

CAS PRATIQUE SSIS

Echo Pilote



UBS Vannes
2022 - 2023

1 TABLE DES MATIERES

2	INTRODUCTION DU CAS PRATIQUE	2
2.1	CAS PRATIQUE	2
2.2	DONNEES UTILISEES	2
3	INITIALISATION DE LA SOLUTION	2
3.1	CREATION DU PROJET	2
3.2	PARAMETRAGE ET CONNEXIONS DE PROJET	3
4	IMPLEMENTATION DES PACKAGES	4
4.1	CANAL	4
4.2	PRODUIT	7
4.3	CLIENT	10
4.4	VENTES	15
5	AMELIORATION DES PACKAGES	19
5.1	TEST DE L'EXISTENCE DU FICHIER	19
5.2	BOUCLE SUR LES FICHIERS DE VENTES.	21
6	PACKAGES TECHNIQUES	23
6.1	MAIN	23
6.2	SEQUENCE SSAS	24

2 INTRODUCTION DU CAS PRATIQUE

2.1 CAS PRATIQUE

Ce que vous allez faire dans cet exercice :

- ✓ Charger des données à partir de fichiers plats (CSV)
- ✓ Utiliser la tâche de dimension à transformation lente
- ✓ Utiliser la tâche de chargement en masse
- ✓ Utiliser la tâche d'exécution SQL
- ✓ Alimenter un DWH

2.2 DONNEES UTILISEES

Les données utilisées sont celles de la base de démonstration Echo Pilote. Il s'agit de données de ventes d'une entreprise fictive.

Les fichiers suivants sont à disposition pour le chargement des données :

Source	Type source	Mode d'alimentation	Table de destination
Canal.csv	csv	full	[ssisdbo].[Canal]
Client.csv	csv	slow	[ssisdbo].[Client]
Commercial.csv	csv	full	[ssisdbo].[Commercial]
Objectif.csv	csv	delta	[ssisdbo].[Objectif]
Produit.csv	csv	delta	[ssisdbo].[Produit]
Succursale.csv	csv	full	[ssisdbo].[Succursale]
TypeClient.csv	csv	full	[ssisdbo].[TypeClient]
Vente.csv	csv	delta	[ssisdbo].[Vente]

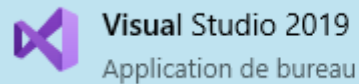
3 INITIALISATION DE LA SOLUTION

3.1 CREATION DU PROJET

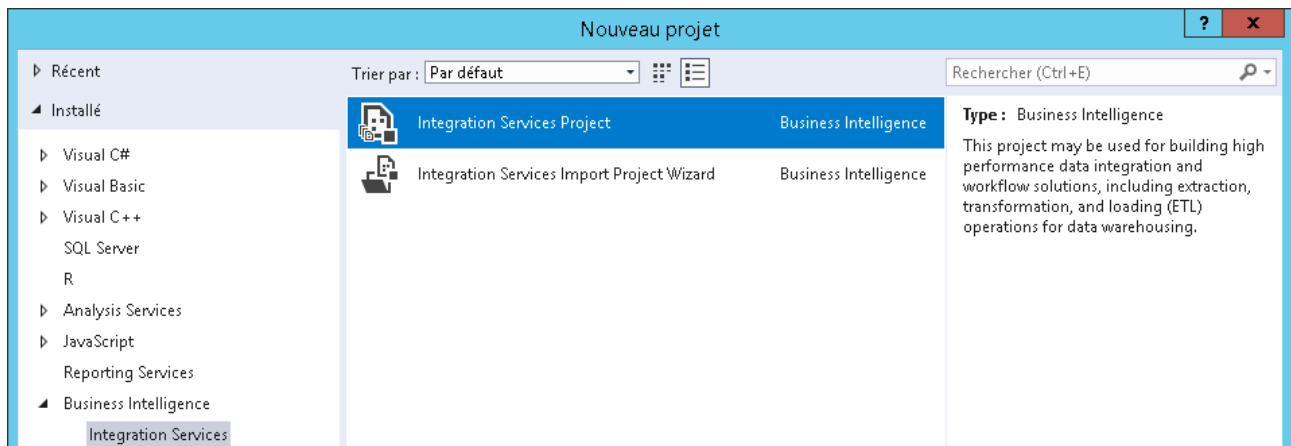
La première étape consiste à créer un projet SSIS. Un projet Visual Studio fait partie d'une solution qui peut embarquer plusieurs projets. Il est fréquent de regrouper dans la même solution le projet SQL du Datawarehouse, le projet SSIS et le projet SSAS.

Créer un nouveau projet « SSIS » dans Visual Studio ou SQL Server Data Tool :

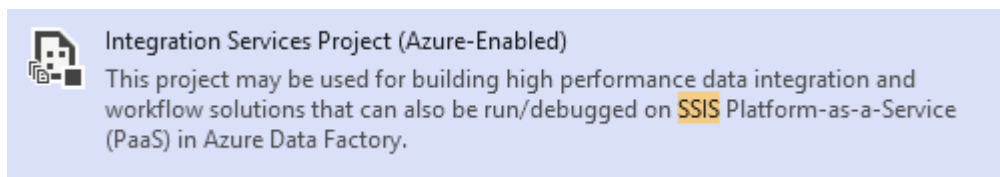
- ✓ Ouvrir Visual Studio ou SSDT



- ✓ Sélectionner « Nouveau projet » dans le menu.
- ✓ Sélectionner comme type de projet « Integration Services Projet » :



Visual studio 2017



Visual studio 2019

- ✓ Nommer le projet et choisir un emplacement, garder cochée l'option « Créer un répertoire pour la solution » (ou si vous utilisez Visual Studio 2019, décocher l'option « Placer la solution et le projet dans le même répertoire »).
- ✓ Sélectionner « Créer ».

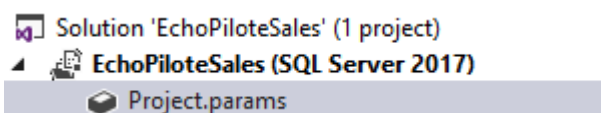
Le projet est créé avec un premier package vide nommé « Package.dtsx »


3.2 PARAMETRAGE ET CONNEXIONS DE PROJET



Les ressources utilisées par les packages d'un projet sont souvent communes ; par exemple les package utilisent la même connexion à la base de données destination.

Créer les paramètres projet.

- ✓ Dans l'explorateur de solution, ouvrir « Project.params ».



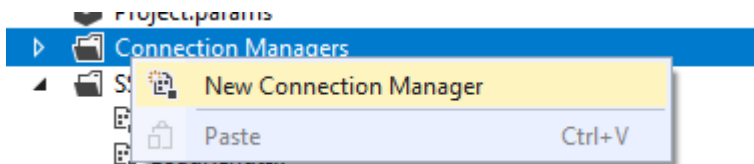
- ✓ Ajouter un nouveau paramètre avec le bouton 
 - Nommer le paramètre « SourceFilePath »
 - Choisir le type « String »
 - Entrer comme valeur le chemin du répertoire où sont stockés les fichiers sources.

Project.params [Design] 					
Name	Data type	Value	Sensitive	Required	Description
 SourceFilePath	String	D:\OneDrive - UMANIS\Trainings\SsisTraining-0.5jour\EchoPilote...	False	True	

- ✓ Enregistrer la page (Ctrl+S).

Créer une connexion vers la base de données de destination.

- ✓ Dans l'explorateur d'objet, sélectionner « Connection Managers ». Faire un clic-droit → « New Connection Manager ».



- Sélectionner le type de connexion « OLE DB » et cliquer sur « Add ».
 - Dans la page suivante, aucune connexion ne devrait être proposée. Sélectionner « New... ».
 - Entrer le nom du serveur, les informations d'authentification et le nom de la base de données à utiliser.
 - Tester la connexion et si c'est OK, valider les différentes étapes.
- ✓ Dans le menu « File », sélectionner « Save All » pour enregistrer l'ajout de la connexion au projet.

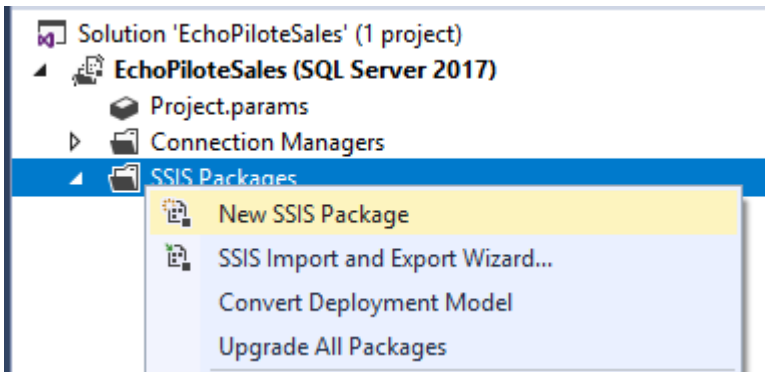
4 IMPLEMENTATION DES PACKAGES

4.1 CANAL

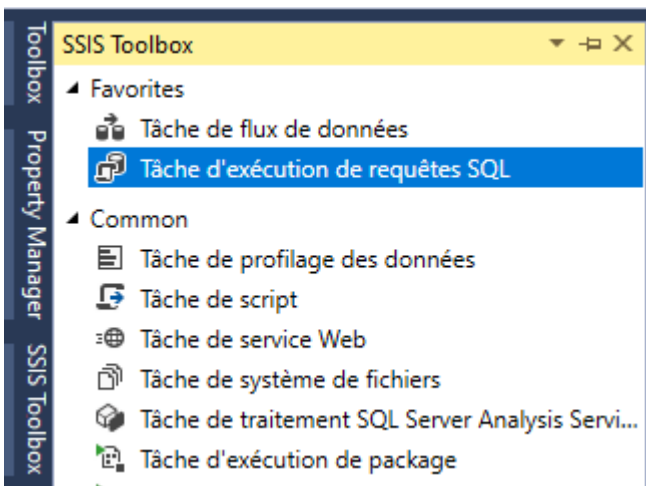
La dimension canal est très simple, avec peu de données. On va l'alimenter en « supprimer et insert », c'est-à-dire qu'on va **supprimer l'ensemble des données déjà chargées pour recharger les données à nouveau**.

Créer et initialiser le package « Canal.dtsx ».

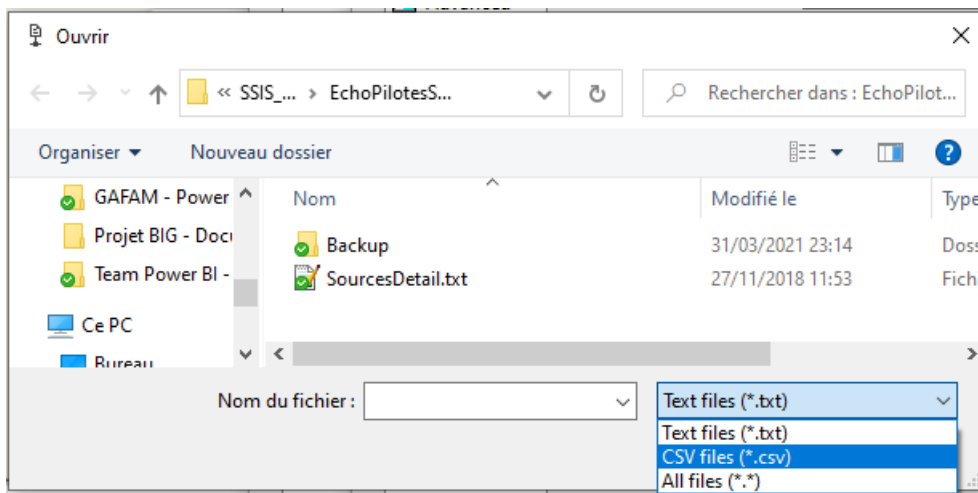
- ✓ Créer un nouveau package et le renommer : Canal.dtsx



- ✓ A partir de la boîte à outil SSIS, ajouter une tâche d'exécution de requêtes SQL :



- ✓ Renommer la tâche : Truncate Canal
- ✓ Double-cliquer sur la tâche pour la configurer. Dans le menu « General »
 - ResultSet : None
 - Connection :
 - SQLSourceType : Direct input
 - SQLStatement : TRUNCATE TABLE [ssisdbo].[Canal]
- ✓ Valider et fermer la fenêtre de configuration avec « OK ».
- ✓ A partir de la boîte à outil SSIS, ajouter une tâche de flux de données : Tâche de flux de données
- ✓ Renommer la tâche de flux de données : Charge Canal
- ✓ Ouvrir le flux de données (double-clic).
- ✓ Ajouter une source de fichier plat Source du fichier plat et l'ouvrir.
 - Pour la connexion à la source sélectionner « New... ».
 - Utiliser « Browse... » pour naviguer jusqu'au fichier « Canal.csv ». Penser à modifier le filtre de type de fichier pour afficher les fichier « .csv ».



- Modifier l'encodage pour « 65001 (UTF-8) ». La connexion doit être configurée ainsi :

Select a file and specify the file properties and the file format.

File name:

Locale: ☐ Unicode

Code page:

Format:

Text qualifier:

Header row delimiter:

Header rows to skip:

☒ Column names in the first data row

- Dans l'onglet « Advanced » modifier les types des colonnes :

CanalID	DT_I4
CanalDesc	DT_WSTR, 50

Configure the properties of each column.

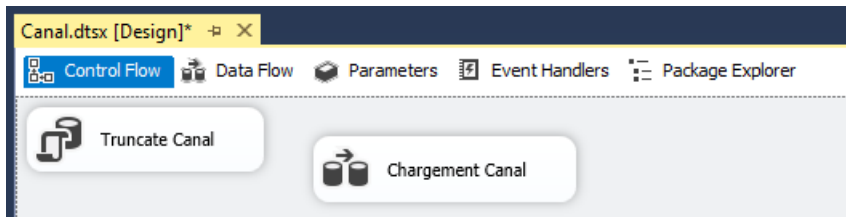
CanalID	CanalDesc
CanalID	CanalDesc

Misc	
Name	CanalDesc
ColumnDelimiter	{CR}{LF}
ColumnType	Delimited
InputColumnWidth	0
DataPrecision	0
DataScale	0
DataType	chaîne Unicode [DT_WSTR]
OutputColumnWidth	50
TextQualified	True

- Valider avec « OK ».
- De retour dans la fenêtre de configuration de la source, vérifier les colonnes dans l'onglet « Columns » :

External Column	Output Column
CanalID	CanalID
CanalDesc	CanalDesc

- ✓ Valider avec « OK » pour fermer la fenêtre de configuration de la source.
- ✓ Ajouter une tâche Destination OLE DB et créer une relation de la source vers la destination.
- ✓ Double-cliquer sur la destination pour la configurer.
- ✓ Sélectionner la connexion vers la base de données à alimenter et la table [ssisdbo].[Canal].
- ✓ Ouvrir l'onglet « Mappings » et vérifier la configuration de mappage entre la source et la destination.
- ✓ Valider la configuration de la destination avec « OK ».
- ✓ Retourner dans le flux de contrôle.



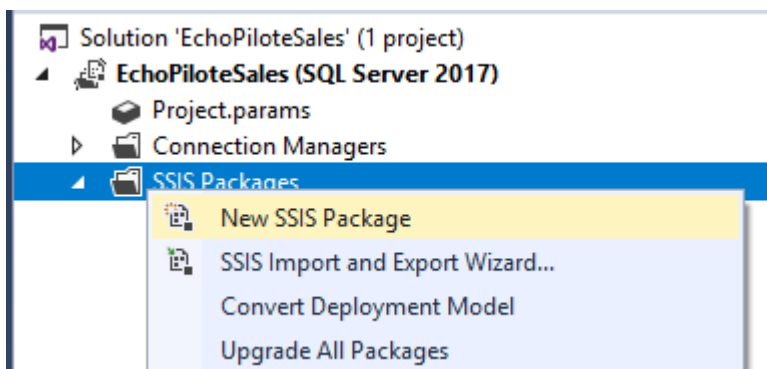
- ✓ Créer une contrainte de précédence de la tâche SQL vers le flux de données.
- ✓ Tester le package avec le bouton « Start » et vérifier si les données sont bien chargées.

4.2 PRODUIT

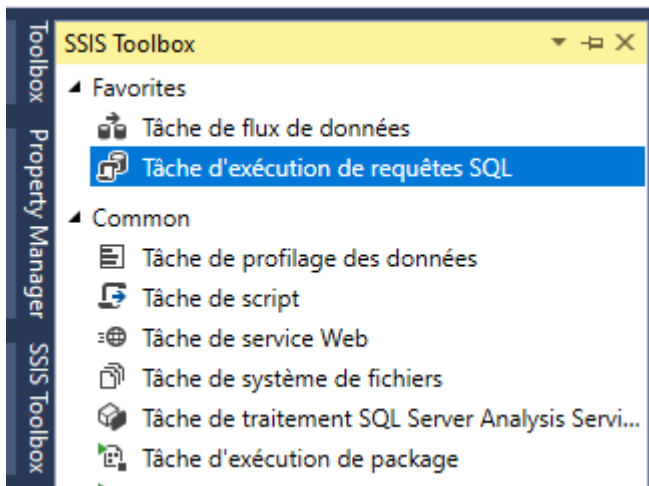
La dimension produit est assez simple mais les valeurs peuvent être mise à jour régulièrement. On va l'alimenter en « annule et remplace ».

Créer et initialiser le package « Produit.dtsx ».

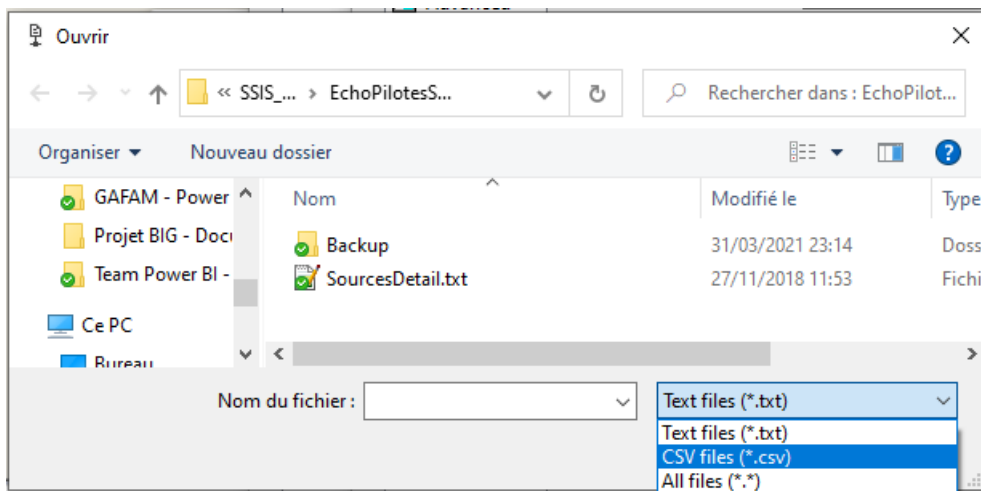
- ✓ Créer un nouveau package et le renommer : Produit.dtsx



- ✓ A partir de la boîte à outil SSIS, ajouter une tâche d'exécution de requêtes SQL :



- ✓ Renommer la tâche : Truncate Produit
- ✓ Double-cliquer sur la tâche pour la configurer. Dans le menu « General »
 - ResultSet : None
 - Connection :
 - SQLSourceType : Direct input
 - SQLStatement : TRUNCATE TABLE [ssisods].[Produit]
- ✓ Valider et fermer la fenêtre de configuration avec « OK ».
- ✓ A partir de la boîte à outil SSIS, ajouter une tâche de flux de données : Tâche de flux de données
- ✓ Renommer la tâche de flux de données : Charge Canal
- ✓ Ouvrir le flux de données (double-clic).
- ✓ Ajouter une source de fichier plat Source du fichier plat et l'ouvrir.
 - Pour la connexion à la source sélectionner « New... ».
 - Utiliser « Browse... » pour naviguer jusqu'au fichier « Produit.csv ». Penser à modifier le filtre de type de fichier pour afficher les fichier « .csv ».



- Modifier l'encodage pour « 65001 (UTF-8) ». La connexion doit être configurée ainsi :

Select a file and specify the file properties and the file format.

File name:

Locale: ☐ Unicode

Code page:

Format:

Text qualifier:

Header row delimiter:

Header rows to skip:

☒ Column names in the first data row

- Dans l'onglet « Advanced » modifier les types des colonnes :

ProduitID	DT_I4
ProduitDesc	DT_WSTR, 50
GammeDesc	DT_WSTR, 50
TypeProduitDesc	DT_WSTR, 50

- Valider avec « OK ».
 - De retour dans la fenêtre de configuration de la source, vérifier les colonnes dans l'onglet « Columns » :
- ✓ Valider avec « OK » pour fermer la fenêtre de configuration de la source.
 - ✓ Ajouter une tâche Destination OLE DB et créer une relation de la source vers la destination.
 - ✓ Double-cliquer sur la destination pour la configurer.
 - ✓ Sélectionner la connexion vers la base de données à alimenter et la table [ssisods].[Produit].
 - ✓ Ouvrir l'onglet « Mappings » et vérifier la configuration de mappage entre la source et la destination.
 - ✓ Valider la configuration de la destination avec « OK ».
 - ✓ Retourner dans le flux de contrôle.
 - ✓ Créer une contrainte de précédence de la tâche SQL vers le flux de données.
 - ✓ A partir de la boîte à outil SSIS, ajouter une tâche d'exécution de requêtes SQL.
 - Configurer la tâche pour les produits présents à la fois dans la table ODS et dans la table DBO soit supprimés de la table DBO :

```
delete from t
from [ssisdbo].[Produit] t
inner join [ssisods].[Produit] s on s.ProduitID = t.ProduitID
```

ou

```
delete from [ssisdbo].[Produit]
where ProduitID in (select ProduitID from [ssisods].[Produit])
```

- ✓ A partir de la boîte à outil SSIS, ajouter une tâche d'exécution de requêtes SQL.
 - Configurer la tâche pour tous les produits de la table ODS soit chargés dans la table DBO :

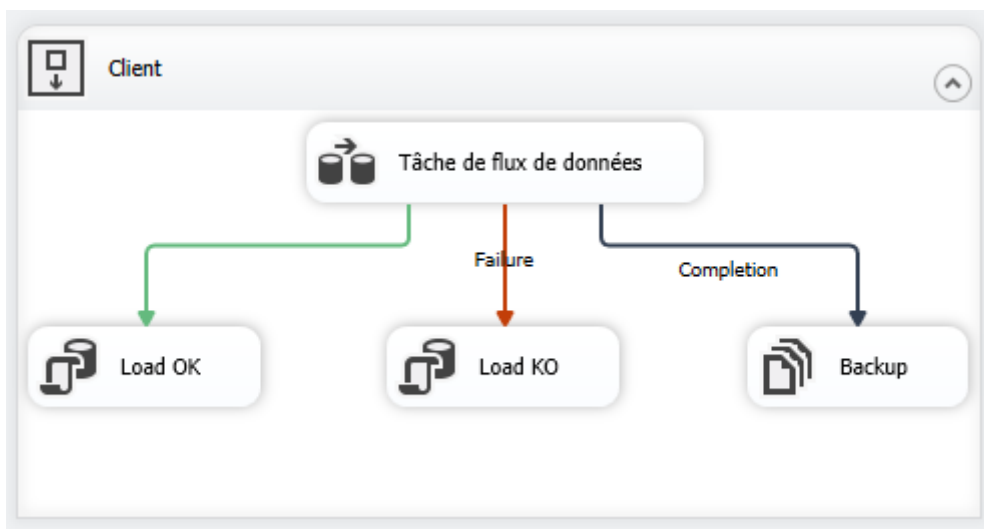
```
insert into [ssisdbo].[Produit] (
[ProduitID],[ProduitDesc],[GammeDesc],[TypeProduitDesc])
select [ProduitID],[ProduitDesc],[GammeDesc],[TypeProduitDesc]
from [ssisods].[Produit]
```

- ✓ Tester le package avec le bouton « Start » et vérifier si les données sont bien chargées.

4.3 CLIENT

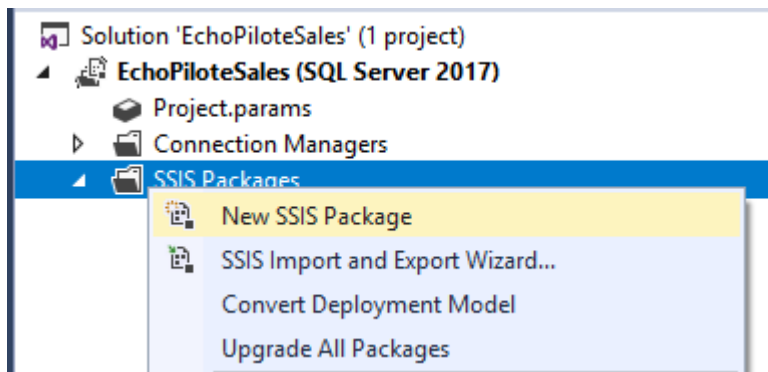
La dimension « Client » va être alimentée avec une transformation de dimension à variation lente. Ce type de transformation permet de gérer simultanément les modifications de valeur d'attribut et l'historisation. La tâche « Dimension à variation lente » va automatiquement générer d'autres tâches dans le flux de données en fonction de sa configuration.

Cette tâche utilise notamment les tâches de commande OLE DB qui peuvent avoir des performances assez mauvaises en cas de nombreux appels. On utilisera donc la « Dimension à variation lente » pour la mise à jour de table de dimension avec un faible taux de modification au cours du temps. Pour les tables avec plus de modification on préférera un chargement en « annule et remplace » ou un MERGE en T-SQL.

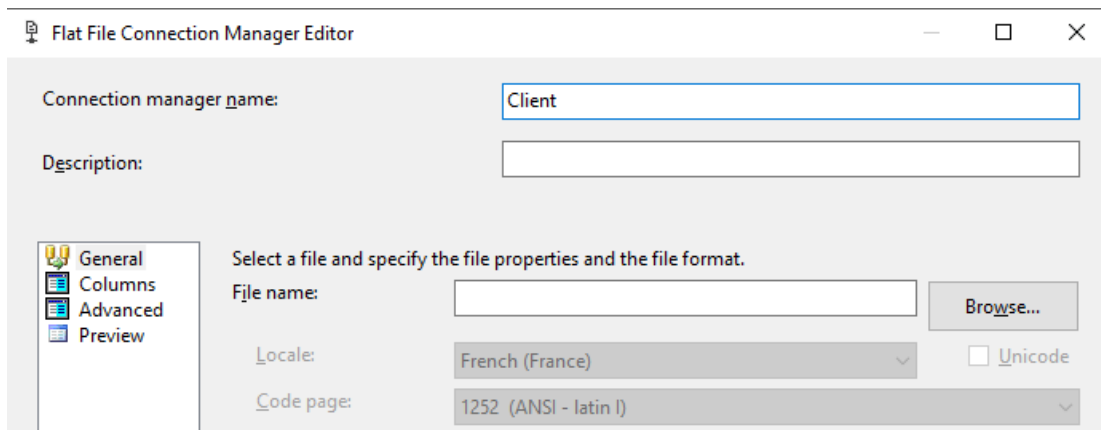


Créer et initialiser le package « Client.dtsx ».

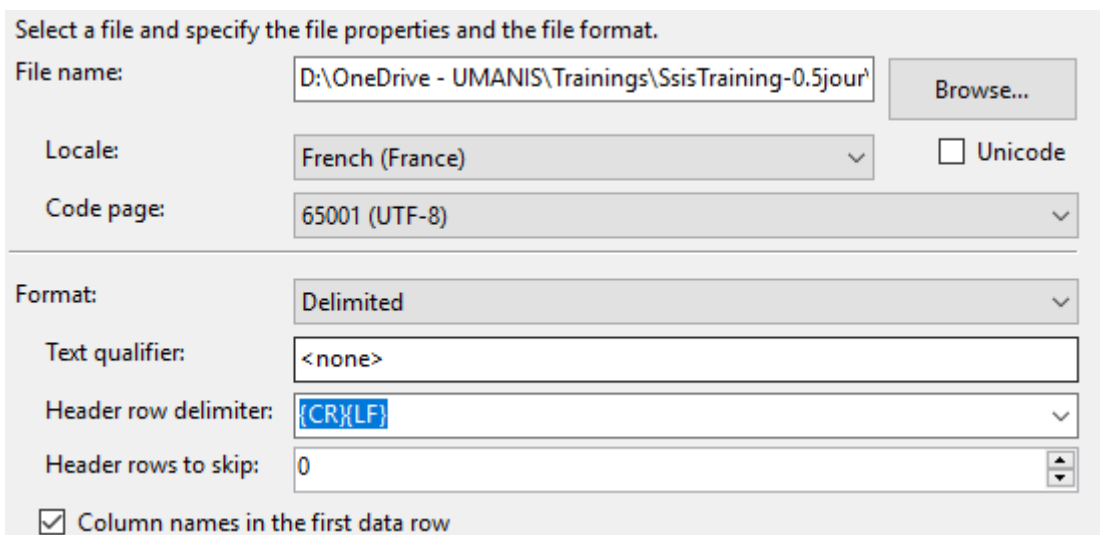
- ✓ Créer un nouveau package et le renommer : Client.dtsx



- ✓ Dans le flux de contrôle, ajouter un conteneur de séquence **Sequence Container** et le renommer : Client.
- ✓ Ajouter une tâche de flux de données au conteneur **Tâche de flux de données**.
- ✓ Ouvrir le flux de données (double-clic).
- ✓ Ajouter une source de fichier plat **Source du fichier plat** et l'ouvrir.
 - Pour la connexion à la source sélectionner « New... ».
 - Nommer la connexion « Client » :



- Utiliser « Browse... » pour naviguer jusqu'au fichier « Client.csv »
- Modifier l'encodage pour « 65001 (UTF-8) ». La connexion doit être configurée ainsi :




- Dans l'onglet « Advanced » modifier les types des colonnes :

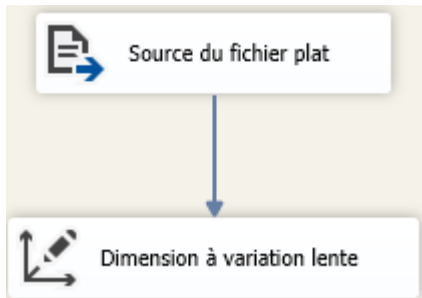
ClientDesc	DT_WSTR, 50
TypeClientID	DT_I4
VilleID	DT_I4

- Valider

- ✓ De retour dans le flux de données, ajouter une tâche de dimension à variation lente :


 Dimension à variation lente

- Créer une contrainte de précédence de la source vers la dimension à variation lente.



- Ouvrir la tâche « Dimension à variation lente »
- Sélectionner la connexion de projet « EchoPilote » et la table « Dbo.Client ».
- Définir la colonne « ClientDesc » comme Business key et laisser les autres inchangées. Valider avec « Next ».
- Définir le type de changement comme « Changing attribute » pour les deux autres colonnes de la table. Valider avec « Next ».
- Ne pas cocher l'option « Change all the matching records ». Valider avec « Next ».
- Décocher l'option « Enable inferred member support ». Valider avec « Next », puis « Finish ».

On souhaite à présent enregistrer le résultat du chargement pour pouvoir faire un suivi de l'alimentation. On va créer deux tâches SQL, une à exécuter en cas de réussite et l'autre en cas d'échec.

- ✓ Revenir dans le flux de contrôle du package « Client.dtsx ».
- ✓ Créer une variable de type texte nommée « tableName ». Mettre comme valeur « Client.csv ».
- ✓ Ajouter une tâche d'exécution de requêtes SQL  Tâche d'exécution de requêtes SQL et la renommer : « Load OK ». Ouvrir la tâche.
 - Configurer la page comme ci-dessous. La requête SQL à utiliser est :

```
update [ssisdbo].[zSourcesListe] set [lastUpdate] = GETDATE(), [lastUpdateStatus] = 'OK' where [tableName] = ?
```

▼ General	
Name	Load OK
Description	Màj de zSourcesListe
▼ Options	
TimeOut	0
CodePage	1252
TypeConversionMode	Allowed
▼ Result Set	
ResultSet	None
▼ SQL Statement	
ConnectionType	OLE DB
Connection	localhost.EchoPiloteSalesTest
SQLSourceType	Direct input
SQLStatement	update [dbo].[zSourcesListe] set [lastUpdate] =
IsQueryStoredProcedure	False
BypassPrepare	True

- La requête utilise un paramètre (« ? »), il faut donc le spécifier dans l'onglet « Parameter Mapping » :

General	Variable Name	Direction	Data Type	Parameter Name	Parameter Size
Parameter Mapping	User::tableName	Input	VARCHAR	@tableName	-1
Result Set					
Expressions					

- Valider la configuration de la tâche avec « OK ».
- ✓ De manière similaire à « Load OK », faire une d'exécution de requêtes SQL pour indiquer dans la table « zSourcesListe » si le chargement est en échec.
- ✓ Créer les contraintes de précédence entre le flux de données, « Load OK » et « Load KO ». La tâche « Load KO » doit être exécutée uniquement si la tâche de flux de données échoue.

On va également ajouter une tâche pour déplacer le fichier dans le répertoire de backup. Dans un premier temps on va configurer la connexion pour laquelle dépend du paramètre de projet « SourceFilesPath ».

- ✓ Dans le manager de connexions, sélectionner la connexion « Client » et presser F4 pour afficher ses propriétés.
- Dans la propriété « Expressions » sélectionner le bouton « ... »



- Dans l'éditeur des expressions de propriétés, ajouter une propriété « ConnexionString ». Utiliser « ... » pour modifier la formule :

Expression:

```
@[$Project::SourceFilesPath] + "\\Client.csv"
```

- La formule peut être testée avec le bouton « Evaluate Expression ».
- Valider les différentes fenêtres pour revenir au flux de contrôle.

- ✓ La connexion « Client » dépend à présent du paramètre de projet « SourceFilePath »
- ✓ Sélectionner à nouveau la connexion « Client » et la dupliquer (Ctrl+C, Ctrl+V).
 - Sélectionner la nouvelle connexion, et la renommer « ClientBack ».
 - Comme pour la connexion « Client », modifier l'expression «ConnectionString» de « ClientBack » pour qu'il pointe vers un fichier Client, postfixé de la date et heure de chargement, dans le sous-répertoire « Backup ». On utilisera les fonctions d'expressions SQL disponible dans

Expression Builder

Specify the expression for the property: ConnectionString.

+ Variables and Parameters

+ Mathematical Functions
+ String Functions
- Date/Time Functions

fx DATEADD(«datepart», «number», «date»)
fx DATEDIFF(«datepart», «startdate», «enddate»)
fx DATEPART(«datepart», «date»)
fx DAY(«date»)
fx GETDATE()
fx GETUTCDATE()
fx MONTH(«date»)

Description:
Returns the current date of the system.

Expression:

```
@[$Project::SourceFilePath] + "\\Backup\\Vente." + Replace( Replace(Left((DT_WSTR, 50)getdate(), 19),":","."),",",".") + ".csv"
```

Evaluated value:

```
D:\OneDrive - UMANIS\Trainings\Ssis Training-0.5jour\EchoPiloteSales\EchoPilotesSalesSources\Backup\Vente.2021-03-31.21.58.02.csv
```

Evaluate Expression OK Cancel

- ✓ De retour dans le flux de contrôle, ajouter une tâche de système de fichiers.
 - Ouvrir la tâche et la configurer pour que le fichier de la connexion « Client » soit déplacé selon la connexion « ClientBak ».

▼ Destination Connection	
IsDestinationPathVariable	False
DestinationConnection	ClientBack
OverwriteDestination	True
▼ General	
Name	Backup
Description	Tâche de système de fichiers
▼ Operation	
Operation	Move file
▼ Source Connection	
IsSourcePathVariable	False
SourceConnection	Client

- Créer une contrainte de précedence entre la tâche de flux de données et la tâche de backup. Le backup doit être fait après le flux de données quel que soit le résultat de l'opération.

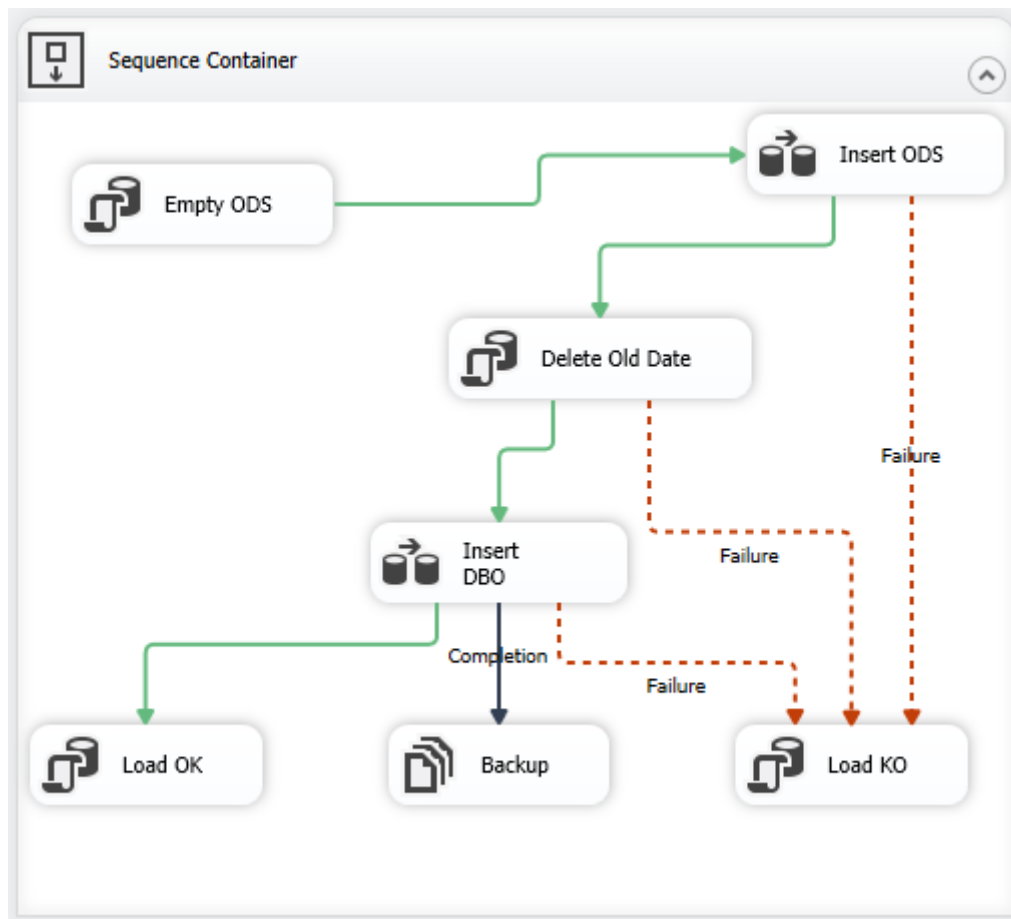
4.4 VENTES

Le package des ventes va permettre d'intégrer les données dans la table de fait. Les transformations à appliquer sur la donnée nécessitent de faire une intégration en deux étapes :

- Chargement des données dans une table d'ODS.
- Mise à jour du Datawarehouse à partir des données de l'ODS.

On sait que le système source revoit toujours des périodes complètes : **si une valeur est modifiée dans le système source il revoit toutes les valeurs à la même date que la valeur modifiée**. Il faut donc faire attention lors de l'intégration à ne pas charger de données en doublon et à prendre en compte les modifications éventuelles.


On va utiliser une tâche de Look-Up pour enrichir les données brutes avec des données de la table « Client » afin d'avoir certaines informations client directement dans la table de faits



On commence par créer un nouveau package :

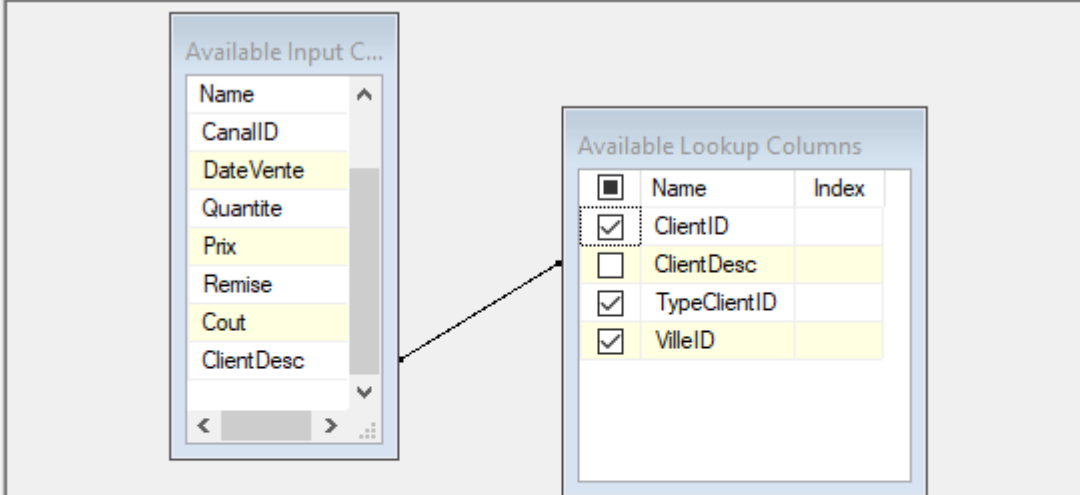
- ✓ Créer un nouveau package et le renommer : Vente.dtsx
- ✓ Ajouter un conteneur de séquences.
 - Dans le conteneur de séquence créer une tâche SQL pour vider la table [ods].[vente].

General	▼ General
Parameter Mapping	Name Empty ODS
Result Set	Description Tâche d'exécution de requêtes SQL
Expressions	▼ Options
	TimeOut 0
	CodePage 1252
	TypeConversionMode Allowed
	▼ Result Set
	ResultSet None
	▼ SQL Statement
	ConnectionType OLE DB
	Connection localhost.EchoPiloteSalesTest
	SQLSourceType Direct input
	SQLStatement truncate table [ods].[Vente]
	IsQueryStoredProcedure False
	BypassPrepare True

- ✓ Ajouter ensuite un flux de données pour charger les données depuis le fichier « Ventes.csv » vers la table [ods].[vente].
 - Dans le flux de données, créer une source de fichier plat. La configurer et créer le gestionnaire de connexion vers le fichier « Ventes.csv ». Dans ce fichier **la première colonne est du texte (DT_WSTR) et les suivantes sont des entiers (DT_I4)**.
 - Ajouter à la suite de la source de données une tâche de recherche  Recherche . Et la configurer ainsi :

Cache Mode	Partial cache
Connexion type	OleDb
No matching rows handling	Fail component
OLE DB Connection Manager	EchoPiloteSalesTest
Use table or view	Dbo.Client

- Dans la page des colonnes, faire un glisser-déposer entre les noms de colonnes « Client Desc » des deux tables pour indiquer que le look-up se fait sur ces colonnes.
- Cocher les colonnes « ClientID », « TypeClientID », « VilleID » pour qu'elles soient ajoutées au flux de données.



Lookup Column	Lookup Operation	Output Alias
ClientID	<add as new column>	ClientID
VilleID	<add as new column>	VilleID
TypeClientID	<add as new column>	TypeClientID

- Ajouter une tâche de colonne dérivée pour ajouter les colonnes :
 - « DateVenteTyped » qui correspond à la colonne « DateVente » converti en type date.
 - « Date » qui correspond à la date-heure du chargement.

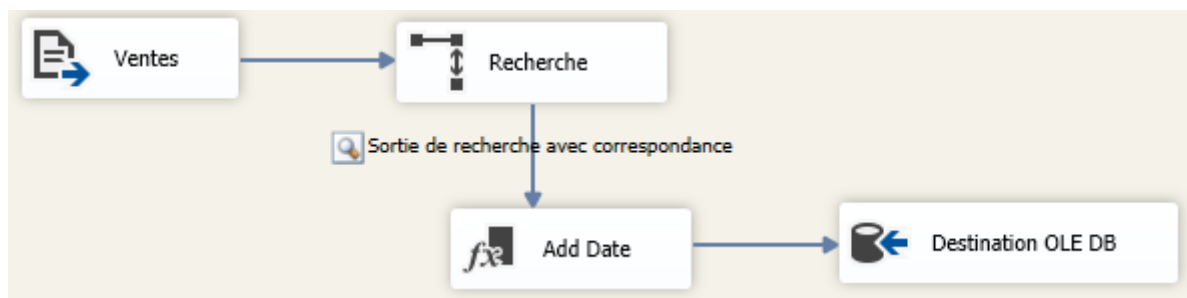
Derived Column Name	Derived Column	Expression	Data Type
DateVenteTyped	<add as new column>	(DT_DATE)(SUBSTRING((DT_WSTR,8)DateVente,1,4) + "-" + SUBSTRING((DT_WSTR,8)DateVente,5,2) + "-" + SUBSTRING((DT_WSTR,8)DateVente,7,2))	date [DT_DATE]

(DT_DATE)(SUBSTRING((DT_WSTR,8)DateVente,1,4) + "-" + SUBSTRING((DT_WSTR,8)DateVente,5,2) + "-" + SUBSTRING((DT_WSTR,8)DateVente,7,2))

Date	<add as new column>	GETDATE()	horodateur base de données [DT_DBTIMESTAMP]
------	---------------------	-----------	---

- Enfin, ajouter une tâche de destination OLE DB pour charger le flux de données dans la table ods.Ventes. Configurer la correspondance des colonnes pour charger les nouvelles colonnes :

Input Column	Destination Column
ClientID	ClientID
ProduitID	ProduitID
CommercialID	CommercialID
CanalID	CanalID
DateVenteTyped	DateVente
Quantite	Quantite
Prix	Prix
Remise	Remise
Cout	Cout



Plutôt que de modifier les lignes à mettre à jour dans le Datawarehouse, on va supprimer ces lignes avant de charger l'intégralité de l'ODS.

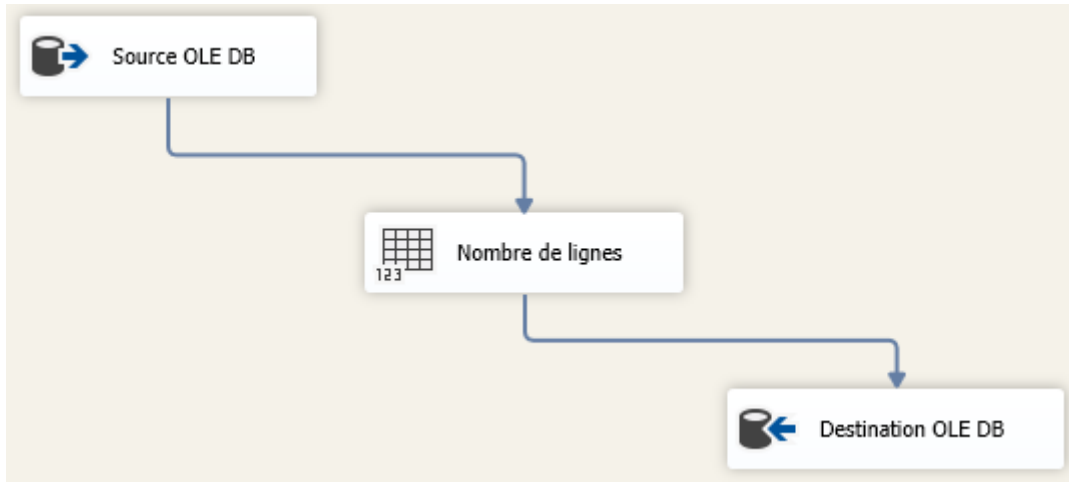
- ✓ Ajouter une tâche SQL pour supprimer de la table [ssisdbo].[vente] toutes les lignes avec des dates contenues dans l'ODS.

General Parameter Mapping Result Set Expressions	<table border="1"> <tr> <td colspan="2"> General Name: Delete Old Date Description: Tâche d'exécution de requêtes SQL </td> </tr> <tr> <td colspan="2"> Options TimeOut: 0 CodePage: 1252 TypeConversionMode: Allowed </td> </tr> <tr> <td colspan="2"> Result Set ResultSet: None </td> </tr> <tr> <td colspan="2"> SQL Statement ConnectionType: OLE DB Connection: localhost.EchoPiloteSalesTest SQLSourceType: Direct input SQLStatement: delete from dfrom [dbo].[Vente] dinner join (se IsQueryStoredProcedure: False BypassPrepare: True </td> </tr> </table>	General Name: Delete Old Date Description: Tâche d'exécution de requêtes SQL		Options TimeOut: 0 CodePage: 1252 TypeConversionMode: Allowed		Result Set ResultSet: None		SQL Statement ConnectionType: OLE DB Connection: localhost.EchoPiloteSalesTest SQLSourceType: Direct input SQLStatement: delete from dfrom [dbo].[Vente] dinner join (se IsQueryStoredProcedure: False BypassPrepare: True	
General Name: Delete Old Date Description: Tâche d'exécution de requêtes SQL									
Options TimeOut: 0 CodePage: 1252 TypeConversionMode: Allowed									
Result Set ResultSet: None									
SQL Statement ConnectionType: OLE DB Connection: localhost.EchoPiloteSalesTest SQLSourceType: Direct input SQLStatement: delete from dfrom [dbo].[Vente] dinner join (se IsQueryStoredProcedure: False BypassPrepare: True									

```

delete from t
from [ssisdbo].[Vente] t
inner join (SELECT distinct [DateVente] FROM [ssisods].[Vente]) s on s.[DateVente] = t.[DateVente]
  
```

- ✓ Ajouter ensuite un flux de données pour charger les données depuis la table [ods].[vente] vers la table [ssisdbo].[ventes]. Lors du chargement, il faut enregistrer le nombre de lignes dans une variable « NbInsert ».




- ✓ Comme pour les packages précédents, ajouter deux tâche « LoadOK » et « LoadKO » pour enregistrer le résultat de l'exécution, et une tâche de backup de fichier.

5 AMELIORATION DES PACKAGES

5.1 TEST DE L'EXISTENCE DU FICHIER

On va ajouter une étape de test au package Client pour qu'il ne soit pas KO si le fichier source n'existe pas.

- ✓ Ouvrir le package « Client.dtsx »
- ✓ Créer une variable « fichierExiste » de type Int32 avec comme valeur initiale « 0 ».
- ✓ Ajouter au flux de contrôle une tâche de script :  Tâche de script.
- ✓ Ouvrir la tâche de script.
 - Vérifier que le langage utilisé est bien « Microsoft Visual C# 2017 ».
 - Dans l'encart « ReadOnlyVariables », sélectionner le paramètre projet « SourceFilePath ».
 - Dans l'encart « ReadWriteVariables », sélectionner la variable « fichierExiste ».

Script	
ScriptLanguage	Microsoft Visual C# 2017
EntryPoint	Main
ReadOnlyVariables	\$Project::SourceFilePath
ReadWriteVariables	User::fichierExiste

- Sélectionner le bouton « Modifier le script » pour ouvrir l'éditeur de script.
- Dans le bloc « region Namespaces », ajouter l'instruction « `using System.IO;` ».
- Au niveau du commentaire « `// TODO: Add your code here` » ajouter le code suivant :

```

string filePath =
Dts.Variables["$Project::SourceFilePath"].Value.ToString() +
"\\client.csv";

if (File.Exists(filePath))
{
    Dts.Variables["User::fichierExiste"].Value = 1;
}

public void Main()
{
    // TODO: Add your code here
    string filePath = Dts.Variables["$Project::SourceFilePath"].Value.ToString() + "\\client.csv";
    if (File.Exists(filePath))
    {
        Dts.Variables["User::fichierExiste"].Value = 1;
    }

    Dts.TaskResult = (int)ScriptResults.Success;
}

```

- Enregistrer le script et fermer l'éditeur de script.
- De retour dans la fenêtre de configuration de la tâche, valider avec le bouton « OK ».
- ✓ Ajouter une contrainte de précédence de la tâche de script sur le conteneur de séquence.
- ✓ Double-cliquer sur la contrainte de précédence pour ouvrir la page de configuration et paramétrer ainsi :

Options de contrainte

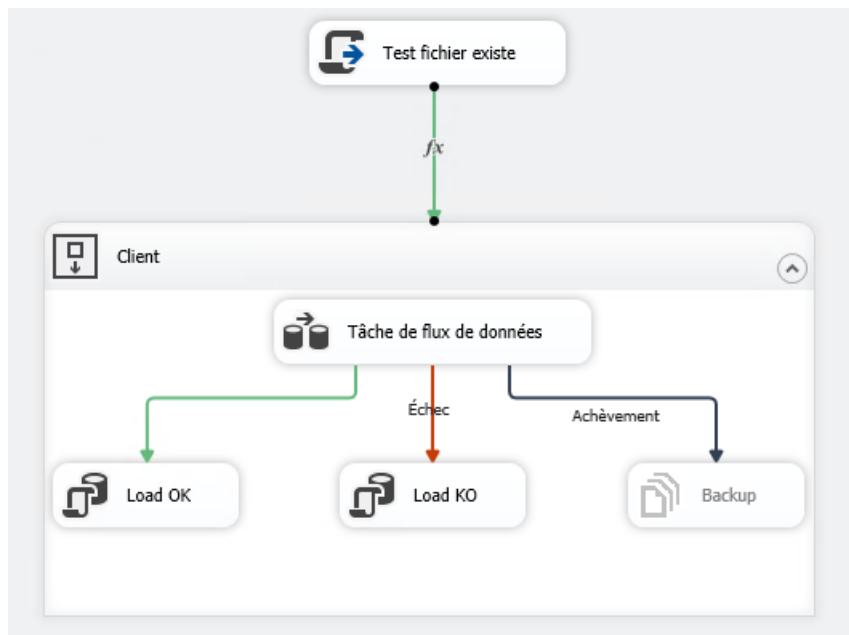
Opération d'évaluation : Expression et contrainte

Valeur : Succès

Expression : @[User::fichierExiste] == 1

... Tester

Le package doit ressembler à ça :



5.2 BOUCLE SUR LES FICHIERS DE VENTES.

- ✓ Ouvrir le package Vente.dtsx
- ✓ Dans le package, créer une variable « FichierVente » de type « String » avec comme valeur le nom du fichier initial :

Variables			
Name	Scope	Data type	Value
FichierVente	Vente	String	Ventes.csv

- ✓ Dans les connections « Ventes » et « VentesBack », modifier l'expression de la chaîne de connexion pour utiliser la variable « FichierVente » à la place du nom de fichier en dur.

Expression:

```
@[$Project::SourceFilePath] + "\\\" + @[User::FichierVente]
```

Expression:

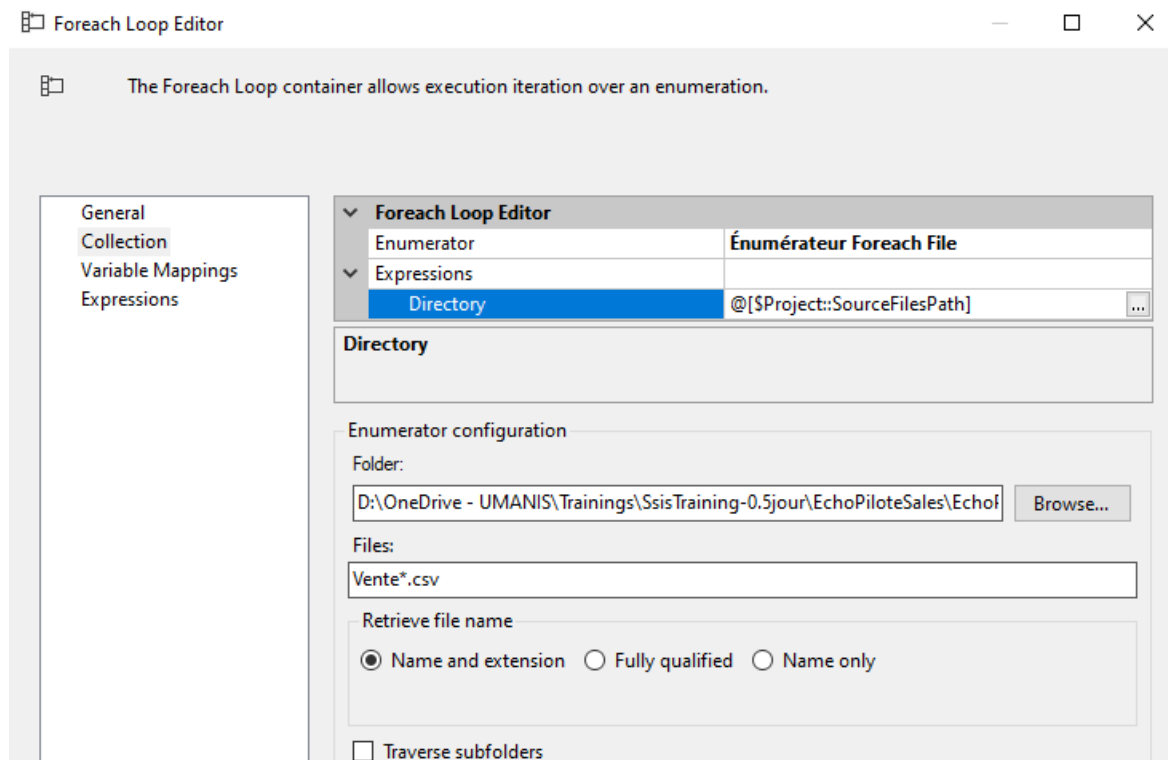
```
@[$Project::SourceFilePath] + "\\Backup\\"
+ Replace( @[User::FichierVente] , ".csv", Replace( Replace( Left((DT_WSTR, 50)getdate(), 19), ":", "."), " ", ".") + ".csv")
```

Evaluated value:

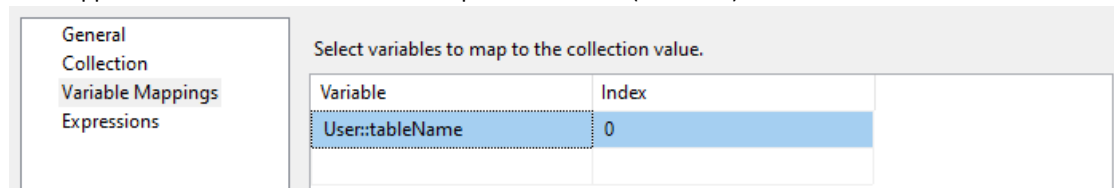
```
D:\OneDrive - UMANIS\Trainings\SsisTraining-0.5jour\EchoPiloteSales\EchoPiloteSalesSources\Backup\Ventes2022-04-07.18.47.26.csv
```

- ✓ Ajouter au package un conteneur boucle Foreach. Le configurer pour qu'il parcoure chaque fichier d'un répertoire en renseignant le répertoire des fichiers sources :
 - Sélectionner le type d'énumérateur : « Énumérateur Foreach File ».
 - Dans le menu « Expressions » juste sous le type d'énumérateur, ajouter une expression pour la propriété « Directory » ; la propriété doit prendre la valeur du paramètre « SourceFilePath »

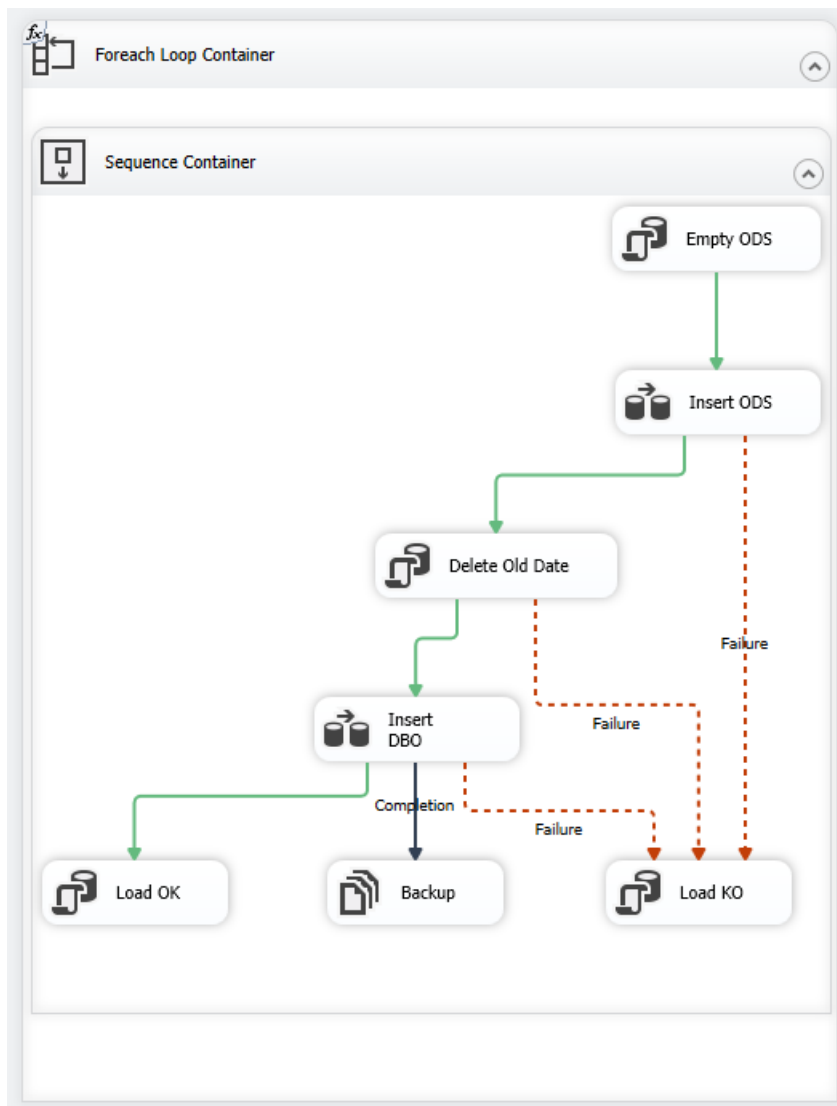
- Et en indiquant comme masque de fichier : « Vente*.csv ».



- ✓ Mapper la variable « FichierVente » à la première valeur (index = 0) de la boucle.



- ✓ Sélectionner le conteneur de séquence qui contient toutes les tâches, et le faire glisser dans le conteneur de boucle Foreach.

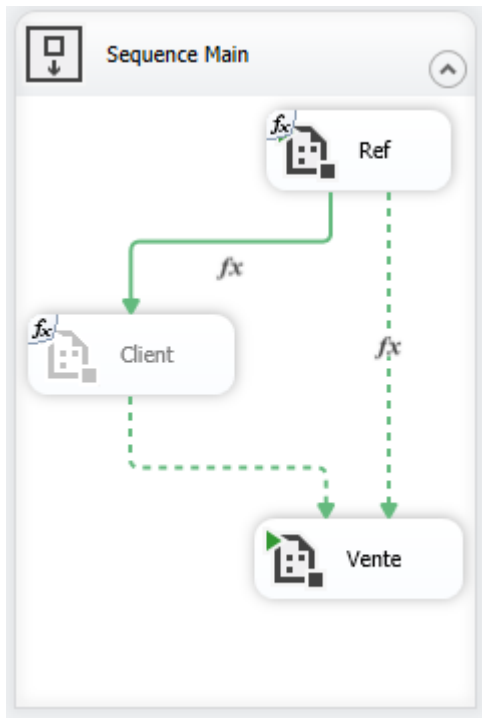



✓

6 PACKAGES TECHNIQUES

6.1 MAIN

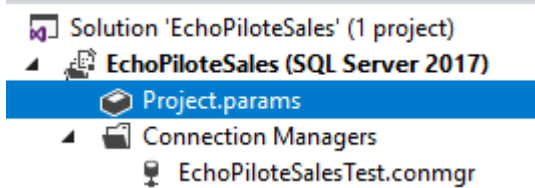
Le package principal permet de coordonner l'alimentation en ordonnant l'exécution des autres packages. Cet ordonnancement peut être statique : les packages sont appelés dans des tâches d'exécution de package successives ; ou dynamique : les packages sont exécutés en fonction d'une liste configurée en base de données. Cette deuxième solution, quoique plus longue à développer au départ, donne une meilleure maintenabilité.



- ✓ Créer un package nommé « Main.dtsx ».
- ✓ Utiliser la tâche d'exécution de package  Tâche d'exécution de package pour appeler successivement les packages « Canal », « Produit », « Client » et « Vente ».

Il est possible de conditionner l'exécution des package en fonction de paramètres de projet.

- ✓ Ouvrir les paramètres de projet.



- Ajouter deux paramètres de type booléen : « fait » et « ref ».
- ✓ Dans le package « Main.dtsx », pour chaque tâche d'exécution de package, modifier l'expression de la propriété « Disable » pour activer ou désactiver les tâches en fonction des paramètres.

6.2 SEQUENCE SSAS

Il est possible d'utiliser des tâches SSIS pour gérer la maintenance d'un cube SSAS. Un package peut exécuter du code XMLA pour modifier la structure du cube (par exemple : ajout de partition) ou pour recharger les données du cube.

