ABEL Josselin

SEGARD Donatien

Master 2 MIAGE SIO

TD1 - Prise en Main

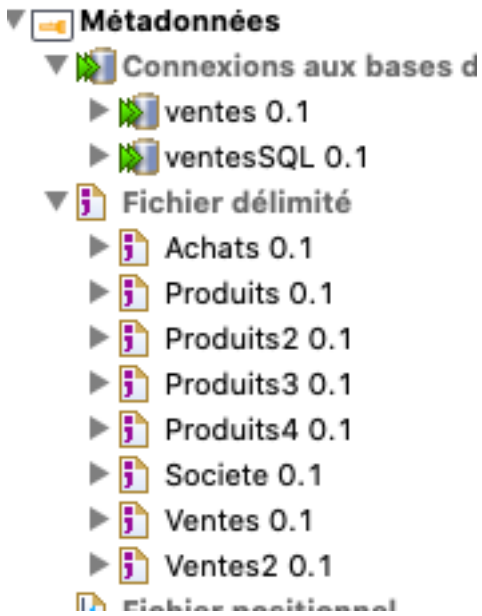
Rapport TD ETL

Etablissement : Université de Picardie Jules Verne

Enseignant : Aurélien BOULET

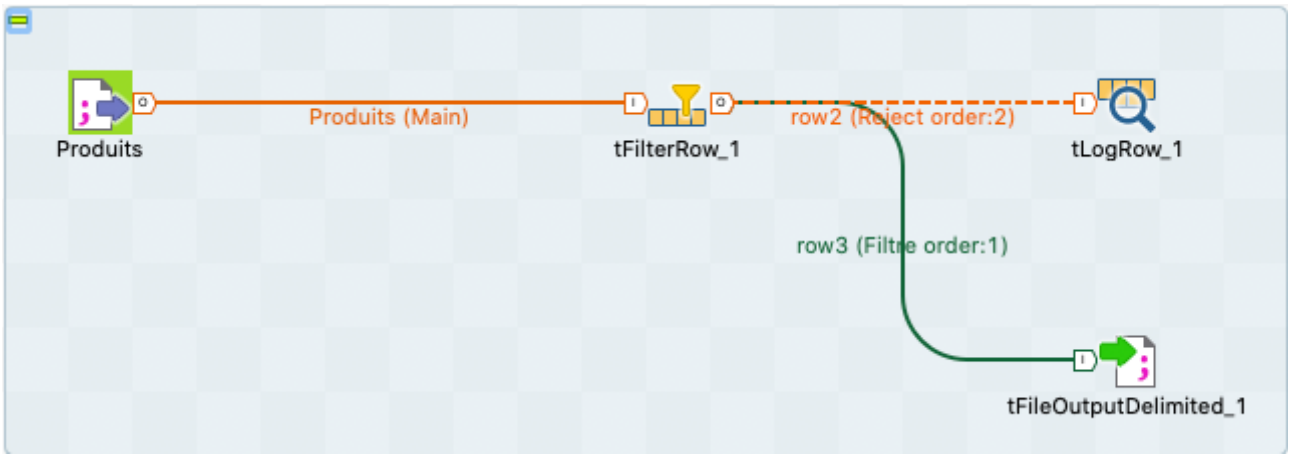
UE : Interopérabilité des SI

Les Métadonnées



TD2 - Filtre de ligne

Le TD2 consiste en la création d'un job talend permettant de filtrer d'un fichier en entrée les produit « Avion ».



Le composant tFileInputDelimited permet de prendre ne entrée un fichier délimité, ici Produits.txt. Ce fichier a été ajouté en tant que métadonnée.

Le composant tFilterRow permet de créer un filtre sur une ou plusieurs colonne(s) du fichier.

Conditions

Colonne d'entrée	Fonction	Opérateur	Valeur
prod_type	Vide	Vaut	"Avion"

Ici, on filtre la colonne « prod_type » pour avoir les lignes comprenant la valeur « Avion ».
Le composant tFileOutputDelimited permet de créer un fichier délimité en sortie pour y stocker, ici, les lignes passant le filter.
Enfin le composant tLogRow permet d'afficher des elements dans la console, ici, les lignes ne passant pas le filtre.

Les rejets dans la console :

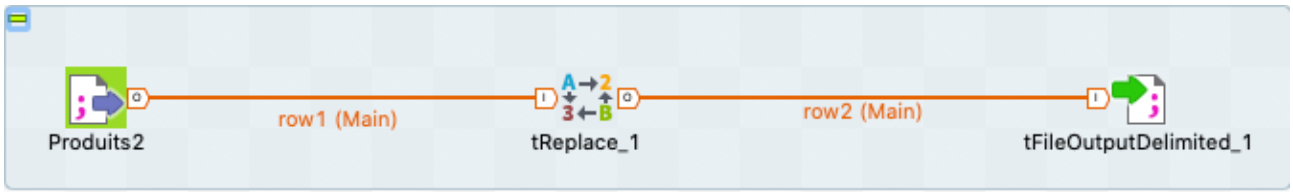
```
Starting job TD2 at 16:01 06/12/2020.
[statistics] connecting to socket on port 3571
[statistics] connected
prod_code   prod_lib    prod_marque prod_type   errorMessage
PRD001;Clio;Renault;Automobile;prod_type.compareTo("Avion") == 0 failed
PRD002;Megane;Renault;Automobile;prod_type.compareTo("Avion") == 0 failed
PRD003;Punto;Fiat;Automobile;prod_type.compareTo("Avion") == 0 failed
PRD004;Panda;Fiat;Automobile;prod_type.compareTo("Avion") == 0 failed
PRD005;Corail;Bombardier;Train;prod_type.compareTo("Avion") == 0 failed
PRD006;TGV;Alstom;Train;prod_type.compareTo("Avion") == 0 failed
[statistics] disconnected

Job TD2 ended at 16:01 06/12/2020. [exit code  = 0]
```

Le fichiers en sortie :

```
prod_code;prod_lib;prod_marque;prod_type
PRD007;A310;Airbus;Avion
PRD008;A320;Airbus;Avion
PRD009;777;Boeing;Avion
PRD007;A310;Airbus;Avion
PRD008;A320;Airbus;Avion
PRD009;777;Boeing;Avion
PRD007;A310;Airbus;Avion
PRD008;A320;Airbus;Avion
PRD009;777;Boeing;Avion
```

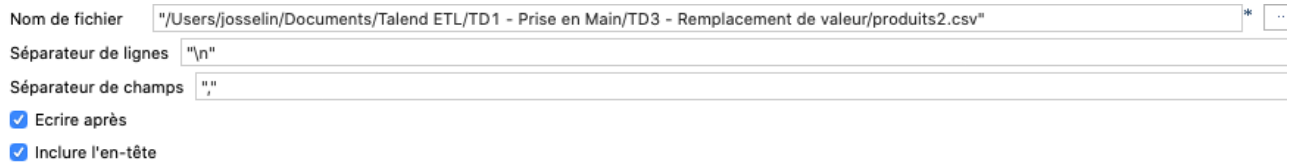
TD3 - Remplacement de valeur

Le TD3 consiste en la création d'un job permettant le remplacement d'une valeur « Airbus » dans le fichier en entrée par la valeur « EADS » fans le fichier en sortie.



Le composant tReplace qui permet de rechercher une valeur dans le flux de données, et de la remplacer par une valeur choisie.

Ici, on recherche la valeur « Airbus » dans la colonne « prod_marque » et on la remplace par la valeur « EADS ».
On peut paramétrer certains element du fichier en sortie dans le tFileOutputDelimited.



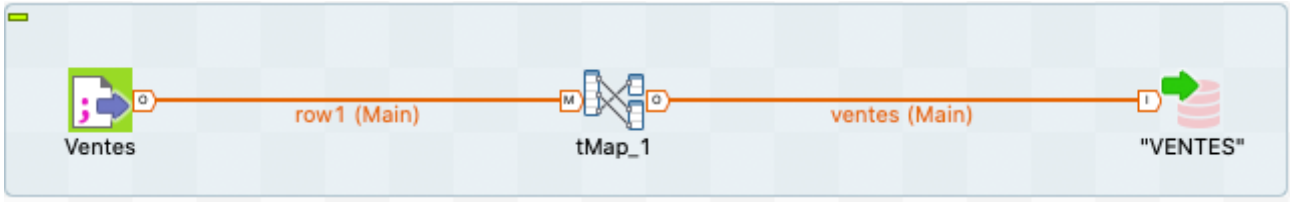
Ici, comme demandé dans l'énoncé, on precise le séparateur de champs « , ».

Le fichier en sortie (extrait) :

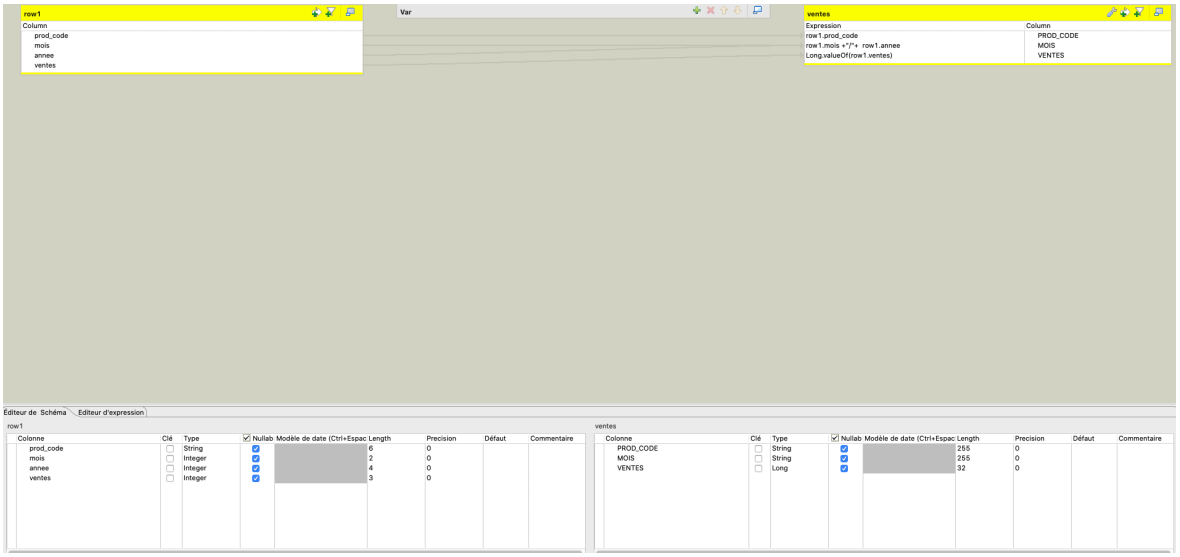
```
PRD001,Clio,Renault,Automobile
PRD002,Megane,Renault,Automobile
PRD003,Punto,Fiat,Automobile
PRD004,Panda,Fiat,Automobile
PRD005,Corail,Bombardier,Train
PRD006,TGV,Alstom,Train
PRD007,A310,EADS,Avion
PRD008,A320,EADS,Avion
PRD008,A320,John deere,Tracteur
PRD009,777,Boeing,Avion
```

TD4 - Utilisation du tMap

Le TD4 consiste en la création d'un job permettant le chargement d'un fichier dans une table ayant un schéma de données different.



Le composant tMap, permet la « mapping » de données. C'est dire le charger un des données provenant d'un ou plusieurs schéma dans un ou plusieurs schéma avec un structure différente.



Ici, on charge les données d'un schéma à quatre colonnes dans un schéma à 3 colonnes. Pour ce faire on drag&drop les données du schéma en entrée dans le schéma en sortie.
On peut aussi transformé les données. Ici, les colonnes « mois » et « année » sont chargé dans un seul colonne « mois » en les concaténant. De plus, la colonnes « ventes » en entrée et la colonne « vente » en sortie n'ont pas le même type, on opère donc une conversion.

Un fois le mapping fini, on utilise un composant tDBOutput pour charger les données dans une base de données.

On précise le liens vers la BDD, le user/password et le nom de la table pour permettre le chargement des données.

TD5 - Agrégation d'information

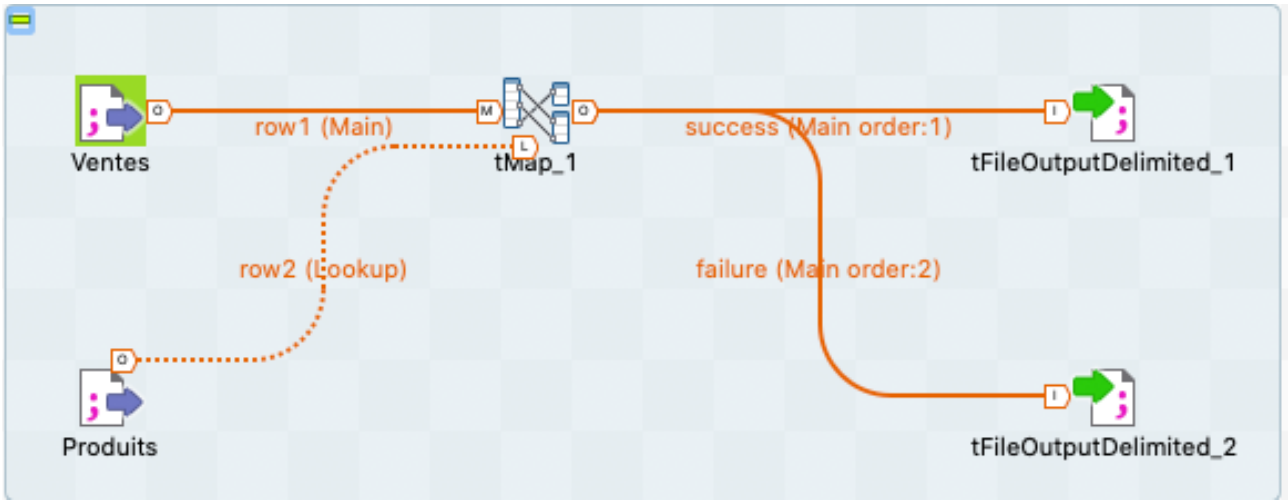
Le TD5 consiste en la création d'un job permettant l'agrégation d'information. Dans ce cas, la sommes des ventes par années.

Le composant tAggragateRow permet l'agrégation de données issue du flux en entrée. Il s'agit d'un simple Group By. Ici, on choisie d'agréger les informations par rapport à la colonne année. L'agrégation de fera via une addition des données dans la colonne vente.

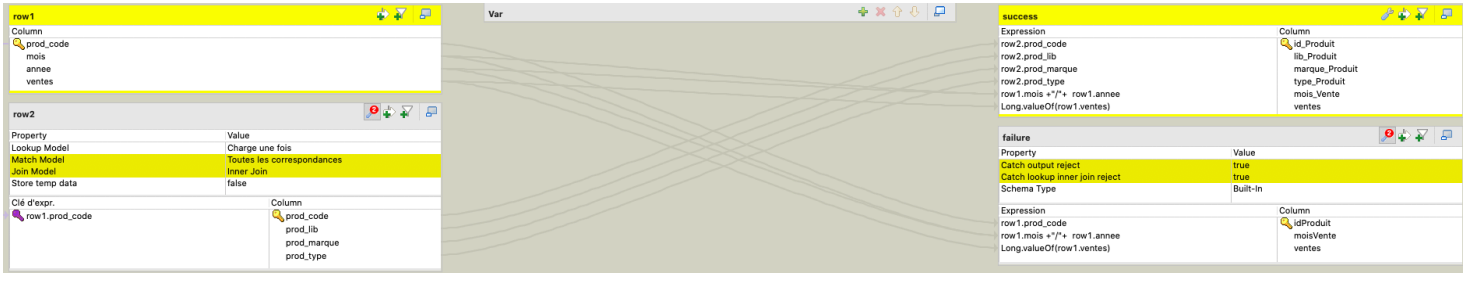
Enfin, on charge les données via un tDBOutput.

TD6 - Contrôle d'intégrité

Le TD6 consiste en la création d'un job permettant le contrôle de l'intégrité de données issue d'un fichier en fonction d'un fichier référentiel.



Ici, on utilise deux fichiers en entrée. Le fichier « Ventes » dont les données seront contrôlées et le fichier « Produits » qui sera le référentiel.



Le composant tMap permet durant le mapping de faire une jointure entre les deux schéma en entrée.
En effectuant une jointure interne sur les colonnes « prod_code » on peut vérifier si le produit présent dans le fichier « Ventes » existe dans le fichier « Produits ».
Enfin, on charge dans le schéma en sortie « success » les données passant la jointure et donc le contrôle d'intégrité. Les autres seront chargées dans le schéma « failure ».
Pour récupérer les rejets de la jointure, ont mets à « true » les champs « Catch output rejets » et « Catch lookup inner join reject ».

Enfin, on charge les données dans des fichiers délimités avec le composant tFileOutputDelimited. Un pour chaque schéma en sortie du tMap.

Le fichier success (extrait) :

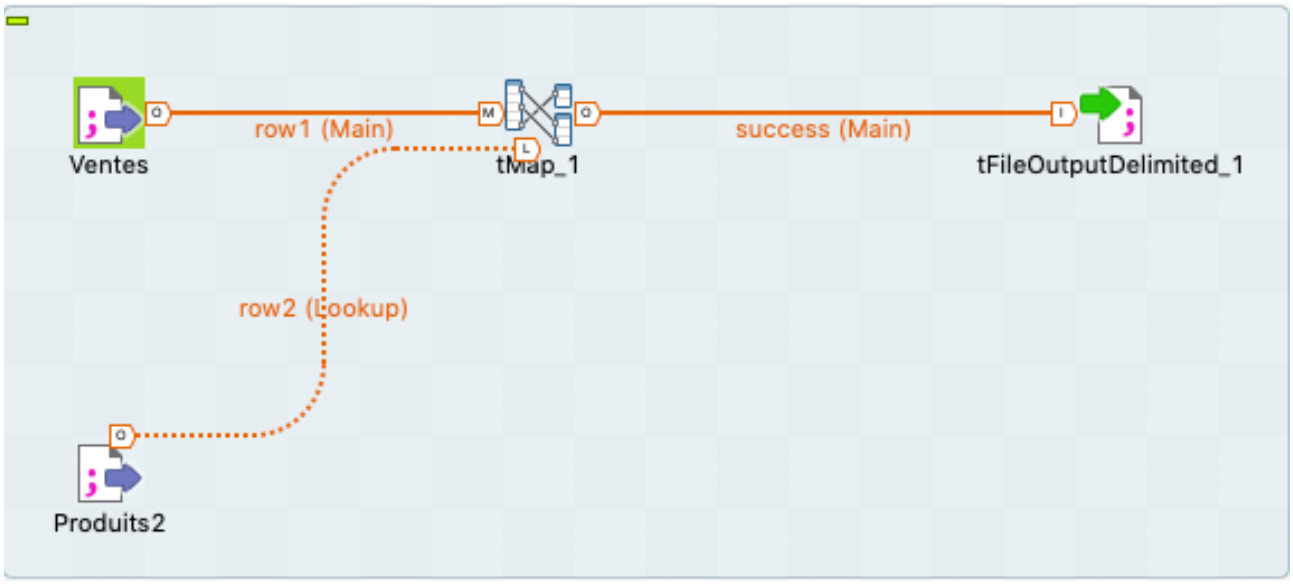
```
id_Produit;lib_Produit;marque_Produit;type_Produit;mois_Vente;ventes
PRD001;Clio;Renault;Automobile;1/2008;125
PRD001;Clio;Renault;Automobile;2/2008;120
PRD001;Clio;Renault;Automobile;3/2008;114
PRD001;Clio;Renault;Automobile;4/2008;107
PRD001;Clio;Renault;Automobile;5/2008;125
PRD001;Clio;Renault;Automobile;6/2008;135
PRD001;Clio;Renault;Automobile;7/2008;147
PRD001;Clio;Renault;Automobile;8/2008;189
PRD001;Clio;Renault;Automobile;9/2008;190
PRD001;Clio;Renault;Automobile;10/2008;187
PRD001;Clio;Renault;Automobile;11/2008;202
```

Le fichier failure (extrait) :

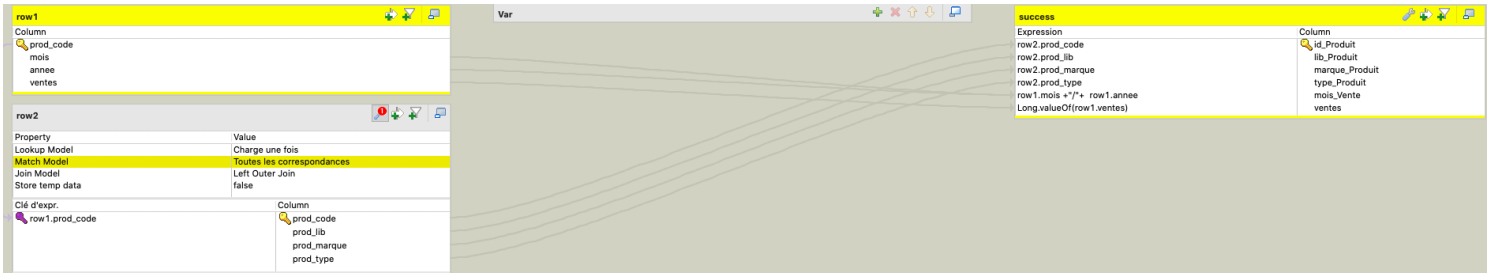
```
idProduit;moisVente;ventes
PRDZZZ;1/2008;251
PRDZZZ;2/2008;241
PRDZZZ;3/2008;229
PRDZZZ;4/2008;215
PRDZZZ;5/2008;251
PRDZZZ;6/2008;272
PRDZZZ;7/2008;296
PRDZZZ;8/2008;380
PRDZZZ;9/2008;382
PRDZZZ;10/2008;376
PRDZZZ;11/2008;406
PRDZZZ;12/2008;503
```

TD7 - Jointure externe

Le TD7 consiste en la duplication du job du TD6, mets en utilisant un jointure externe, en ne gardant que le données passant le contrôle d'intégrité.
Le job est donc essentiellement le même.



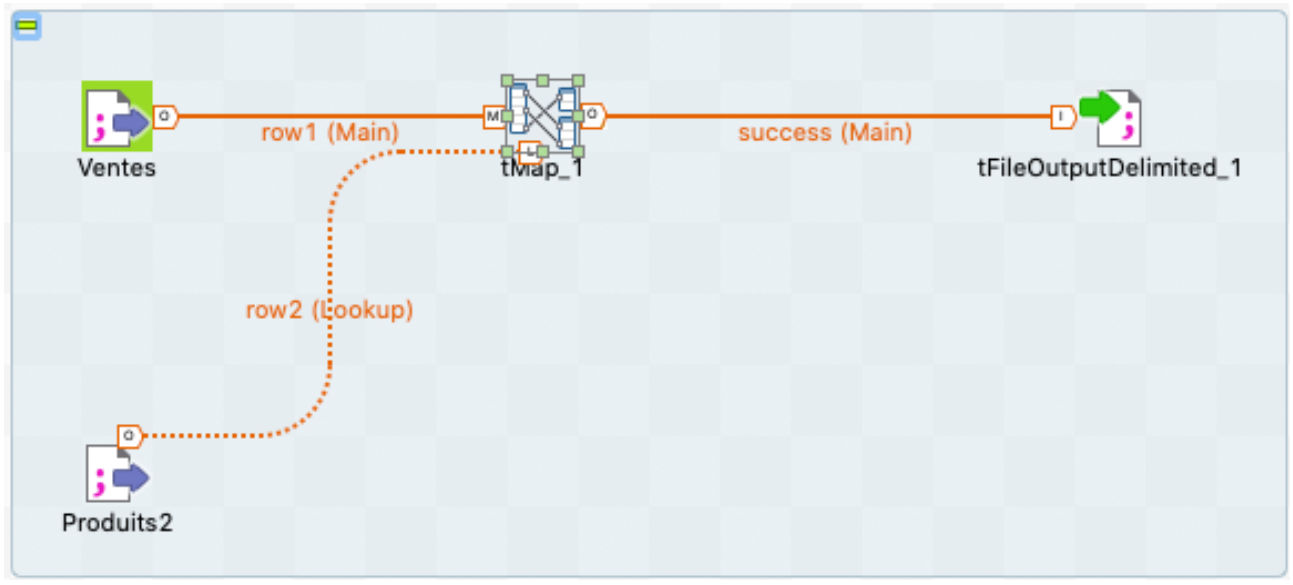
On utilise donc un left outer join et non un inner join.



La suite dû est le test des différents « Match Model ».

TD8 - If then else dans un tMap

Le TD8 consiste en la duplication du job du TD7 mais en ajoutant un « If » pour rajouter la libelle « Produit Inconnu » si la produit du fichier « Ventes » n'existe pas dans le fichier « Produits2 »



Ici, le job est le même que lors du TD7.

success

Expression	Column
row2.prod_code	id_Produit
row2.prod_lib==null ? "Produit Inconnu" : row2.prod_lib	lib_Produit
row2.prod_marque	marque_Produit
row2.prod_type	type_Produit
row1.mois +"/"+ row1.annee	mois_Vente
Long.valueOf(row1.ventes)	ventes

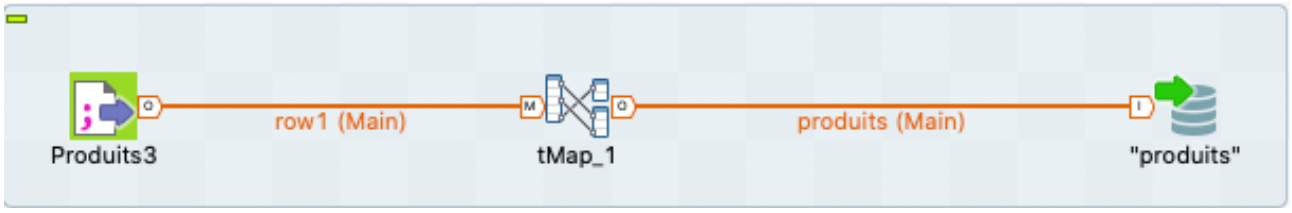
On à rajouter dans le mapping de la colonne « lib_produit » un « If » permettant de charger « Produit Inconnu » si le libelle dans le schéma en entrée est « null ».

Le fichier en sortie (extrait) :

```
PRD008;A320;John deere;Tracteur;10/2009;477
PRD008;A320;John deere;Tracteur;11/2009;507
PRD008;A320;John deere;Tracteur;12/2009;603
PRDZZZ;Produit Inconnu;;;1/2008;251
PRDZZZ;Produit Inconnu;;;2/2008;241
PRDZZZ;Produit Inconnu;;;3/2008;229
PRDZZZ;Produit Inconnu;;;4/2008;215
PRDZZZ;Produit Inconnu;;;5/2008;251
PRDZZZ;Produit Inconnu;;;6/2008;272
PRDZZZ;Produit Inconnu;;;7/2008;296
PRDZZZ;Produit Inconnu;;;8/2008;380
PRDZZZ;Produit Inconnu;;;9/2008;382
```

TD9 - Gestion des formats

Le TD9 consiste en la création d'un job permettant le chargement d'un schéma en entrées dans un schéma en sortie ayant des formats différents.



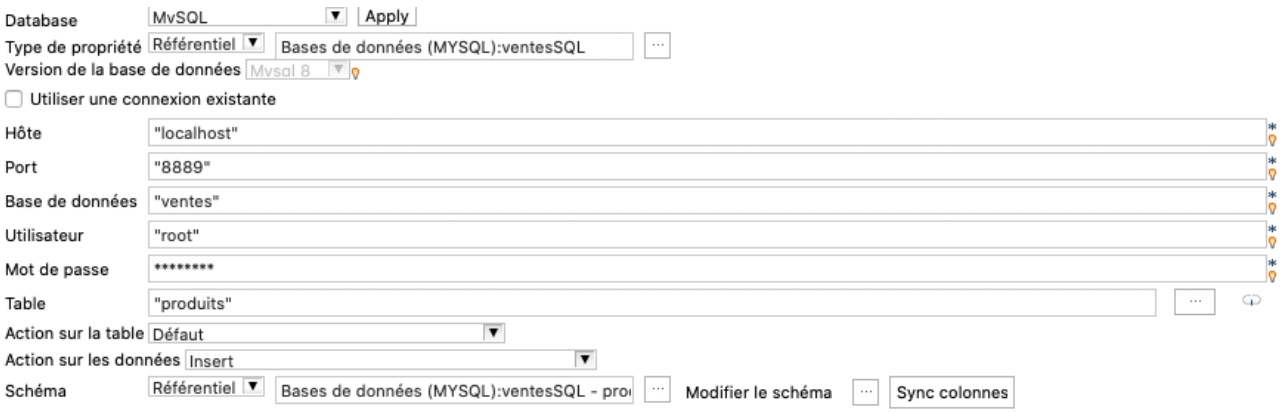
On utilise le composant tMap pour convertir les données au moment du mapping.

produits

Expression	Column
Integer.parseInt(row1.prod_id)	id_produit
Float.parseFloat(row1.prix)	prix

On utilise du code JAVA au moment du mapping des colonnes pour convertir les données en entrée au format en sortie.

Enfin, on utilise le composant tDBOutput pour charger les données dans une table.



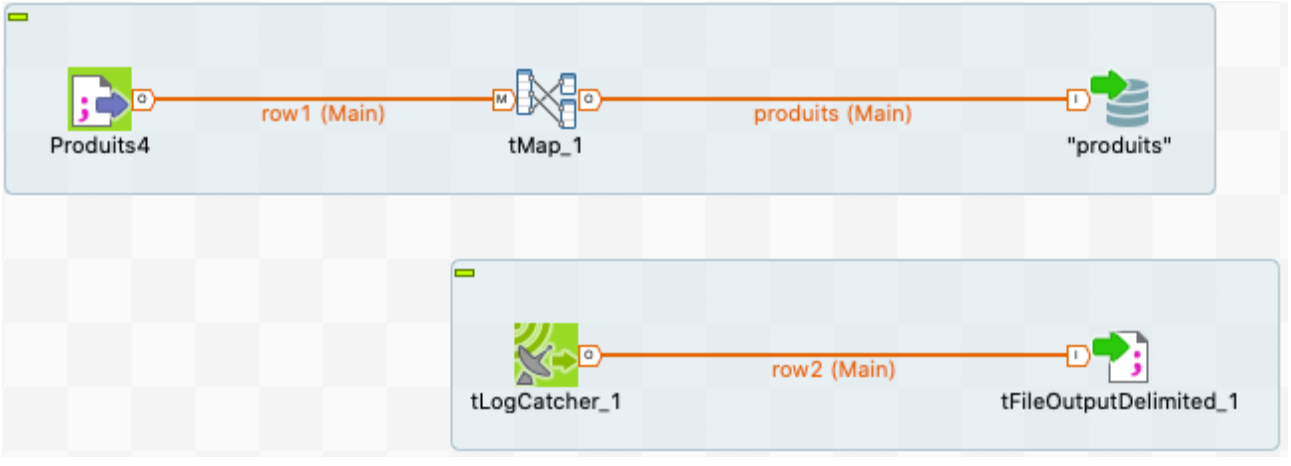
Ici, la BBD n'est plus « Access » mais « MySQL ». On doit donc préciser l'hôte, le port, la nom de la BDD, le nom de la table ainsi que le user/password du serveur.

Le table en sortie :

id_produit	prix
1	150.33
2	399.9
3	233.3
4	155.5

TD10 - Gestion des erreurs

Le TD10 consiste en la duplication du job du TD9, pour y ajouter la gestion des erreurs.



On utilise le composant tLogCatcher pour récupérer les erreurs JAVA.

On coche la case « Capturer les exceptions Java » pour cela.
Enfin, on charge les erreurs dans un fichier à l'aide du composant tFileOutputDelimited.

Fichier en sortie :

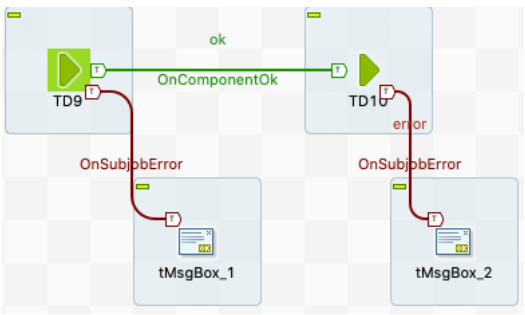
```
2020-12-06 16:25:18;3WKrni;3WKrni;3WKrni;TD1___PRISE_EN_MAIN;TD10;Default;6;Java
Exception;tMap_1;java.lang.NumberFormatException:For input string: "NC";1
```

Pour permettre la chargement des lignes dans la BDD avant l'erreur :

J'ai préciser dans les paramètres avancés du tDBOutput, qu'il fallait commiter à chaque enregistrement. Cependant dans le cadre de grosse volumétrie cent serait pas bon au niveau des performances.

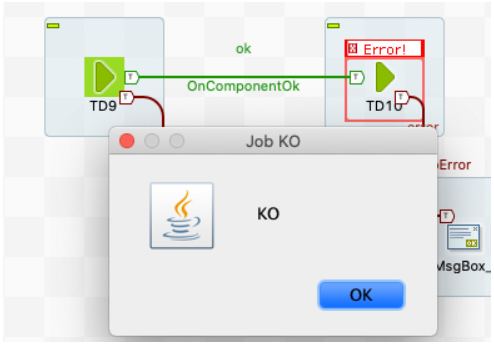
TD11 - Enchainement de job

Le TD11 consiste en la création d'un job permettant l'execution de deux job l'un après l'autre, ainsi que l'affiche de « KO » dans un fenêtre de dialogue en cas d'erreur dans l'un des jobs.



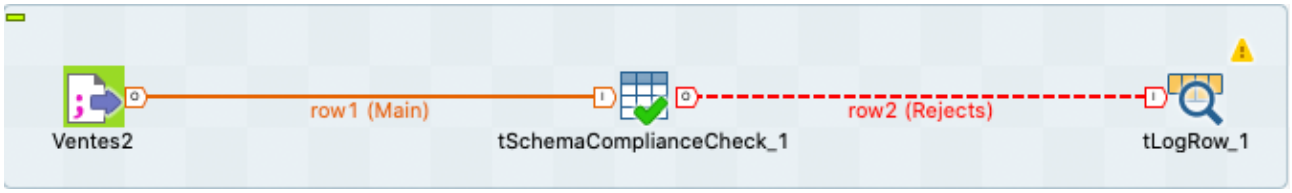
Le composant tMsgBox permet l'affichage d'un fenêtre de dialogue.

On droit préciser un titre pour la fenêtre, ainsi que son contenu, son icône et les boutons.
Les tMsgBox son lié par un lien OnSubjobError pour ne ce lancer qu'en cas d'erreur des jobs. A noter que dans ce cas un lien OnComponentError aurait aussi marché.

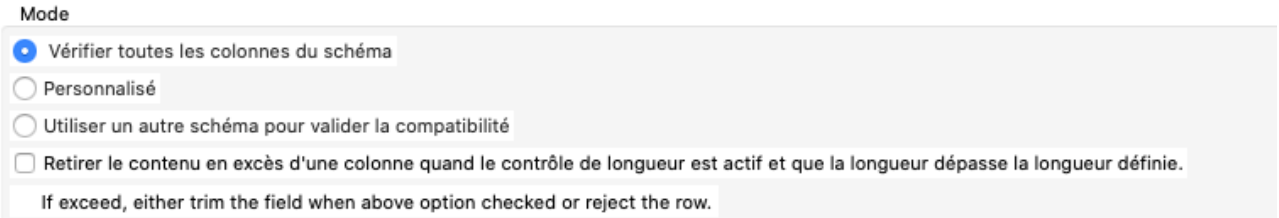


TD12 - Contrôle de format

Le TD12 consiste en la création d'un job permettant le contrôle du format des données d'un fichier en entrée.



Le composant tSchemaComplianceCheck permet de vérifier si les données correspondent au format attendu en sortie.



On lui demande de vérifier toutes les colonnes.

Colonne	Clé	Type	Nullab	Modèle de (	Length	Precisi	Défau
prod_code	☐	String	☑		6	0	
mois	☐	String	☑		2	0	
annee	☐	Integer	☑		4	0	
ventes	☐	String	☑			0	

On precise le format attendu dans le schéma attendu.

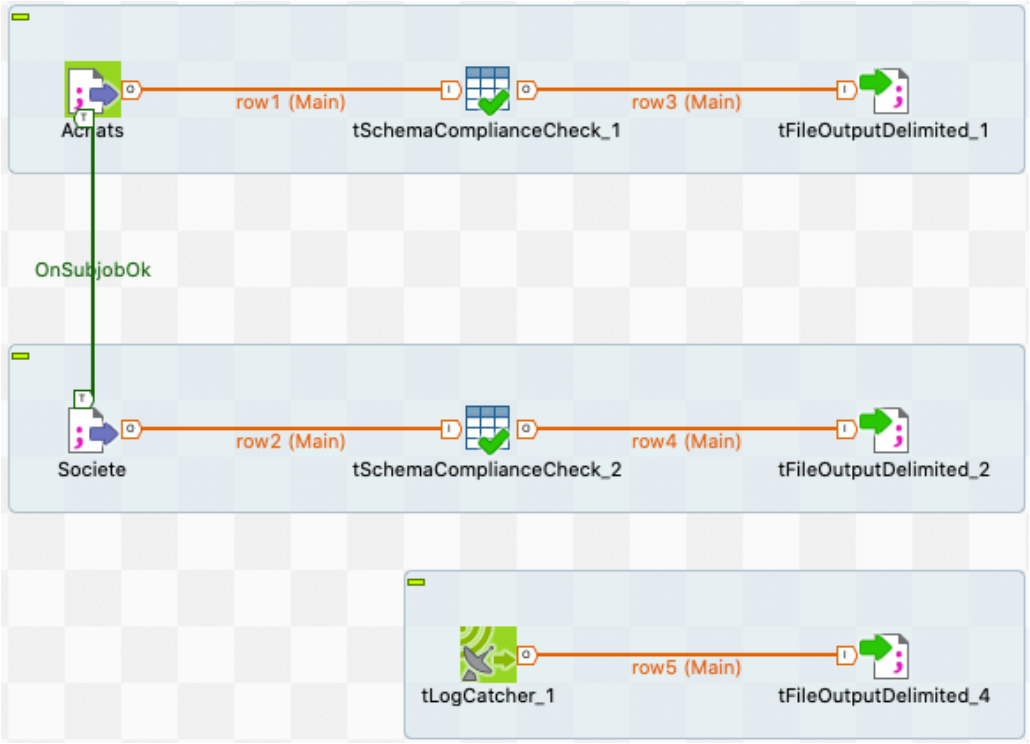
```
Starting job TD12 at 15:58 06/12/2020.
[statistics] connecting to socket on port 3361
[statistics] connected
PRD001|EADS|2008|114|8|mois:exceed max length
PRD001|4|20038|107|8|annee:exceed max length
PRDZZZZ|12|2009|603|8|prod_code:exceed max length
[statistics] disconnected

Job TD12 ended at 15:58 06/12/2020. [exit code  = 0]
```

Le tLogRow affiche dans la console les lignes ne correspondant pas au format attendu.

TD13 - Exercice Récapitulatif

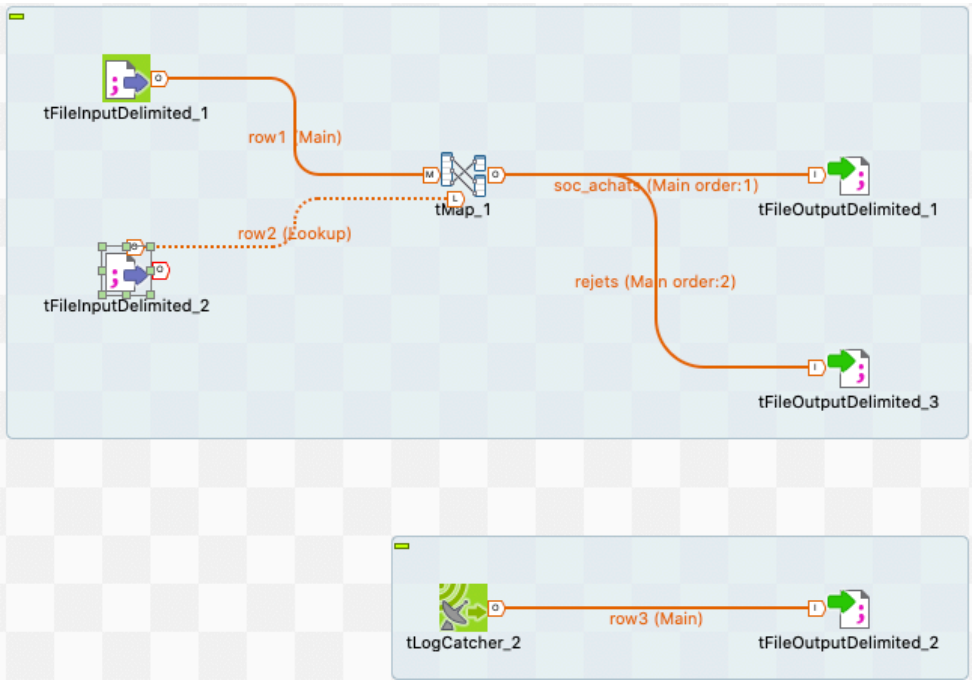
Le TD13 consiste en la création d'un job permettant le chargement d'un fichier en sortie « soc_achats » depuis les fichiers en entrée « Achats » et « Societe » avec un contrôle de format au préalable. Le tout repartie dans plusieurs job. Il faut aussi créer un fichier journal contenant les informations de traitement.
J'ai donc un job job « père » lançant à la suite deux job « fils ».

Le premier job permet le contrôle de format :

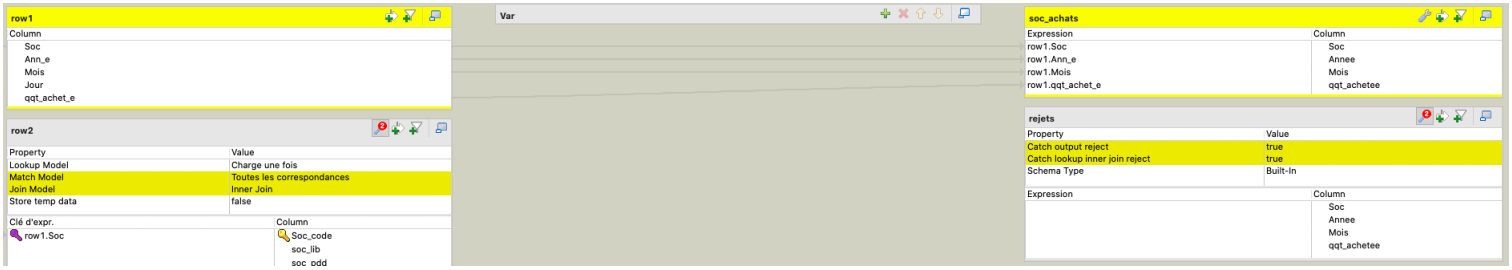


J'ai deux Subjob qui utilise le composant tSchemaComplianceCheck pour contrôler le format des deux fichiers en entrée. Les Subjob sont liés par un lien OnSubjobOk pour permettre l'execution du second en cas de réussite du premier.
Je crée deux fichiers en sortie qui contiendrons les lignes des deux fichiers respectifs qui ont passé le contrôle de format.
Enfin pour créer le fichier journal, j'utilise un tLogCatcher.

Le second job effectue le chargement :



Les fiches en entré sont les fichiers en sortie du premier job.
J'utilise un composant tMap pour faire la jointure interne entre les deux fichiers en entrée.



J'effectue une jointure interne en gardant toutes les correspondance. J'ai créer le schéma de sortie « soc_achats » en me basant sur le schéma donné dans l'énoncé. J'ai aussi créé un schéma « rejets » pour stocker les lignes ne passant pas la jointure. Le tMap permet aussi via la jointure interne d'effectuer un contrôle d'intégrité pour ne garder que les sociétés du fichier « achats » existant dans le fichier « société ».

Enfin toujours pour le fichier journal, je charge la erreur potentiel avec un tLogCatcher est un tFileOutputDelimited.

Fichier « soc_achats » (extrait) :

```
Soc;Annee;Mois;qqt_achetee
001;2008;01;1
001;2008;01;114
001;2008;01;68
001;2008;01;105
001;2008;01;36
001;2008;01;157
001;2008;01;133
001;2008;01;33
001;2008;01;235
001;2008;01;24
001;2008;01;38
001;2008;01;207
001;2008;01;90
001;2008;01;15
001;2008;01;200
001;2008;01;235
001;2008;01;28
001;2008;01;3
001;2008;01;44
001;2008;01;35
001;2008;01;139
001;2008;01;30
001;2008;01;138
001;2008;01;150
```