R5C06 Exploitation de bases de données S5

-

Compte-rendu TP2 – Construction d’un entrepôt de données avec Talend

Aminata OUMOU RASSOUL NGOM

Groupe 13

S5 ~ Parcours C

BUT Informatique

Introduction **3**
1. Alimentation de la table DW13_MARQUES 3
1.1. Objectif 3
1.2. Construction du job Talend 3
1.3. Vérifications dans SQL Developer 5
2. Alimentation de la table DW13_CATEGORIES 6
2.1. Objectif 6
2.2. Job d'alimentation et gestion de la contrainte 6
2.3. Vérifications dans SQL Developer 9
3. Alimentation de la table DW13_GEOGRAPHIES 10
3.1. Objectif 10
3.2. Job avec jointure dans Talend 10
3.3. Vérifications dans SQL Developer 12
4. Alimentation de la table DW13_MAGASINS 13
4.1. Objectif 13
4.2. Création d'une séquence dans SQL Developer 13
4.3. Mise en place du job Talend 14
4.4. Mappings et transformations dans le tMap 14
4.5. Vérifications dans SQL Developer 15
5. Alimentation de la table DW13_PRODUITS 16
5.1. Objectif 16
5.2. Construction du job Talend 16
5.3. Vérifications dans SQL Developer 17
6. Alimentation de la table DW13_VENTES 17
6.1 Objectifs 17
6.2. Construction du job Talend 17
6.3. Vérifications dans SQL Developer 19
7. Job global de chargement du Data Warehouse 19
7.1 Objectif 19
7.2 Construction du job global dans Talend 19
7.3 Exécution et erreurs rencontrées 20
7.4 Vérifications dans SQL Developer 21
7.5 Remarques sur l'avancement 21
Conclusion **22**

Introduction

Ce TP a pour objectif de mettre en œuvre un processus complet d'alimentation d'un entrepôt de données en utilisant Talend Open Studio. Dans un premier temps, les bases de production PROD_SRC1 et PROD_SRC2 servent de sources, tandis que la base DW13 constitue la cible. Le travail consiste à extraire les données, à les transformer lorsque cela est nécessaire (mappings, jointures, colonnes calculées) et à les charger dans les différentes tables de l'entrepôt.

Les tâches réalisées couvrent plusieurs situations typiques d'un flux ETL : alimentation simple, gestion de contraintes de clé étrangère, jointure entre deux tables sources, contrôle des types, ainsi que vérification des résultats dans SQL Developer. Enfin, un job maître permet d'enchaîner l'ensemble des alimentations selon les dépendances entre les tables.

Ce compte rendu présente donc, de manière progressive, les différents jobs construits, les mappings définis, les actions particulières nécessaires, ainsi que les vérifications effectuées après chaque chargement.

1. Alimentation de la table DW13_MARQUES

1.1. Objectif

La première étape de l'ETL consiste à alimenter la table de dimension DW13_MARQUES à partir de la table source correspondante de la base PROD_SRC1. Le but est de rapatrier la liste des marques, sans transformation particulière, en respectant la structure du schéma DW13.

1.2. Construction du job Talend

Pour cette alimentation, un job dédié a été créé dans Talend, nommé « Job_DW13_MARQUES_0.1 ». Il est constitué des composants suivants reliés en chaîne :

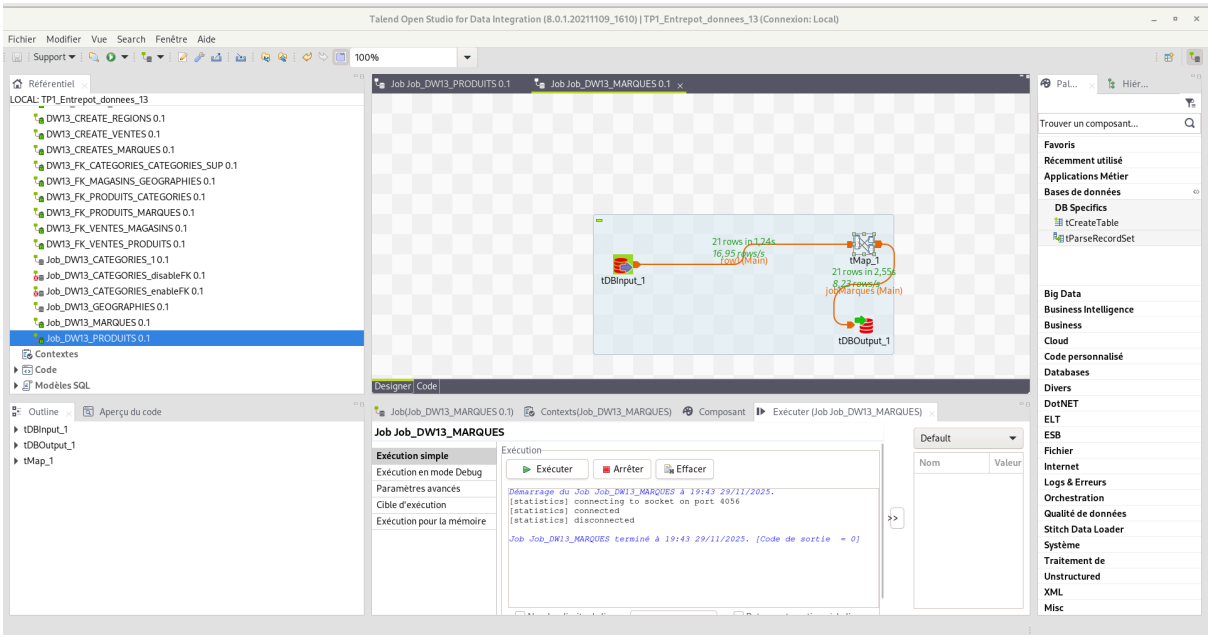
- un composant tDBInput connecté à la base PROD_SRC1, configuré pour lire la table source des marques (SRC1_MARQUES) via une requête simple sélectionnant les colonnes de code et de désignation ;

- un composant tMap, utilisé ici pour effectuer un mapping direct entre les colonnes de la source et celles de la table cible ;

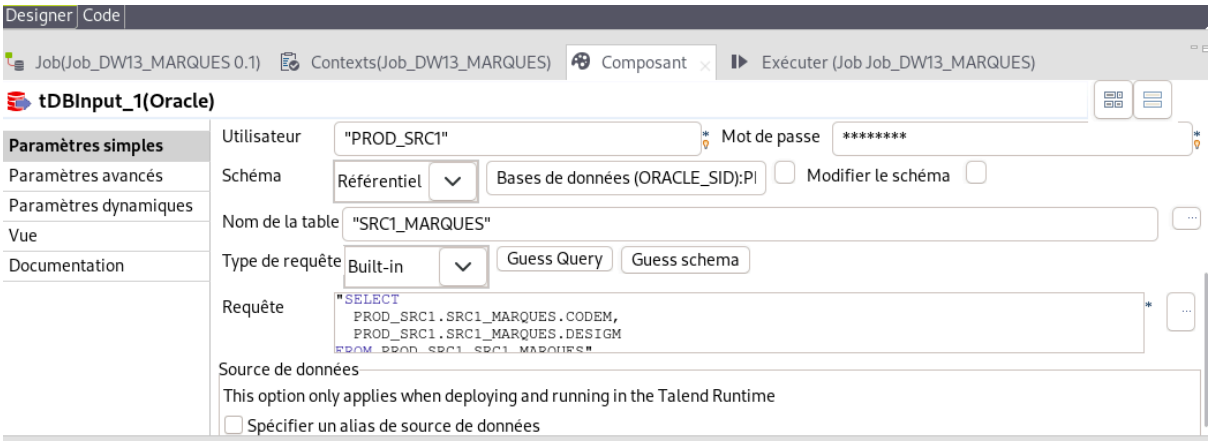
- un composant tDBOutput connecté au compte DW13, configuré en insertion (mode INSERT) sur la table DW13_MARQUES, avec un schéma conforme au dictionnaire de données de l'entrepôt.

Le tMap réalise une correspondance colonne à colonne entre la source et la cible. La clé de la marque en entrée est reliée à la clé CODE_MARQ de la table DW13_MARQUES, et le libellé de la marque est relié à la colonne DESIG_MARQ. Aucune expression spécifique ni transformation supplémentaire n'est nécessaire pour cette table.

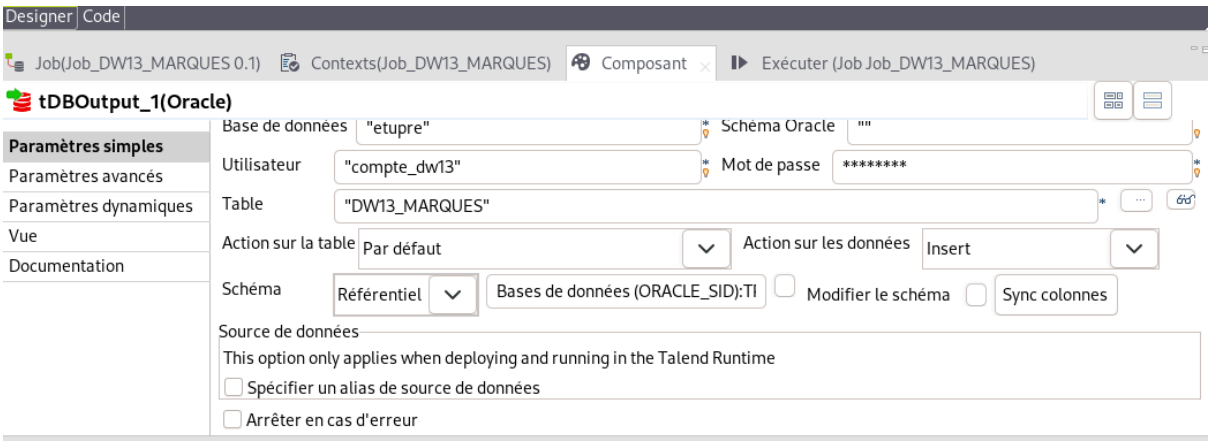
Lors de l'exécution du job, Talend affiche un nombre de lignes traitées cohérent avec la table source, confirmant le bon fonctionnement du flux tDBInput → tMap → tDBOutput.



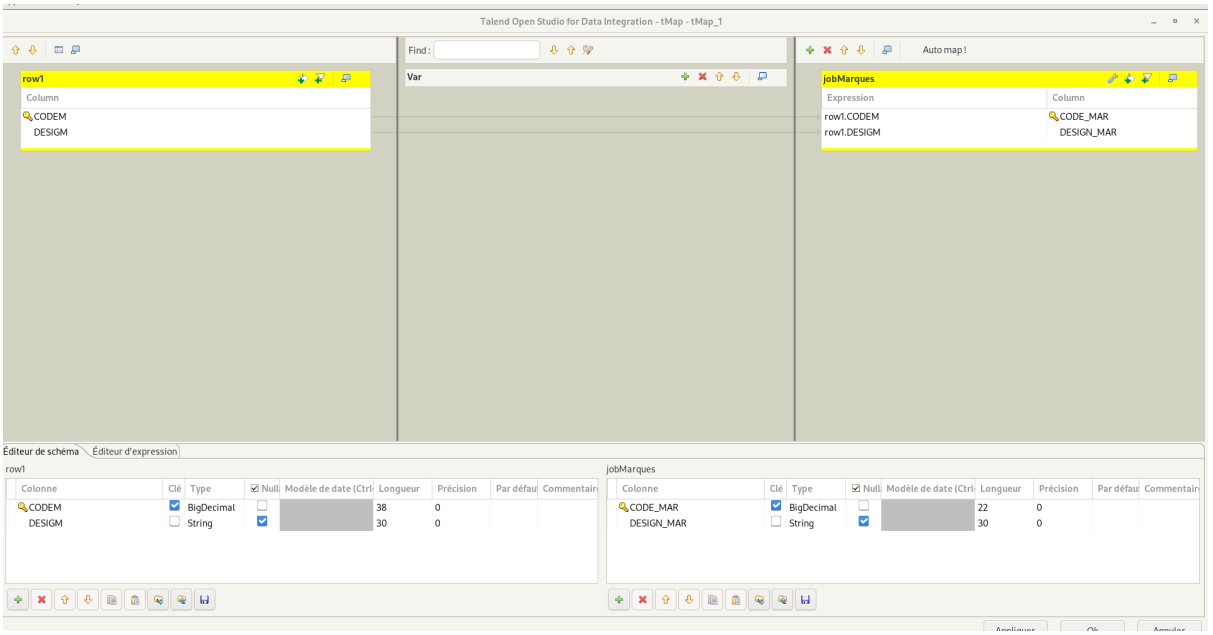
Vue d'ensemble du job Job_DW13_MARQUES avec l'exécution réussie



Détail tDBInput



Détail tDBOutput



Fenêtre du tMap montrant le mapping simple entre les colonnes source et cible

1.3. Vérifications dans SQL Developer

Après l'exécution du job, des requêtes de contrôle ont été réalisées dans SQL Developer sur le compte DW13. Elles permettent de vérifier :

- le nombre total de lignes présentes dans DW13_MARQUES (par exemple via un comptage du nombre d'enregistrements) ;
- la cohérence des valeurs insérées (échantillon de lignes pour vérifier que les codes et libellés des marques sont bien conformes à la source) ;
- le maintien de la structure de la table (types de colonnes et contraintes inchangés après l'alimentation).

Même si toutes ces requêtes n'apparaissent pas systématiquement en capture d'écran, elles ont été exécutées pour s'assurer de la bonne insertion des données et de la stabilité du schéma.

2. Alimentation de la table DW13_CATEGORIES

2.1. Objectif

La table DW13_CATEGORIES représente la dimension des catégories de produits. Son alimentation est un peu plus délicate, car elle contient une clé étrangère (CODE_CAT_SUP) qui pointe sur elle-même. Comme indiqué dans l'énoncé, il est nécessaire de désactiver temporairement cette contrainte de clé étrangère avant l'insertion, puis de la réactiver une fois le chargement terminé.

2.2. Job d'alimentation et gestion de la contrainte

Pour cette table, l'alimentation a été organisée autour d'un job « maître » dédié aux catégories, par exemple nommé « Job_DW13_CATEGORIES ». Ce job enchaîne trois phases :

1. Désactivation de la contrainte de clé étrangère sur la table DW13_CATEGORIES :
   Un composant tDBRow est connecté au compte DW13. Il exécute une instruction de type ALTER TABLE permettant de désactiver la contrainte de clé étrangère sur la colonne CODE_CAT_SUP. Cette étape garantit que les enregistrements pourront être insérés même si les catégories parente et enfant ne sont pas encore toutes présentes.

2. Alimentation de la table DW13_CATEGORIES :
   Un second sous-job réunit un tDBInput, un tMap et un tDBOutput :
   – le tDBInput lit la table source des catégories dans PROD_SRC1 (SRC1_CATEGORIES) à l'aide d'une requête sélectionnant les colonnes de code de catégorie, de libellé et de code de catégorie supérieure ;
   – le tMap effectue le mapping entre ces colonnes sources et les colonnes CODE_CAT, DESIG_CAT et CODE_CAT_SUP de la table de l'entrepôt ;
   – le tDBOutput insère les données dans DW13_CATEGORIES en mode INSERT.

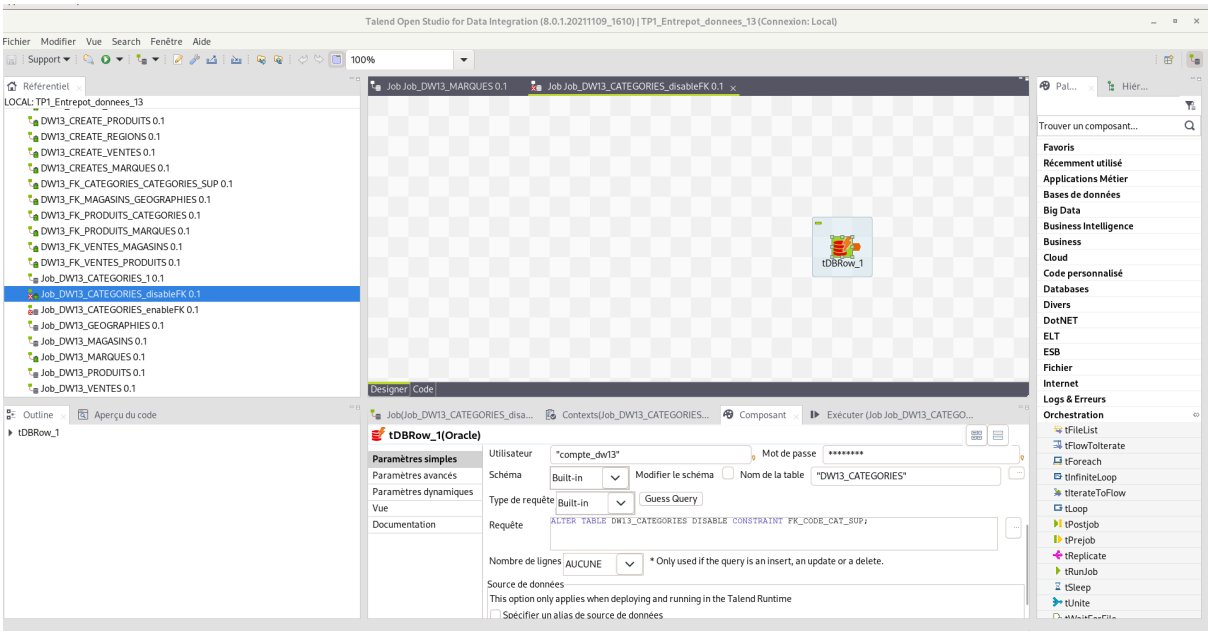
   Les types et longueurs des colonnes ont été soigneusement alignés entre le schéma d'entrée et le schéma de sortie pour éviter les erreurs de compatibilité de données.

3. Réactivation de la contrainte de clé étrangère :
   Un second tDBRow, toujours connecté à DW13, exécute une instruction ALTER TABLE inverse, qui réactive la contrainte de clé étrangère sur CODE_CAT_SUP. Cette étape est réalisée uniquement après que toutes les lignes ont été insérées

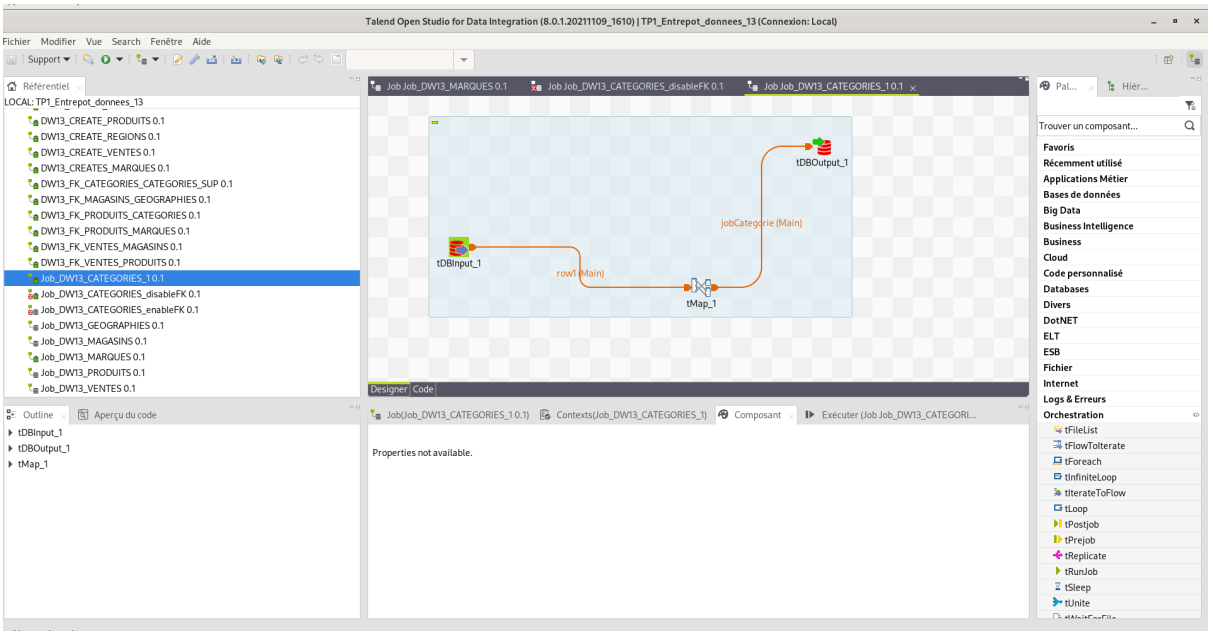
dans la table.

Au cours de la mise au point de cette alimentation, plusieurs erreurs successives sont apparues (erreurs Oracle liées à la contrainte encore active, incompatibilités de schémas entre Talend et Oracle, et erreurs lors de la compilation du job). Une difficulté supplémentaire venait du fait que Talend renvoyait parfois un code de sortie égal à 0. Ce code aurait normalement indiqué une exécution correcte, mais dans ce cas précis, il masquait une anomalie persistante dans le job, rendant le diagnostic plus complexe.

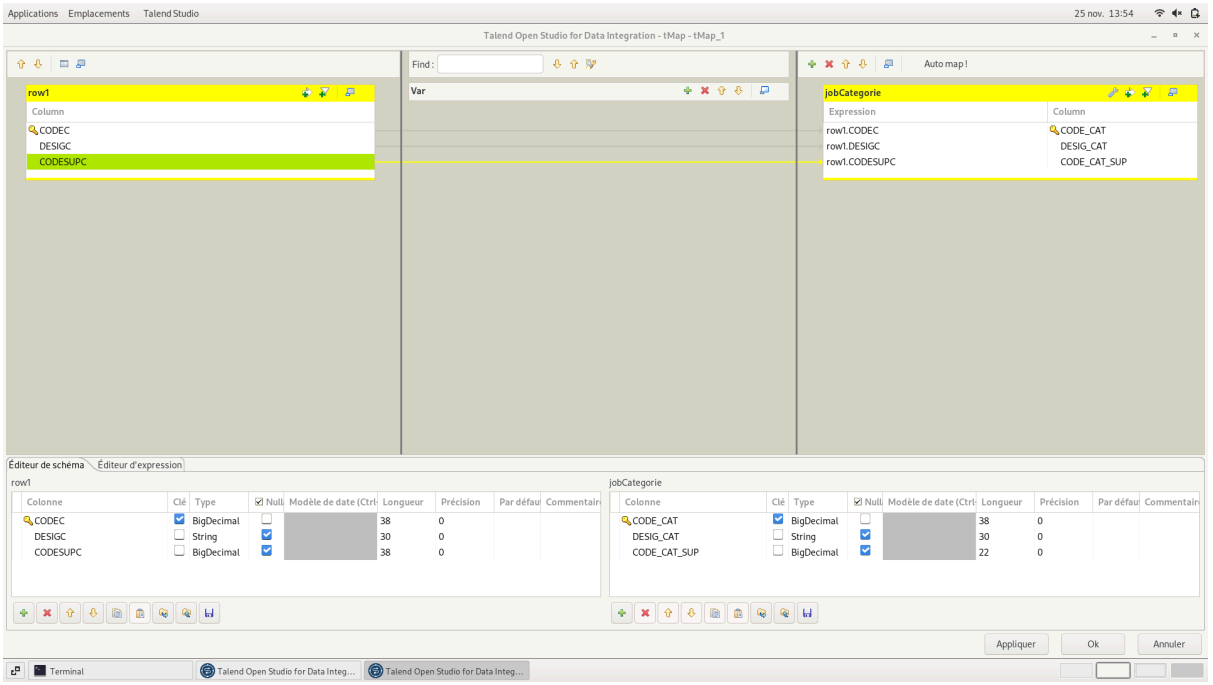
Pour pouvoir continuer à avancer malgré ces blocages, la désactivation et la réactivation de la contrainte de clé étrangère ont finalement été réalisées directement dans SQL Developer. Cette approche a permis de poursuivre le chargement de la table, mais elle aura des conséquences visibles dans la troisième tâche du TP, où certaines dépendances entre tables (et notamment les liens fondés sur les clés et sous-clés) ne se comporteront pas comme prévu à cause de ces manipulations manuelles.



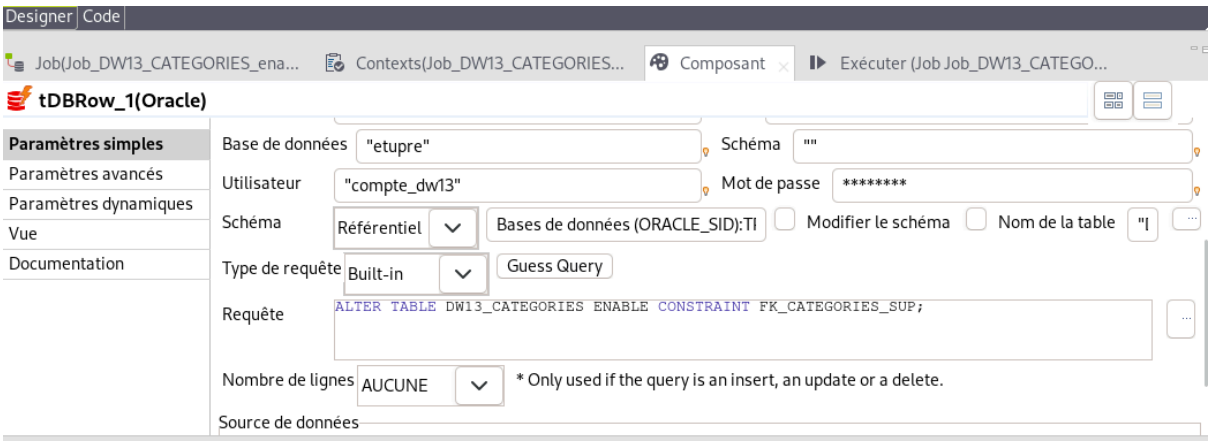
Vue d'ensemble du job servant à désactiver la clé étrangère de la table



Vue d'ensemble du job Job_DW13_CATEGORIES



Vue d'ensemble du Tmap



Détails tDBRow du job jobD13_CATEGORIES_enableFK

2.3. Vérifications dans SQL Developer

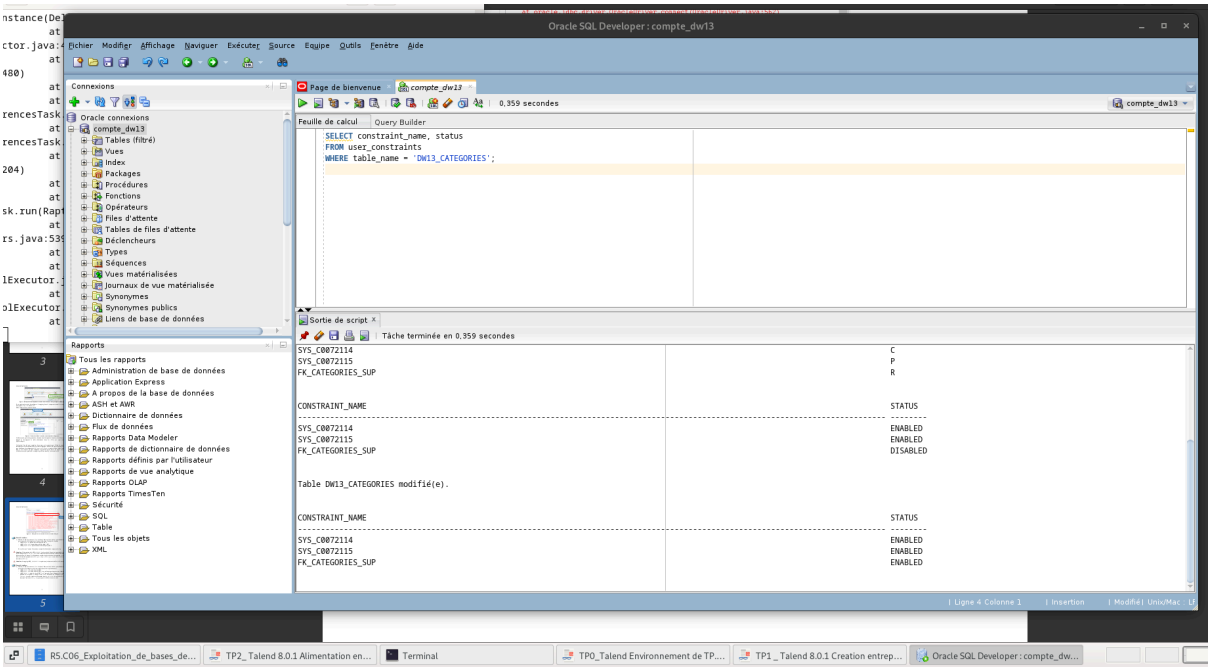
Les vérifications SQL pour cette table ont été particulièrement importantes afin de s'assurer que la contrainte était bien manipulée et que la table était correctement alimentée.

Dans SQL Developer, plusieurs requêtes ont été exécutées :

- un comptage du nombre de lignes présentes dans DW13_CATEGORIES après l'exécution du job, permettant de vérifier que le nombre d'enregistrements correspond au nombre attendu dans l'énoncé ;

- une requête sur le dictionnaire de données (user_constraints/user_cons_columns) pour vérifier que la contrainte de clé étrangère associée à CODE_CAT_SUP était bien repassée à l'état ENABLED après la réactivation ;

- quelques requêtes de sélection sur un échantillon de catégories, pour vérifier la cohérence des codes de catégories et des catégories supérieures.



Vérification de l'état des FK pour l'exécution du job

3. Alimentation de la table DW13_GEOGRAPHIES

3.1. Objectif

La table DW13_GEOGRAPHIES définit les départements géographiques de l'entrepôt. Son alimentation nécessite une jointure entre deux tables de la base PROD_SRC2 : une table des régions et une table des départements. Cette jointure permet de récupérer le code de département et sa désignation, en respectant la structure attendue dans DW13_GEOGRAPHIES.

3.2. Job avec jointure dans Talend

Pour cette tâche, un job spécifique a été créé, par exemple « Job_DW13_GEOGRAPHIES_0.1 ». Il comporte :

- un premier composant tDBInput connecté à PROD_SRC2 et configuré pour interroger la table des régions (avec le code région et le nom de la région) ;
- un second tDBInput sur PROD_SRC2 lisant la table des départements (avec le code département, le nom du département, le code de la région, etc.) ;
- un composant tMap recevant en entrée ces deux flux : l'un est défini comme flux principal (Main), l'autre comme flux de jointure (Lookup) ;

- un tDBOutput alimentant la table DW13_GEOGRAPHIES du compte DW13.

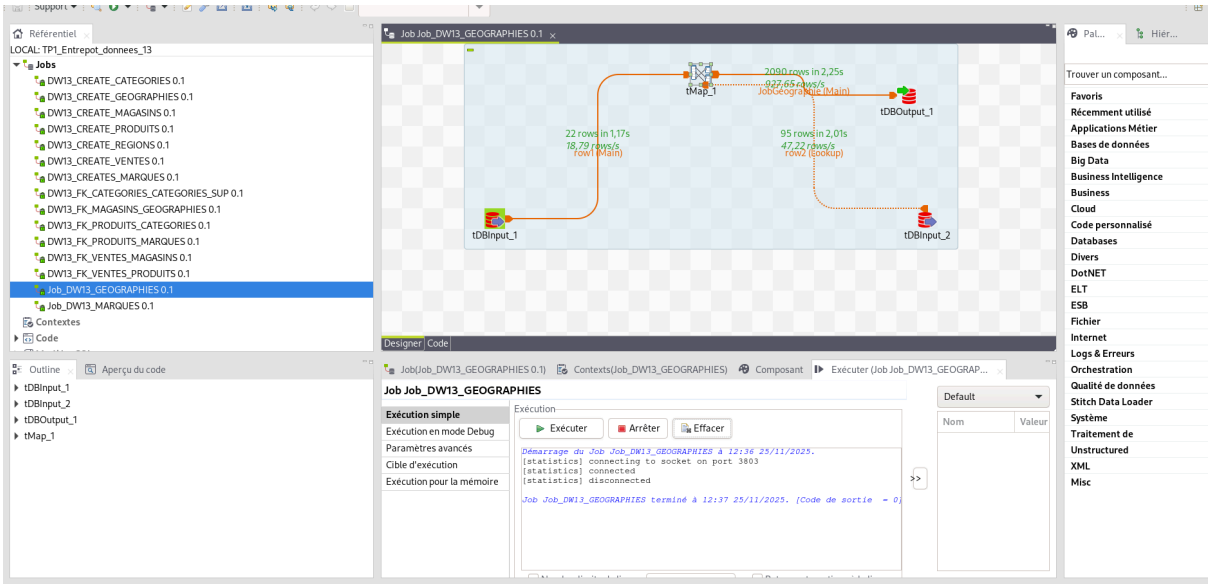
Dans le tMap, une jointure est définie entre la clé de région de la table des régions et la clé de région de la table des départements. Le paramétrage par défaut (jointure externe gauche) est conservé, ce qui correspond à la recommandation du sujet. La jointure est matérialisée dans la partie centrale du tMap par une liaison entre les deux colonnes de code région.

Le schéma de sortie du tMap correspond au schéma de la table DW13_GEOGRAPHIES. Les deux colonnes principales du tMap sont :

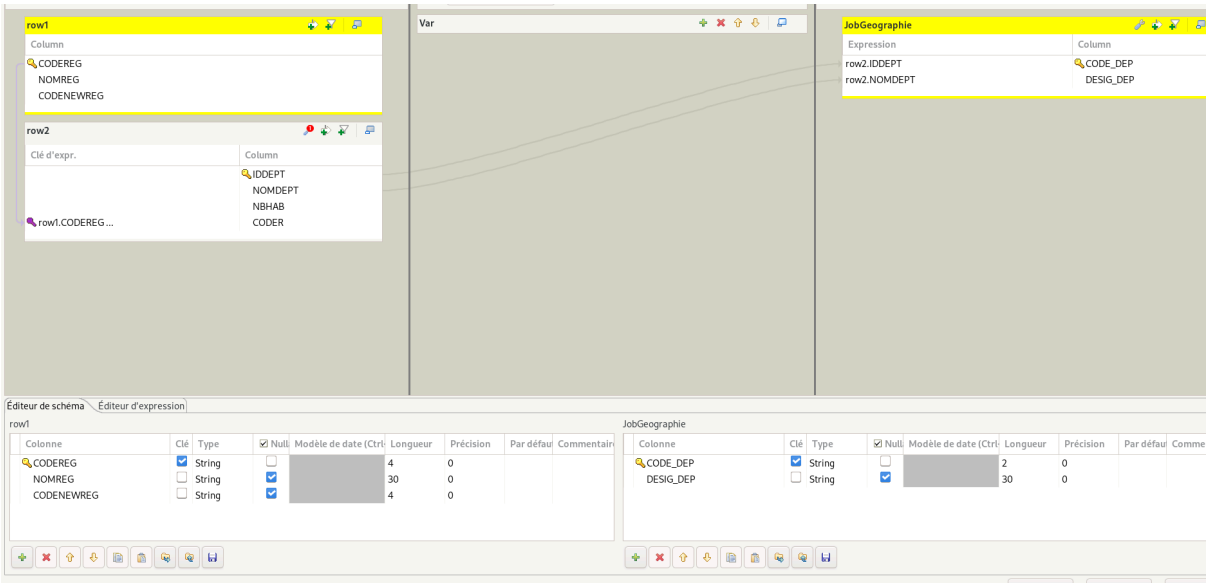
- un champ de sortie correspondant au code du département, alimenté à partir de la colonne identifiant le département dans la table des départements ;

- un champ de sortie correspondant à la désignation du département, alimenté à partir du libellé de la même table.

Le tDBOutput est ensuite configuré pour insérer ces colonnes dans DW13_GEOGRAPHIES, avec les bons types et longueurs.



Vue d'ensemble du job Job_DW13_GEOGRAPHIES avec les deux tDBInput, le tMap et le tDBOutput



Fenêtre du tMap montrant clairement les deux flux d'entrée

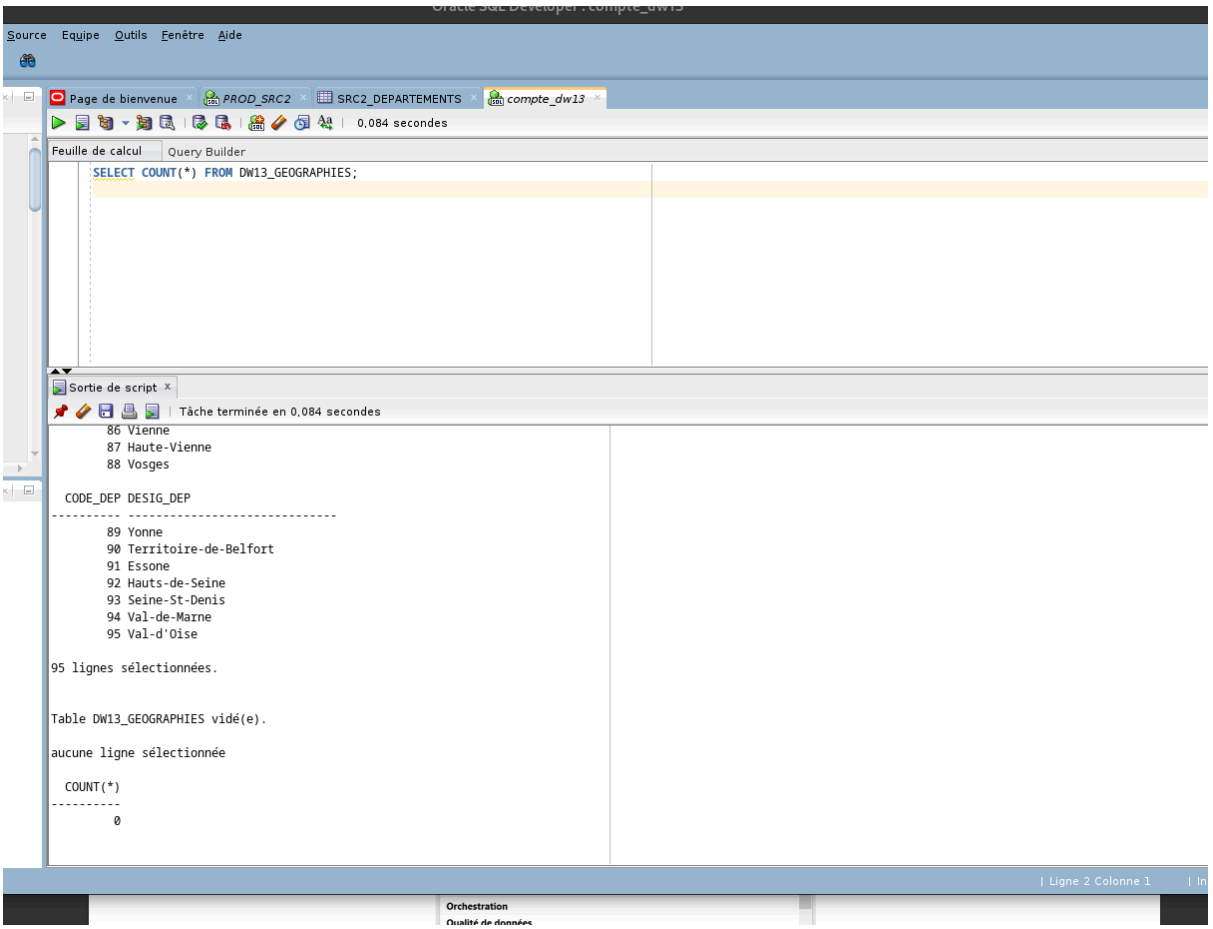
3.3. Vérifications dans SQL Developer

Côté base de données, différentes requêtes ont permis de contrôler le bon remplissage de DW13_GEOGRAPHIES :

- une requête de comptage du nombre d'enregistrements de la table, afin de vérifier que le nombre de départements correspond au nombre attendu après chargement ;

- des requêtes de sélection sur quelques départements, pour s'assurer que les codes et libellés sont cohérents (par exemple contrôle des départements de 89 à 95 visibles sur la capture) ;

- une vérification de la structure de la table (types de données des colonnes CODE_DEP et DESIG_DEP) pour s'assurer que les modifications dans Talend n'ont pas impacté le schéma Oracle.



sélection des lignes de DW13_GEOGRAPHIES, vidage éventuel de la table pour rejouer le job, puis contrôle du nombre de lignes après réexécution du chargement.

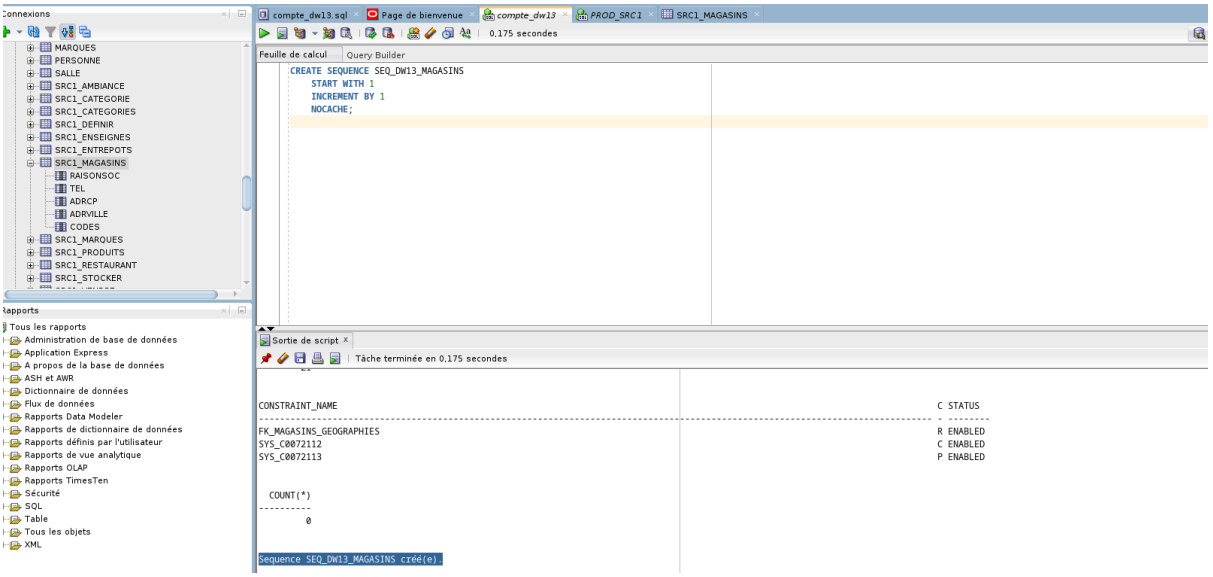
4. Alimentation de la table DW13_MAGASINS

4.1. Objectif

Cette étape consiste à alimenter la table DW13_MAGASINS à partir de la table source SRC1_MAGASINS. La particularité de cette table est qu'elle ne possède pas de clé primaire dans la source. Il est donc nécessaire de générer un identifiant unique pour chaque magasin dans l'entrepôt.

4.2. Création d'une séquence dans SQL Developer

Avant la construction du job Talend, une séquence a été créée directement dans SQL Developer afin de générer les codes magasins (CODE_MAG).
Cette séquence SEQ_DW13_MAGASINS démarre à 1 avec un incrément de 1.
La création a aussi été l'occasion de vérifier les contraintes existantes sur DW13_MAGASINS (toutes à l'état ENABLED).



Création de la séquence

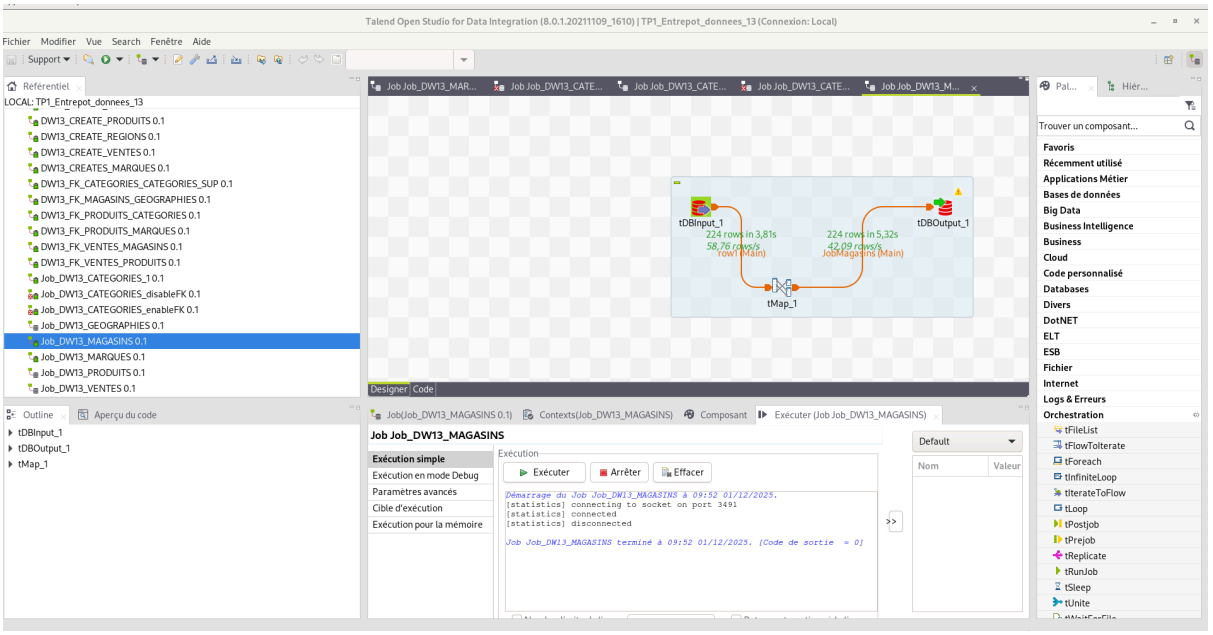
4.3. Mise en place du job Talend

Le job « Job_DW13_MAGASINS » repose sur trois composants :
– un tDBInput lisant SRC1_MAGASINS ;
– un tMap réalisant les transformations ;
– un tDBOutput insérant les données dans DW13_MAGASINS.

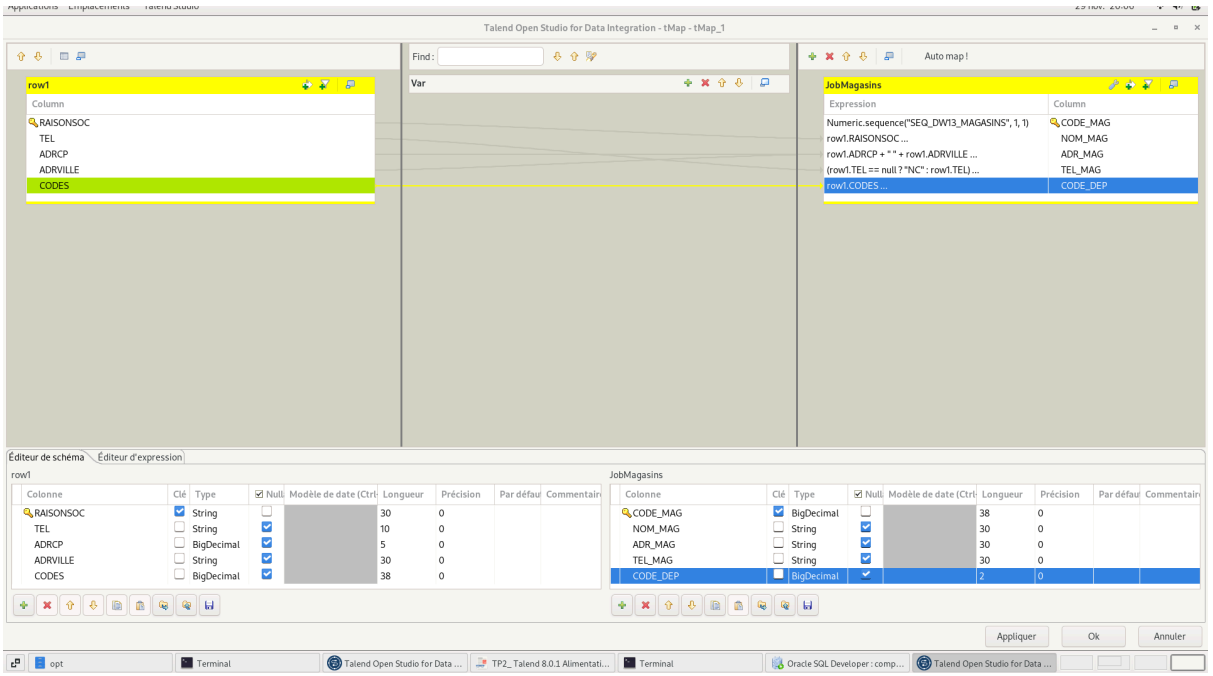
4.4. Mappings et transformations dans le tMap

Le tMap comporte plusieurs règles importantes :
– génération du CODE_MAG à l'aide de la fonction Numeric.sequence() associée à SEQ_DW13_MAGASINS ;
– récupération du nom, de l'adresse et de la ville directement depuis les colonnes de la source ;
– extraction du code département (CODE_DEP) depuis la colonne CODES de la table SRC1_MAGASINS.

Le schéma cible a été rigoureusement aligné avec celui de DW13_MAGASINS, notamment pour les longueurs de chaînes (30 caractères par exemple).



Vue d'ensemble du job magasins avec une exécution réussie



Tmap après mapping

4.5. Vérifications dans SQL Developer

Après exécution :

- 224 lignes ont bien été insérées
- aucun doublon sur CODE_MAG

- correspondance correcte entre les données sources et les valeurs transformées
- contrôle de quelques lignes (nom, adresse, code département, téléphone)

5. Alimentation de la table DW13_PRODUITS

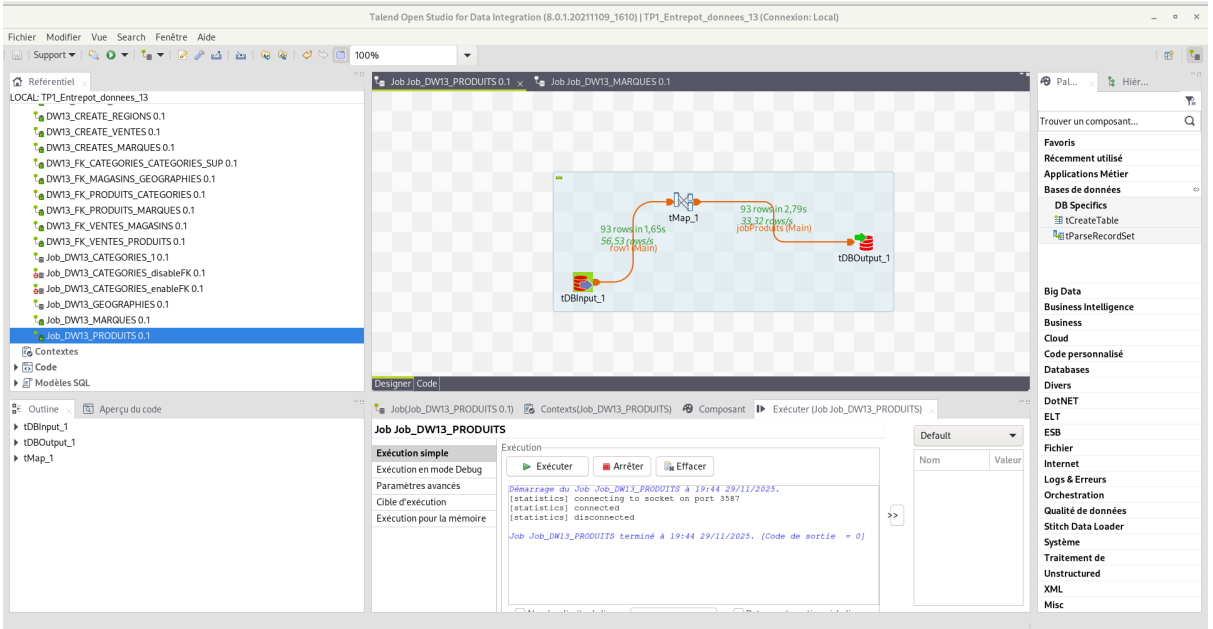
5.1. Objectif

L'objectif est d'alimenter la table DW13_PRODUITS à partir de la table SRC1_PRODUITS. Ce chargement est direct car aucune transformation complexe ni jointure n'est nécessaire. Il s'agit simplement de transférer les informations de code produit, désignation, code catégorie et code marque vers l'entrepôt en respectant le schéma du dictionnaire DW13.

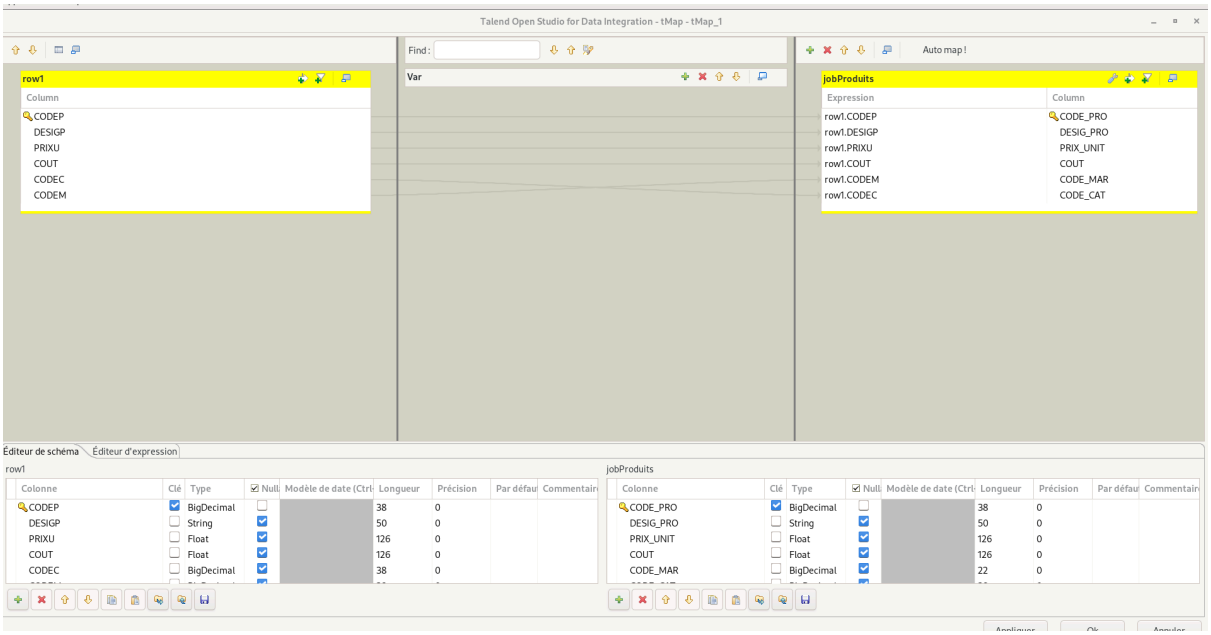
5.2. Construction du job Talend

Le job Job_DW13_PRODUITS utilise trois composants :
– un tDBInput qui lit la table SRC1_PRODUITS
– un tMap pour réaliser l'association de colonnes
– un tDBOutput pour l'insertion dans DW13_PRODUITS

Dans le tMap, chaque colonne de la source est reliée directement à la colonne correspondante dans la table cible.
Les types de données ont été contrôlés afin que les valeurs respectent les types Oracle (BigDecimal, String, etc.).
Aucune modification ou calcul intermédiaire n'a été ajouté dans cette étape.



Vue d'ensemble du job correspondant à la table PRODUITS



Détail du Tmap du job Job_DW13_PRODUITS après mapping

5.3. Vérifications dans SQL Developer

Après exécution, un contrôle a été effectué directement dans SQL Developer. Le nombre de lignes insérées dans DW13_PRODUITS correspond exactement au nombre de lignes de SRC1_PRODUITS. Un échantillon a été vérifié pour s'assurer que les valeurs CODE_PRO, CODE_CAT et CODE_MARQUES sont cohérentes avec la source.

6. Alimentation de la table DW13_VENTES

6.1 Objectifs

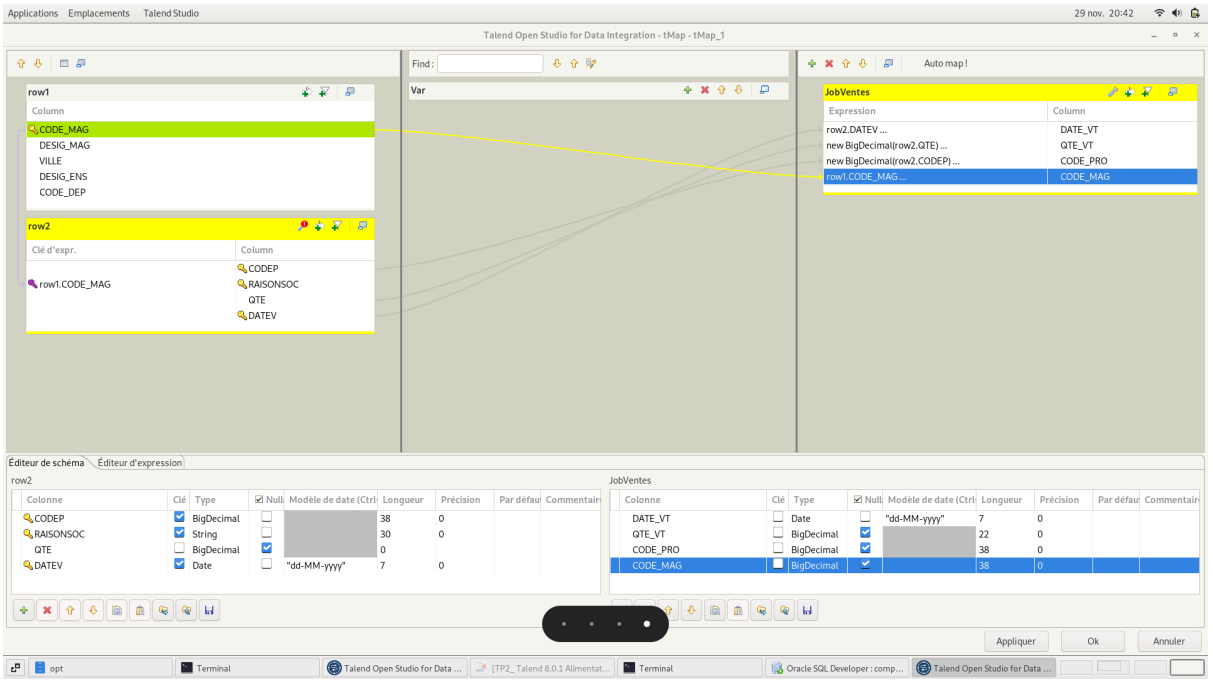
Cette étape consiste à charger les ventes du système de production dans DW13_VENTES.
Elle nécessite une jointure entre les magasins déjà chargés dans DW13_MAGASINS et les ventes présentes dans SRC2_VENTES.
L'objectif est de retrouver le magasin correspondant à chaque vente afin de renseigner correctement CODE_MAG dans la table cible.

6.2. Construction du job Talend

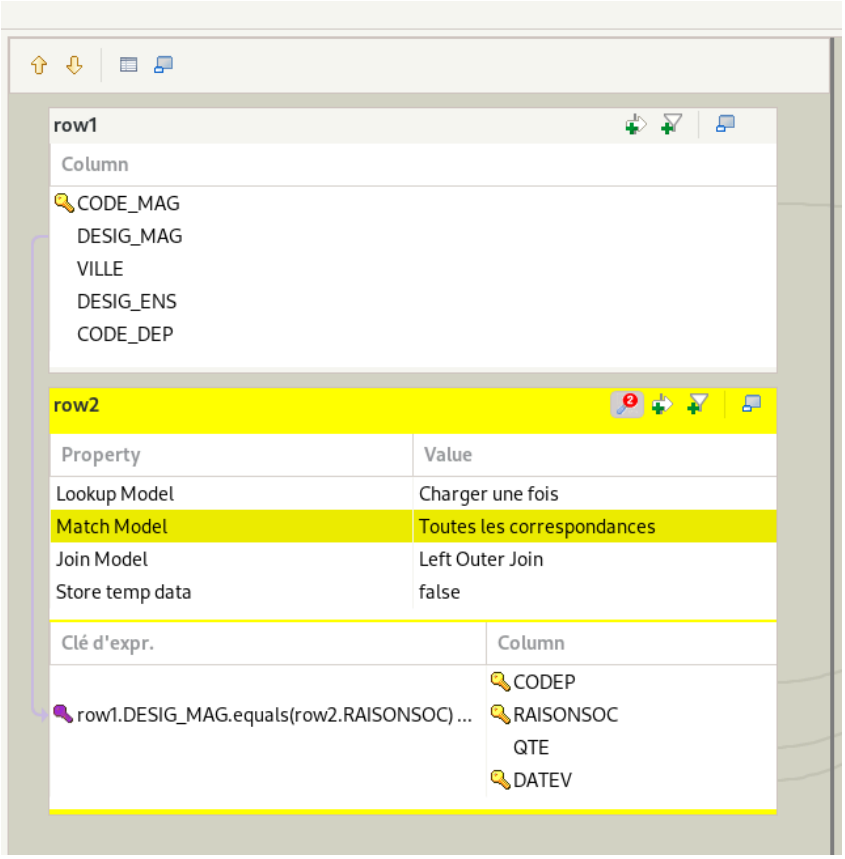
Le job a été construit avec deux tDBInput :
– le premier lit DW13_MAGASINS (flux principal)
– le second lit SRC2_VENTES (flux lookup)

Dans le tMap, une jointure a été configurée sur la correspondance entre DESIG_MAG (côté DW13) et RAISONSOC (côté SRC2).
Le modèle de jointure utilisé est Left Outer Join afin de conserver l'ensemble des magasins même si certaines ventes ne trouvent pas de correspondance.

Les colonnes DATE_VT, QTE_VT, CODE_PRO et CODE_MAG ont ensuite été alimentées à partir des valeurs issues du flux lookup et du flux principal.



Vue d'ensemble du Tmap correspondant au job Job_DW13_VENTES



Le schéma ci-dessus montre la configuration de la jointure dans le tMap. C'est précisément à ce niveau que plusieurs erreurs sont apparues pendant l'exécution du job. L'expression row1.DESIG_MAG.equals(row2.RAISONSOC) imposait une correspondance exacte entre les deux champs, ce qui déclenchait systématiquement une erreur dans le Match Model. Tant que la valeur exacte n'était pas trouvée, Talend considérait qu'il n'y avait aucune correspondance valide, ce qui empêchait le chargement et provoquait l'échec du job. C'est pour cette raison qu'il a été nécessaire de revoir la configuration de la jointure et de vérifier les valeurs présentes dans les deux tables avant de relancer le job.

6.3. Vérifications dans SQL Developer

Les vérifications ont porté sur le nombre total de ventes insérées ainsi que sur la cohérence des valeurs.
Un contrôle a été effectué sur les dates, les quantités et les correspondances entre CODE_PRO et CODE_MAG.
Des requêtes de comptage et quelques sélections ont été réalisées pour confirmer que la jointure avait été correctement appliquée.

7. Job global de chargement du Data Warehouse

7.1 Objectif

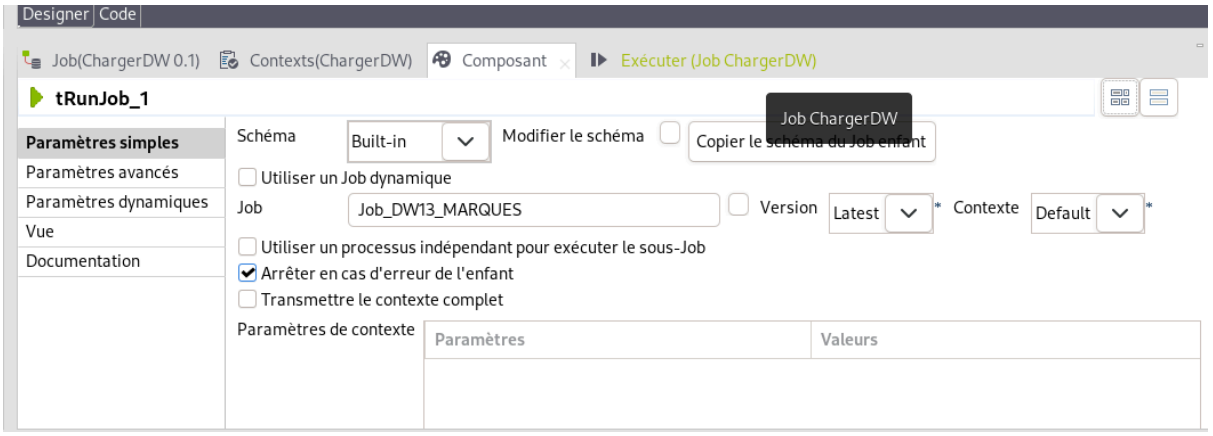
L'objectif de cette partie est de regrouper plusieurs jobs d'alimentation dans un seul job maître afin de lancer automatiquement les chargements successifs. Le job global permet d'enchaîner l'exécution des jobs déjà réalisés dans les parties précédentes et de gérer leur dépendance logique, notamment grâce aux déclencheurs OnSubjobOk.

7.2 Construction du job global dans Talend

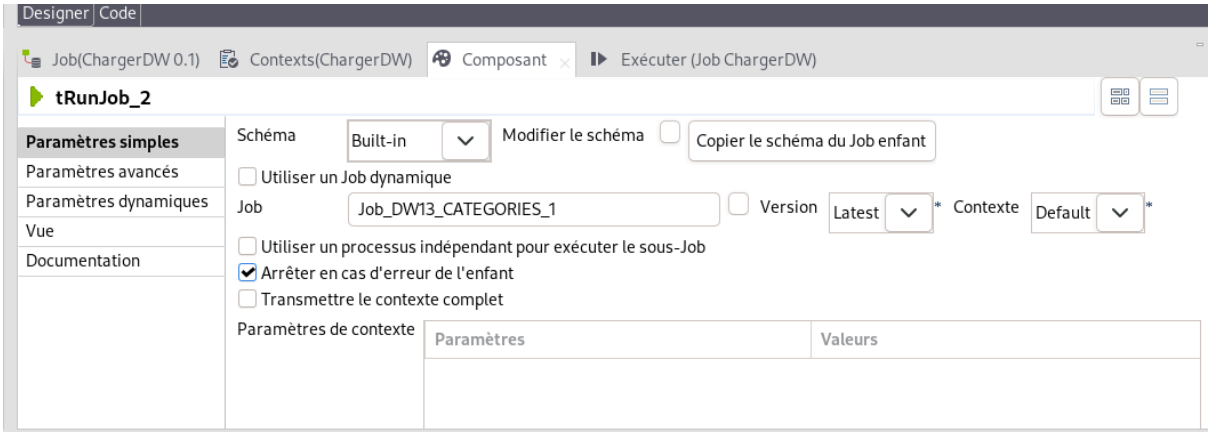
Le job global a été construit en utilisant plusieurs composants tRunJob. Chaque tRunJob appelle un job enfant déjà configuré dans les étapes précédentes. Les liens OnSubjobOk permettent de définir l'ordre d'exécution.

Insérer la capture du job global :
(Insérer la capture : vue complète du job avec tRunJob_1, tRunJob_3 et tRunJob_2)

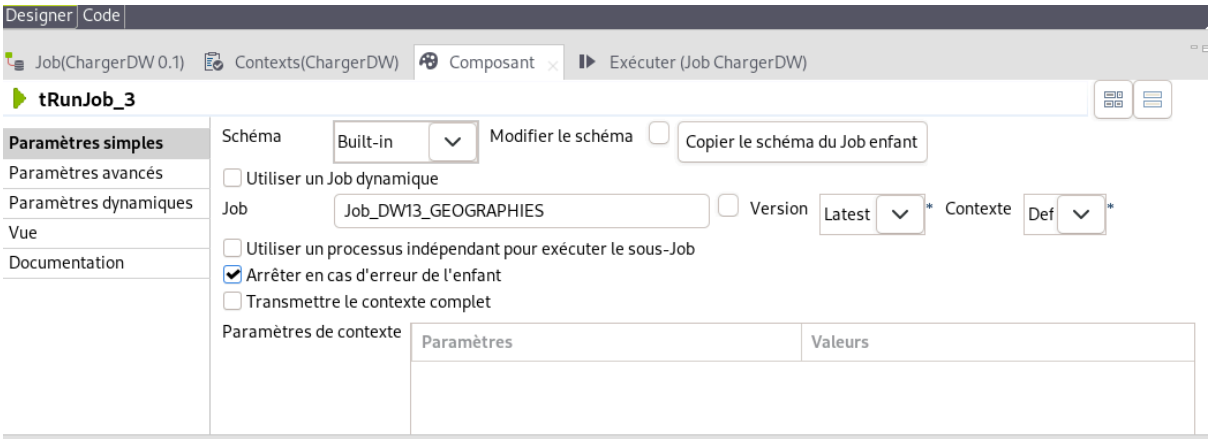
Chaque tRunJob est paramétré pour lancer un job enfant précis. Par exemple :
– Le premier tRunJob exécute le job d'alimentation de DW13_MARQUES.
– Le second tRunJob exécute DW13_CATEGORIES_1.
– Le troisième tRunJob exécute un autre job d'alimentation prévu dans la chaîne.



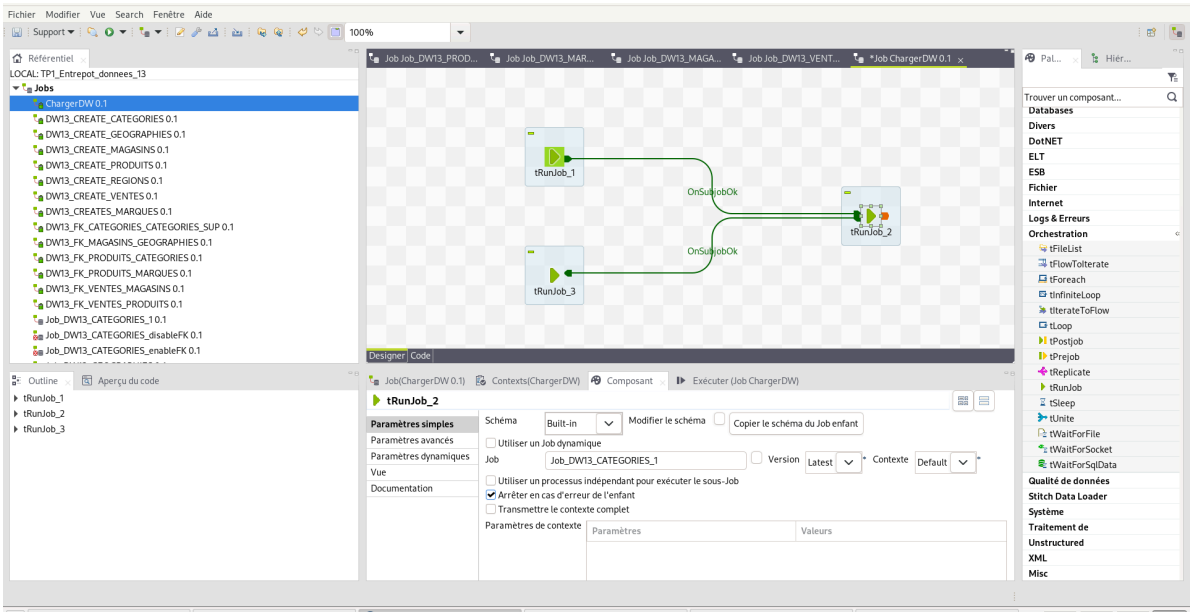
Détail du premier tRunJob (Job_DW13_MARQUES)



Détail du deuxième tRunJob (Job_DW13_CATEGORIES)



Détail du troisième TRunJob(Job_DW13_GEOGRAPHIES



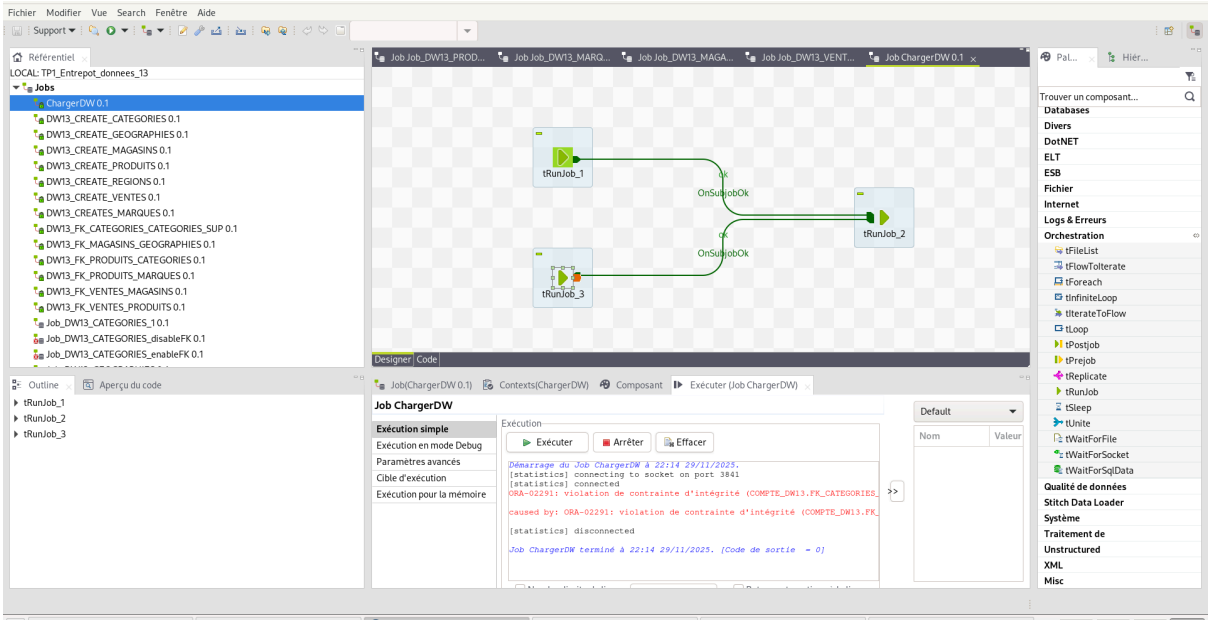
Vue d'ensemble de chargerDW

7.3 Exécution et erreurs rencontrées

Lors du lancement du job global, des erreurs liées aux contraintes de clés étrangères sont apparues. Ces erreurs se produisent car certaines dépendances entre tables ne sont pas respectées dans l'ordre d'exécution ou que certaines données chargées en amont ne correspondent pas encore aux contraintes d'intégrité définies dans le schéma Oracle.

Insérer la capture de l'erreur ORA-02291 :
(Insérer la capture : erreur ORA-02291 violation de contrainte FK_CATEGORIES)

L'origine du problème provient du job chargé d'alimenter la table des catégories, appelé depuis tRunJob_3. La contrainte en question nécessite que la table référencée soit correctement alimentée en amont. Le job s'arrête donc automatiquement, conformément au paramétrage "Arrêter en cas d'erreur de l'enfant".



Erreur d'exécution de ChargerDW

7.4 Vérifications dans SQL Developer

Une vérification a été faite dans SQL Developer pour s'assurer de l'état des contraintes et pour confirmer que l'erreur provient bien d'un problème d'intégrité référentielle. L'analyse a confirmé que la contrainte FK_CATEGORIES_SUP continuait à bloquer certaines insertions.

7.5 Remarques sur l'avancement

Pour des raisons de temps, l'ensemble des jobs prévus dans la chaîne n'a pas pu être finalisé. Le premier job a pu être exécuté avec succès, mais les suivants ont généré plusieurs erreurs de contraintes. Une partie importante du travail a consisté à analyser ces erreurs et à vérifier les dépendances entre tables, ce qui n'a pas permis de terminer toute la chaîne d'exécution dans le délai imparti.

Conclusion

La majorité du travail d'alimentation de l'entrepôt de données a pu être réalisée avec succès. Les principaux jobs ont été construits, exécutés et vérifiés, ce qui a permis de valider l'essentiel du processus ETL attendu. La gestion des schémas, des transformations dans Talend et des contrôles dans SQL Developer a constitué la base du travail et a permis de bien comprendre le fonctionnement de chaque étape du chargement.

Certaines difficultés sont néanmoins apparues, en particulier au niveau des contraintes d'intégrité et de l'ordre de traitement des tables. Afin d'avancer plus rapidement, certaines opérations ont dû être effectuées directement dans SQL Developer, notamment pour la table des catégories, ce qui a permis de continuer à alimenter l'entrepôt malgré les blocages

rencontrés dans Talend. Ces ajustements ont été nécessaires pour respecter le temps disponible, mais le travail sera repris et corrigé entièrement pour assurer la continuité du projet et garantir une base propre pour la suite du TP, en particulier pour le TP3.

Dans l’ensemble, les objectifs principaux ont été atteints et le travail effectué fournit une base solide pour terminer les derniers chargements et stabiliser les relations entre tables dans la prochaine étape du projet.