un rôle central dans cette étape.
Il permet d'effectuer les correspondances entre les champs issus de l'entrepôt et ceux attendus dans la table DM13_D_PRODUITS, tout en appliquant certaines règles de gestion.

Le code produit est directement utilisé comme clé primaire de la dimension.
Le libellé du produit est récupéré sans modification particulière.
La catégorie est extraite depuis la table des catégories, tandis que la classe correspond à la catégorie parente associée.
La marque est issue de la table des marques, et lorsqu'aucune correspondance n'est trouvée, une valeur par défaut est affectée afin d'éviter les valeurs nulles.
De la même manière, le groupe de marque est renseigné lorsque l'information est disponible, sinon une valeur générique est utilisée.

Ces choix permettent de garantir l'intégrité de la dimension PRODUITS, même lorsque certaines données sont absentes ou incomplètes dans les sources.



Vue du tMap montrant les correspondances des champs PRODUITS

3.4 Chargement dans la table DM13_D_PRODUITS

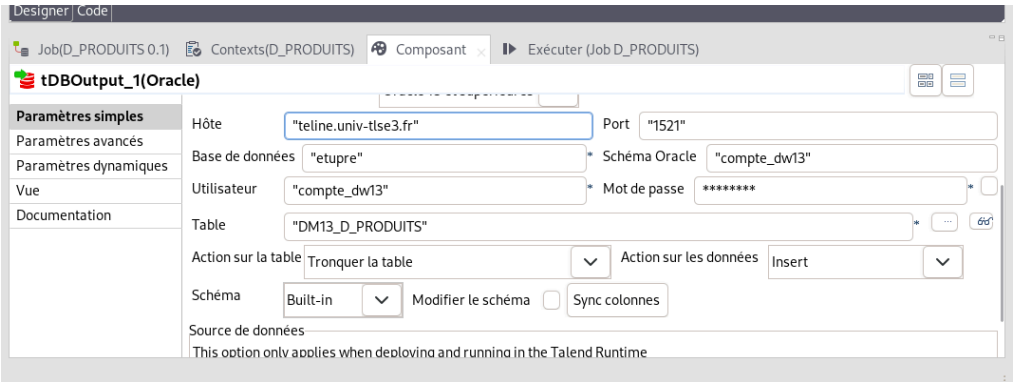
Une fois les transformations réalisées, les données sont insérées dans la table DM13_D_PRODUITS à l'aide d'un composant tDBOutput.

Le composant est configuré avec :

- *une connexion vers le compte* `compte_dm13`*,*
- *la table cible* `DM13_D_PRODUITS`*,*

- *une action de type Truncate + Insert, afin de garantir un rechargement propre de la dimension à chaque exécution du job.*

Le schéma est récupéré depuis le référentiel Talend, ce qui permet d'éviter les erreurs de typage et d'assurer la cohérence entre la structure de la table et les données insérées.



Paramétrage du tDBOutput de la dimension PRODUITS

4. Alimentation de la table de faits DM13_F_VENTES

4.1 Objectif

L'objectif de cette étape est d'alimenter la table de faits **DM13_F_VENTES**, qui constitue le cœur du magasin de données. Contrairement aux dimensions, cette table ne stocke pas des informations descriptives mais des mesures quantitatives issues de l'activité de vente.

Dans le magasin de données, les ventes ne sont plus stockées au niveau journalier comme dans l'entrepôt, mais agrégées à un niveau mensuel. Chaque ligne de la table de faits correspond donc à une combinaison unique entre un magasin, un produit et un mois, associée à des indicateurs numériques tels que la quantité vendue et le bénéfice.

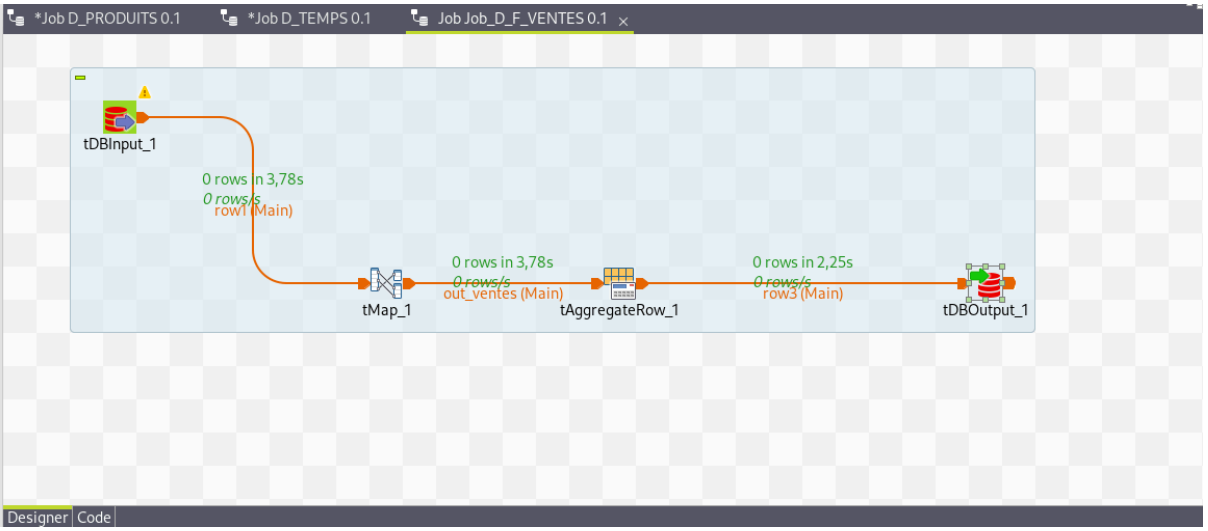
Cette étape est essentielle car elle permet de passer d'une logique transactionnelle à une logique décisionnelle, adaptée à l'analyse.

4.2 Mise en place du job d'alimentation

Pour réaliser cette alimentation, un job spécifique nommé **Job_D_F_VENTES** a été créé dans Talend.
Ce job repose sur une chaîne de traitement plus complète que pour les dimensions, car il nécessite une agrégation des données.

Le flux général du traitement est le suivant :

– extraction des ventes depuis la table DW13_VENTES,
– transformation et préparation des données dans un tMap,
– agrégation des données via un composant tAggregateRow,
– chargement du résultat agrégé dans la table DM13_F_VENTES.



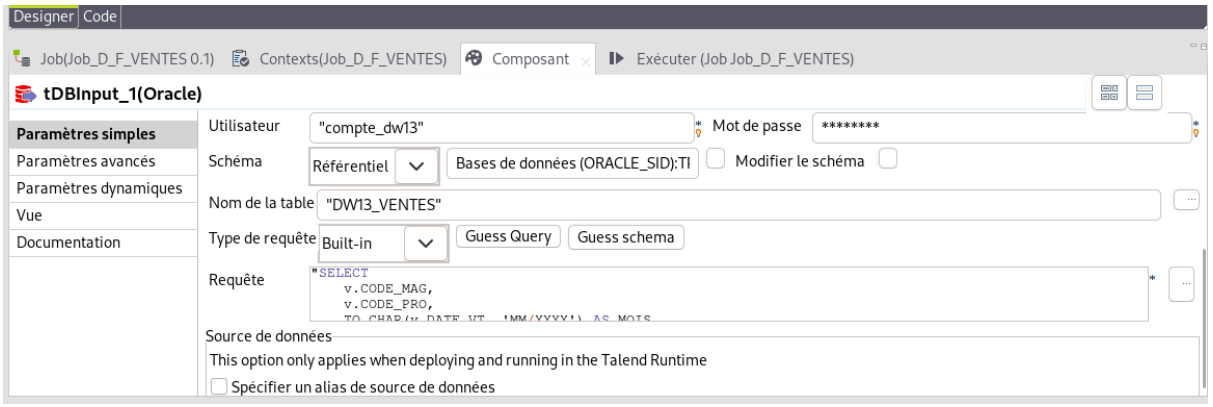
vue globale du job Job_D_F_VENTES dans Talend

4.3 Extraction des données depuis l'entrepôt (tDBInput)

Le composant tDBInput est configuré pour se connecter à la base DW13 et interroger la table DW13_VENTES.
La requête permet d'extraire les informations nécessaires à la construction de la table de faits, à savoir :

– le code du magasin,
– le code du produit,
– la date de vente,
– la quantité vendue,
– le bénéfice associé à la vente.

La date de vente est conservée sous forme de date à ce stade, car la transformation en mois est réalisée ultérieurement.



paramétrage du tDBInput avec la requête SQL

4.4 Transformation des données dans le tMap

Le composant tMap permet de préparer les données avant leur agrégation.

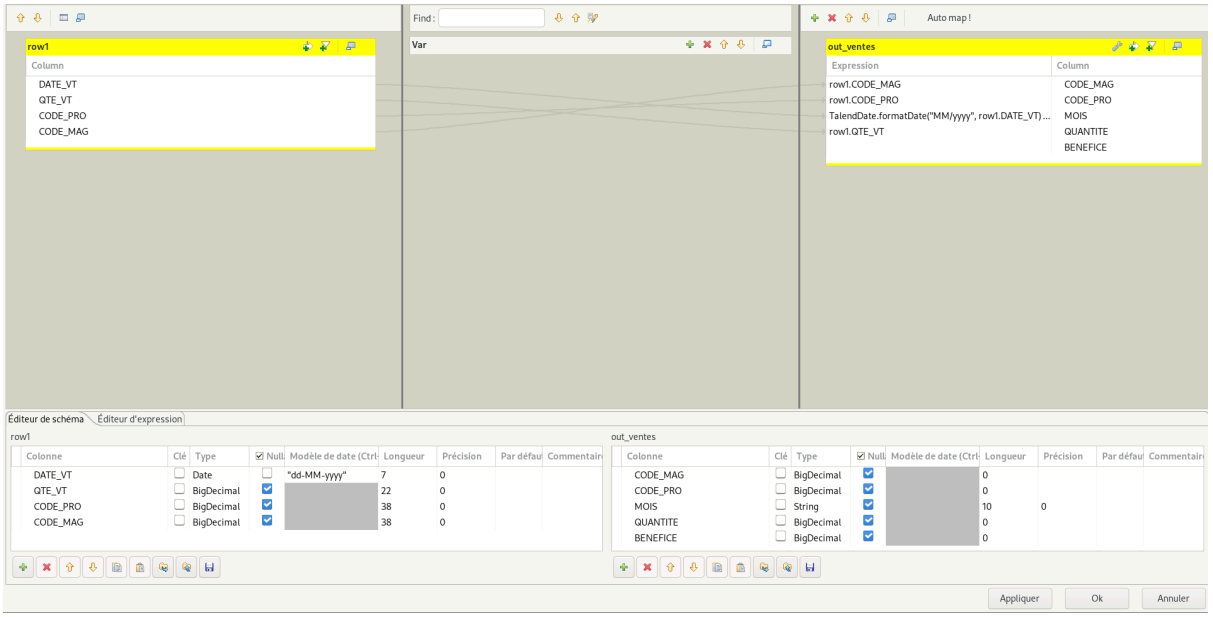
Dans ce composant, les champs issus du tDBInput sont mappés vers une sortie intermédiaire appelée **out_ventes**.
La transformation principale réalisée ici concerne la date de vente : celle-ci est convertie en un format mensuel à l'aide de la fonction suivante :

```
TalendDate.formatDate("MM/yyyy", row1.DATE_VT)
```

Cette transformation permet d'obtenir une granularité mensuelle conforme au modèle du magasin de données.

Les autres champs sont transmis tels quels :
– CODE_MAG
– CODE_PRO
– QTE_VT
– BENEFICE

Le tMap ne réalise pas encore d'agrégation, mais prépare les données pour le composant suivant.



tMap montrant la transformation de la date et le mapping vers out_ventes

4.5 Agrégation des données avec tAggregateRow

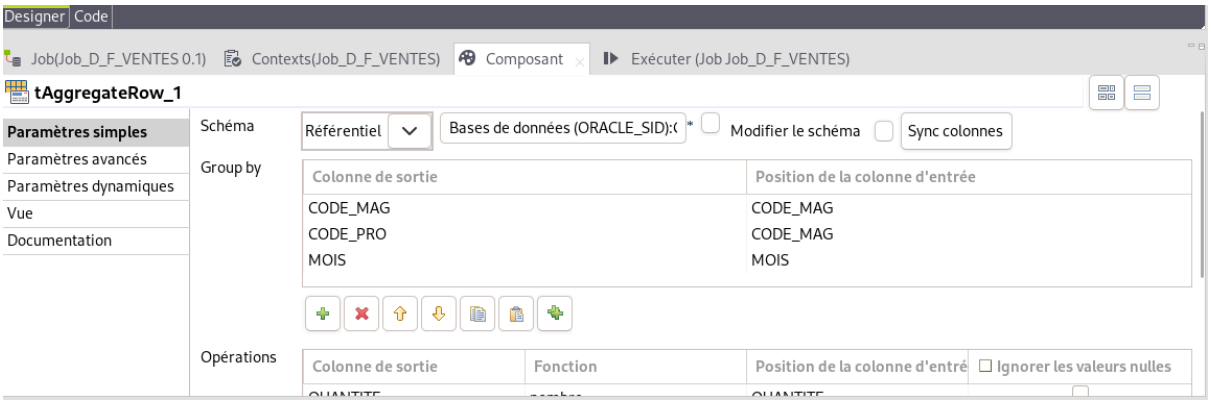
L'agrégation des ventes est réalisée à l'aide du composant **tAggregateRow**, qui constitue l'élément central de ce traitement.

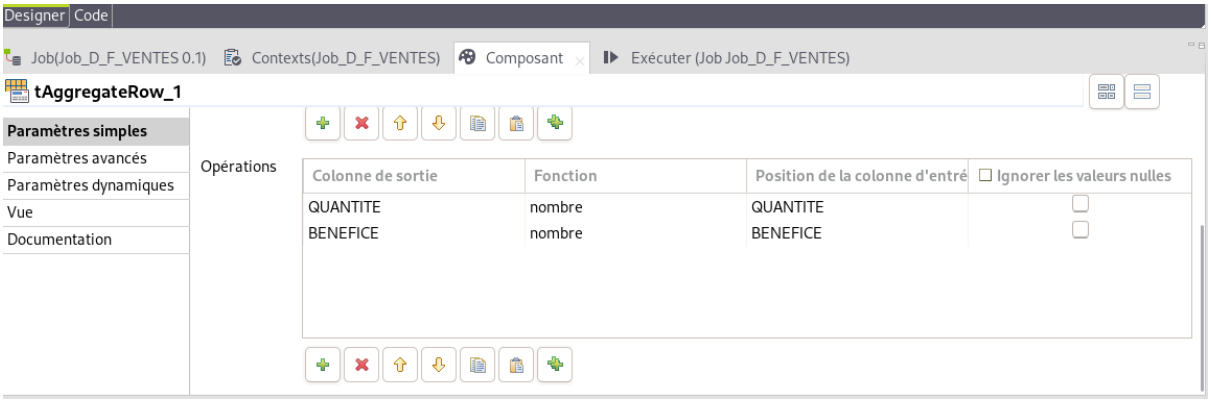
Dans ce composant, les données sont regroupées selon les champs suivants :
– CODE_MAG
– CODE_PRO
– MOIS

Ces trois champs définissent la granularité finale de la table de faits.

Les mesures sont ensuite agrégées à l'aide des fonctions suivantes :
– QUANTITE : somme des quantités vendues
– BENEFICE : somme des bénéfices

Le composant tAggregateRow permet ainsi de transformer les ventes unitaires issues de l'entrepôt en indicateurs consolidés par mois, par produit et par magasin.





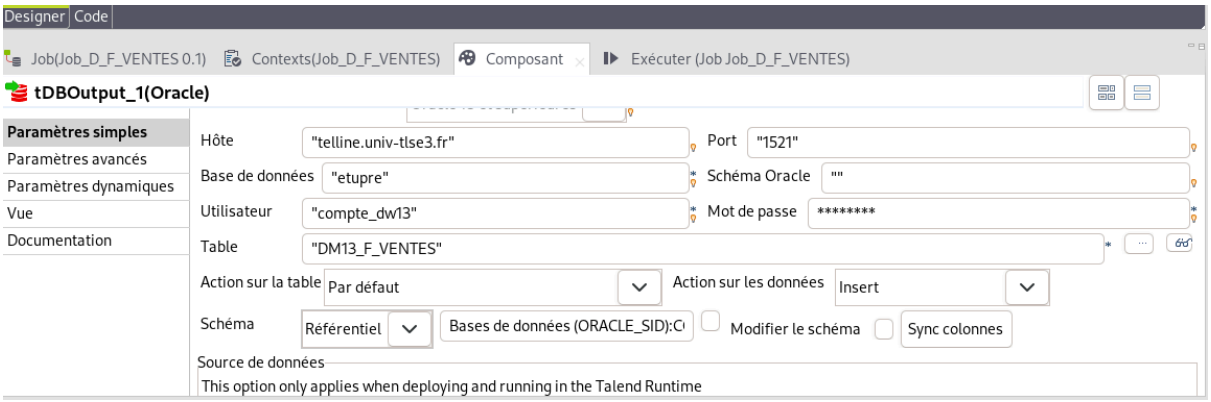
Paramétrage du tAggregateRow avec les champs de groupement et les fonctions SUM

4.6 Chargement dans la table DM13_F_VENTES

Une fois les données agrégées, celles-ci sont envoyées vers le composant tDBOutput afin d'être insérées dans la table DM13_F_VENTES.

Le tDBOutput est configuré avec :
– la connexion vers le schéma COMPTE_DW13,
– la table cible DM13_F_VENTES,
– une action de type **Insert**,
– un schéma issu du référentiel Talend.

Cette configuration permet d'insérer directement les résultats de l'agrégation sans modification supplémentaire.



Paramétrage du tDBOutput pour la table DM13_F_VENTES

4.7 Exécution du job et vérification

Après exécution du job, le traitement se termine sans erreur et les données sont correctement insérées dans la table de faits.

Exécution du Job réussi

Conclusion

L'ensemble des trois TP réalisés dans ce module a permis de comprendre concrètement les différentes étapes de mise en place d'un système décisionnel, depuis la construction de l'entrepôt de données jusqu'à l'alimentation d'un magasin de données exploitable pour l'analyse.

Au fil des séances, nous avons appris à structurer les données, à gérer les relations entre les tables, à appliquer des transformations adaptées et à automatiser les traitements à l'aide de Talend.

Le dernier TP, consacré au magasin de données, a permis de consolider ces acquis en mettant en œuvre une logique décisionnelle complète : création des dimensions, gestion des agrégations, utilisation du composant **tAggregateRow** et alimentation d'une table de faits à

une granularité mensuelle. Cette étape finale illustre bien la différence entre un système transactionnel et un système orienté analyse.

Malgré certaines contraintes techniques, notamment liées aux accès aux bases de données, les objectifs ont pu être atteints grâce à une organisation rigoureuse et à une bonne compréhension des mécanismes d'ETL.
Ces travaux ont permis de mieux appréhender le rôle d'un entrepôt et d'un magasin de données dans un contexte professionnel, tout en développant une réelle autonomie dans l'utilisation de Talend et des bases de données décisionnelles.