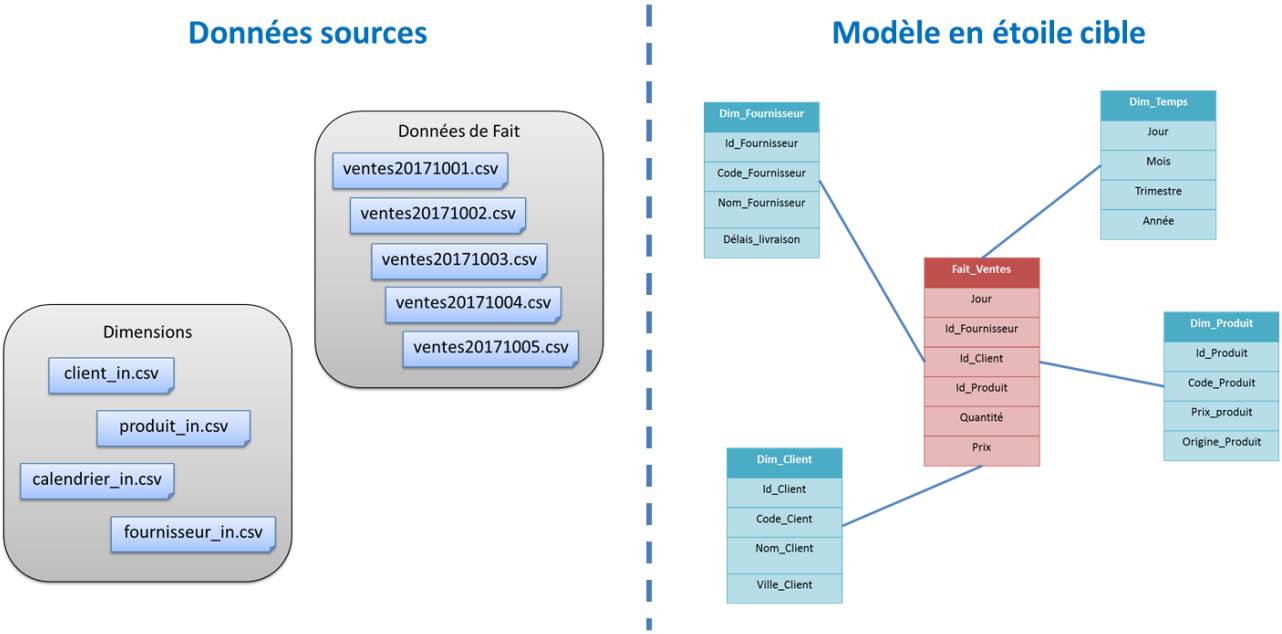
Cas étude Talend

Alimentation d'une étoile de gestion des ventes

1. Objectif

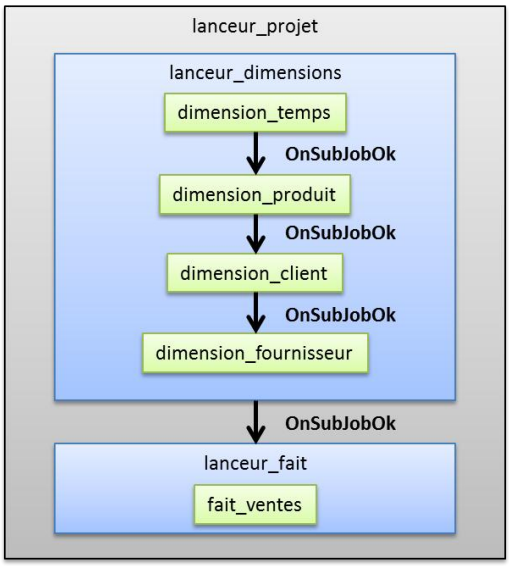
L'objectif de ce cas étude est de développer une chaine de traitement ETL sous Talend qui va permettre d'alimenter une étoile de gestion des ventes.

Le modèle cible à alimenter est le suivant :



Liste des fichiers sources et modèle en étoile cible

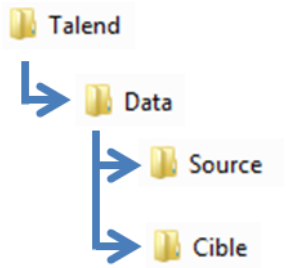
Ce projet va vous faire développer quatre jobs d'alimentations pour les tables de dimensions, un job pour la table de fait et des jobs lanceurs. La figure ci-dessous montre les différents jobs que vous allez développer avec leur ordonnancement.



Liste des jobs avec cinématique

2. Préparation de l'environnement de travail

Du fait des contraintes techniques, nous allons manipuler dans ce projet uniquement des fichiers csv que ce soit en source ou en cible. Afin de différencier les fichiers et simuler le fait de travailler sur deux systèmes différents, il est nécessaire de créer le répertoire Data qui contient les répertoires Source et Cible dans votre dossier Talend.



Architecture des dossiers

Créer un nouveau projet dans Talend appelé projet_NOM.

3. Développement du job d'alimentation de la dimension Temps

Créer un job appelé dimension_temps.

Placer le fichier calendrier_in.csv dans le dossier Data/Source et créer les métadonnées du fichier calendrier_in.csv.

Lire le contenu du fichier à l'aide d'un tFileInputDelimited et envoyer son contenu un tMap. La structure du fichier est la suivante : Jour, Mois, Trimestre, Année. Vu qu'il n'y a aucune transformation ni renommage à réaliser faire glisser l'ensemble du flux d'entrée vers la sortie (dans les bonnes partiques Talend, un tMap est systématiquement placé entre une source et une cible au cas où des règles de renommage ou transformation soient à mettre en place ultérieurement).

Ecrire le flux de sortie du tMap dans le fichier dim_temps.csv à l'aide d'un tFileOutputDelimited. Paramétrer le composant pour faire en sorte que les en-têtes soient présentes dans le fichier de sortie et que celui-ci soit vidé à chaque exécution. Le fichier dim_temps.csv doit être écrit dans le répertoire Data/Cible.

4. Développement du job d'alimentation de la dimension Client

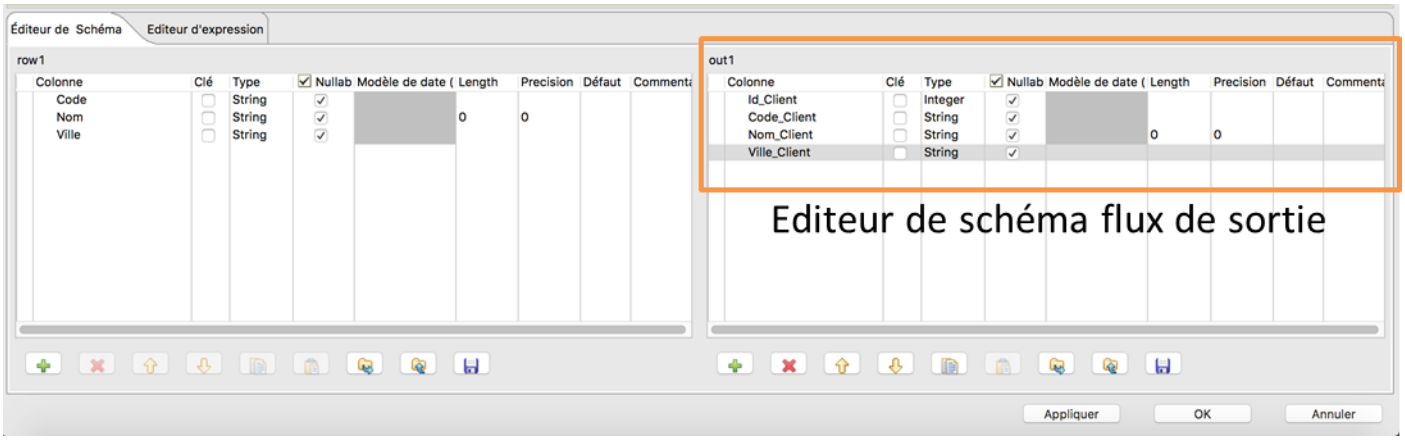
La dimension client est alimentée à partir du fichier client_in.csv.

Créer un nouveau job appelé dimension_client.

Placer le fichier client_in.csv dans le dossier Data/Source et créer les métadonnées du fichier client_in.csv.

Lire le contenu du fichier à l'aide d'un tFileInputDelimited et envoyer son contenu dans un tMap qui va permettre de réaliser les transformations suivantes :

- Créer un flux de sortie et créer une colonne Id_Client de type Integer qui contiendra une séquence qui commencera à 1 avec un pas de 1.
- Faire glisser les 3 champs en entrée de fournisseur afin de les placer sous la colonne Id_Client. Renommer les colonnes Code, Nom, Ville afin qu'elles s'appellent Code_Client, Nom_Client et Ville_Client. Pour renommer une colonne, il faut utiliser l'éditeur de schéma du flux de sortie du tMap.



Editeur de schéma du tMap

Ecrire le flux de sortie du tMap dans le fichier dim_client.csv. Paramétrer le composant pour faire en sorte que les en-têtes soient présentes dans le fichier de sortie et que celui-ci soit vidé à chaque exécution (on ne veut pas écrire à la suite à chaque nouvelle exécution). Le fichier dim_client.csv doit être écrit dans le répertoire Data/Cible.

Exécuter le job et vérifier le bon contenu du fichier dim_client.csv.

5. Développement du job d'alimentation de la dimension Fournisseur

La dimension fournisseur est alimentée à partir du fichier fournisseur_in.csv.

Créer un nouveau job appelé dimension_fournisseur.

Placer le fichier fournisseur_in.csv dans le dossier Data/Source et créer les métadonnées du fichier fournisseur_in.csv (la colonne délais doit être en integer).

Lire le contenu du fichier à l'aide d'un tFileInputDelimited et envoyer son contenu dans un tMap qui va permettre de réaliser les transformations suivantes :

- Créer un flux de sortie et créer une colonne Id_Fournisseur de type Integer qui contiendra une séquence qui commencera à 1 avec un pas de 1.
- Faire glisser les 3 champs en entrée de fournisseur afin de les placer sous la colonne Id_Fournisseur. Renommer les colonnes Code, Nom, Delais afin qu'elles s'appellent Code_Fournisseur, Nom_Fournisseur et Delais_Fournisseur.

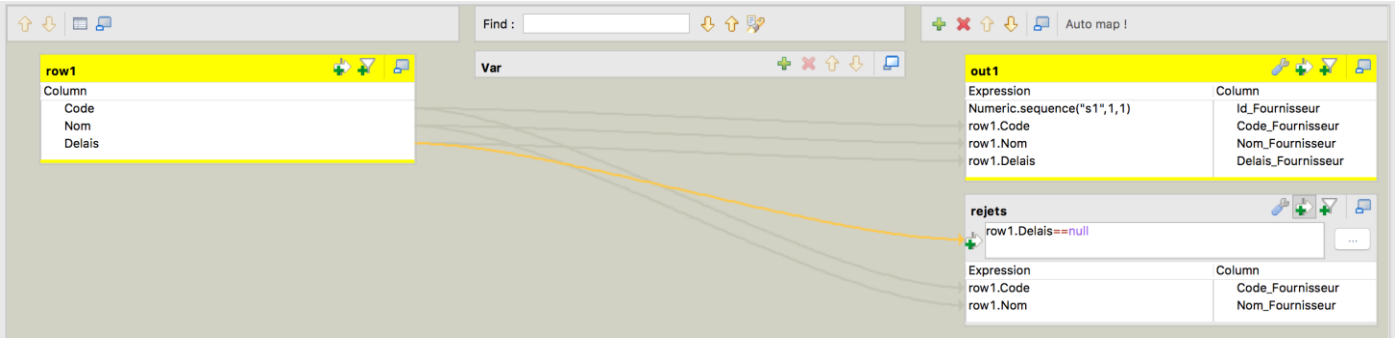
Ecrire le flux de sortie du tMap dans le fichier dim_fournisseur.csv. Paramétrer le composant pour faire en sorte que les en-têtes soient présentes dans le fichier de sortie et que celui-ci soit vidé à chaque exécution (on ne veut pas écrire à la suite à chaque nouvelle exécution). Le fichier dim_fournisseur.csv doit être écrit dans le répertoire Data/Cible.

Exécuter le job et vérifier le bon contenu du fichier dim_fournisseur.csv.

En observant le fichier de sortie, on se rend compte que le délai n'est pas toujours renseigné pour certains fournisseurs. Nous allons donc mettre en place un contrôle qui va permettre d'identifier les fournisseurs qui n'ont pas de délai de renseigné et créer un fichier de rejet qui contiendra le code et le nom de ces fournisseurs. Nous souhaitons cependant continuer à écrire ces fournisseurs dans le fichier dim_fournisseur.csv (nous ne rejetons pas les lignes qui ne respectent pas les conditions, nous souhaitons juste connaitre les lignes en erreur afin de pouvoir faire remonter l'information).

Pour cela, dans le tMap créer un second flux de sortie appelé rejets. Positionner un contrôle sur cette sortie afin de ne laisser passer dans ce flux uniquement les lignes dont le délai est null (utiliser l'opérateur == null). Ne placer dans le flux de sortie « rejets » uniquement le code et le nom du fournisseur et écrire le contenu du flux dans un fichier rejets_fournisseur.csv dans le répertoire Data/Cible.

Le tMap doit avoir la structure suivante :



tMap dimension fournisseur

Exécuter le job et vérifier la création des deux fichiers : dim_fournisseur.csv et rejets_fournisseur.csv.

6. Développement du job d'alimentation de la dimension Produit

La dimension produit est alimentée à partir du fichier produit_in.csv.

Créer un nouveau job appelé dimension_produit.

Placer le fichier produit_in.csv dans le dossier Data/Source et créer les métadonnées du fichier produit_in.csv (les colonnes prix et origine doivent être en integer).

Créer le job sur le même modèle que celui pour charger la dimension fournisseur.

Le tableau ci-dessous montre la correspondance des noms entre les données sources et cibles dans le tMap.

Entrée	Sortie
	Id_Produit
Code	Code_produit
Prix	Prix_produit
Origine	Origine_produit

Mapping entrée/sortie dimension Produit

Vous devez cette fois mettre en place un contrôle afin de vérifier que le prix et l'origine sont bien renseignés. A la différence du job de chargement de la dimension fournisseur, les produits qui n'ont pas de prix ou d'origine de renseigné ne doivent pas être écrits dans le fichier dim_produit.csv mais uniquement dans rejets_produit.csv.

Dans le flux de sortie des rejets, on souhaite récupérer le code produit ainsi que le motif du rejet qui peut-être soit l'absence de prix, soit l'absence d'origine. Pour cela, on crée dans le flux de sortie des rejets deux colonnes. Dans la première colonne appelée code_produit on fait glisser le code. Dans la seconde colonne appelée cause_rejet, vous devez mettre en place une expression qui fasse que : si le prix n'est pas renseigné alors le message est « Prix absent » sinon si l'origine n'est pas renseignée alors le message est « Origine absente » sinon le message est «».

Exécuter le job et vérifier le contenu des deux fichiers dim_produit.csv et rejets_produit.csv (si vous avez bien réussi à paramétrer les rejets, le nombre de ligne du fichier dim_fournisseur.csv + le nombre de ligne du fichier rejets_fournisseur.csv doit être égal au nombre de lignes du fichier source).

7. Développement du job lanceur des dimensions

Maintenant que les quatre jobs d'alimentation des dimensions sont terminés et fonctionnels nous allons créer un job lanceur pour les dimensions.

Créer un job appelé lanceur_dimensions.

Faire en sorte que ce job exécute dans l'ordre les dimensions Temps, Produit, Client et Fournisseur à l'aide de composants tRunJob. Paramétrer les tRunJob pour que le traitement s'arrête en cas d'erreur.

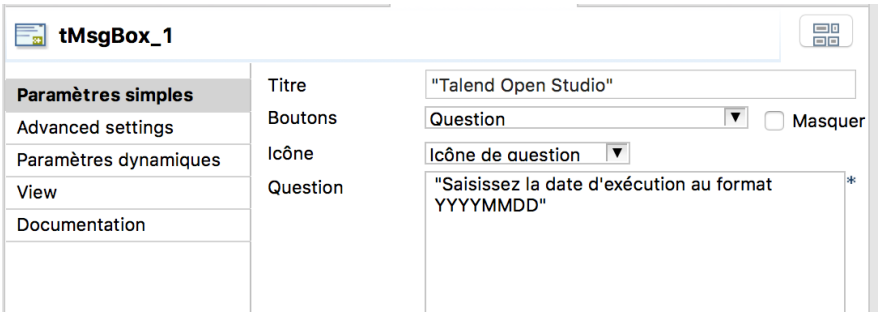
Exécuter le job et vérifier que les quatre sous-jobs se sont bien exécutés.

8. Développement du job lanceur_projet

Afin de simuler l'exécution de plusieurs jours, nous allons mettre en place un mécanisme qui va nous demander une date d'exécution en début de job. Nous allons voir que cette date nous permettra d'identifier le fichier des ventes à charger.

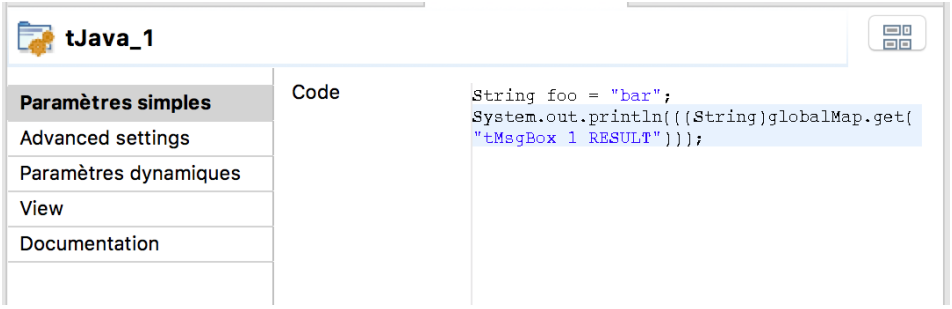
Créer un job lanceur_projet.

Insérer un composant tMsgBopx qui permet d'ouvrir une invite et de saisir une valeur. Nous allons l'utiliser ici pour saisir notre date. Cette date sera alors chargée dans une variable propre au composant que nous pourrons utiliser par la suite. Paramétrez le composant comme le montre la figure suivante.



Configuration du tMsgBox

Vu que c'est la première fois que nous utilisons ce composant, nous allons tester son fonctionnement. Insérer un composant tJava connecté après le tMsgBox à l'aide d'un lien tComponentOk. Paramétrez le composant jTava comme dans la figure suivante afin d'afficher la valeur saisie dans le composant tMsgBox.



Configuration du tJava

Exécuter le job et vérifier le fonctionnement. La date que vous avez saisi dans l'invite doit s'afficher dans la console.

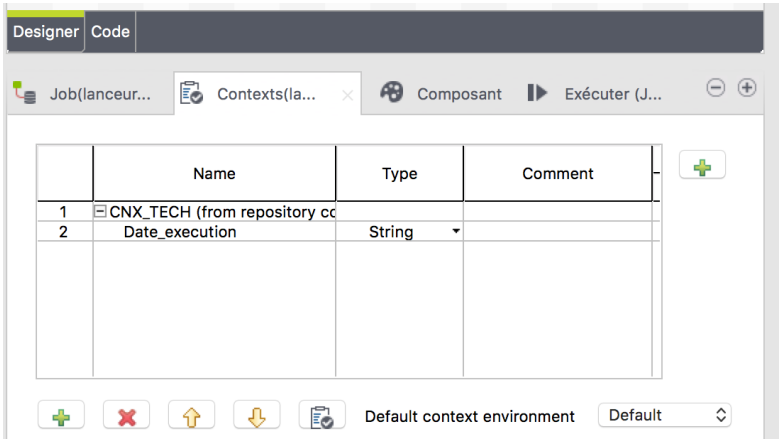
Maintenant que nous savons récupérer le résultat de l'invite, nous allons la charger dans une variable de contexte car nous allons l'utiliser dans le job de chargement de la table de faits des ventes (le job de chargement de la table

de fait est un job fils et nous ne pouvons donc pas utiliser une variable globale uniquement utilisable dans un même job).

Nous allons tout d'abord créer une variable de contexte que nous pourrons utiliser dans les différents jobs du projet.

Faites un clique droit sur contexte et sélectionnez « Créer un groupe de contexte ». Appelé le CNX_TECH et cliquez sur Suivant. Dans la nouvelle fenêtre, cliquez sur le + afin d'ajouter une nouvelle variable et nommez la Date_execution et cliquez sur terminez.

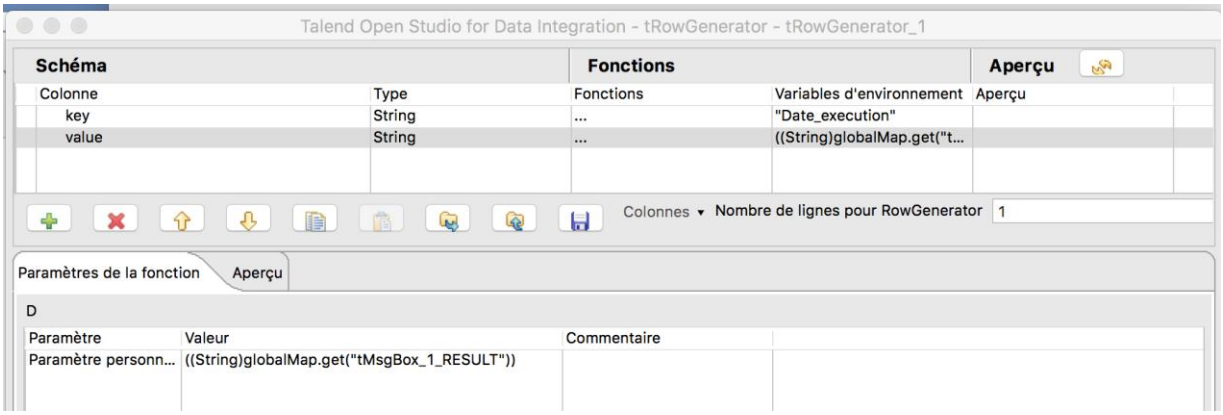
Pour ajouter la variable de contexte à votre job lanceur, ouvrez votre job et cliquez sur l'onglet Contexte. Faites ensuite glissez le groupe de contexte CNX_TECH dans l'onglet. Vous devez alors obtenir le même résultat que ci-dessous.



Onglet Contexte du job lanceur

Maintenant que la variable de contexte est créée et disponible dans le job nous allons pouvoir l'alimenter avec la date de l'invite.

Supprimez le tJava et insérez un composant tRowGenerator qui va nous permettre d'envoyer une ligne au composant tContextLoad. Paramétrez-le comme le montre la figure ci-dessous. Pour rappel, le composant tContextLoad attend un flux composé de deux colonnes key et value. Nous allons donc indiquer dans key le nom du contexte ("Date_execution" ° et dans value la valeur récupérée du tMsgBox ((String)globalMap.get("tMsgBox_1_RESULT")).



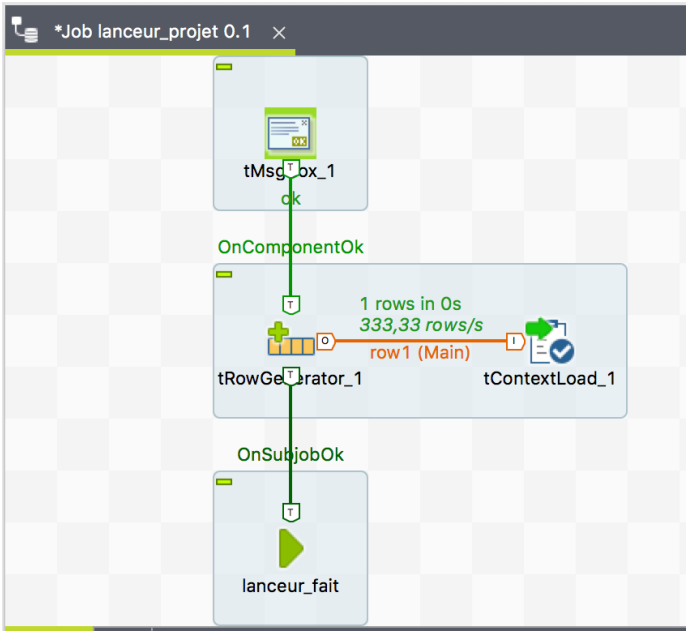
tRowGenerator pour charger le contexte

Envoyez le flux du tRowGenerator dans un composant tContextLoad. Cochez la case « Afficher les opérations » du tContextLoad et exécutez le job. Si tout est bien paramétré vous devez avoir dans la console le message : `tContextLoad_1 set key "Date_execution" with value "20171001".`

9. Développement du job lanceur_fait

Le job lanceur_fait ne va faire que lancer le job de chargement du fichier fait_ventes. Il doit cependant transmettre la variable de contexte du job lanceur_projet à notre futur job d'alimentation de la table de faits. Pour cela, il est nécessaire de lui définir le groupe de contextes. Comme pour le job lanceur, faites glissez le contexte CNX_TECH dans l'onglet Contexte du job.

Modifier le job lanceur_projet afin d'exécuter le job lanceur_fait une fois que la variable de contexte est bien chargée (Cochez les cases « Arrêt en cas d'erreur » et « Transmettre tout le contexte »).



Job lanceur_projet

10. Développement du job d'alimentation du fait ventes

Nous allons maintenant développer le job d'alimentation de la table de fait des ventes.

Créer un job fait_ventes.

Glissez le groupe de contexte CNX_TECH dans l'onglet Contexte.

Créer la métadonnée du fichier des ventes à partir d'un fichier des ventes (le choix du fichier n'est pas important car nous souhaitons uniquement récupérer le schéma).

Insérez un composant tFileInputDelimited basé sur la métadonnée que vous venez de créer. Nous allons maintenant modifier le chemin de lecture du fichier afin que celui-ci contienne la date du fichier en paramètre. Cliquez sur le chemin du fichier puis sur « Basculer la propriété en mode Built-In ». Remplacer le nom du fichier " …/ventes_20171001.csv" par "…/ventes_"+context.Date_execution+".csv".

Envoyez le flux de sortie dans un tLogRow qui va nous permettre de tester le fonctionnement dans un premier temps.

Modifiez le job lanceur_fait afin d'y insérer l'appel du job fait_ventes (pensez à cocher les cases nécessaires !).

La chaîne est maintenant prête donc vous pouvez retourner au niveau du job lanceur_projet et l'exécuter. Saisissez la date 20171001 et vérifiez que le contenu du fichier ventes_20171001.csv est bien envoyé à la console.

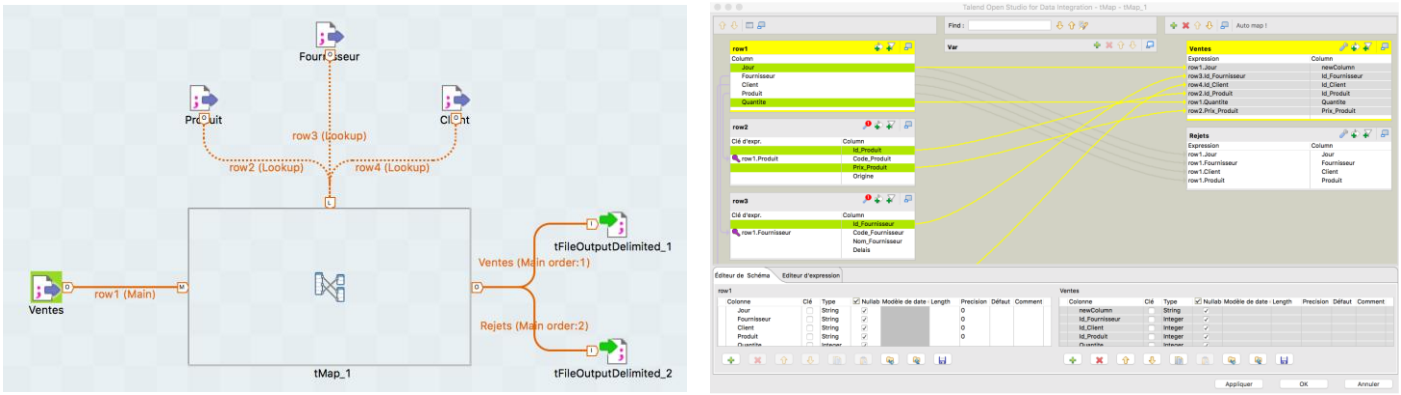
Maintenant que vous savez lire le fichier, vous pouvez continuer le développement du job d'alimentation de la table de faits en mettant en place les jointures suivantes :

- Jointure ventes – client afin de récupérer le l'id client. La jointure se fait sur le code_client.
- Jointure ventes – fournisseur afin de récupérer le l'id fournisseur. La jointure se fait sur le code_fournisseur.
- Jointure ventes – produit afin de récupérer le l'id produit et le prix du produit. La jointure se fait sur le code_produit.

Nous souhaitons charger uniquement les lignes pour lesquelles nous trouvons des correspondances dans l'ensemble des dimensions afin d'assurer la cohérence de notre étoile. L'ensemble des jointures doivent donc être de type « inner join ».

Redirigez les lignes de rejet des jointures dans une sortie appelée rejets dans le tMap.

Votre job et votre tMap doivent ressembler à l'image ci-dessous :



Job fait_ventes et tMap

Ecrivez le contenu de vos sorties dans les fichiers fait_vente.csv et rejets_vente.csv. A la différence des fichiers de sortie des dimensions, vous devez paramétrer vos composants de sortie de sorte que le fichier ne soit pas vidé à chaque nouvelle exécution.

Exécutez le job lanceur en saisissant la date 20171001 et vérifiez le contenu des fichiers fait_vente.csv et rejets_vente.csv. Testez avec une autre date.

11. Modification du job lanceur_projet

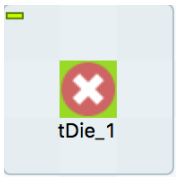
Modifiez le job lanceur afin d'ajouter un contrôle de présence des différents fichiers nécessaires à l'exécution avec des composants tFileExist. Si un fichier n'est pas présent affichez un message indiquant le nom du fichier absent à l'aide d'un tJava et faites-en sorte que le traitement ne continue pas.

Ajoutez le lancement du job lanceur_dimension à la suite des contrôles puis le lancement de lanceur_faits.

Le développement de la chaine est maintenant terminée et vous allez pouvoir la tester. Videz votre répertoire où se trouvent vos fichiers de sortie et exécutez le traitement pour les dates du 01/10/2017 au 05/10/2017. Vérifiez vos différents fichiers de sortie.

12.Mise en place des logs

Maintenant que la chaine d'alimentation est fonctionnelle nous allons la modifier afin d'y ajouter quelques logs. Pour cela, nous allons utiliser de nouveaux composants :

tLogCatcher 	Capture les messages provenant des composants tDie, du tWarn ainsi que les exceptions Java (erreur d'exécution).
tStatCatcher 	Capture le temps d'exécution du job et des composants si la case tStatCatcher Statistics est cochée.
tFlowMeterCatcher 	Capture le nombre d'enregistrements passant par les liens Main monitorés.
tWarn 	Déclenche un message d'avertissement.
tDie 	Arrête le Job en cours d'exécution et génère un message.
tPreJob 	Permet d'exécuter une pré-chaîne de traitement en début de job.
tPostJob 	Permet d'exécuter une post-chaîne de traitement en début de job.

Nous allons maintenant modifier l'ensemble des jobs. Afin de comprendre le fonctionnement, commencez par un job de dimension.

En début de traitement nous allons placer un composant tPreJob relié à un tWarn par un lien tComponentOk qui va nous permettre d'afficher le nom du job et l'heure d'exécution. Pour cela, remplacez le message d'avertissement du tWarn par : "Début du job " + jobName + " : " + TalendDate.getCurrentDate().

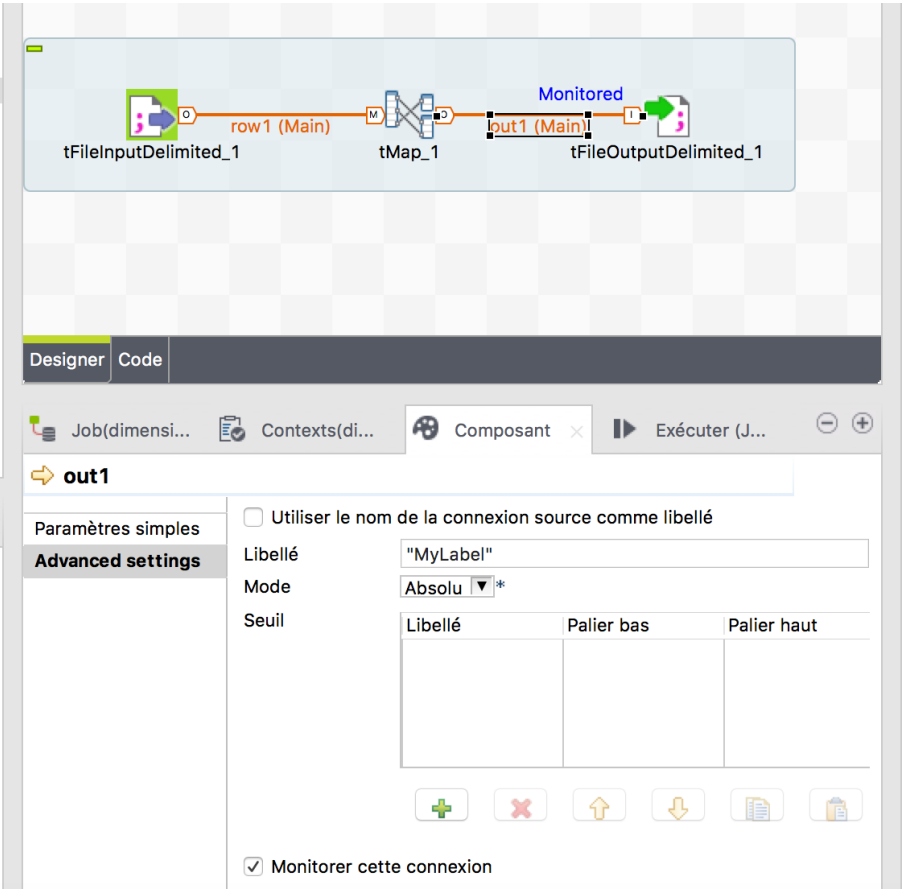
On va également placer en fin de job un composant tPostJob relié à un tWarn par un lien tComponentOk avec le message suivant : "FIn du job " + jobName + " : " + TalendDate.getCurrentDate().

Pour récupérer les messages des tWarn, placez dans le job un composant tLogCatcher que vous allez relier à un tFileOutputDelimited. Paramétrez le pour écrire dans un fichier log.csv, cochez la case écrire après afin de garder l'historique.

Ouvrez le composant tFIleOutputDelimited qui écrit votre fichier de dimension et dans l'onglet « Paramètres avancés » cochez la case « Statistiques du tStatCatcher ». Cela va permettre de récupérer les stats.

Pour récupérer les stats, placez dans le job un composant tStatCatcher que vous allez relier à un tFileOutputDelimited. Paramétrez le pour écrire dans un fichier stat.csv, cochez la case écrire après afin de garder l'historique.

Afin de récupérer des informations sur le volume de données traitées, cliquez sur le lien main qui arrive au tFIleOutputDelimited qui écrit votre fichier de dimension et dans l'onglet « Paramètres avancés » cochez la case « Monitorer cette connexion ».

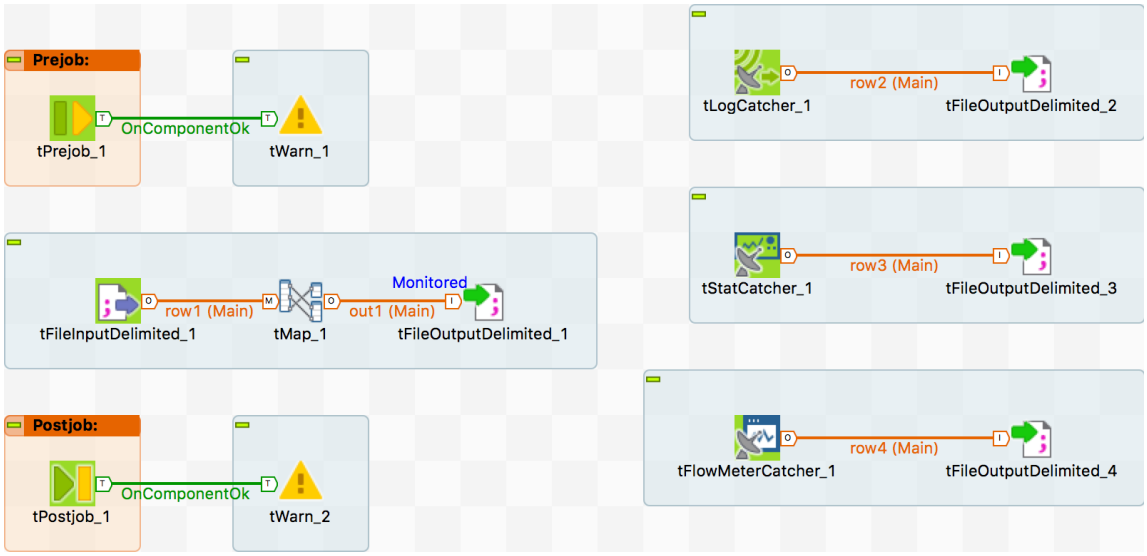


Paramétrage pour Monitorer une connexion

Pour récupérer ces infos, placez dans le job un composant tFlowMeter.+Catcher que vous allez relier à un tFileOutputDelimited. Paramétrez le pour écrire dans un fichier flow.csv, cochez la case écrire après afin de garder l'historique.

Dans le job lanceur, remplacez les tJava qui permettent de signaler l'absence des fichiers sources par des tWarn.

Au final, vos traitements doivent avoir la structure suivante.



Structure d'un job de dimension

Mettez en place les logs sur l'ensemble des jobs. Faites des copier/coller afin de gagner du temps.

13.Exercice Bonus

Créez un job fait_ventes_agrégées.

Ce traitement doit charger dans un fichier ventes_agregees.csv qui contient la somme des ventes par années et par fournisseur. Le fichier doit être écrasé à chaque run.

Colonnes de sorties : Année / Id Fournisseur / Somme des ventes

Ajouter ce traitement dans le lanceur fait.