Лабораторна робота № 10-11

Дослідження роботи SSIS

Мета: освоїти інструментарій інтеграції даних корпоративних інформаційних систем

ПО і інструменти: необхідно встановити SSDT (SSQTBI), поддежівающіе наступні версії SQL Server: SQL Server 2019 року, SQL Server 2019 SQL.

Теоретичні відомості

SSIS - це інструмент, який дозволяє в зручному вигляді реалізувати інтеграцію, тобто реалізувати процес перенесення даних з одного джерела в інший. Цей процес іноді називають ETL (від англ. Extract, Transform, Load - дослівно «витяг, перетворення, завантаження»).

Необхідні інструменти для вивчення SSIS

SSIS буде розглядатися на прикладі SQL Server 2019 Developer Edition.

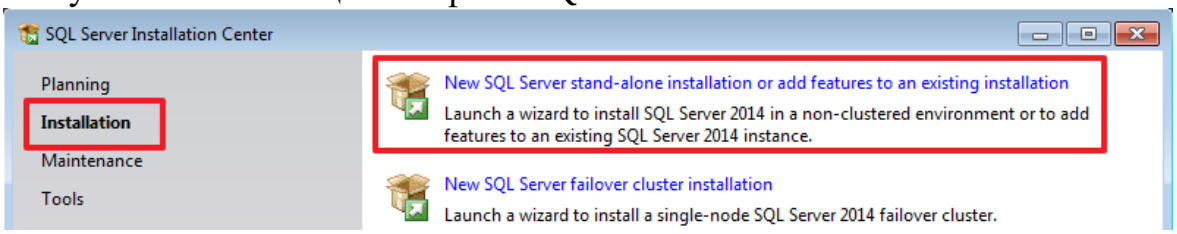
Служби Integration Services доступні SQL Server

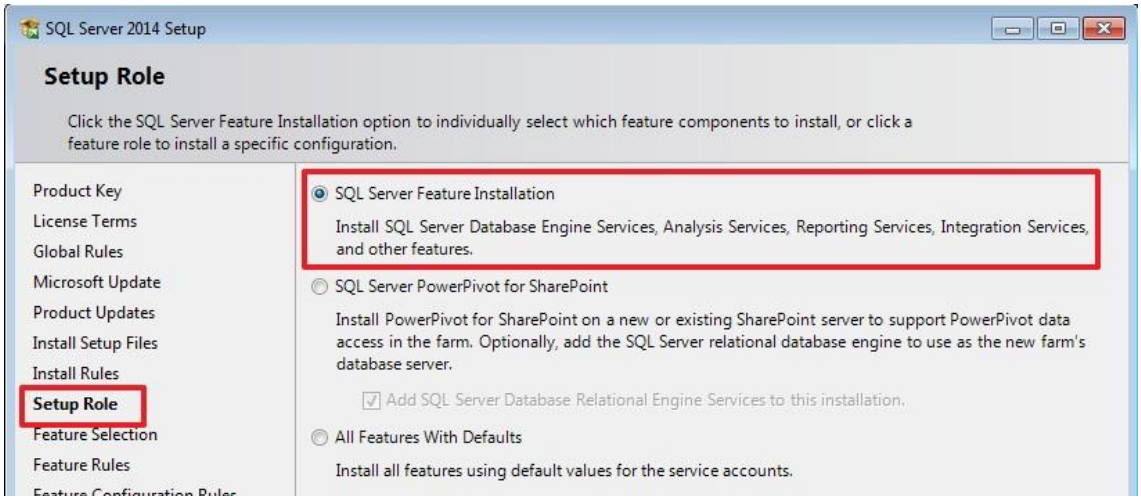
Додатково необхідно буде завантажити і встановити інструмент розробникаSQL Server Data Tools (SSDT).

SSDT - це розширення для Visual Studio, яке дозволить створювати проектинеобхідного нам типу.

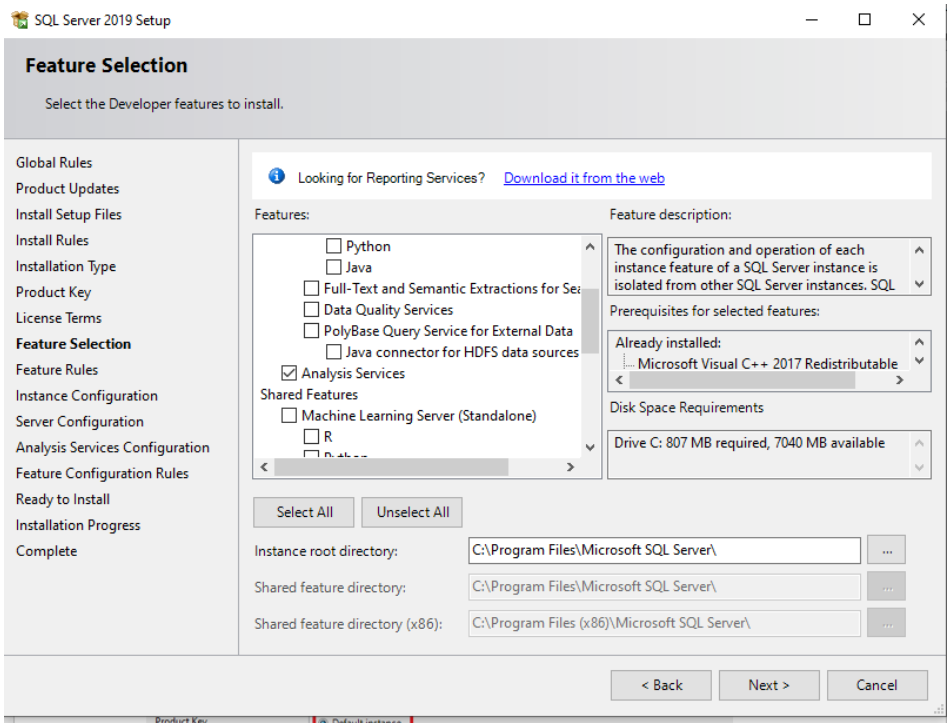
Встановлення SQL Server і SSDT

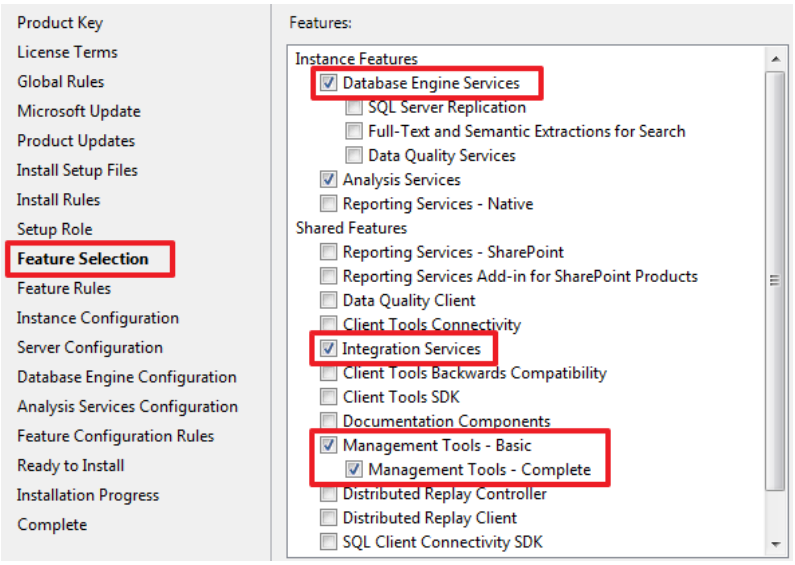
Запускаємо інсталяційний файл SQL Server 2019:





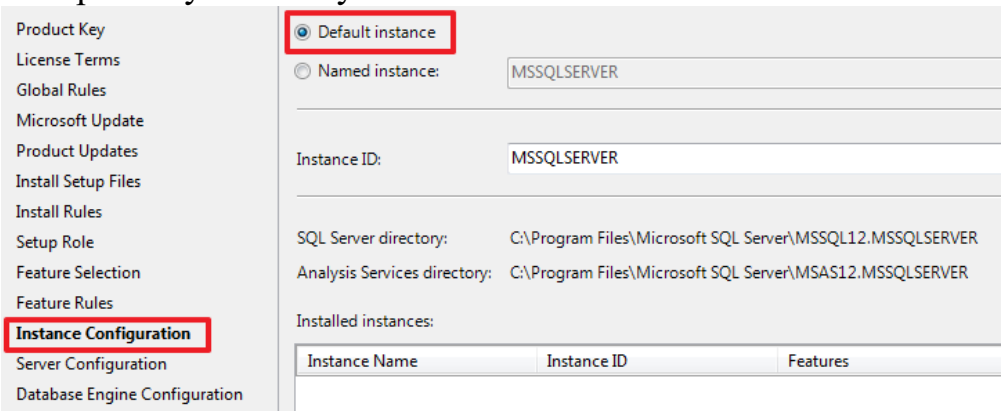
Для роботи SSIS досить буде вибрати такі компоненти:



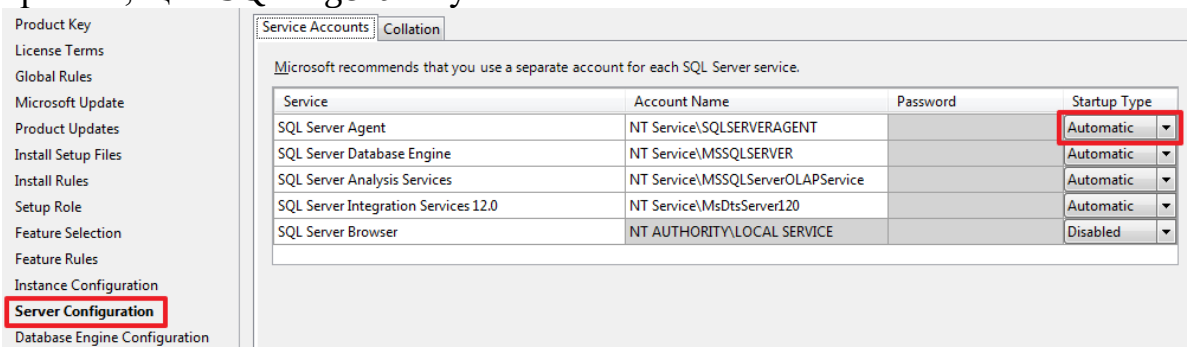


Оскільки мені в подальшому знадобиться Analysis Services (SSAS), то я відзначив і його, якщо він вам не потрібен ви можете не вибирати даний компонент.

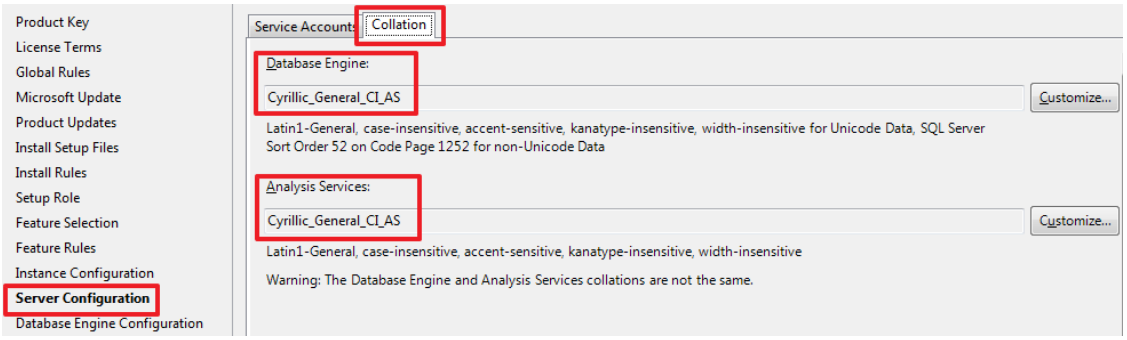
У мене немає інших встановлених SQL Server, і я зроблю цей екземпляр використовується за умовчанням:



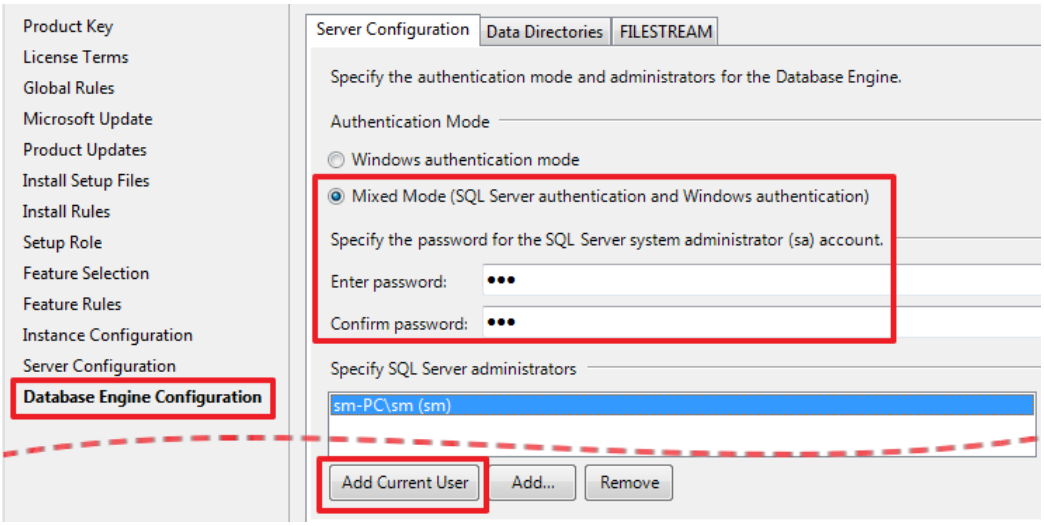
Зроблю, щоб SQL Agent запускався автоматично:



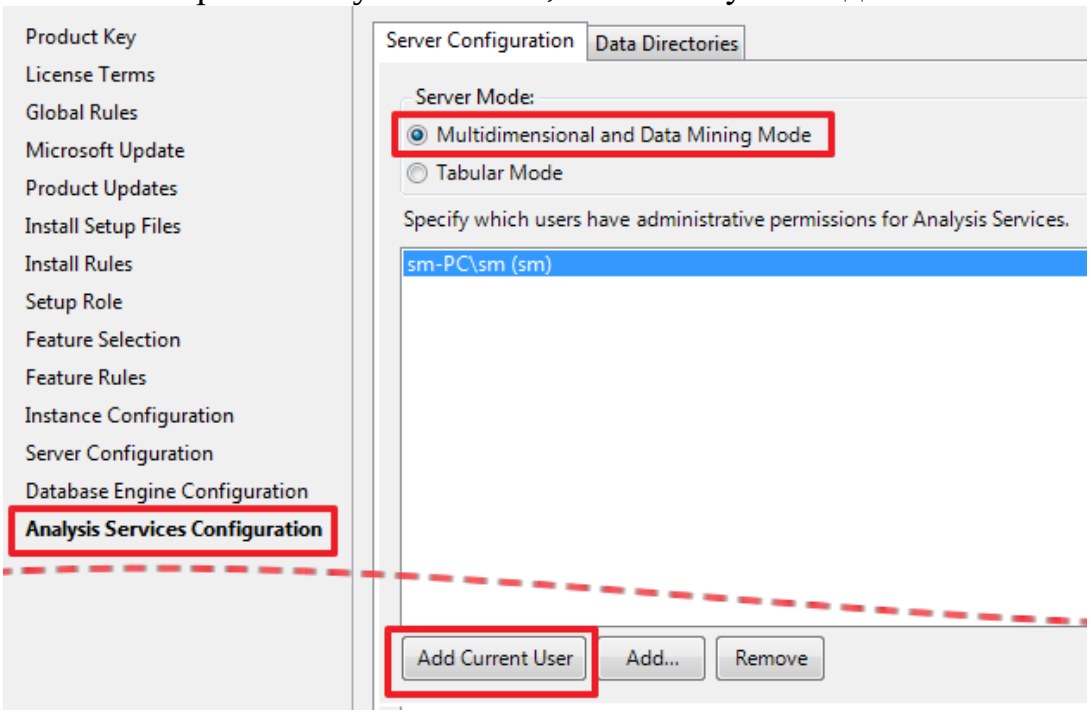
При необхідності можна змінити Collation, який буде використовуватися за замовчуванням:



Встановлю змішаний режим аутентифікації, вказавши свій пароль для користувача sa:



Оскільки обрано Analysis Services, то налаштування для нього:



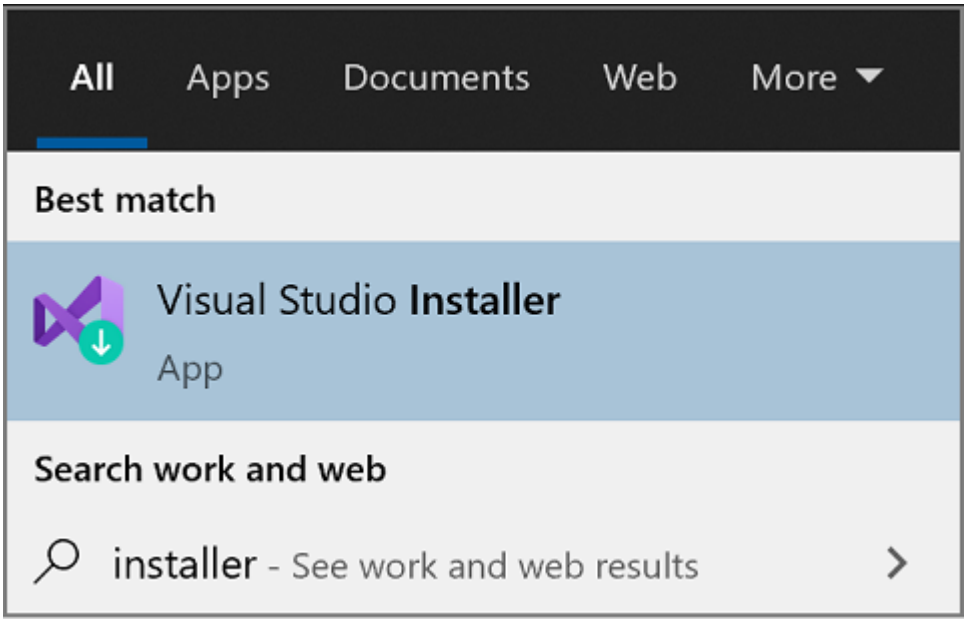
Натискаючи Next і Install запускаємо установку SQL Server і його компонент.Так як у мене на комп'ютері всього один диск, то все директорії я залишив за замовчуванням, при необхідності ви можете змінити їх на більш зручні.

Наступним кроком встановимо SSDT - це розширення для Visual Studio, якедасть нам можливість створювати проекти SSIS.

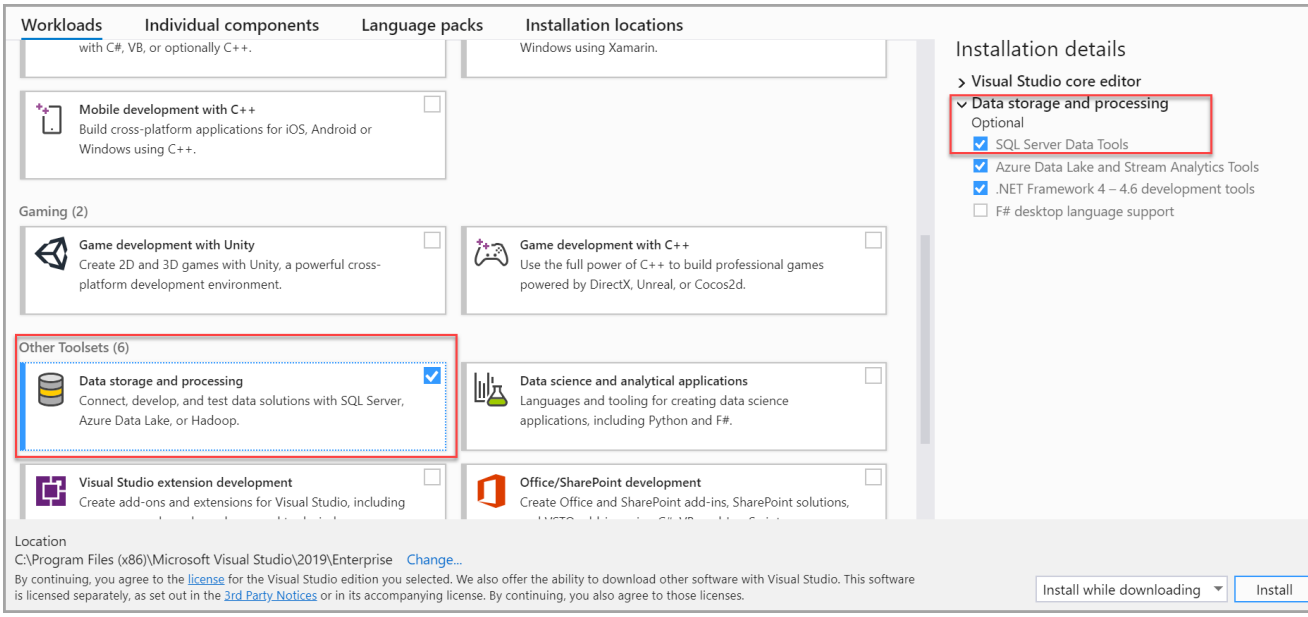
Встановлення SSDT в Visual Studio 2019

Щоб змінити встановлені робочі навантаження Visual Studio для включення SSDT, потрібно використовувати Visual Studio Installer.

1. Запустити Visual Studio Installer.



1. Вибрати Visual Studio, в якій необхідно додати SSDT, а потім вибрати **Змінити**.
2. Вибрати **SQL Server Data Tools** в розділі **Зберігання і обробка даних** в списку рабочих навантажень.



Завдання на лабораторну роботу

1. Реалізувати інтеграцію 2 джерел даних і імпорт даних в БД.
2. Індивідуальне завдання. Реалізувати заповнення БД з альтернативного джерела (таблиці Excelілі MySQLServer)

Хід роботи

Створення демонстраційних баз даних
Запустимо SQL Server Management Studio (SSMS) і за допомогою скрипта створимо 3 бази даних - перші дві (DemoSSIS_SourceA і DemoSSIS_SourceB) виступатимуть в ролі джерел даних, а третя (DemoSSIS_Target) в ролі одержувача даних:

```
-- перша БД виступає в ролі джерела даних
CREATE DATABASE DemoSSIS_SourceA
GO

ALTER DATABASE DemoSSIS_SourceA SETRECOVERY
SIMPLEGO

-- друга БД виступає в ролі джерела даних
CREATE DATABASE DemoSSIS_SourceB
GO

ALTER DATABASE DemoSSIS_SourceB SETRECOVERY
SIMPLEGO

-- БД виступає в ролі одержувача даних
CREATE DATABASE
DemoSSIS_TargetGO

ALTERDATABASE DemoSSIS_Target SETRECOVERY SIMPLE
GO
```

У базах джерелах створимо тестові таблиці і наповнимо їх тестовими даними:

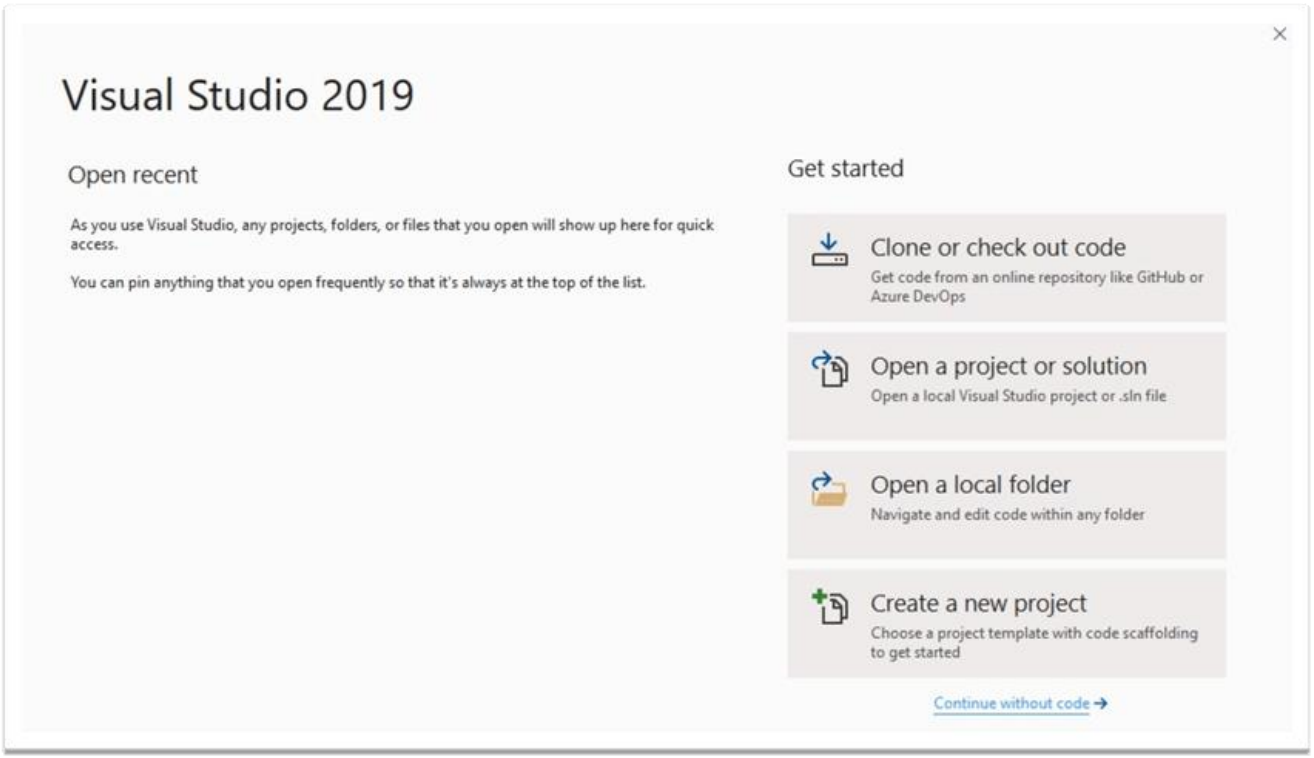
```
USE DemoSSIS_SourceA
GO

-- продукти з джерела A
CREATETABLEProducts(
```

```
IDintNOTNULLIDENTITY,
  Title nvarchar(50) NOTNULL,
  Price money,
CONSTRAINT PK_Products PRIMARY KEY(ID)
)
GO

-- наповнюємо таблицю тестовими даними
SET IDENTITY_INSERT Products ON

INSERT Products(ID,Title,Price)VALUES
(1,N'Клей',20),
(2,N'Корректор',NULL),
(3,N'Скотч',100),
(4,N'Стикеры',80),
(5,N'Скрепки',25)

SET IDENTITY_INSERT Products OFF
GO


USE DemoSSIS_SourceB
GO

-- продукти з джерела B
CREATETABLEProducts(
IDintNOTNULLIDENTITY,
  Title nvarchar(50) NOTNULL,
  Price money,
CONSTRAINT PK_Products PRIMARY KEY(ID)
)
GO

-- наповнюємо таблицю тестовими даними
SET IDENTITY_INSERT Products ON

INSERT Products(ID,Title,Price)VALUES
(1,N'Ножницы',200),
(2,N'Ножканцелярский',70),
(3,N'Дырокол',220),
(4,N'Степлер',150),
(5,N'Шариковаяручка',15)

SET IDENTITY_INSERT Products OFF
GO
```

Створимо таблицю в приймаючій базі:

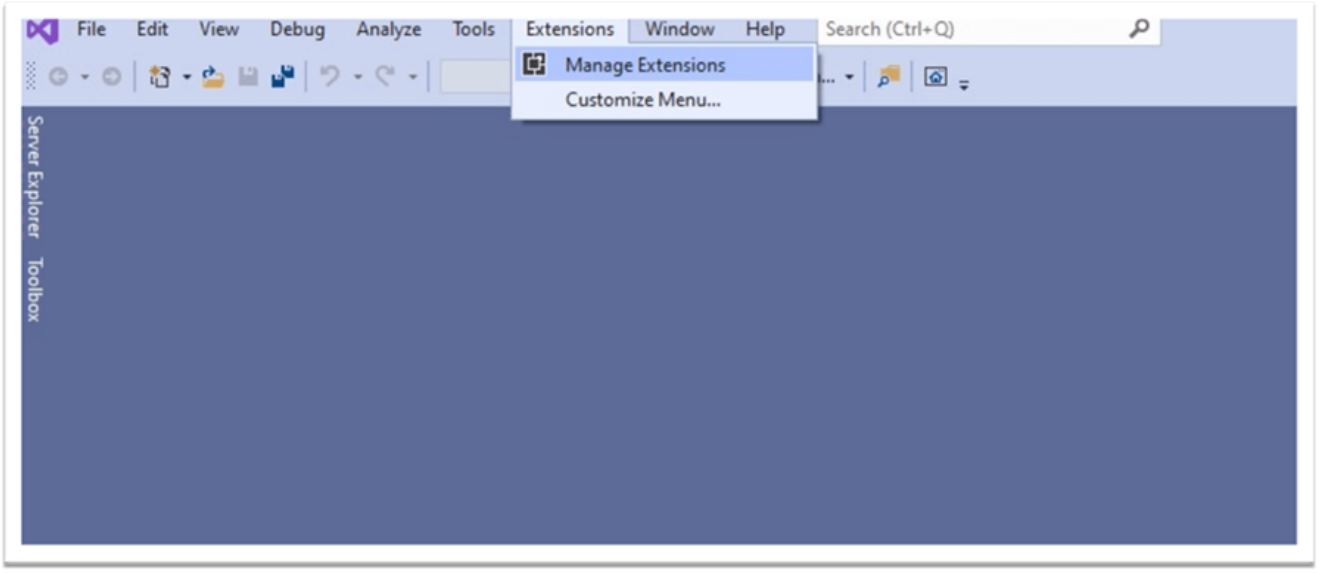
```
USE DemoSSIS_Target
GO

-- приймаюча таблиця
CREATE TABLE Products(
  ID int NOT NULL IDENTITY,
  Title nvarchar(50) NOT NULL,
  Price money,
  SourceID char(1) NOT NULL, -- використовується для ідентифікації джерела
  SourceProductID int NOT NULL, -- ID в джерелі
CONSTRAINT PK_Products PRIMARY KEY(ID),
CONSTRAINT UK_Products UNIQUE(SourceID,SourceProductID),
CONSTRAINT CK_Products_SourceID CHECK(SourceID IN('A','B'))
 )
```

Створення SSIS проекту
Запустимо Visual Studio 2012 і виберемо один з видів пропонованих нам налаштувань середовища

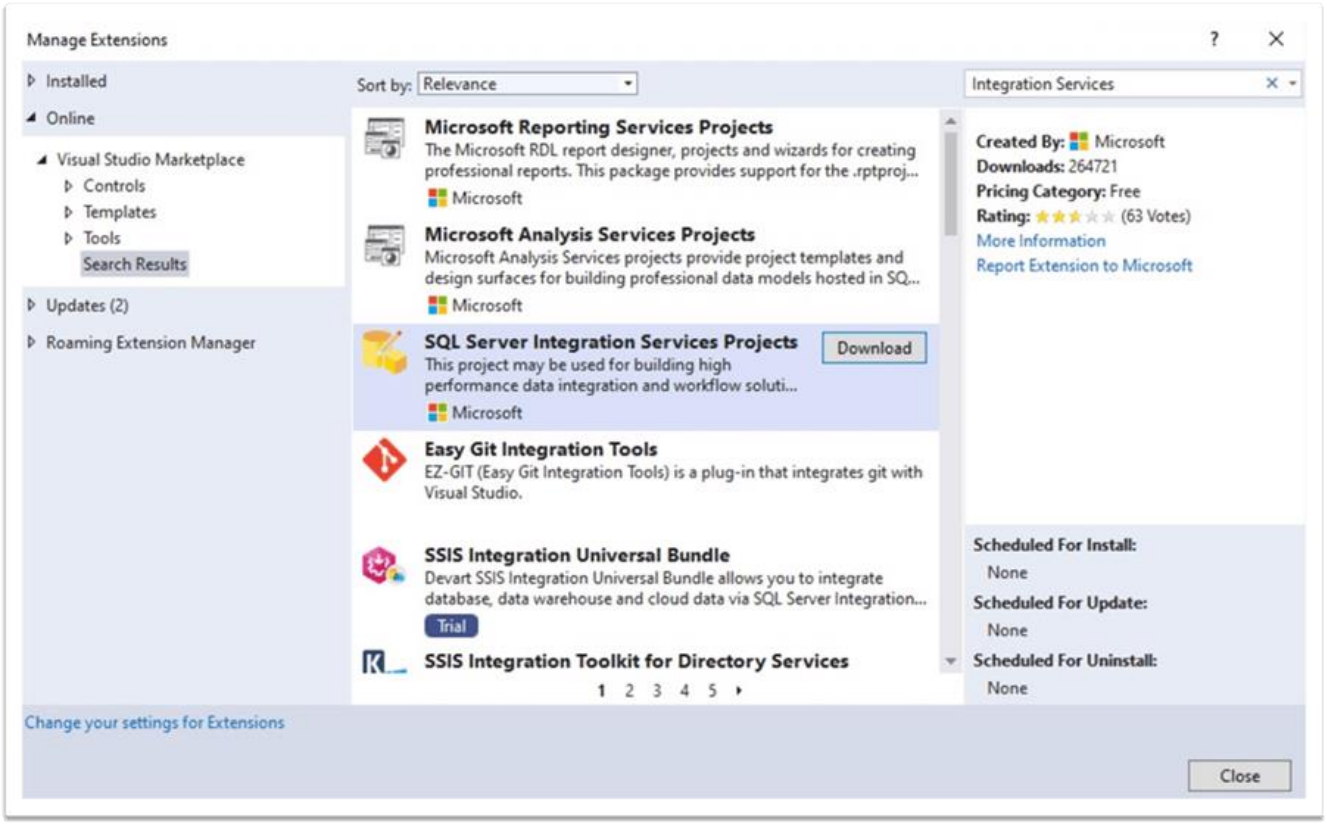


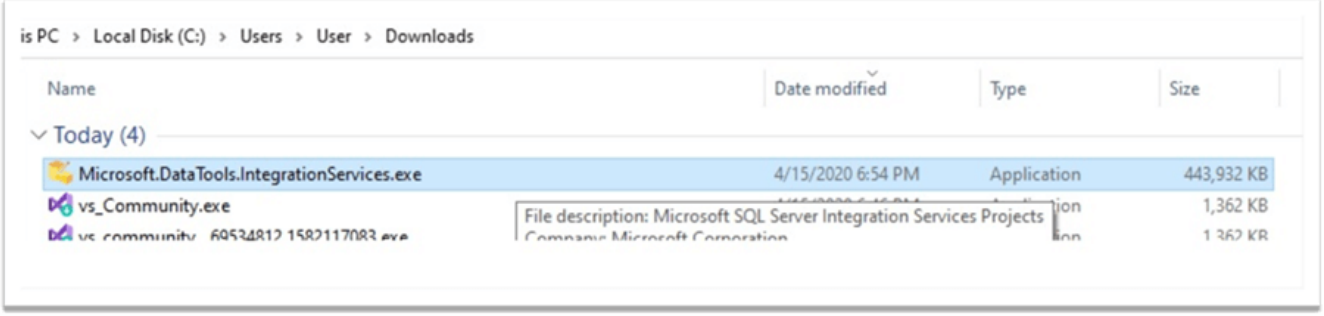
Створимо новий проект (**File -> New -> Project…**):

Extensions" > "Manage Extensions"

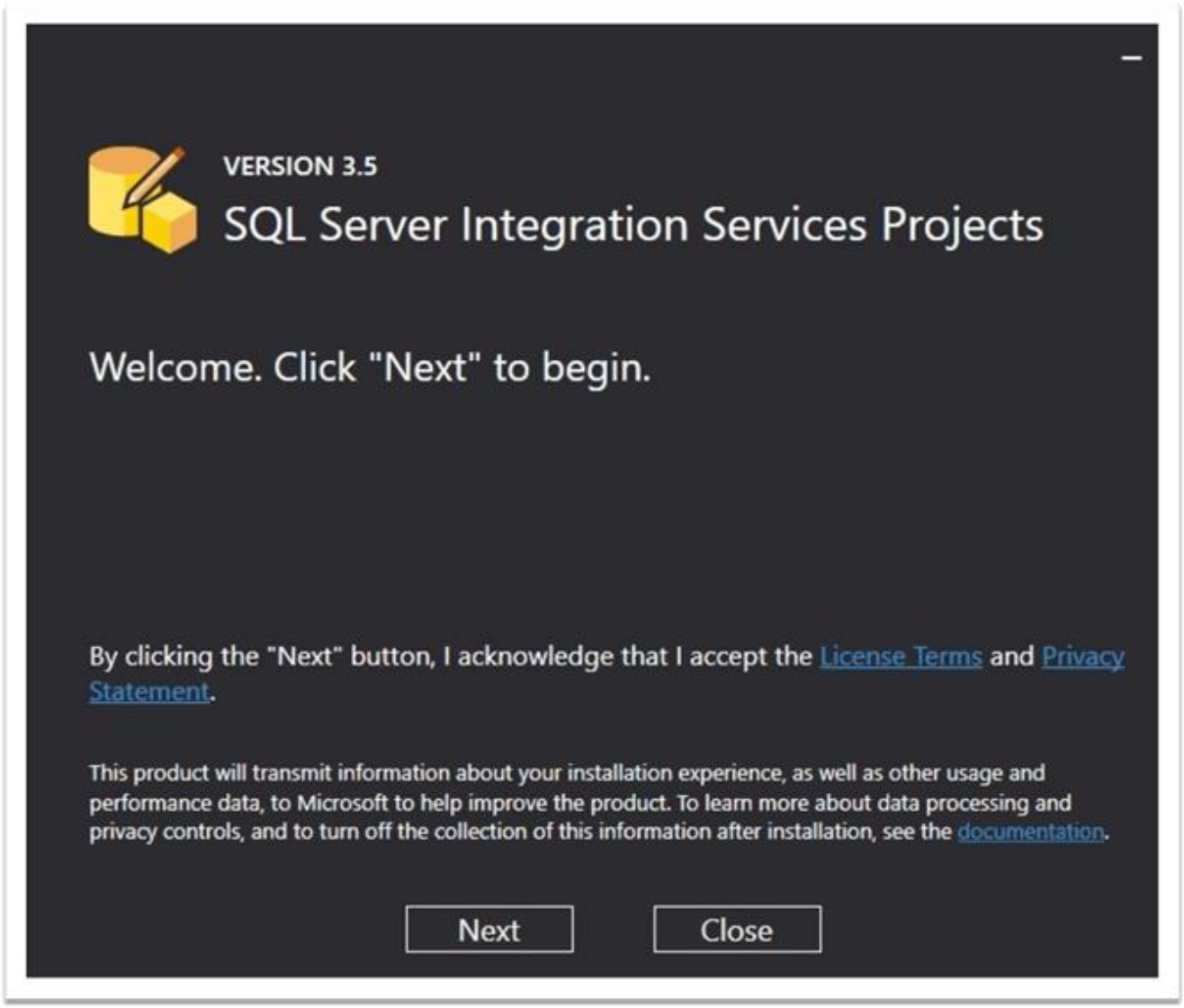


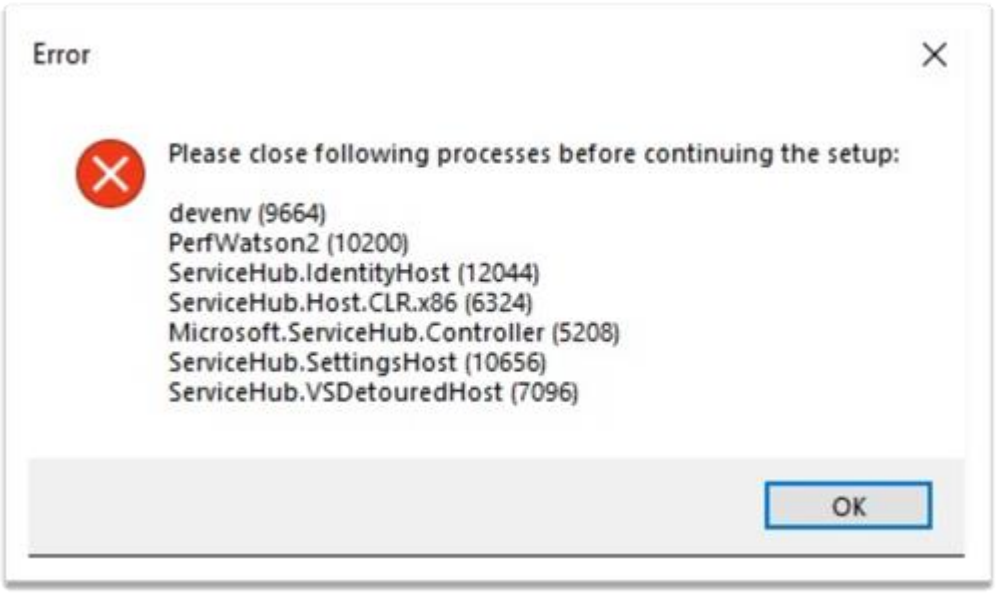
Integration Services

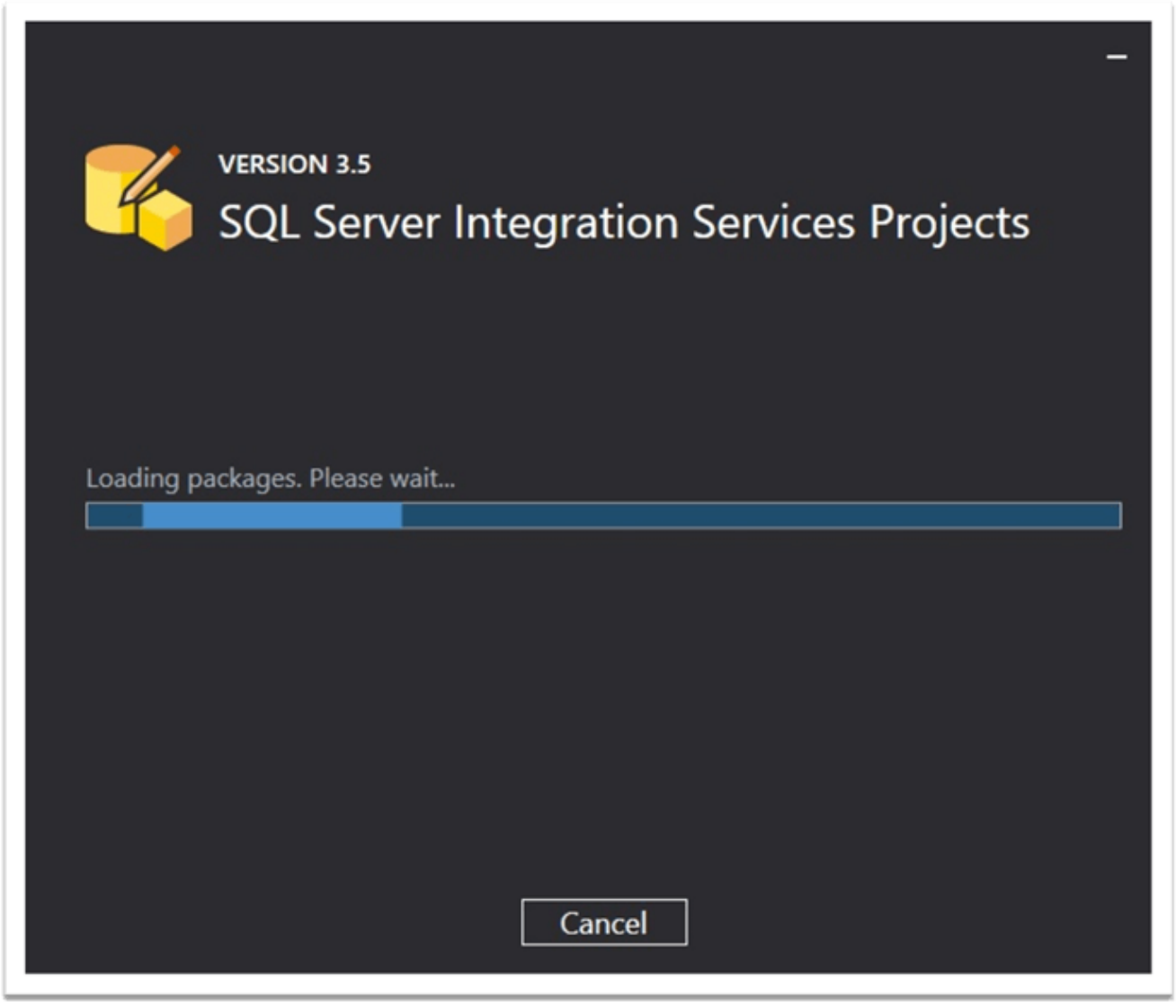


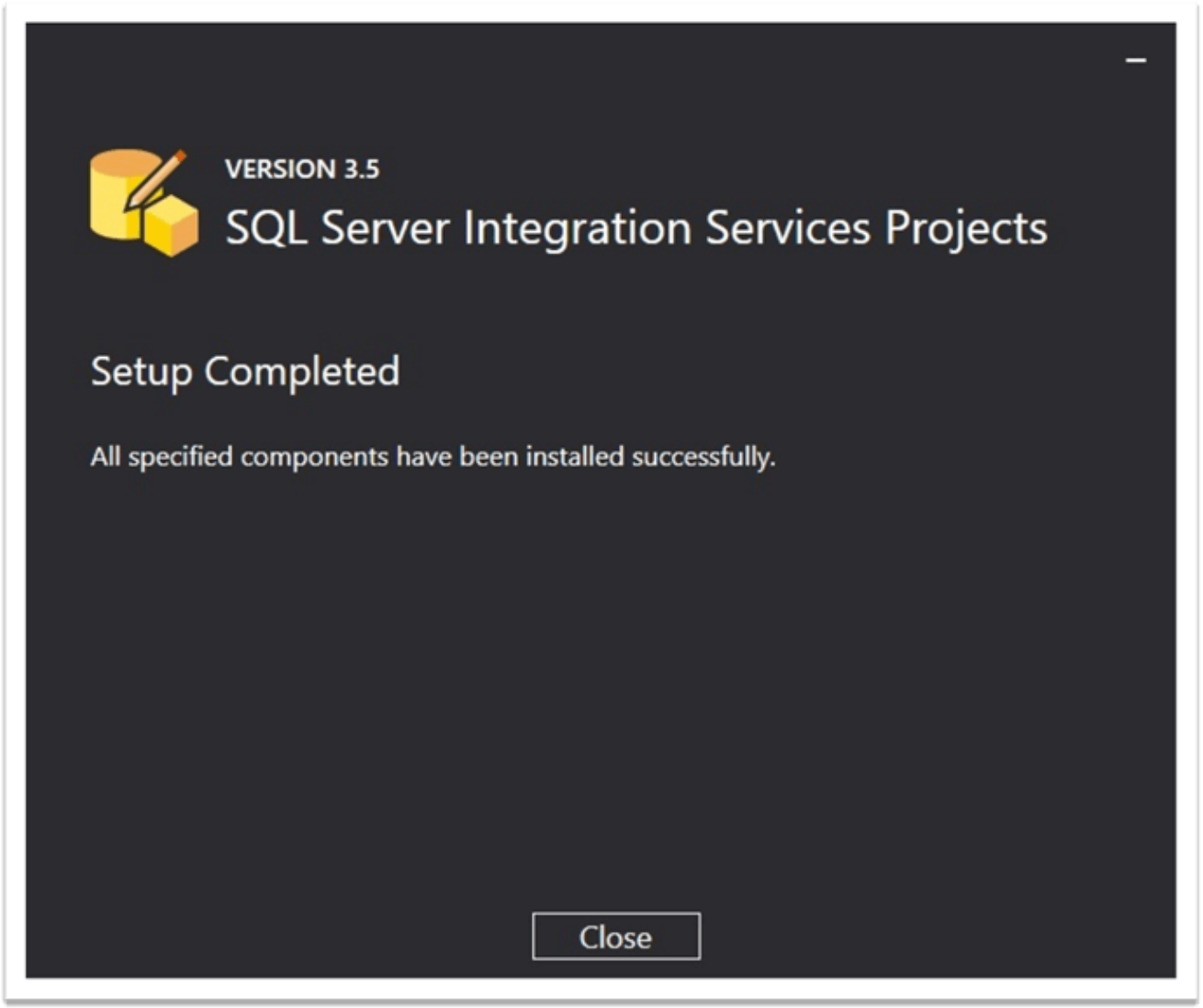


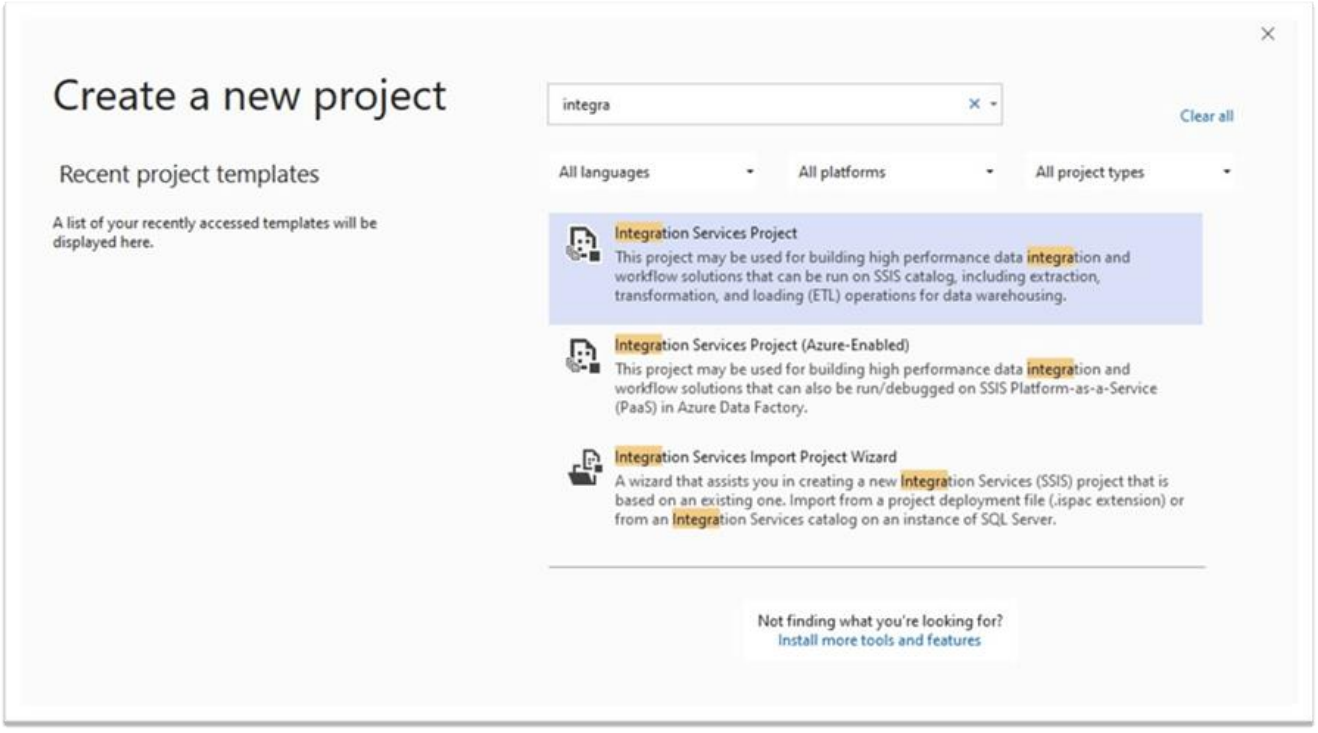

Please select the language of the installer:
English (United States)
OK
Cancel

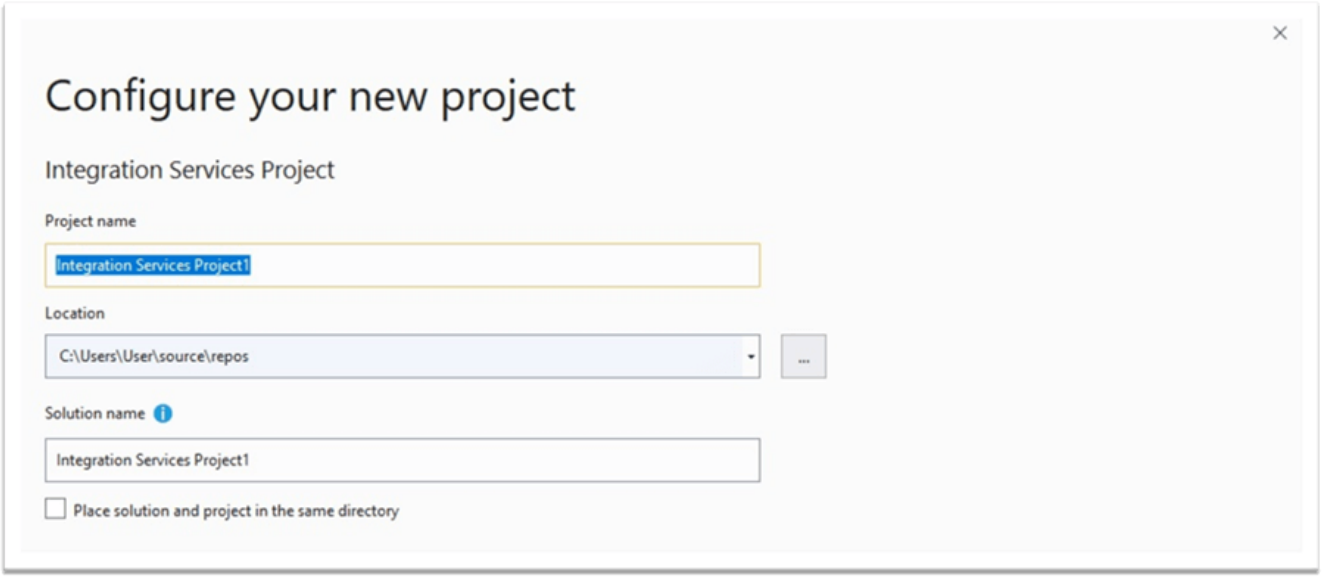

VERSION 3.5
SQL Server Integration Services Projects
Welcome. Click "Next" to begin.
By clicking the "Next" button, I acknowledge that I accept the License Terms and Privacy Statement.
This product will transmit information about your installation experience, as well as other usage and performance data, to Microsoft to help improve the product. To learn more about data processing and privacy controls, and to turn off the collection of this information after installation, see the documentation.
Next
Close

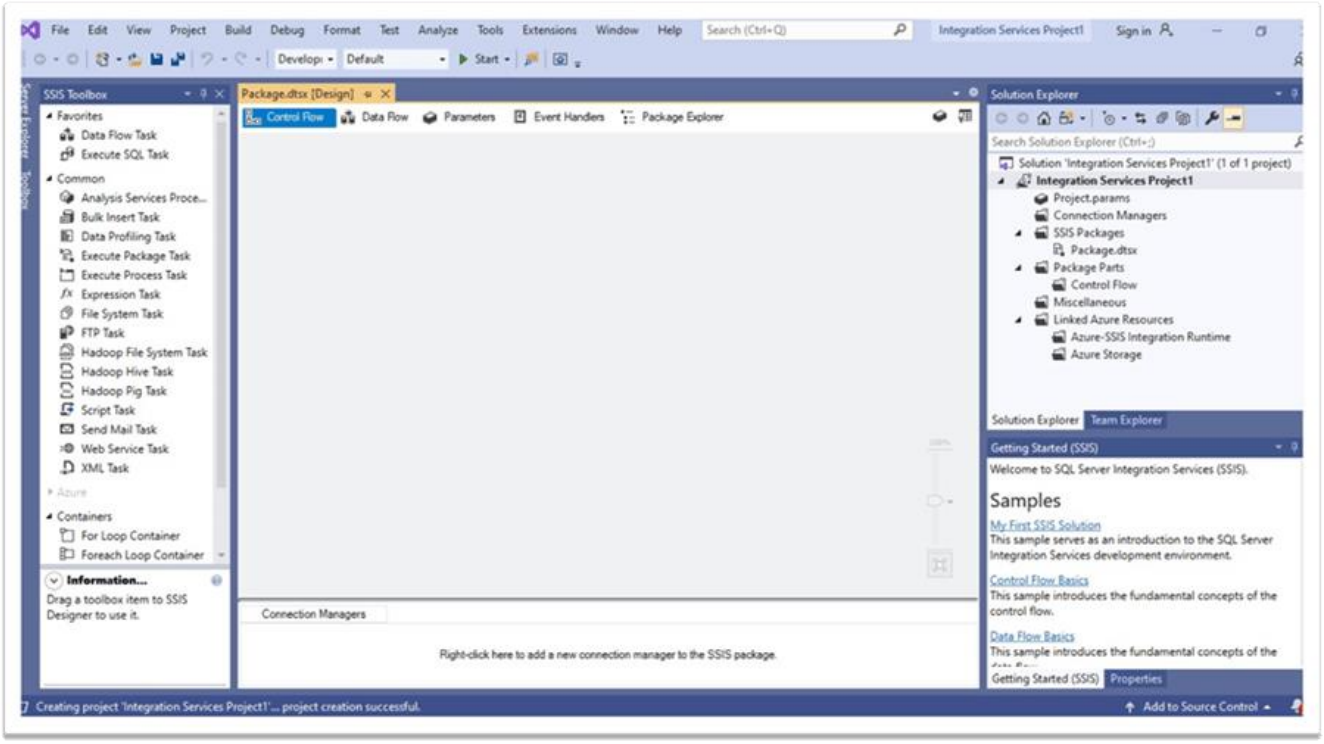

Error
Please close following processes before continuing the setup:
devenv (9664)
PerfWatson2 (10200)
ServiceHub.IdentityHost (12044)
ServiceHub.Host.CLR.x86 (6324)
Microsoft.ServiceHub.Controller (5208)
ServiceHub.SettingsHost (10656)
ServiceHub.VSDetouredHost (7096)
OK


VERSION 3.5
SQL Server Integration Services Projects
Loading packages. Please wait...
Cancel

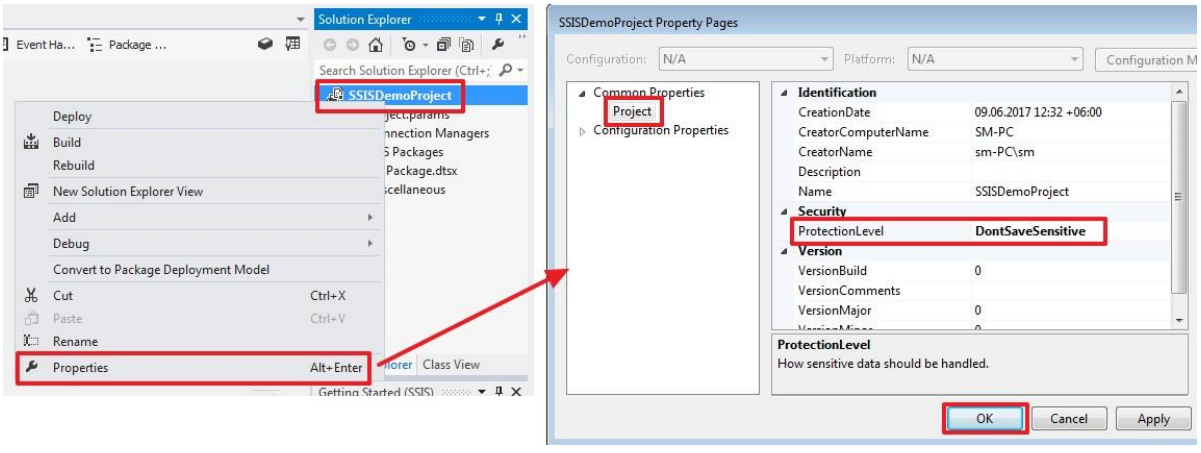

VERSION 3.5
SQL Server Integration Services Projects
Setup Completed
All specified components have been installed successfully.
Close


Create a new project
integra
Clear all
Recent project templates
All languages
All platforms
All project types
A list of your recently accessed templates will be displayed here.
Integration Services Project
This project may be used for building high performance data integration and workflow solutions that can be run on SSIS catalog, including extraction, transformation, and loading (ETL) operations for data warehousing.
Integration Services Project (Azure-Enabled)
This project may be used for building high performance data integration and workflow solutions that can also be run/debugged on SSIS Platform-as-a-Service (PaaS) in Azure Data Factory.
Integration Services Import Project Wizard
A wizard that assists you in creating a new Integration Services (SSIS) project that is based on an existing one. Import from a project deployment file (.ispac extension) or from an Integration Services catalog on an instance of SQL Server.
Not finding what you're looking for?
Install more tools and features

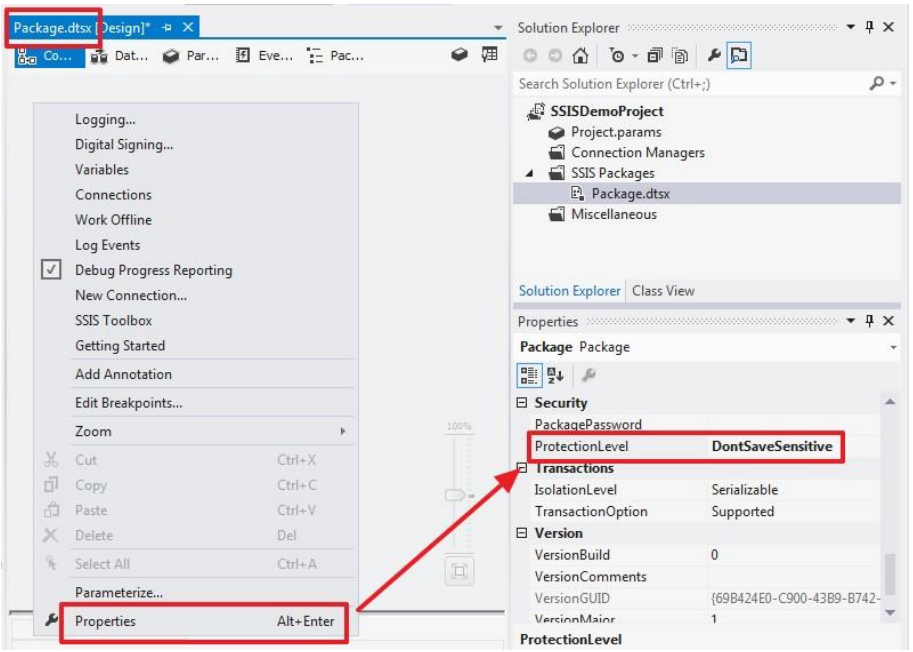




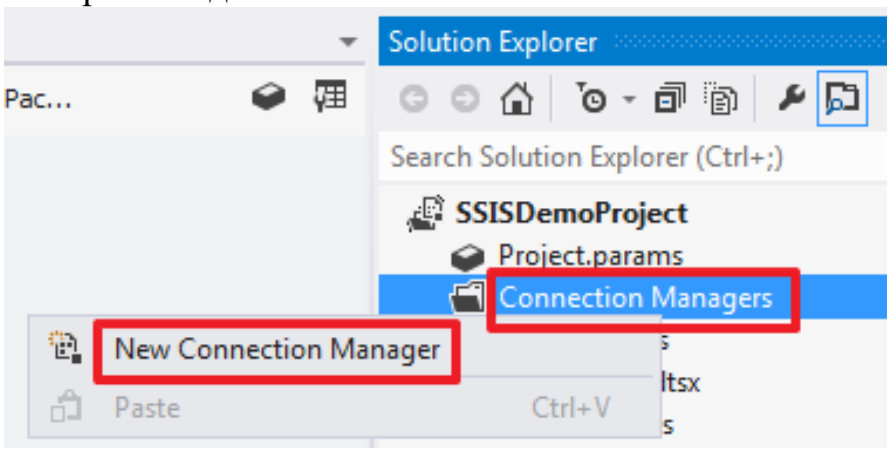
Для подальшого полегшення розгортання зайдемо в властивості проекту та змінимо опцію **ProtectionLevel** на **DontSaveSensitive**:

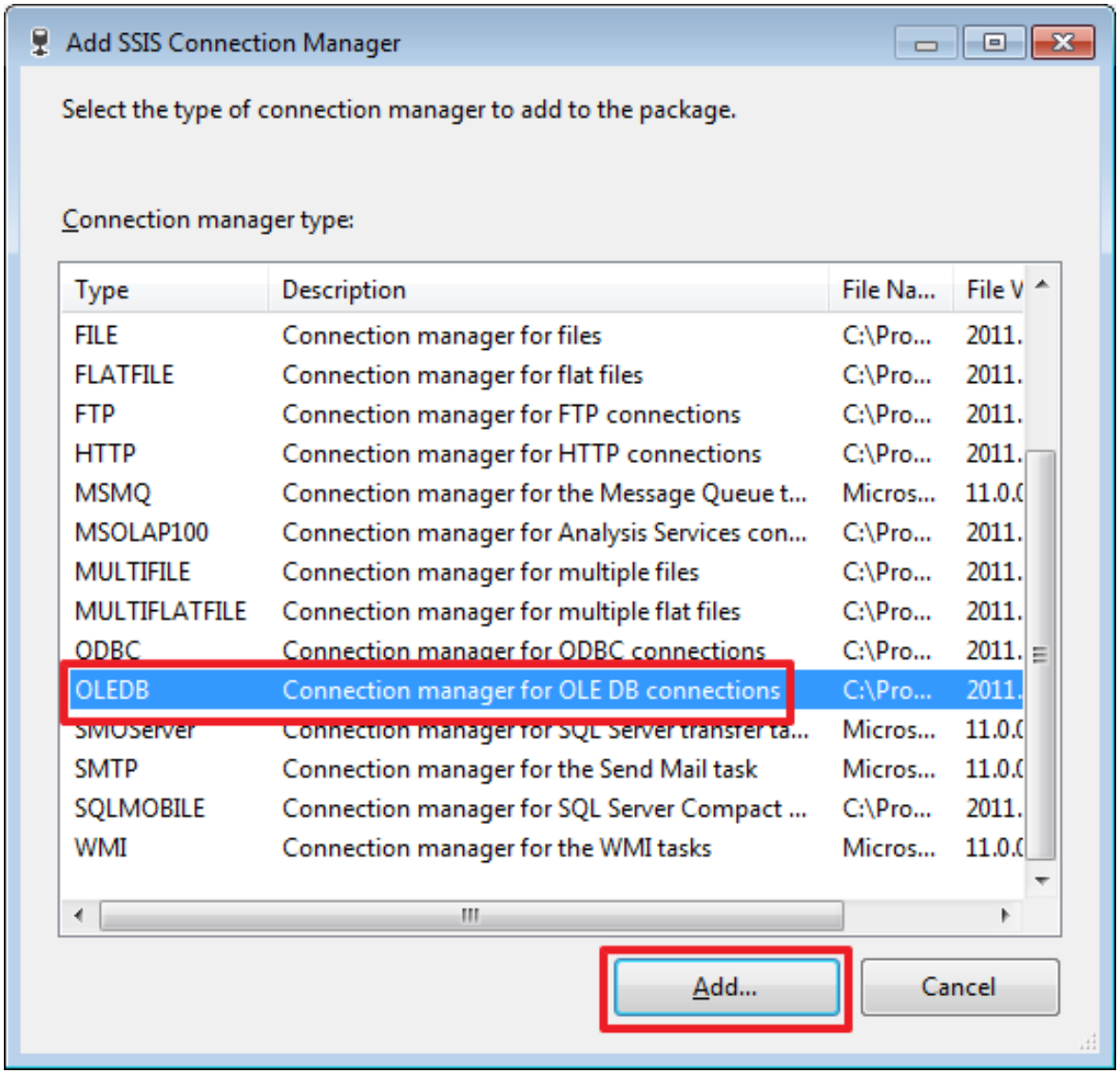


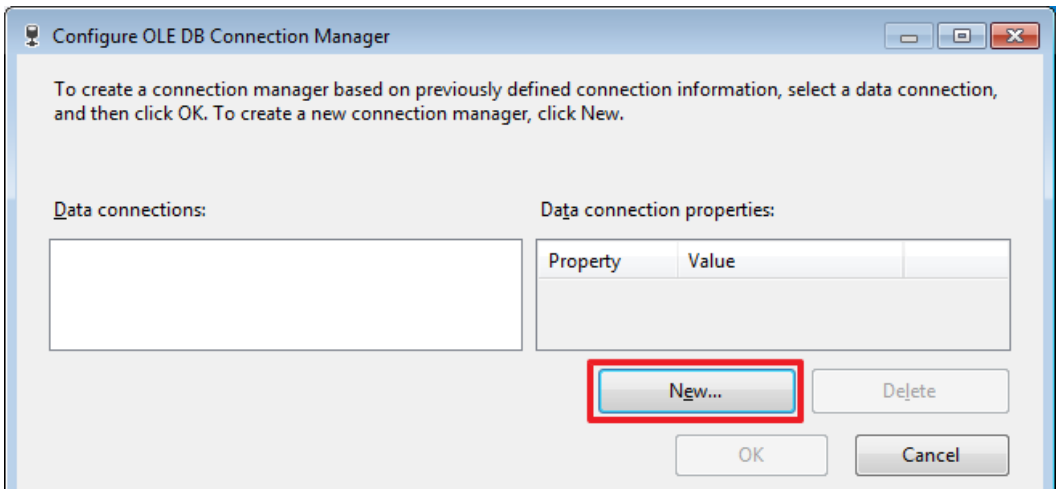
Те ж саме зробимо в властивості пакета, який створився за замовчуванням:


Package.dtsx [Design]*
Co...
Dat...
Par...
Eve...
Pac...
Logging...
Digital Signing...
Variables
Connections
Work Offline
Log Events
Debug Progress Reporting
New Connection...
SSIS Toolbox
Getting Started
Add Annotation
Edit Breakpoints...
Zoom
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Select All Ctrl+A
Parameterize...
Properties Alt+Enter
Solution Explorer
Search Solution Explorer (Ctrl+;)
SSISDemoProject
Project.params
Connection Managers
SSIS Packages
Package.dtsx
Miscellaneous
Solution Explorer
Class View
Properties
Package Package
Security
PackagePassword
ProtectionLevel DontSaveSensitive
Transactions
IsolationLevel Serializable
TransactionOption Supported
Version
VersionBuild 0
VersionComments
VersionGUID {69B424E0-C900-43B9-B742-
VersionMajor 1
ProtectionLevel

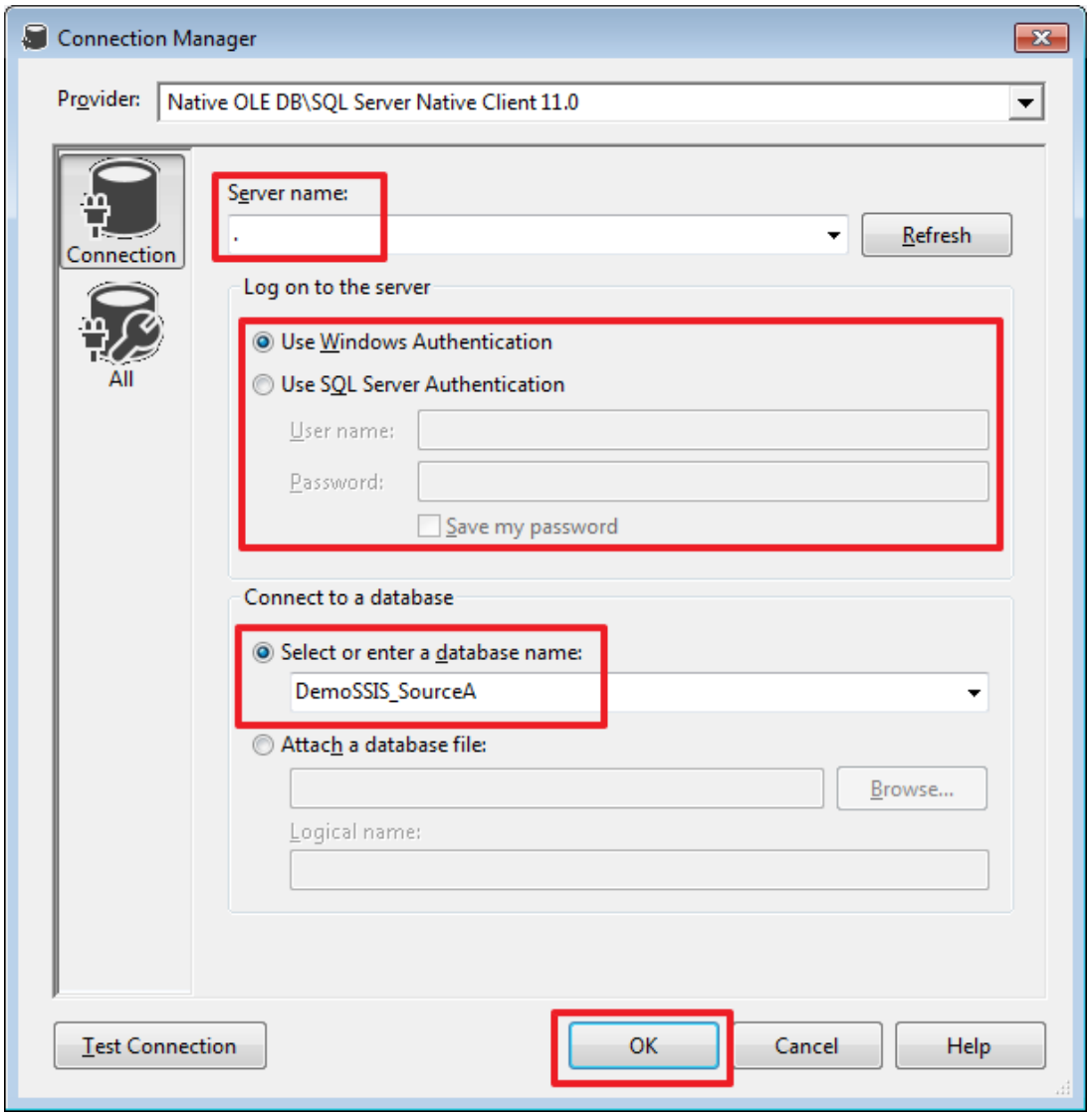
Для всіх нових пакетів ця властивість буде заповнюватися значенням з властивості проекту.
Створимо з'єднання:



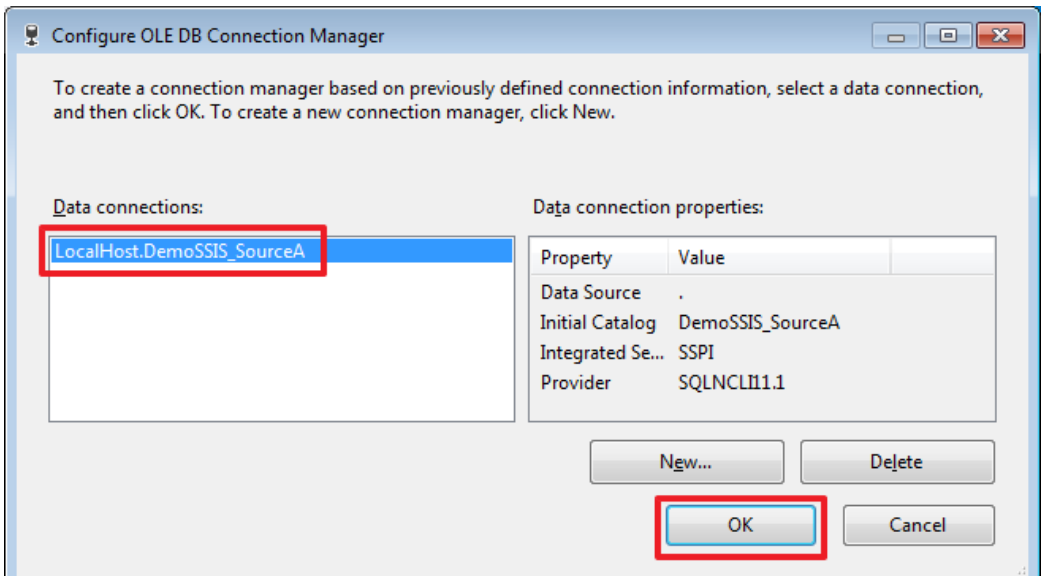




Заповнюємо параметри з'єднання з БД:

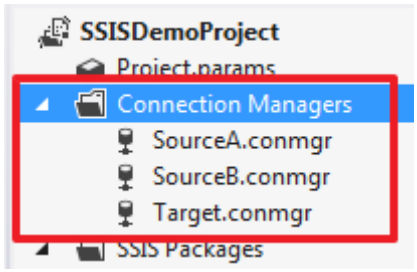


Бойові параметри з'єднання в подальшому можна буде налаштувати при створенні завдання SQL Server Agent.

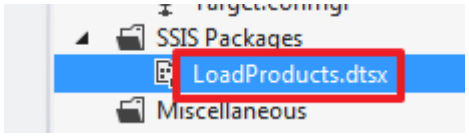


Для зручності я перейменують назву з'єднання на **SourceA**:

Таким же чином створимо і перейменуємо з'єднання для баз **DemoSSIS_SourceB** и **DemoSSIS_Target**:

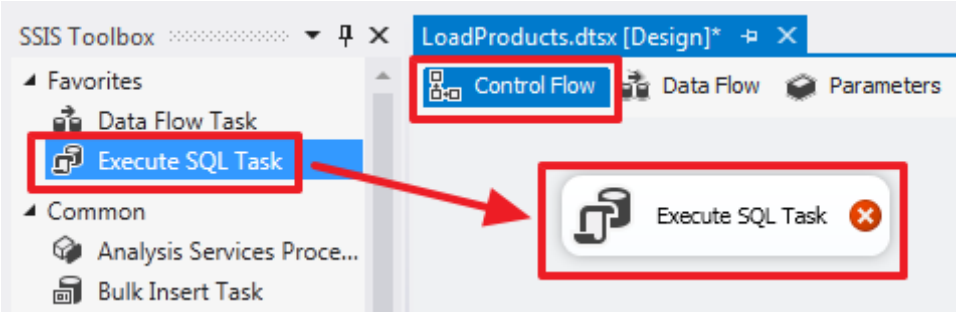


Перейменуємо пакет, створений за замовчуванням, в «**LoadProducts.dtsx**»:

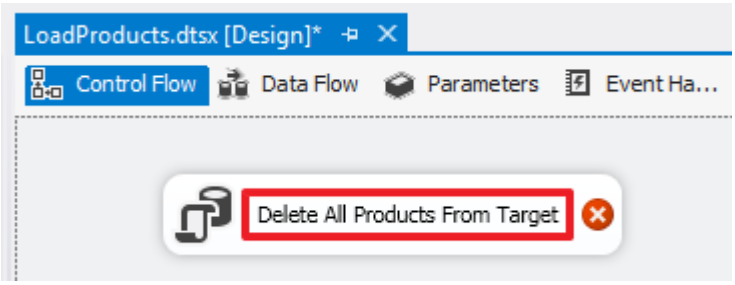


Спочатку напишемо просту логіку, яка буде повністю очищати таблицю Products в базі DemoSSIS_Target і знову завантажувати в неї дані з двох баз даних DemoSSIS_SourceA і DemoSSIS_SourceB.

Для очищення скористаємося компонентом «Execute SQL Task», який ми за допомогою миші створимо в області «Control Flow»:

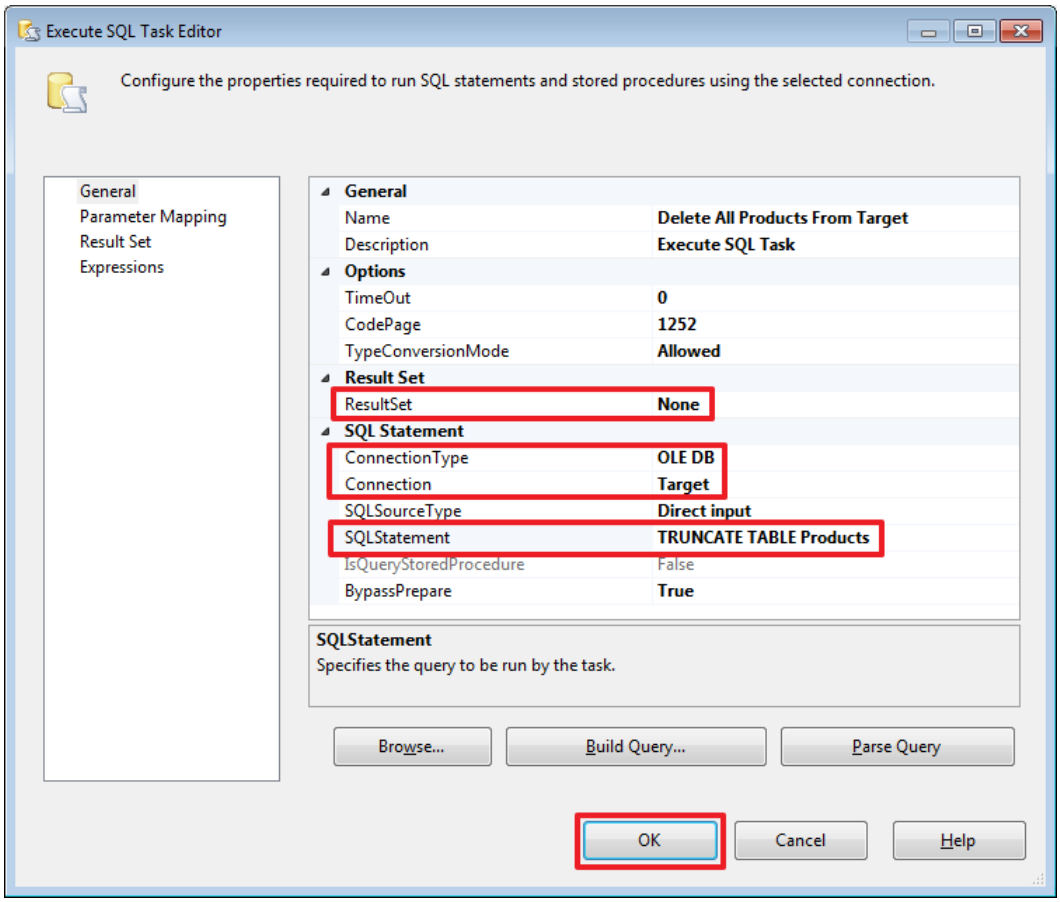


Для наочності можна перейменувати назву компонент. Задамо йому ім'я «**Delete All Products From Target**»:

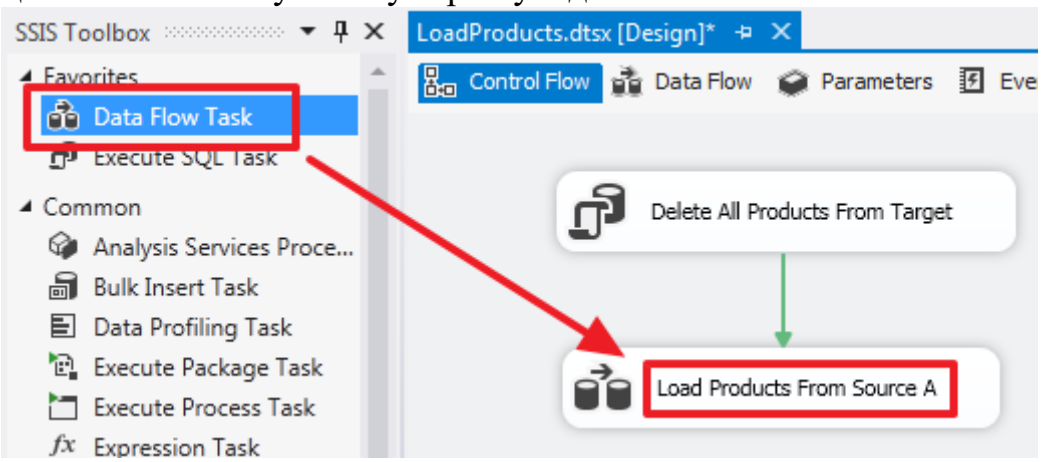


Для цієї мети використовується властивість Name.

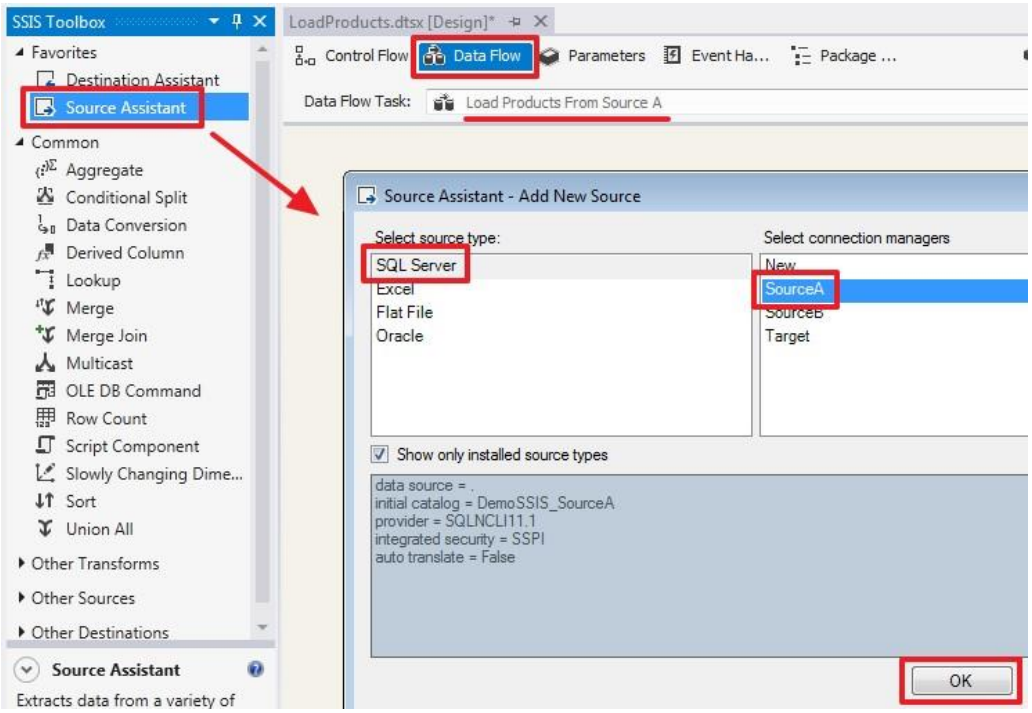
Двічі клацнемо на цьому елементі і пропишемо такі властивості:

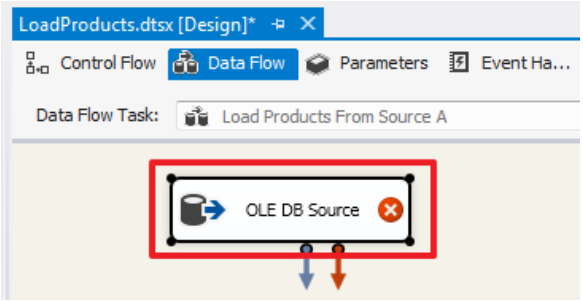


Оскільки TSQL команда «TRUNCATE TABLE Products» нічого не повертає залишимо властивості ResultSet рівним None.
Надалі ми розглянемо, як користуватися параметрами і яким чином можна скористатися результатом виконання команди, записаної в SQLStatement, а поки спробуємо побачити всю картину як це працює в цілому.
Тепер скинемо в область «Control Flow» компонент «Data Flow Task» і перейменуємо його у «Load Products From Source A», а також протягнемо до цього компоненту зелену стрілку від «Delete All Products From Target»:

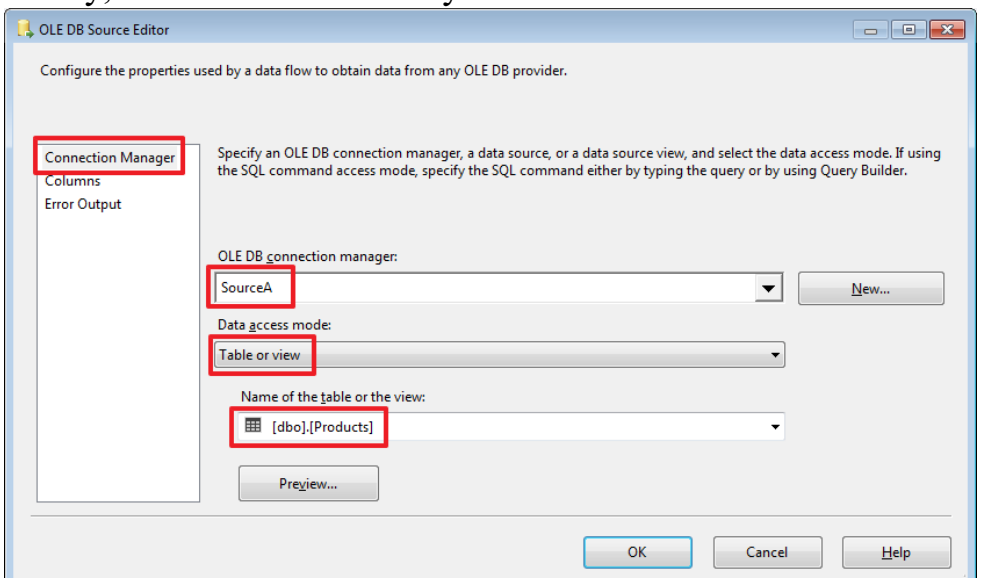


Таким чином ми створили ланцюжок, яка буде виконуватися послідовно.
Клацнувши двічі на «Load Products From Source A» ми потрапляємо в область «Data Flow» цього елемента.
Data Flow Task - це складний компонент, який має свою область, в якій створюються вкладені елементи для роботи з потоком даних.
Скинемо в цю область компонент «Source Assistant»:

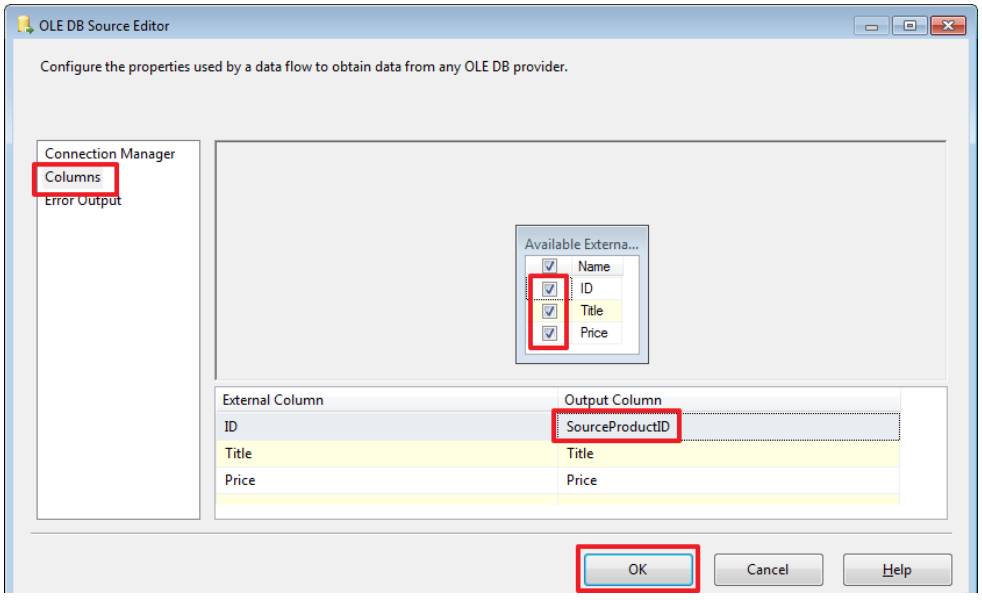




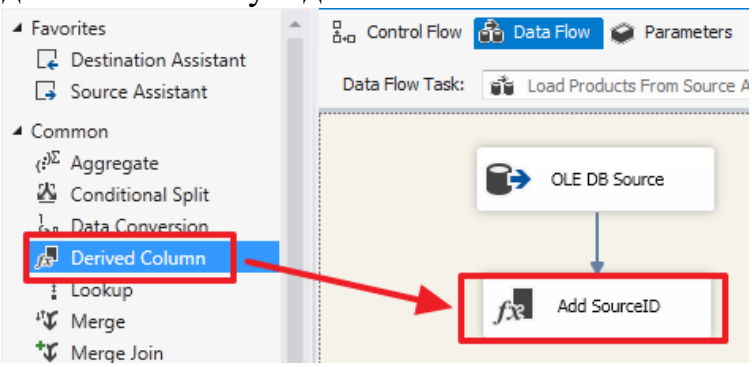
Цей компонент відповідає за отримання даних з джерела. Двічі клацнувши по ньому, ми зможемо налаштувати його:



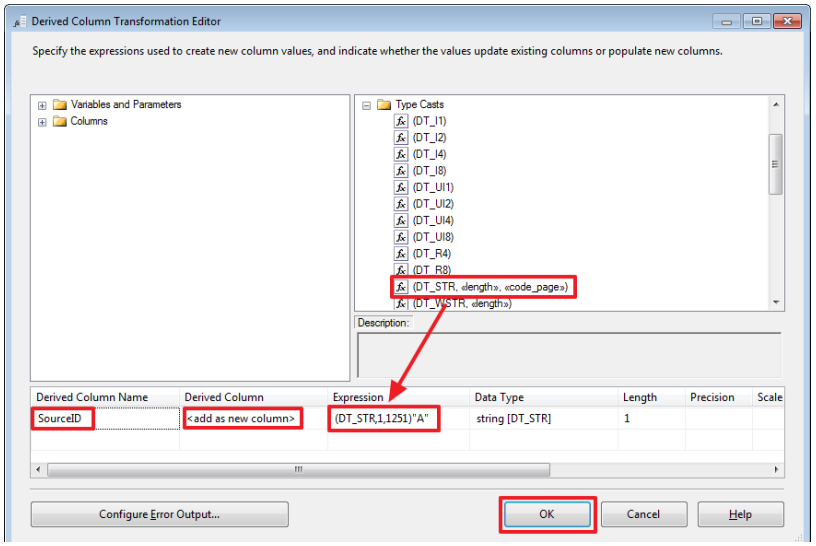
Поки скористаємося режимом «Data access mode» рівним «Table or view». Це призведе до отримання всіх рядків з таблиці Products. Подивитися дані можна натиснувши на «Preview ...».
На закладці Columns ми можемо вибрати тільки необхідні нам колонки і при необхідності перейменувати їх прописавши нове ім'я в колонці «Output Columns»:



Для одержувача потрібна ще одна додаткова колонка SourceID, додамо її до вихідного набору за допомогою компонента «Derived Column», який перейменуємо в «Add SourceID», так само протягнемо синю стрілку до даного елементу від «OLE DB Source»:

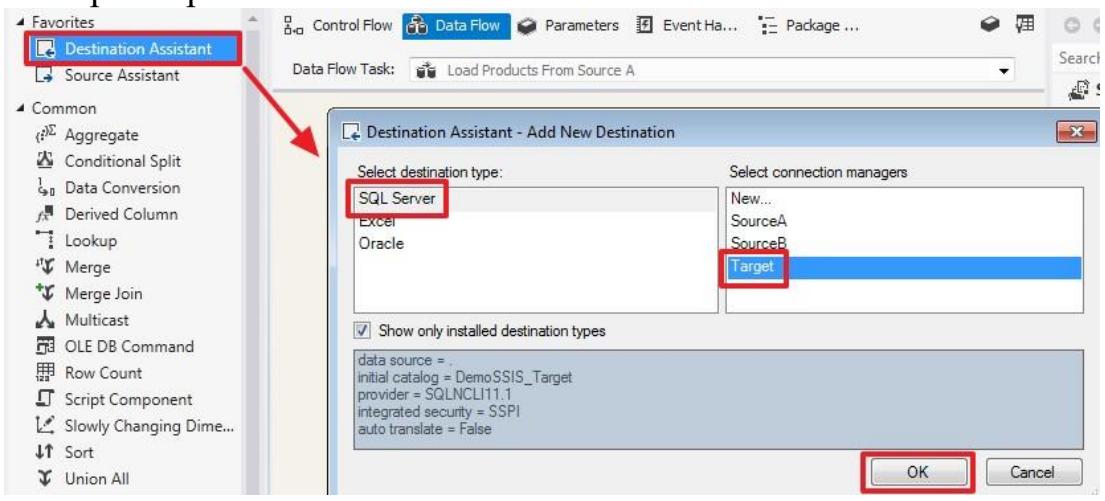


Двічі клацнемо по елементу «Add SourceID» і пропишемо значення «A» у вигляді константи:

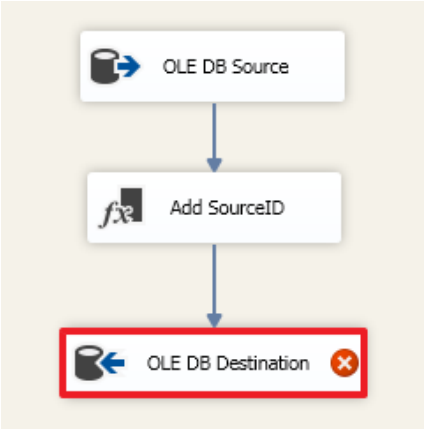


Тут я скористався функцією перетворення типу (DT_STR, 1,1251) для того щоб перетворити Unicode рядок в ANSI.

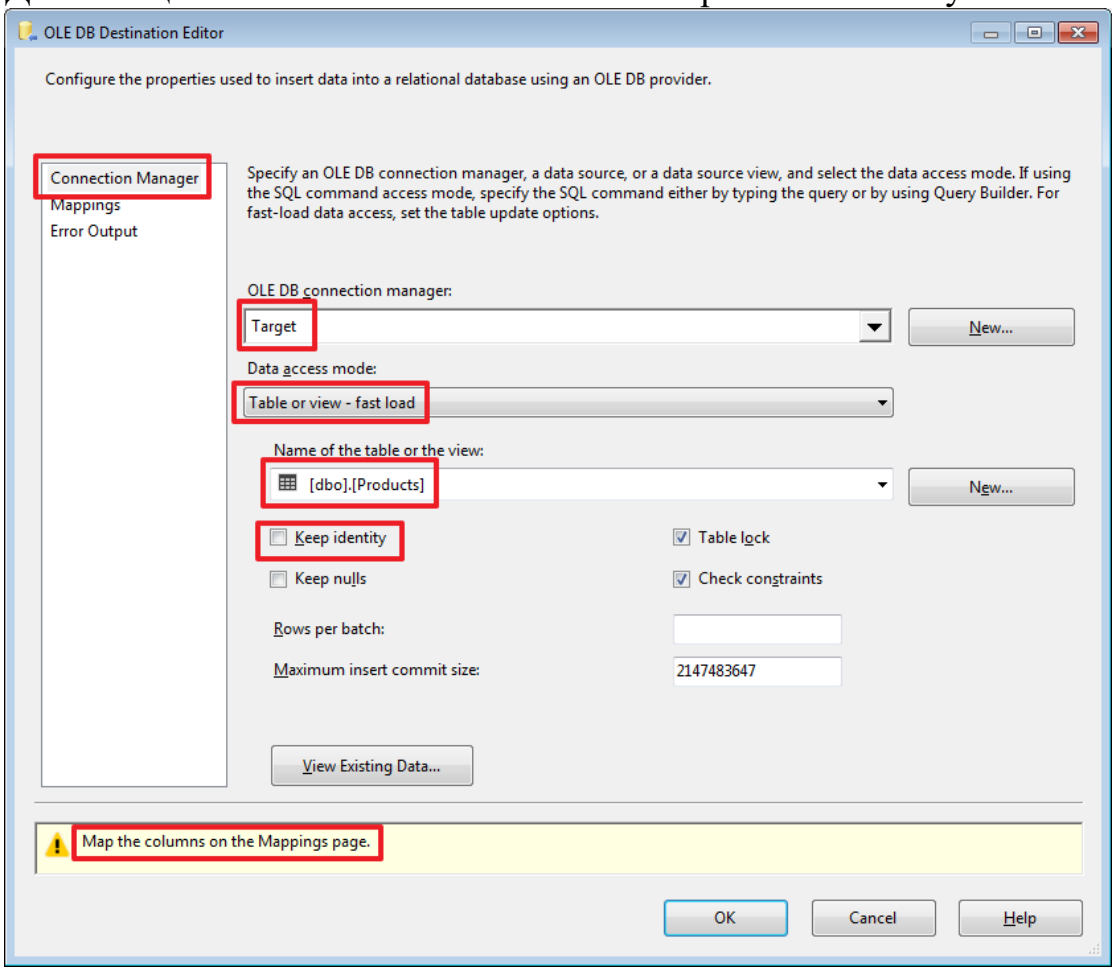
Тепер створимо компонент «Destination Assistant»:



Направимо в нього потік від «Add SourceID»:



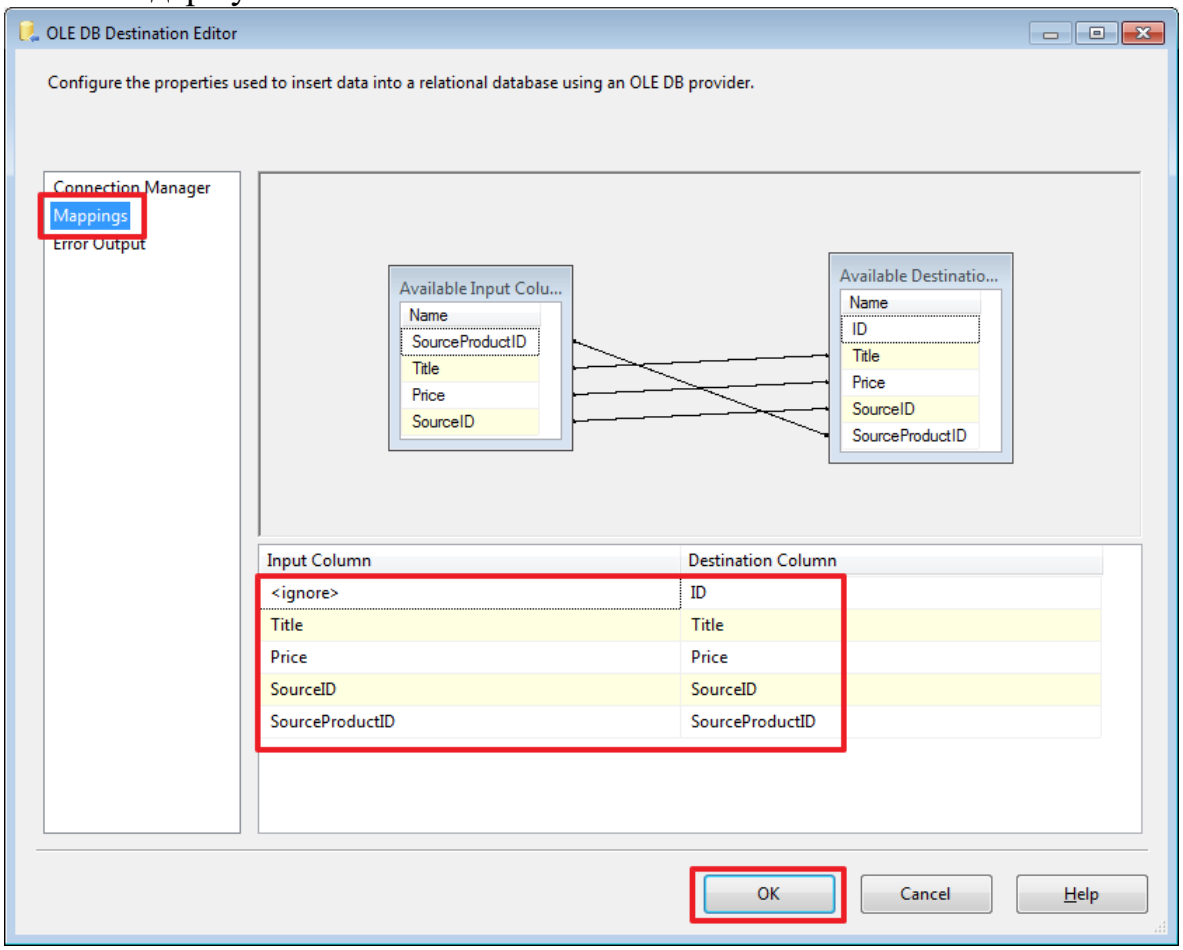
Двічі клацнемо по «OLE DB Destination» і зробимо налаштування



Тут ми показуємо в яку таблицю буде записуватися отриманий набір.

«Keep identity» використовується в разі якщо в приймаючій таблиці є поле з прапором IDENTITY і ми хочемо, щоб значення в нього теж записувалися з джерела (це аналогічно включенню опції SET IDENTITY_INSERT Products ON).

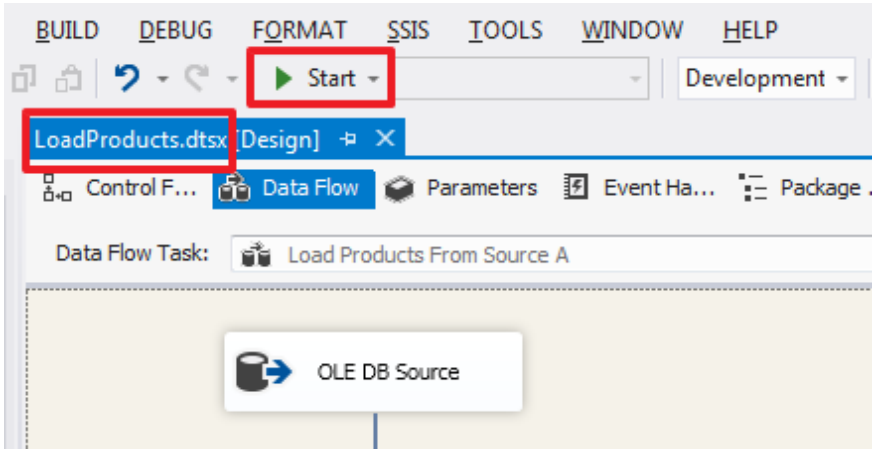
Перейшовши на закладку Mappings здійснимо прив'язку полів джерела з полями одержувача:



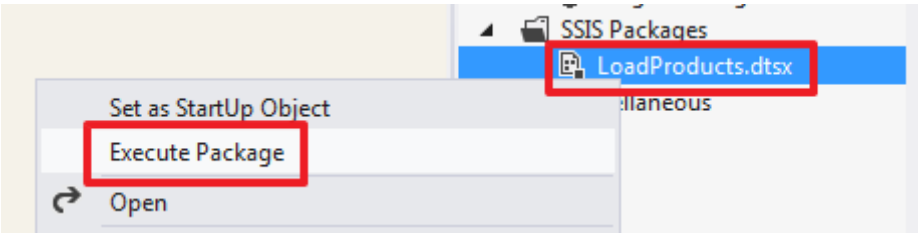
Так як у нас поля джерела і приймача іменуються однаково, то прив'язка здійснилася автоматично.

Чи можемо протестувати роботу пакета і переконатися, що дані залились в таблицю Products бази DemoSSIS_Target.

Запускаємо пакет на виконання з Visual Studio натиснувши **Start** або клавішу **F5**:

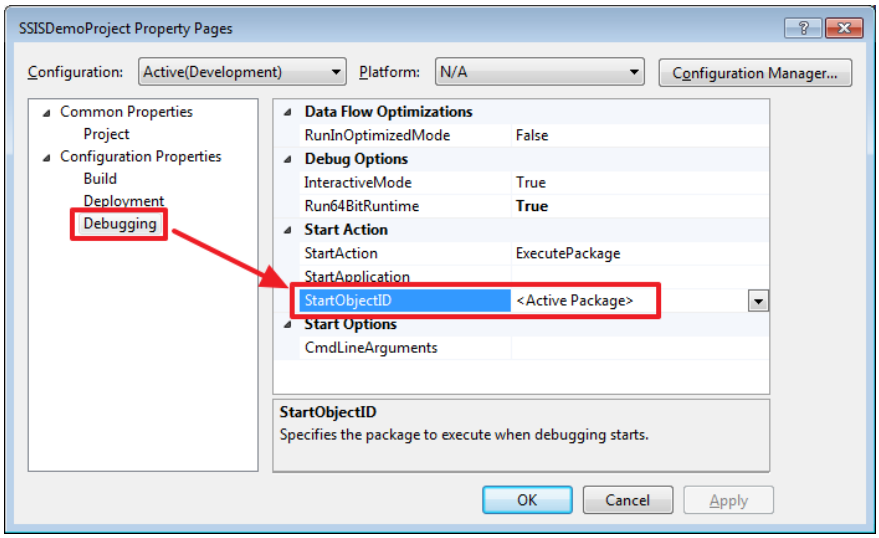


Так само пакет можна виконати, скориставшись командою з контекстного меню:



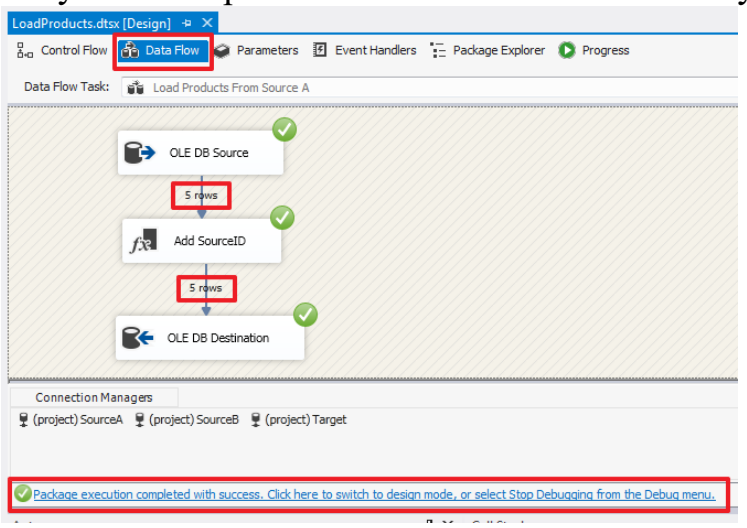
За допомогою «**Set as StartUp Object**» можна задати пакет, який буде запускатися після натискання на **Start (F5)**.

Який пакет буде запускатися при натисканні на **Start (F5)** можна перевизначити у властивостях проекту:



За замовчуванням буде запускатися пакет відкритий в поточний момент, про це говорить значення StartObjectID рівне <Active Package>.

Запустивши проект ми повинні побачити наступну картину:



Пакет виконався без помилок, про що говорить зелений значок і текст в нижній частині.

У разі наявності помилок їх можна буде побачити вкладці Progress.

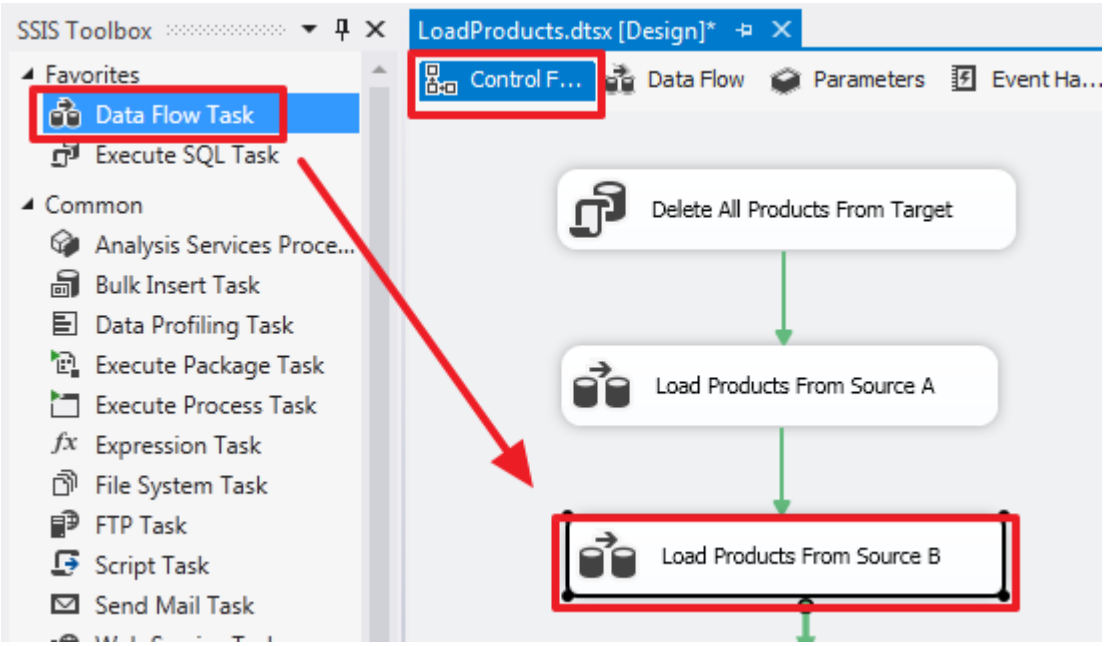
Натиснемо на посилання «Package execution completed ...» або на кнопку «Stop Debugging» розташовану на панелі інструментів для зупинення виконання пакету.

Виконуємо запит:

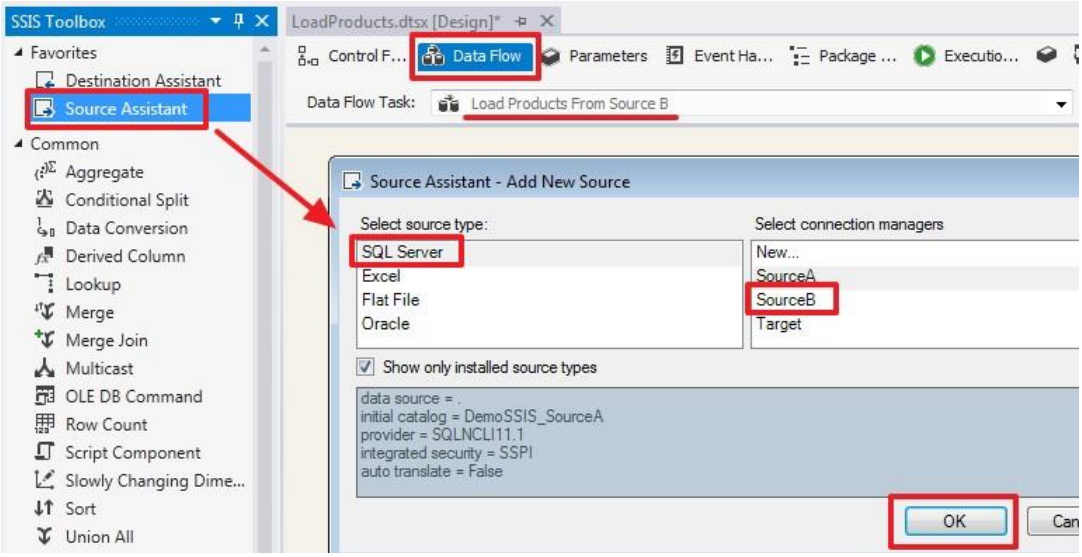
```
USE DemoSSIS_Target
GO

SELECT *
FROM Products
```

І переконаємося, що дані були записані в приймаючу таблицю.

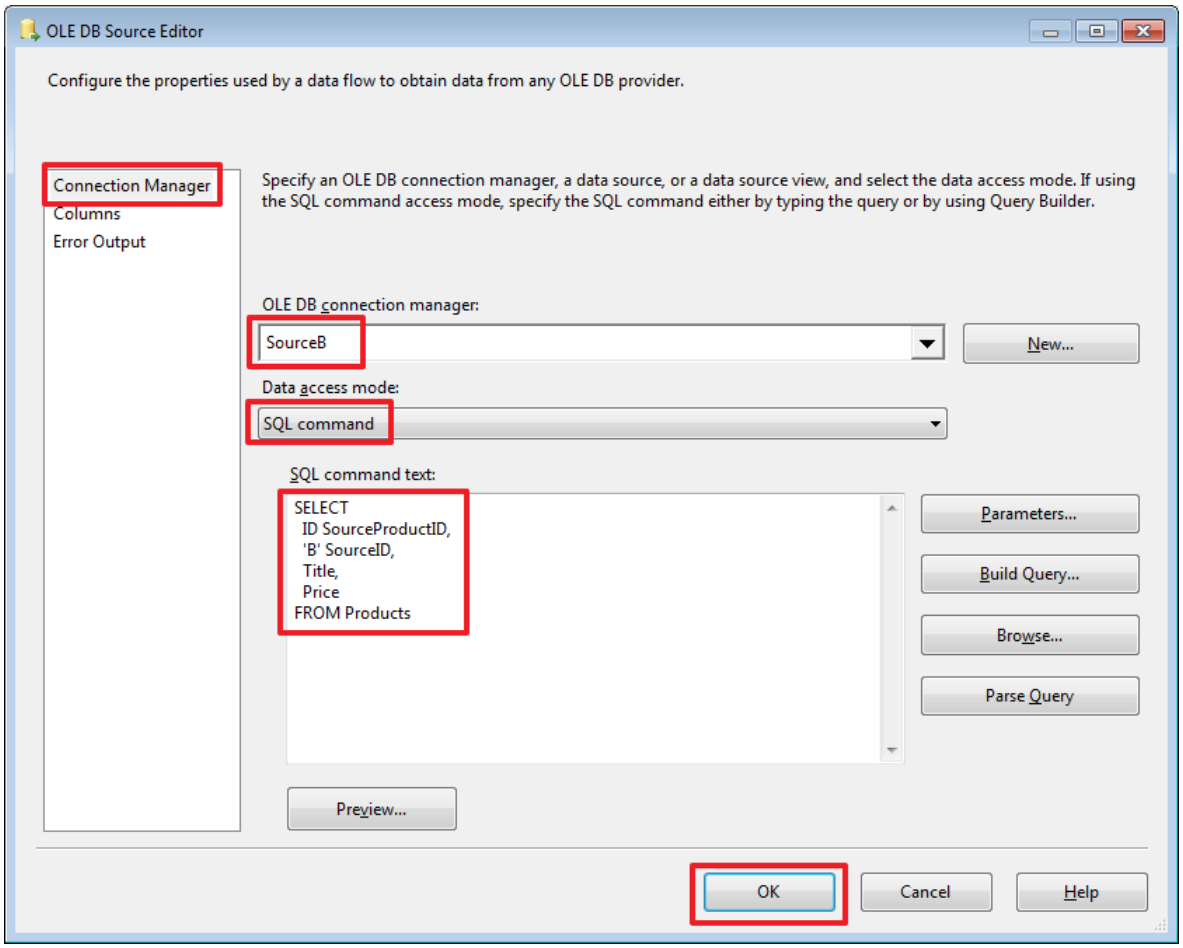
Перейдемо в область «Control Flow» і створимо ще один компонент «Data Task Flow», який назвемо «Load Products From Source B», протягнемо на нього зелену стрілку від «Load Products From Source A»:



Подвійним клацанням зайдемо в область «Data Flow» цього елемента і створимо «Source Assistant» :



Двічі клацнувши на цьому елементі, налаштуємо його по-іншому:

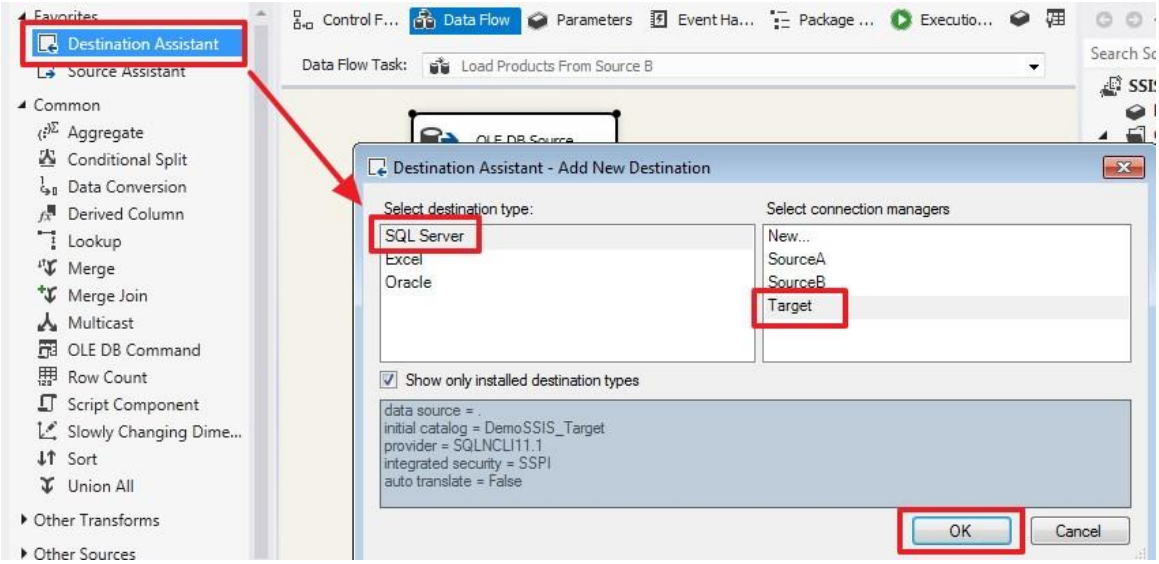


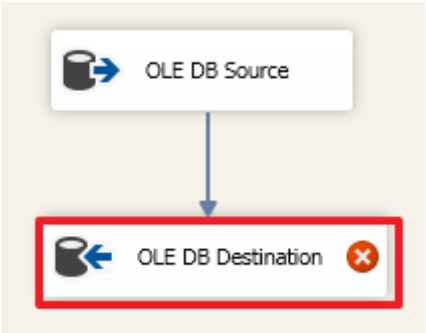
Віберемо режим «SQL command» и пропішемо Наступний запит

```
SELECT
ID SourceProductID,
'B' SourceID,
 Title,
```

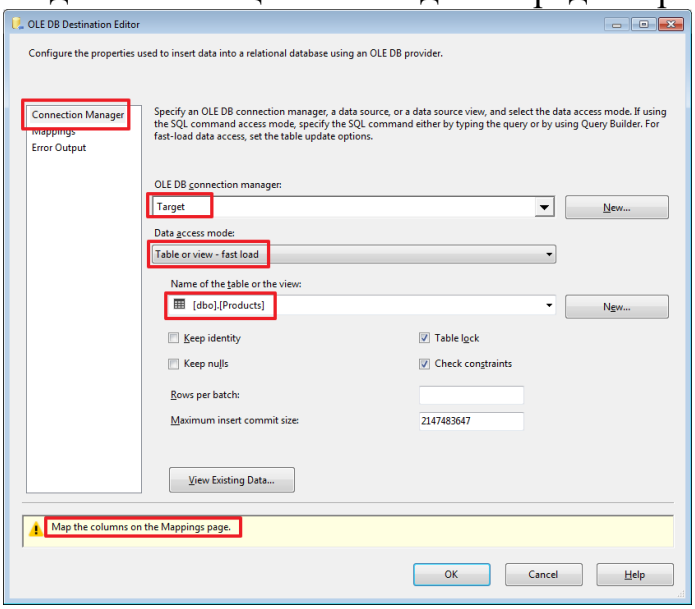
```
Price
FROM Products
```

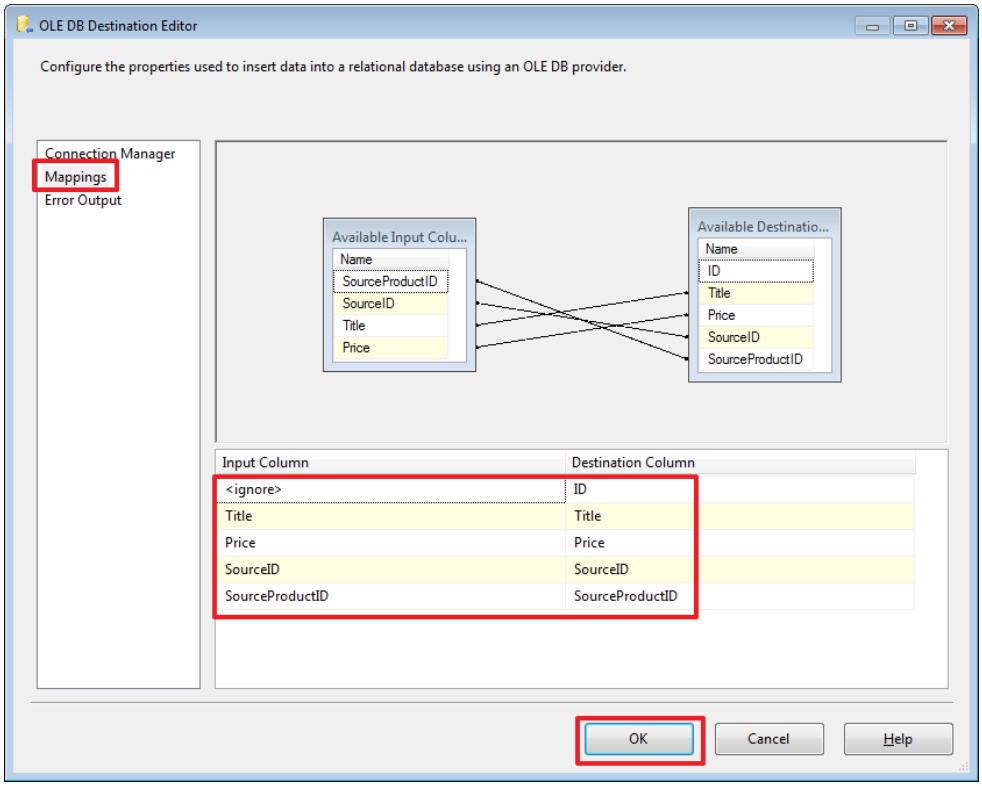
Далі відразу створімо компонент «Destination Assistant» и протягнемо на нього синю стрілку від «OLE DB Source»:





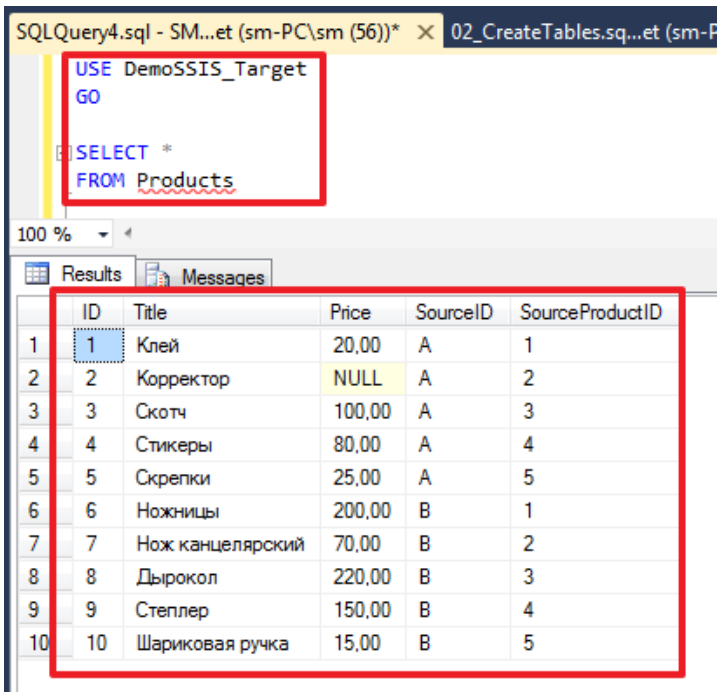
Подвійним клацанням зайдемо в редактор цього елемента і налаштуємо його:





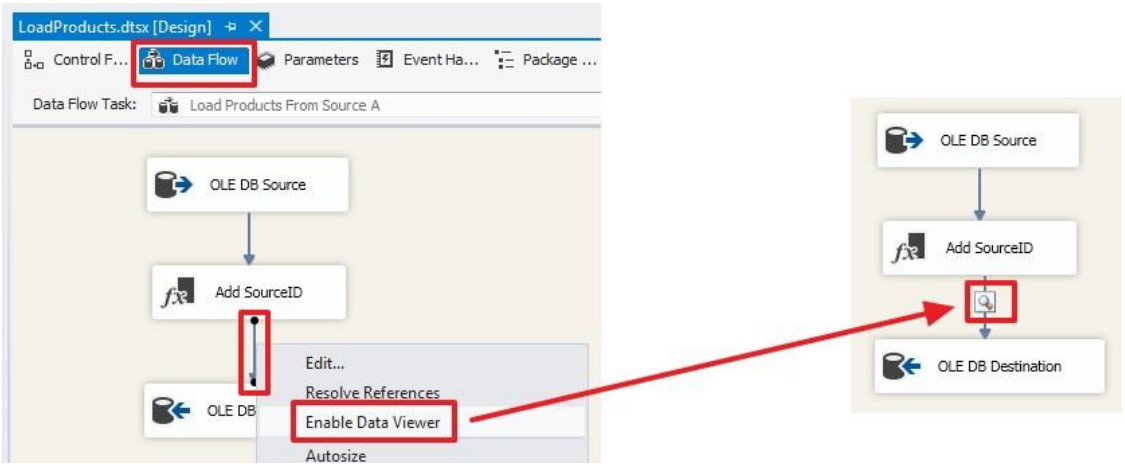
Запустимо проект на виконання і переконаємося, що дані з двох джерел потрапили в таблицю в базі Target:

```
USE DemoSSIS_Target
GO

SELECT *
FROM Products
```

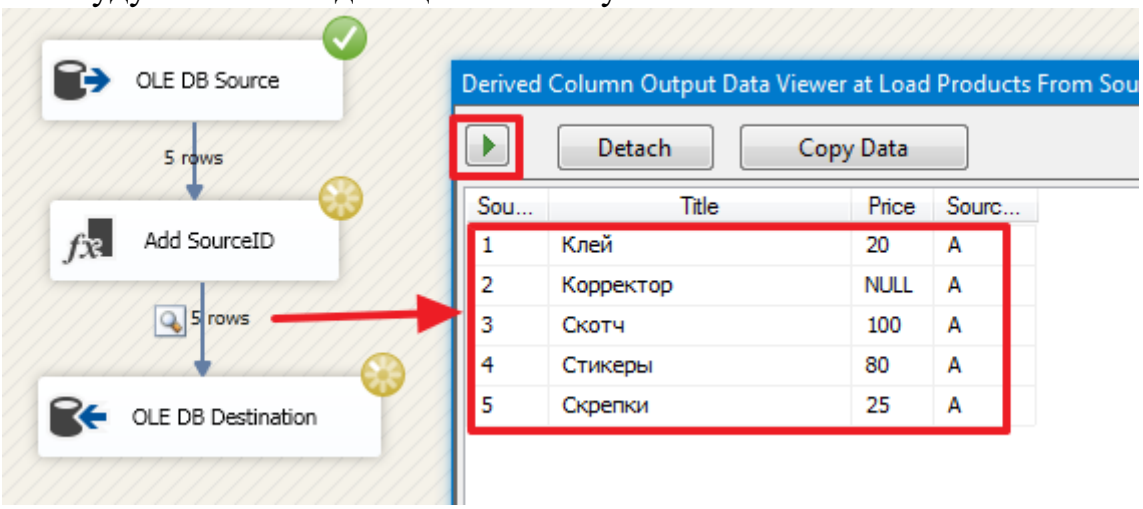


	ID	Title	Price	SourceID	SourceProductID
1	1	Клей	20,00	A	1
2	2	Корректор	NULL	A	2
3	3	Скотч	100,00	A	3
4	4	Стикеры	80,00	A	4
5	5	Скрепки	25,00	A	5
6	6	Ножницы	200,00	B	1
7	7	Нож канцелярский	70,00	B	2
8	8	Дырокол	220,00	B	3
9	9	Степлер	150,00	B	4
10	10	Шариковая ручка	15,00	B	5

Додатково в контекстному меню стрілки можна активізувати «Data Viewer»:

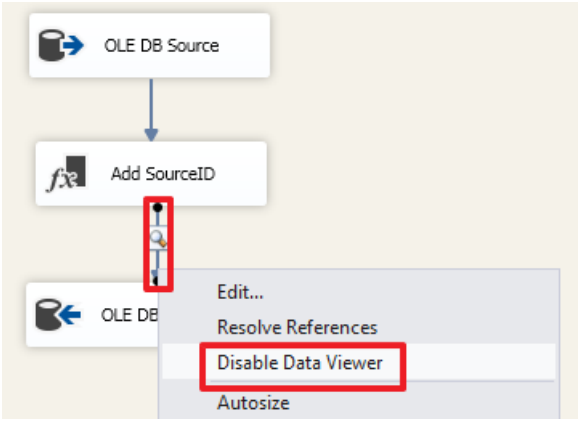


Тепер при запуску пакета на виконання в цій точці буде зроблена зупинка і нам будуть показані дані цього потоку:



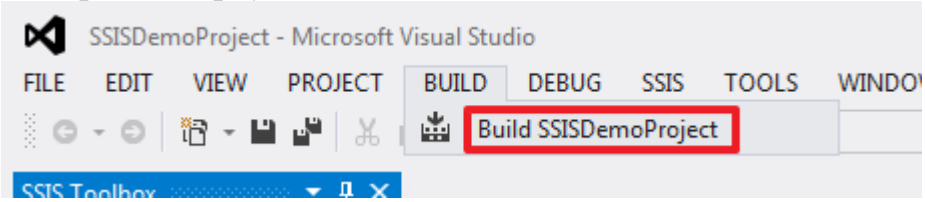
Для продовження виконання пакету потрібно натиснути на кнопку зі стрілкою або просто закрити вікно перегляду даних.

Для відключення цієї функції в контекстному меню стрілки вибираємо «Disable Date Viewer»:



Для першої частини думаю цього буде достатньо.

Створимо збірку:



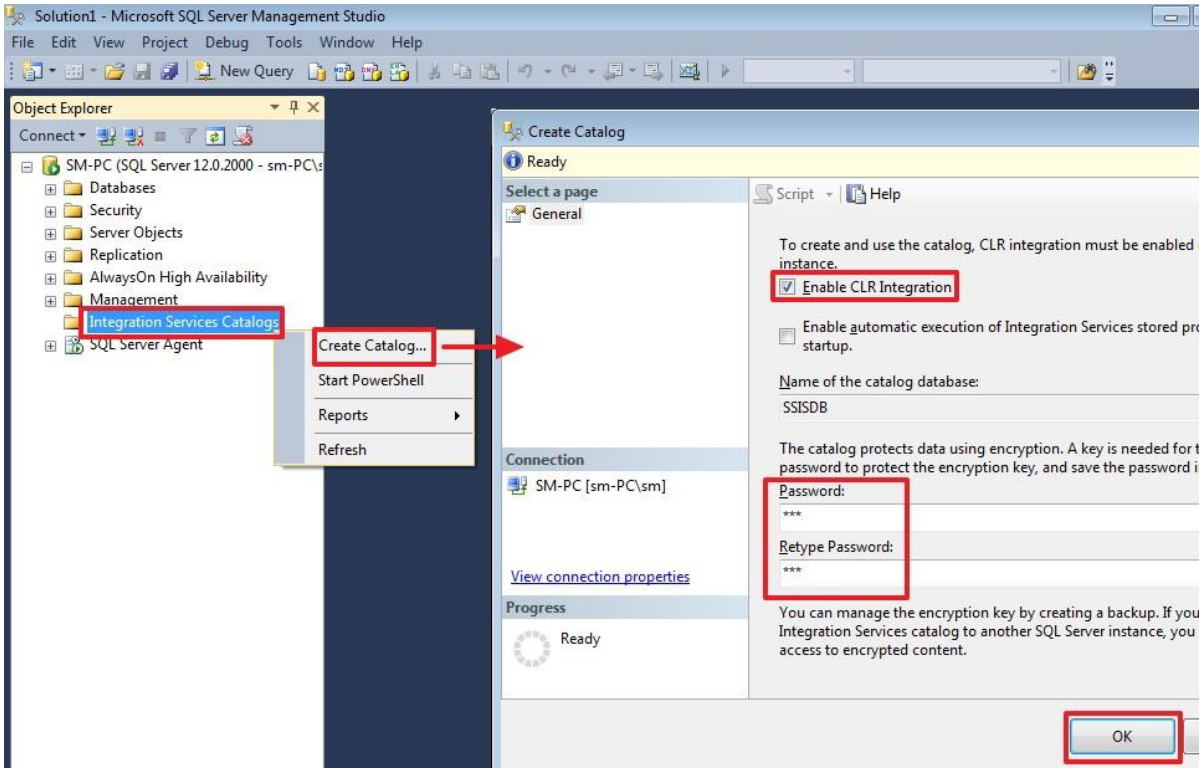
В результаті ми отримаємо файл «C:\SSIS\SSISDemoProject\bin\Development**SSISDemoProject.ispac**».

Розглянемо яким чином робиться розгортання цього проекту на SQL Server.

Розгортання SSIS

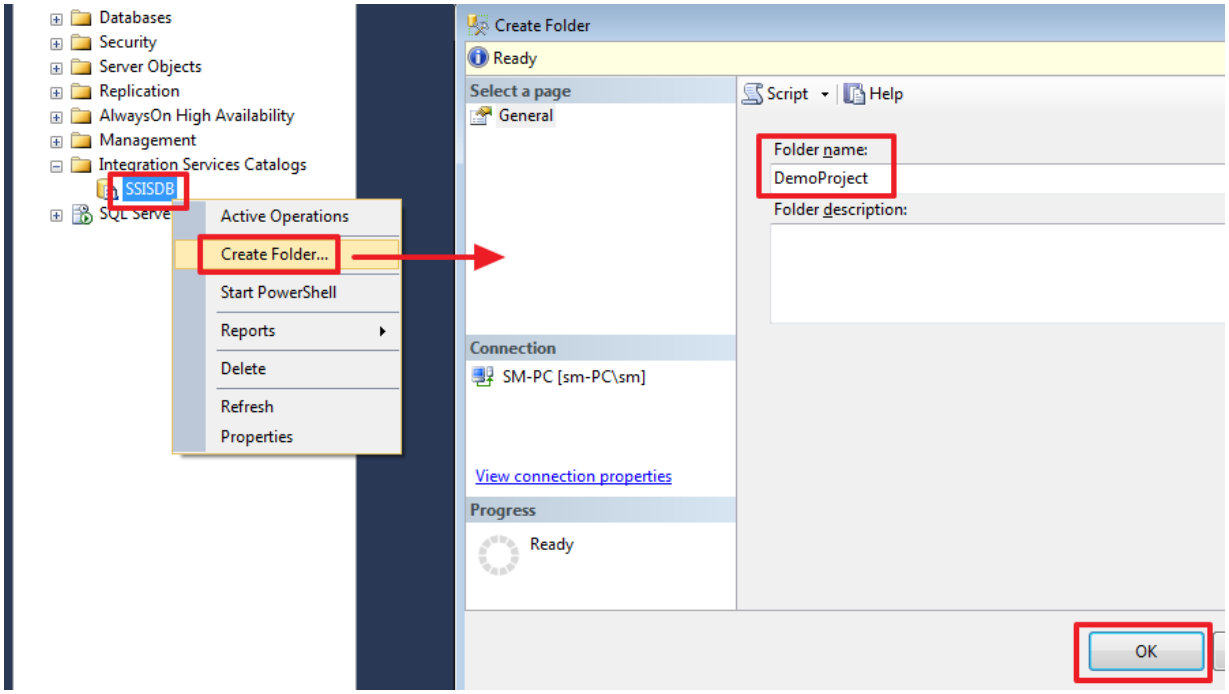
Всі подальші дії будемо робити в SSMS.

Створення каталогу SSISDB:

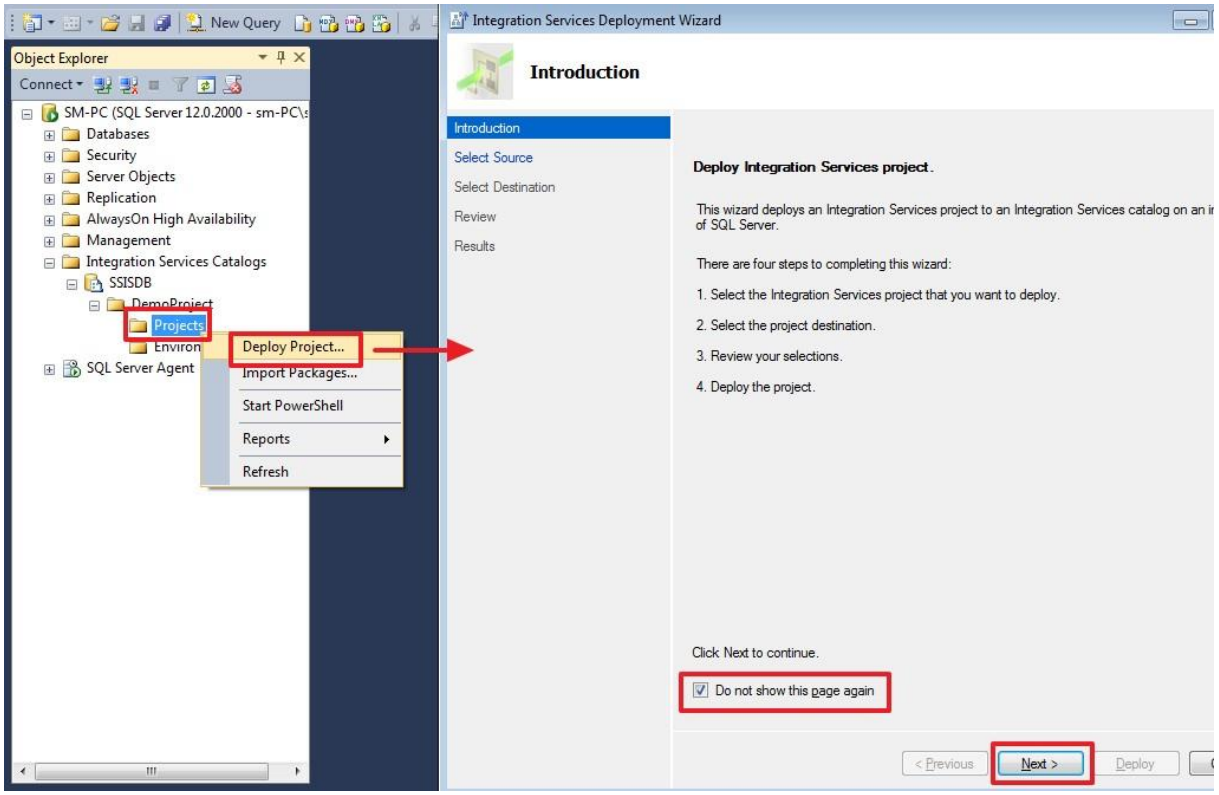


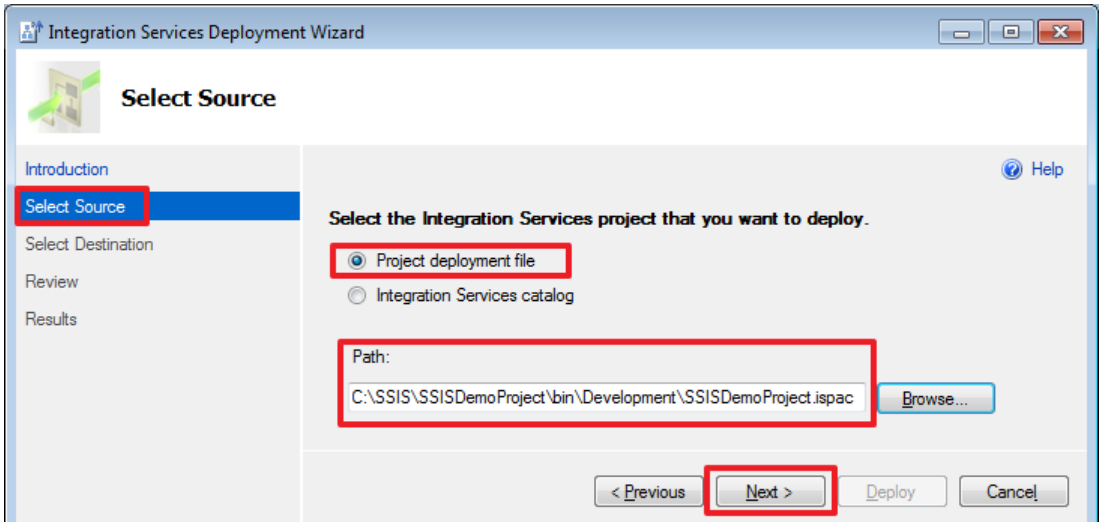
Тут вводимо будь-пароль.

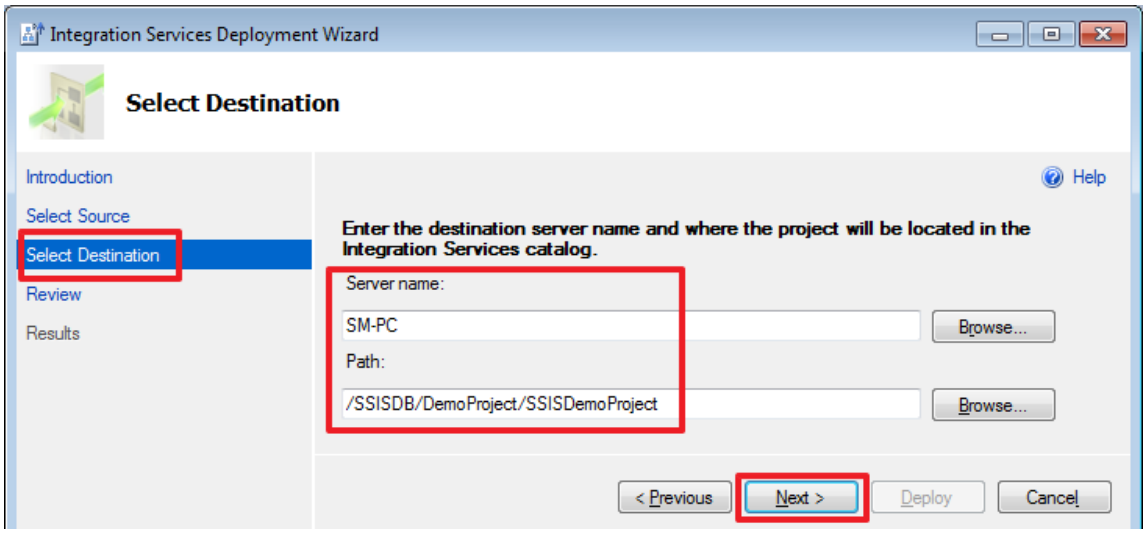
Тепер створюємо папку, в якій буде розташовуватися наш проект:

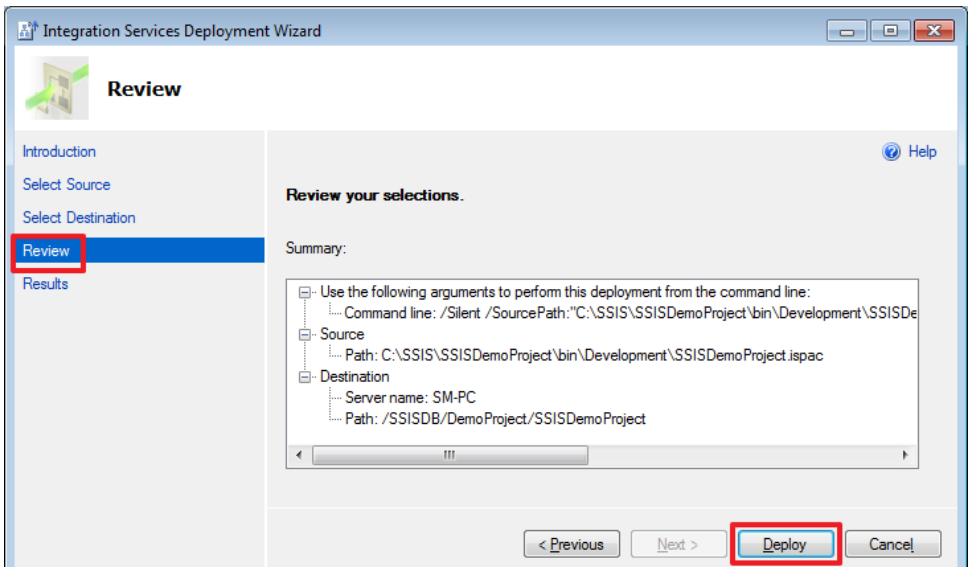


Розгортаємо сам проект:

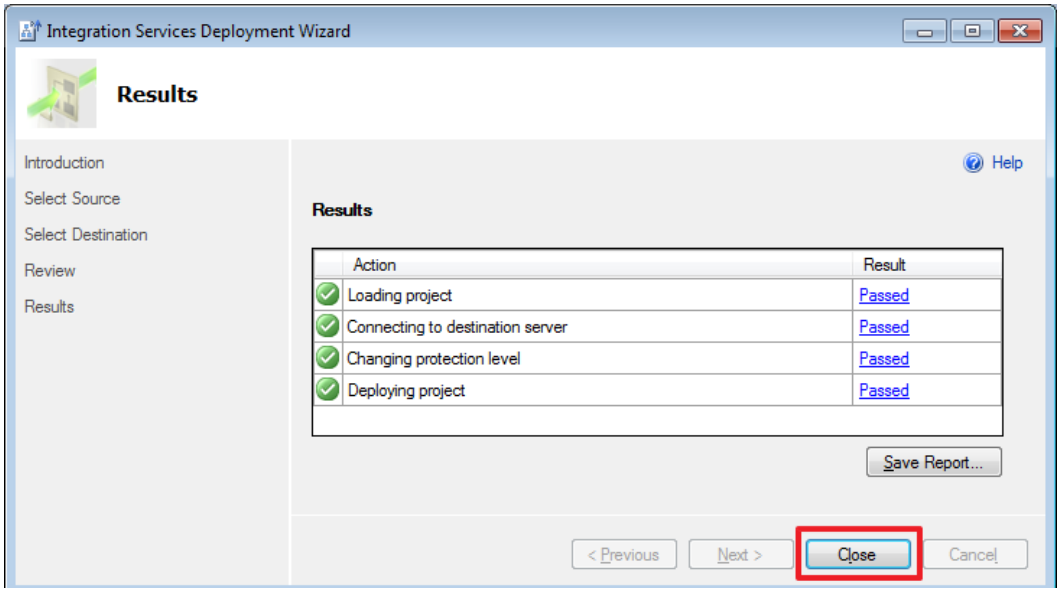




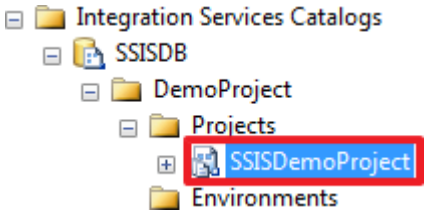




На завершення ми повинні побачити наступну картину:

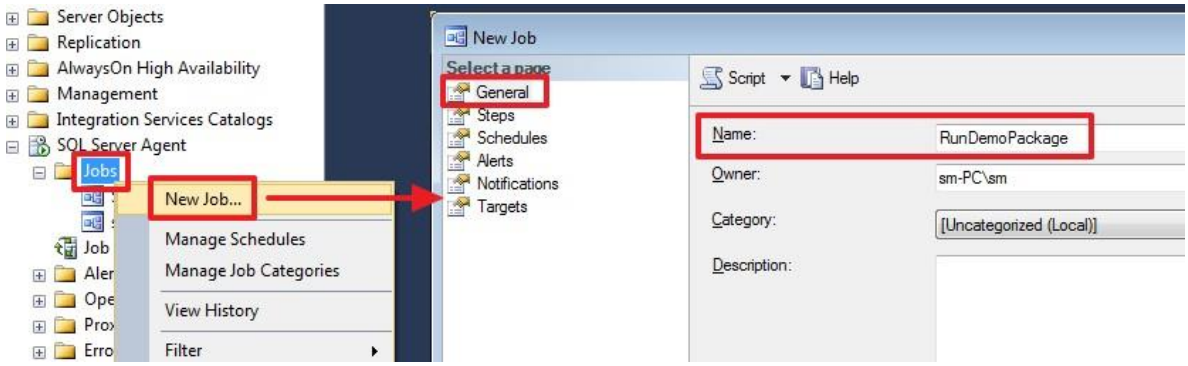


Після поновлення (F5) ми побачимо наш проект:

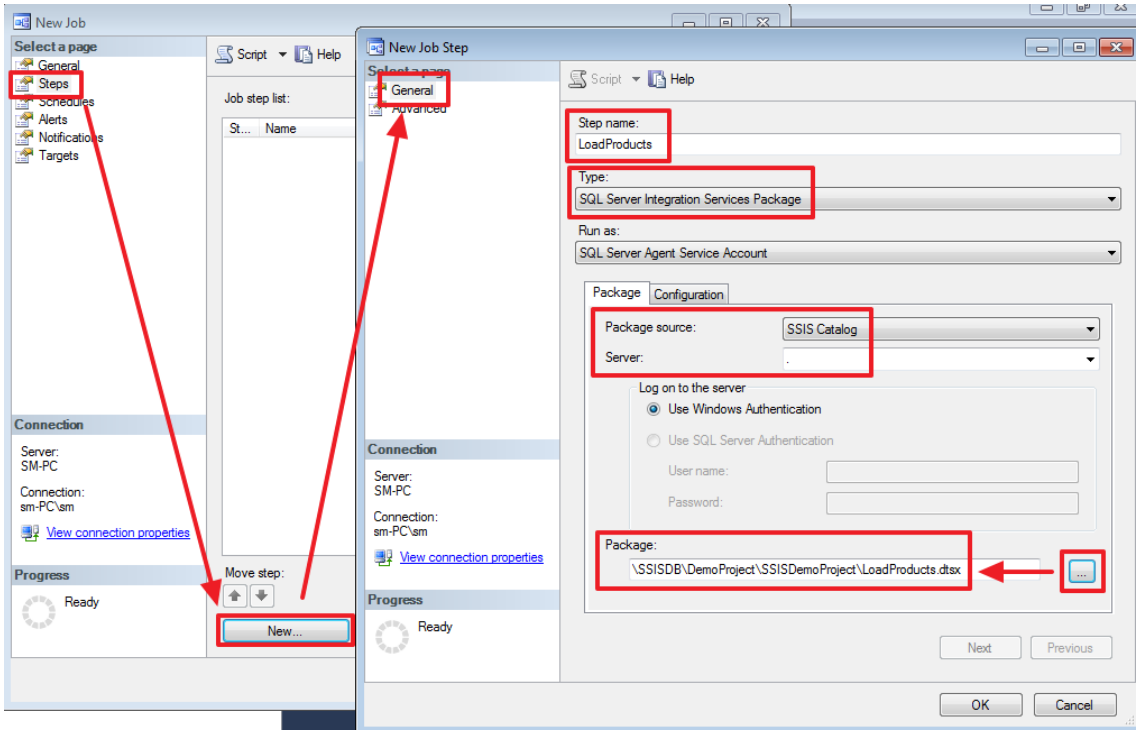


Створення завдання в SQL Server Agent

Створимо завдання в SQL Agent, для виконання пакету за розкладом:

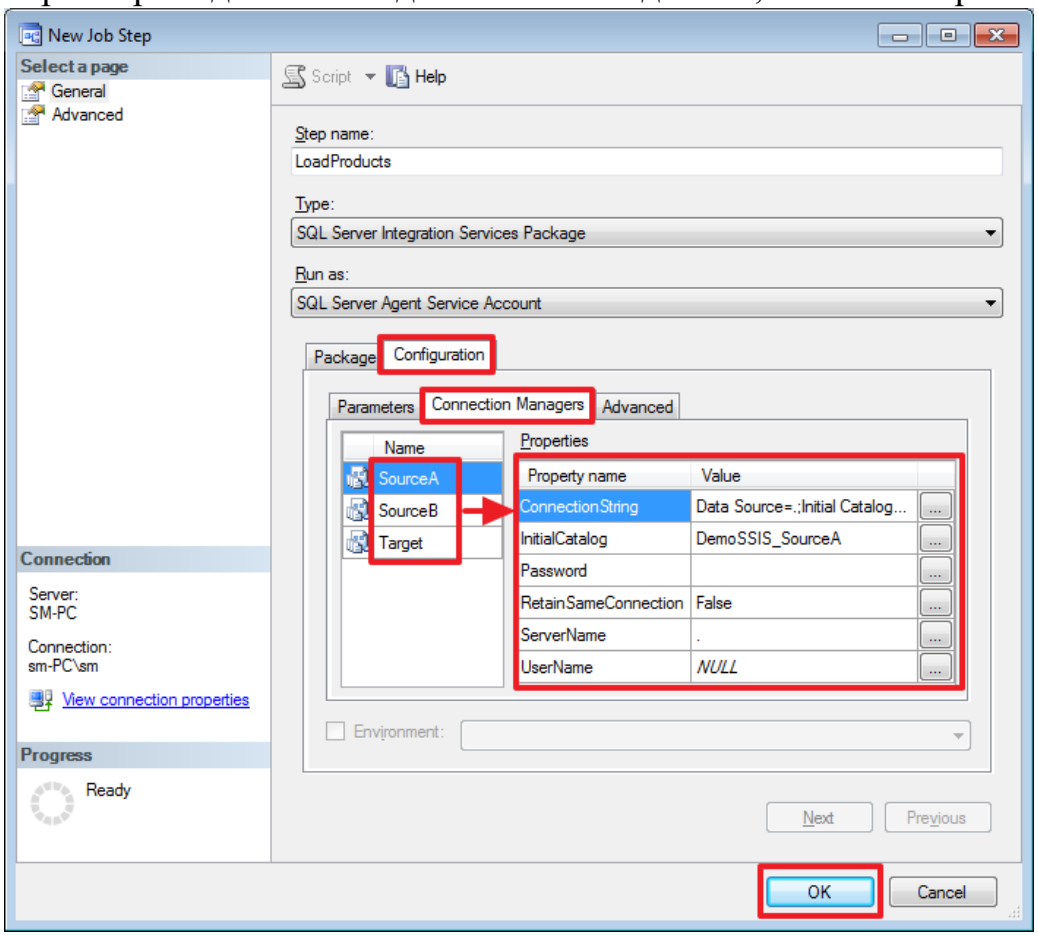


Создаем новый шаг:

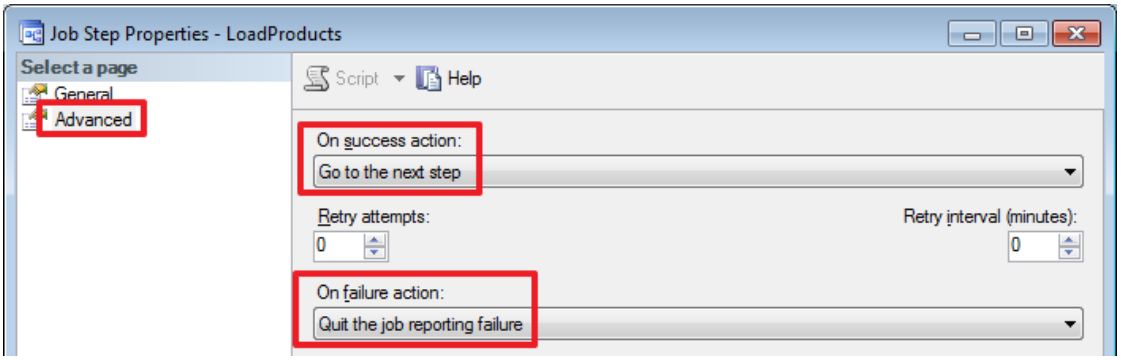


На вкладці «**Configuration → Parameters**» можна задати параметри пакета (їх розглянемо в наступних частинах).

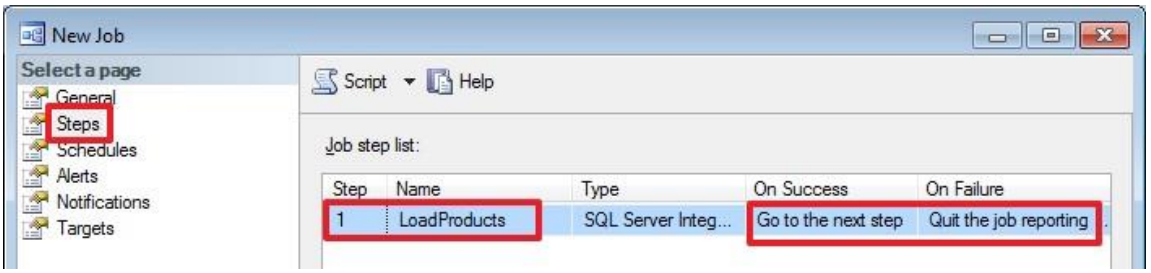
На вкладці «**Configuration → Connection Manager**» ми можемо змінити параметри підключення для кожного з'єднання, яке ми створили в проекті:



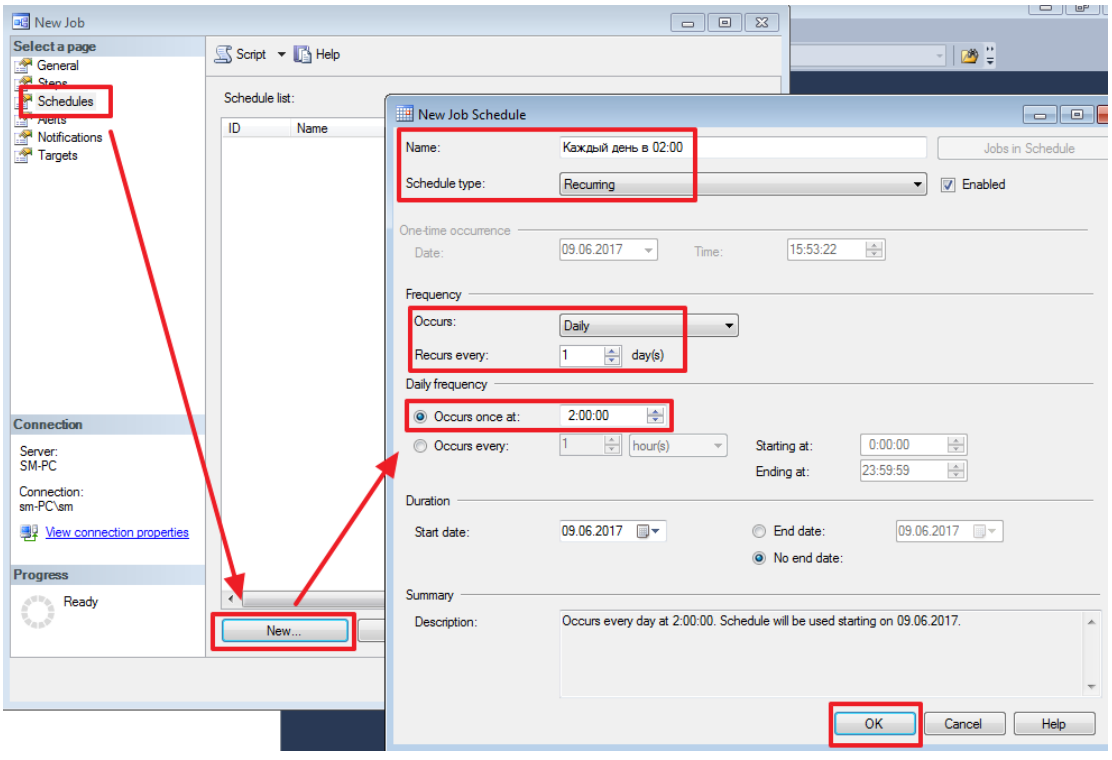
На закладці Advanced можна змінити логіку, яка буде використовуватися при успішному або неуспішному завершення кроку:



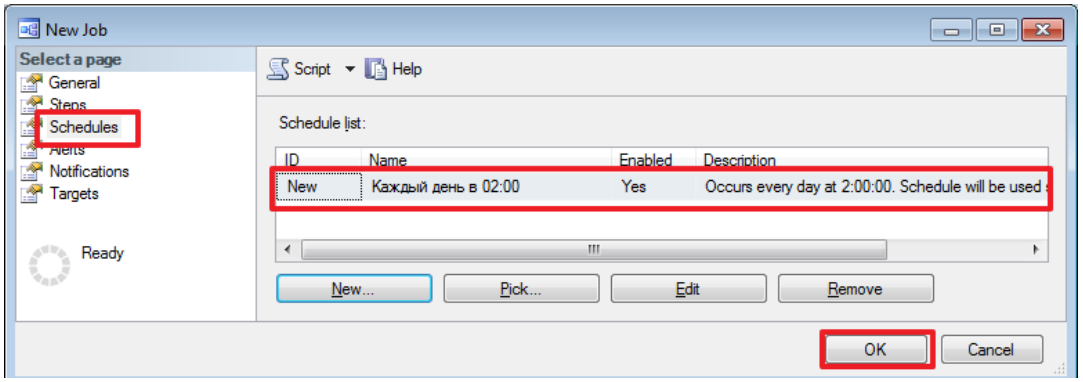
Крок створено:



Залишилося створити розклад для даного завдання:

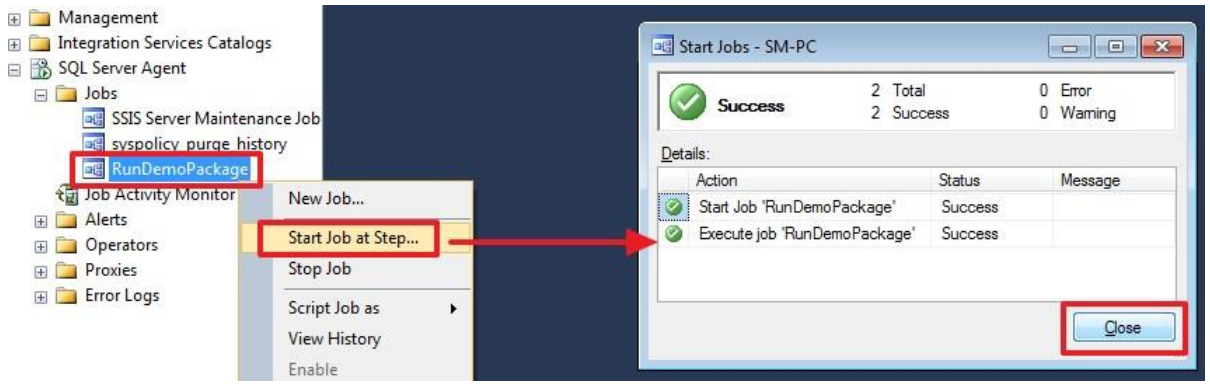


Розклад можна задати різноманітним чином. Думаю, тут все повинно бути інтуїтивно зрозуміло:



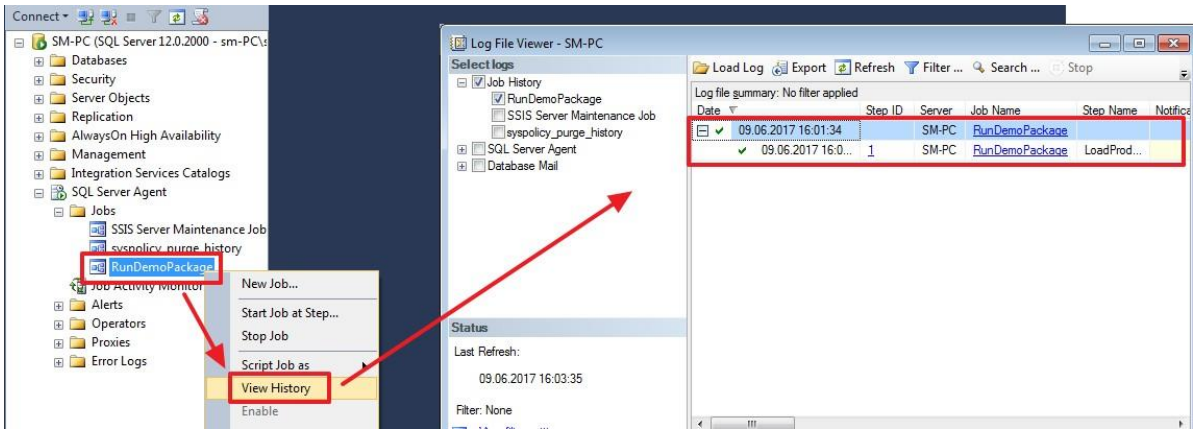
Все, задача створена.

Робимо тестовий запуск:



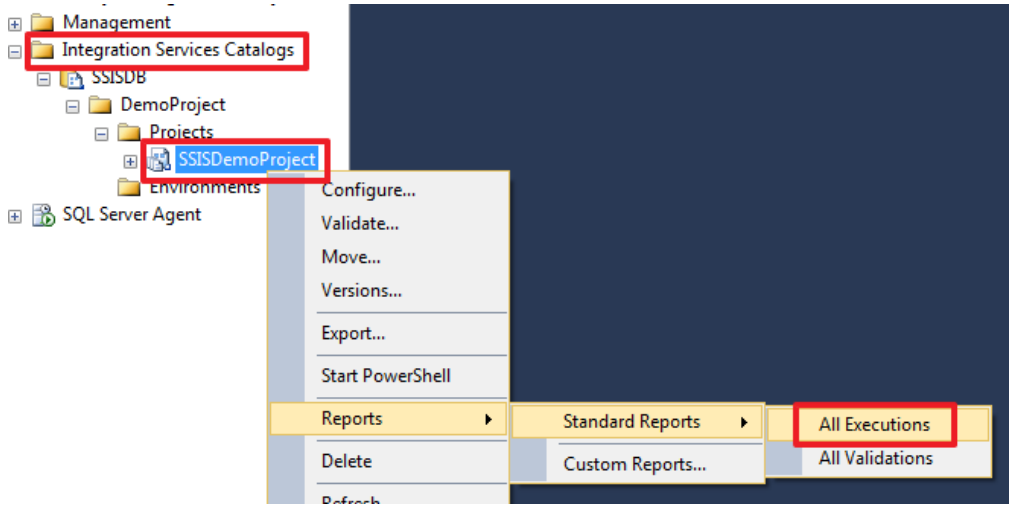
Так як крок у нас всього один, то задача запуститься відразу, інакше потрібно було б вказати з якого кроку потрібно почати виконання.

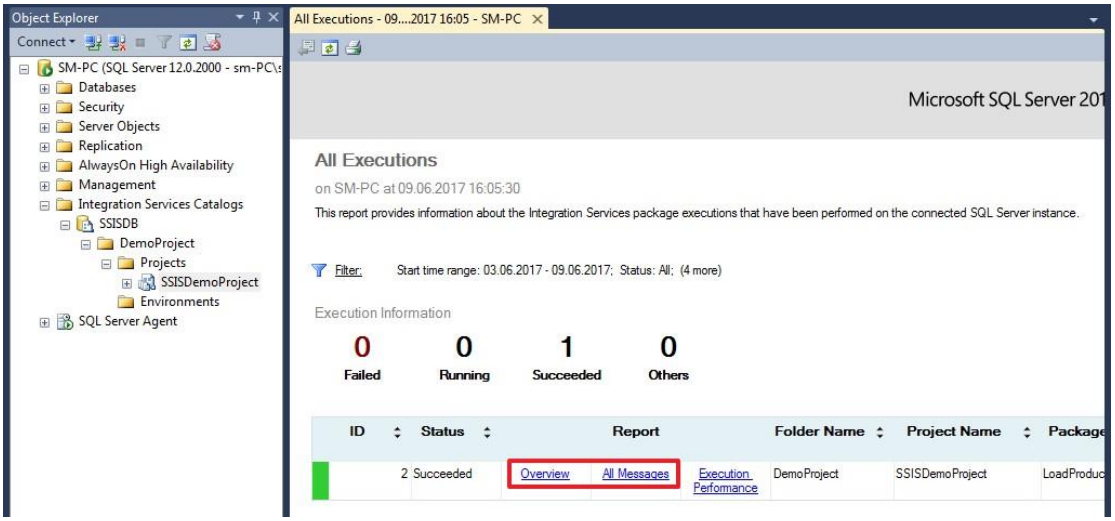
Результат виконання завдання можна побачити в наступному журналі:

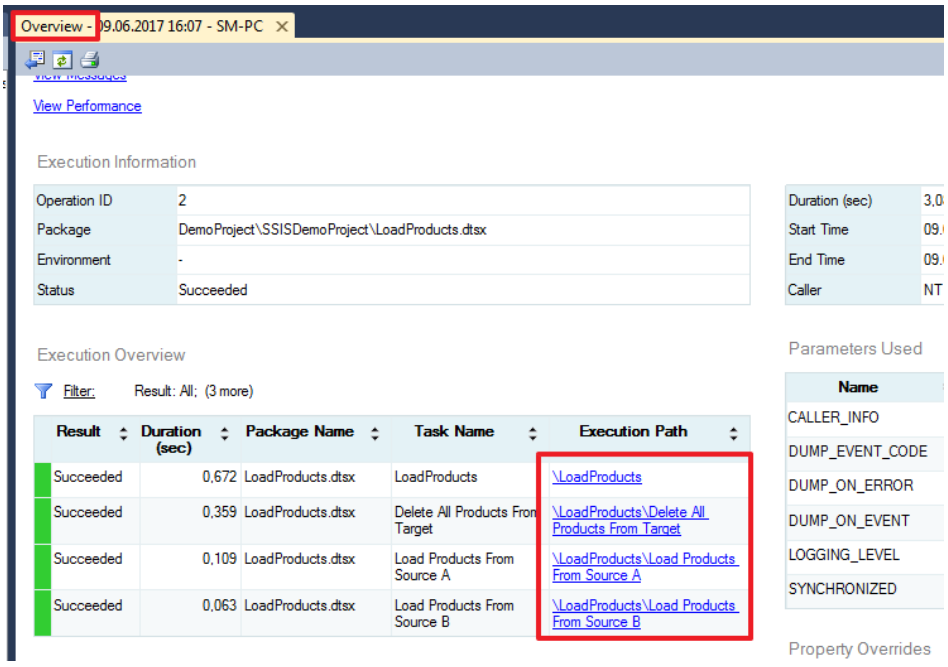


В даному журналі можна побачити успішність завершення кожного кроку, а також час виконання та інші параметри.

Докладніший звіт про виконання пакету можна подивитися за допомогою наступного звіту:







У цій частині змінимо логіку завантаження довідника Products:

За допомогою компонента «Union All» об'єднаємо два вхідних потоку в один; Для нових записів будемо робити вставку, а для записів, які вже були додані раніше будемо робити оновлення. Для поділу записів на додаються і оновлювані скористаємося компонентом Lookup;

Для поновлення записів застосуємо компонент «OLE DB Command».

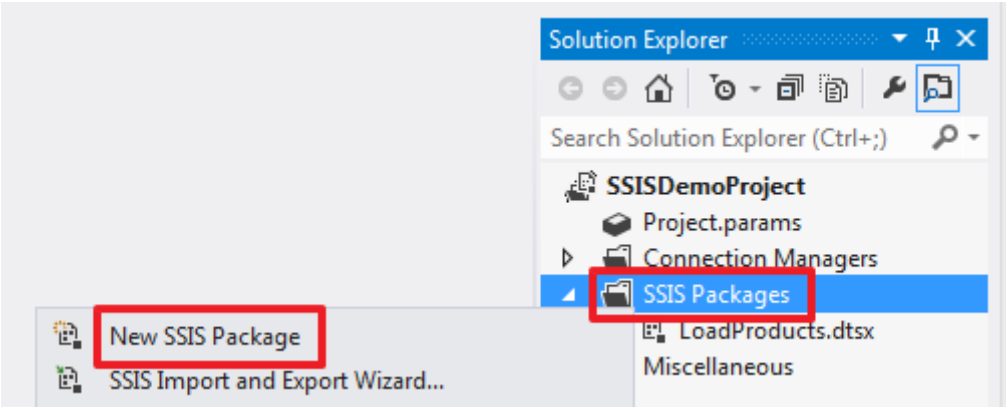
На завершення цієї частини розглянемо компонент Multicast для того щоб распараллелить виходить набір.

Разом в цій частині ми познайомимося з чотирма новими компонентами: Union All, Lookup, OLE DB Command і Multicast.

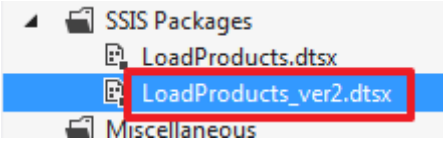
Далі так само буде дуже багато картинок.

Продовжуємо знайомство з SSIS

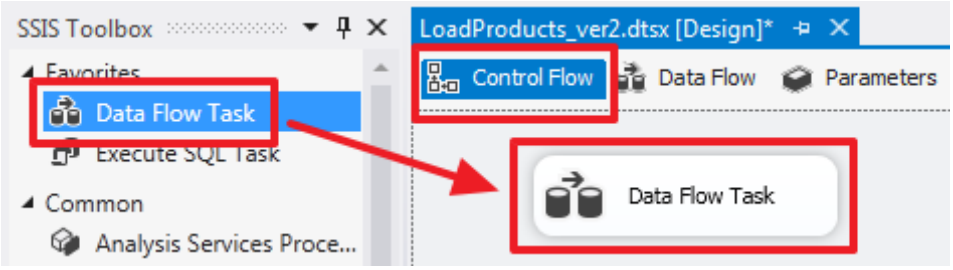
Створимо новий пакет:



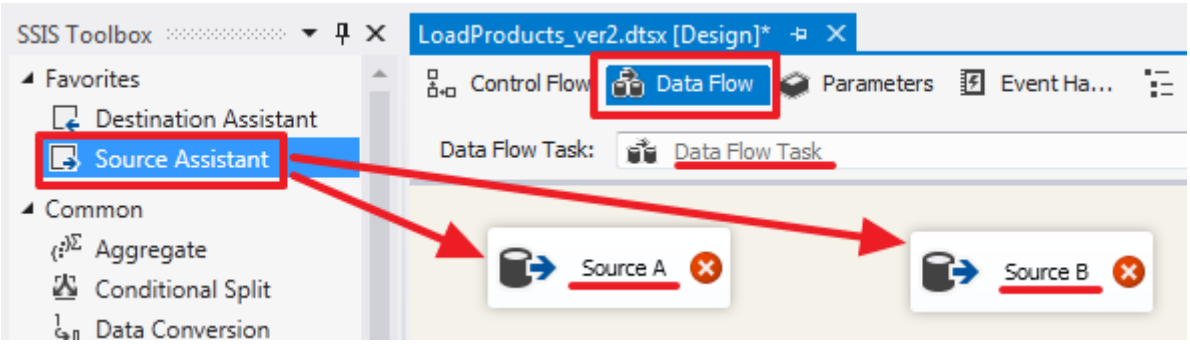
І перейменуємо його у «**LoadProducts_ver2.dtsx**»:



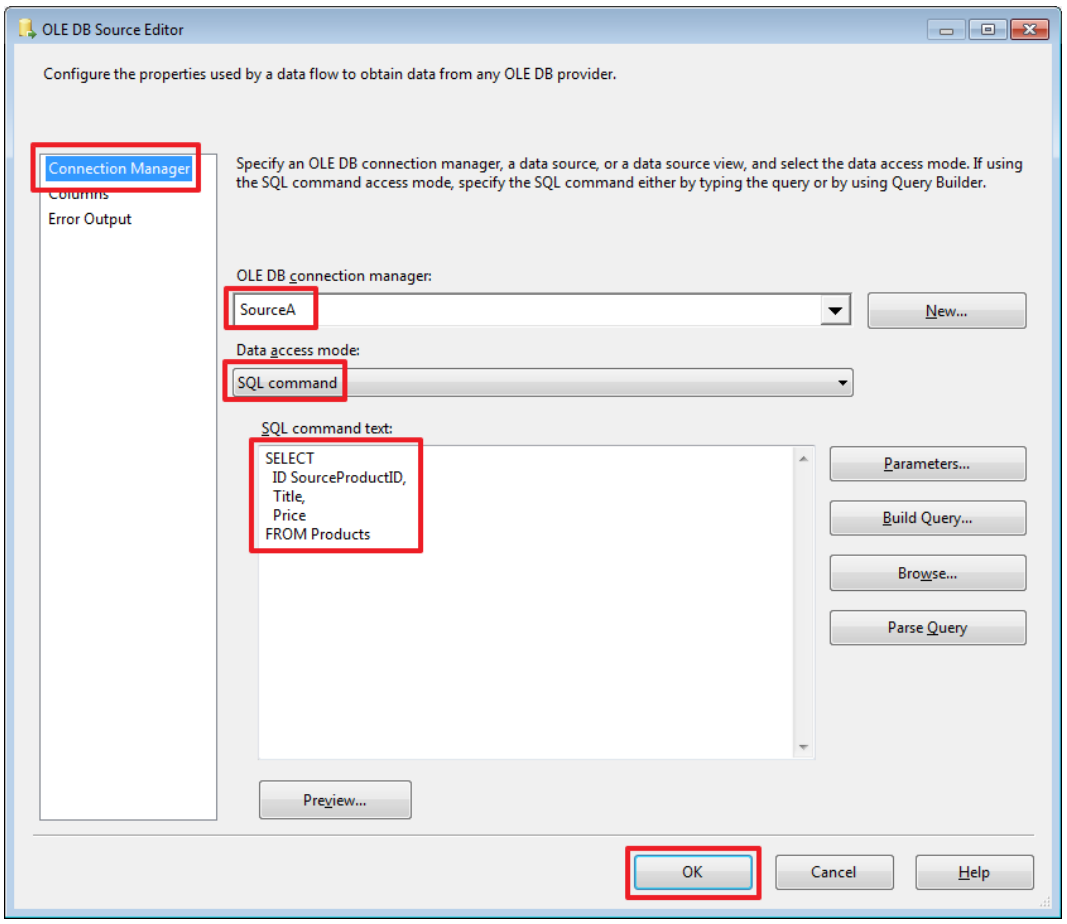
В області «**Control Flow**» створимо «**Data Flow Task**»:



Подвійним клацанням по елементу «Data Flow Task» зайдемо в його область «Data Flow». Створимо два елементи «Source Assistant» для з'єднань SourceA і SourceB. Перейменуємо ці елементи в «Source A» і «Source B» відповідно:



«Source A» налаштуємо наступним чином:

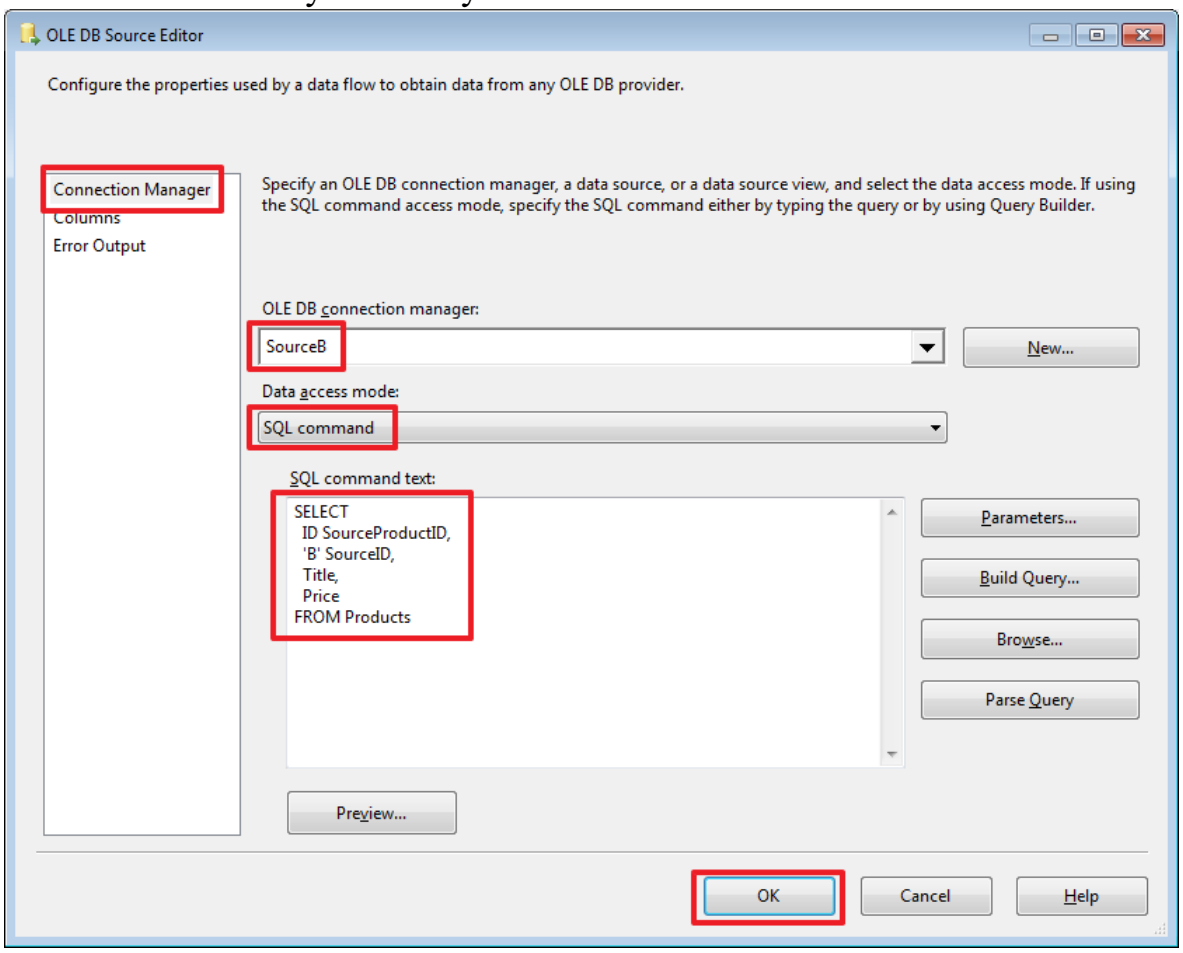


Текст запиту:

```
SELECT
ID SourceProductID,
  Title,
Price
FROMProducts
```

З метою демонстрації великих можливостей за раз, тут я навмисно відпустив SourceID.

«Source B» налаштуємо наступним чином:

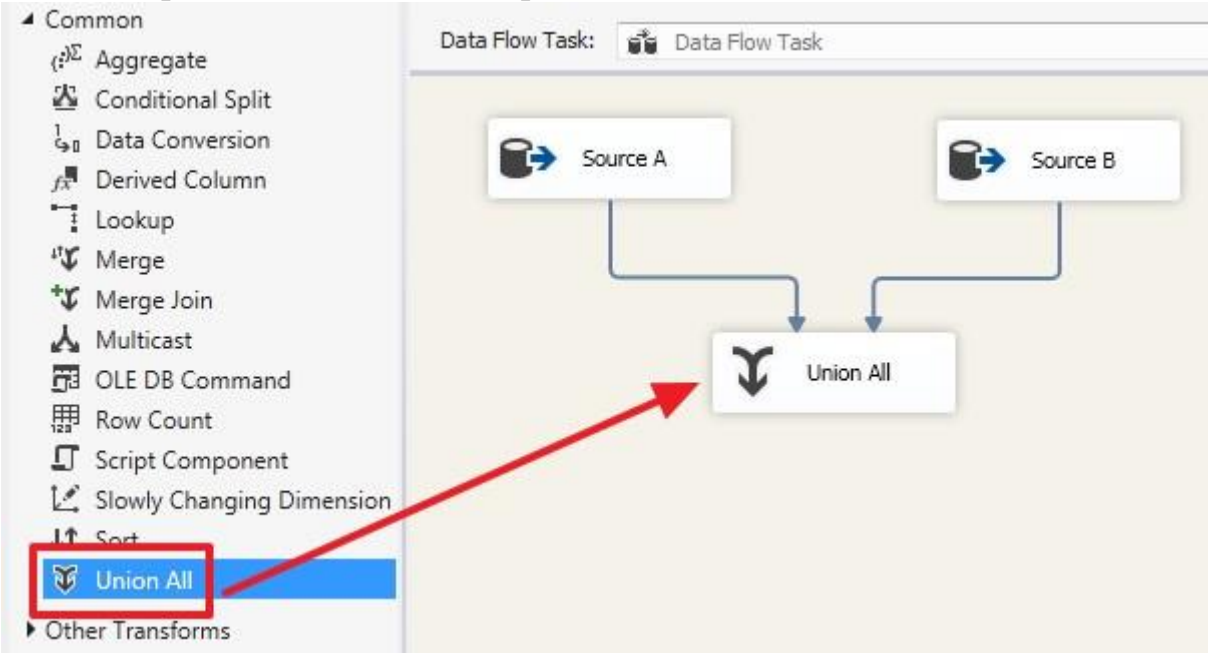


Текст запита:

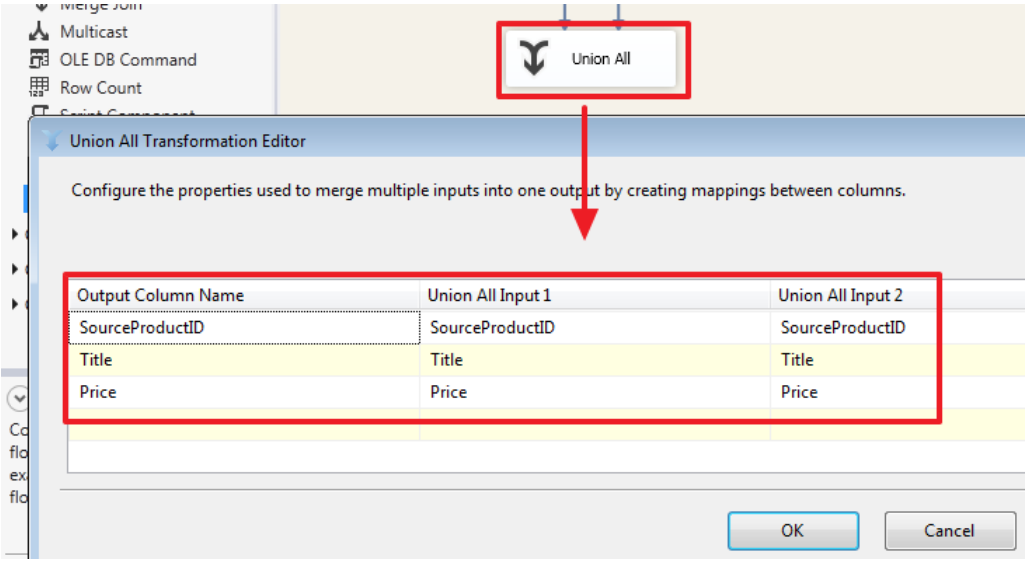
```
SELECT
ID SourceProductID,
'B' SourceID,
  Title,
  Price
FROM Products
```

В результаті набір A у нас буде мати 3 колонки [SourceProductID, Title, Price], а набір B матиме 4 колонки [SourceProductID, SourceID, Title, Price].

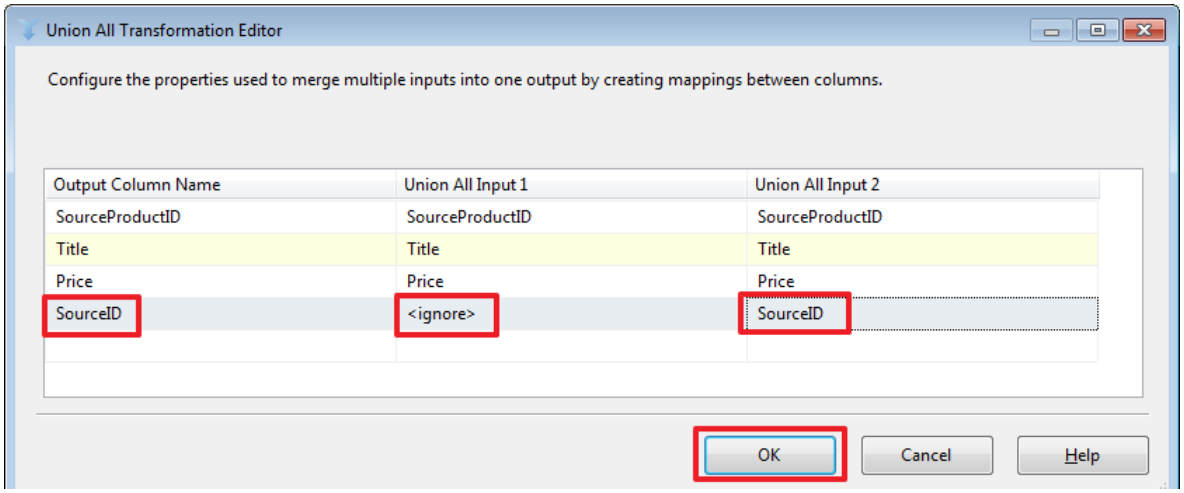
Скористаємося елементом «Union All», щоб об'єднати дані з 2-х наборів в один. Направимо в нього сині стрілки з «Source A» і «Source B»:



Яким чином робиться зіставлення колонок двох вхідних наборів, можна побачити двічі клацнувши на елементі «Union All»:



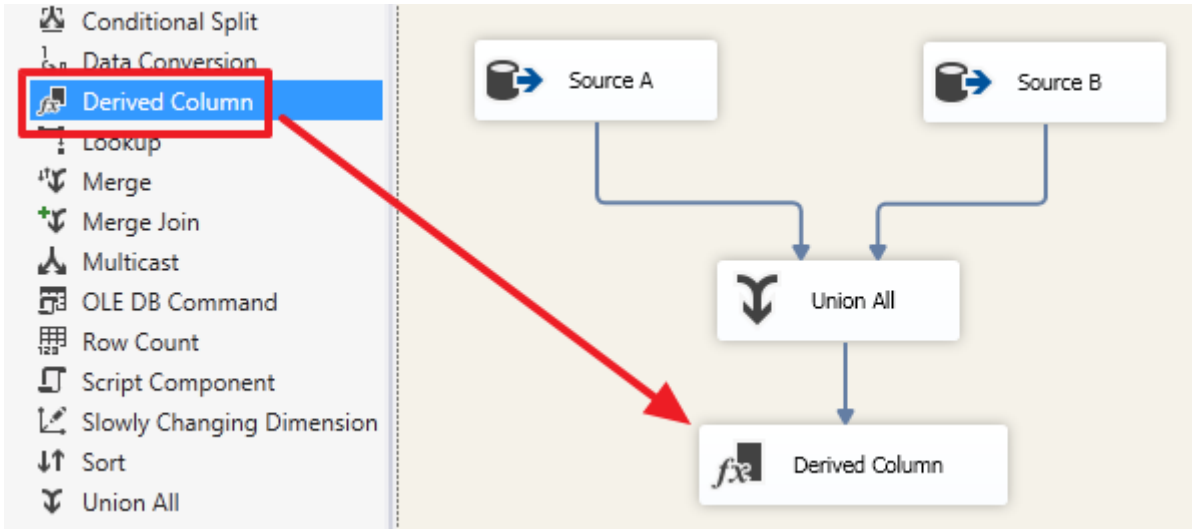
Як ми бачимо, тут зробилося автоматичне співставлення колонок імена яких збігаються. При необхідності ми можемо зробити своє зіставлення, для прикладу додамо колонку SourceID з другого набору:



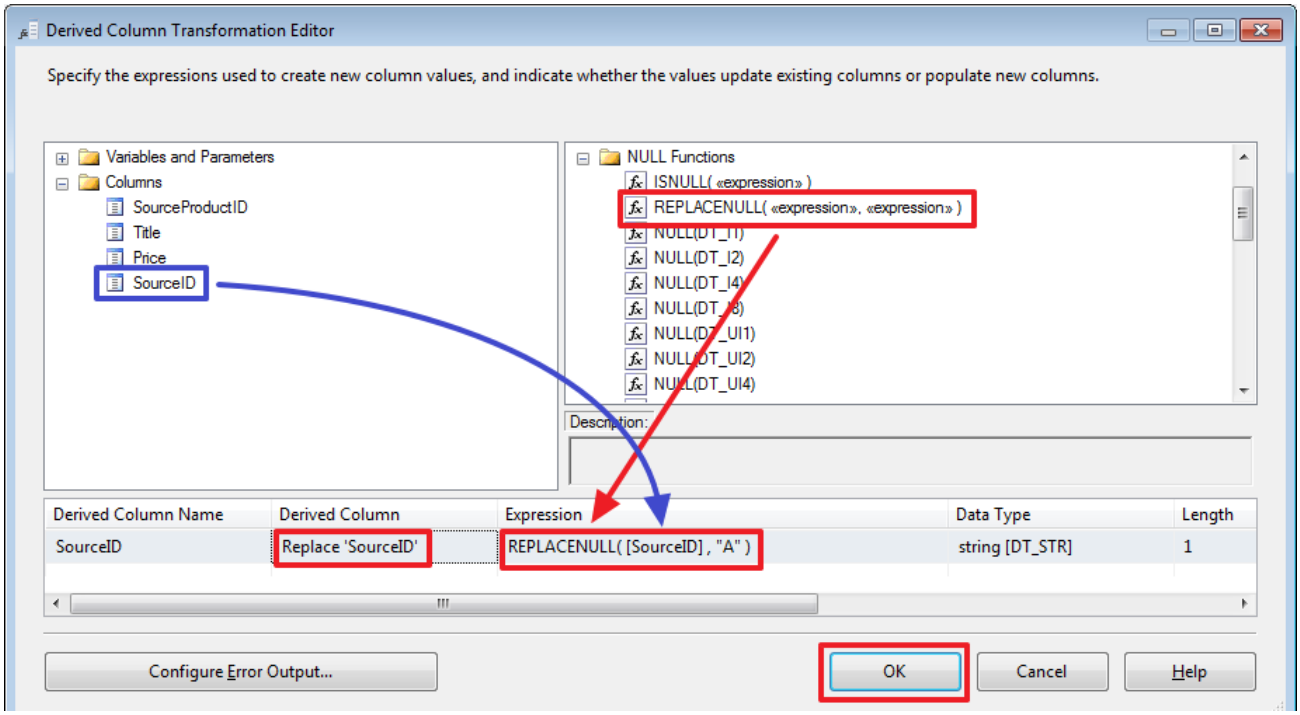
В даному випадку значення SourceID набору «Source A» дорівнюватимуть NULL.

Об'єднання двох наборів в даному випадку робиться на стороні SSIS. Тут варто звернути увагу на те, що бази джерела і приймаюча база можуть розташовуватися на різних серверах / примірниках SQL Server, з цієї причини ми не завжди зможемо так просто написати SQL запит використовуючи в ньому таблиці з різних баз із застосуванням SQL-операції UNION або JOIN (який можна було використовувати замість Lookup описаного нижче).

Для того щоб замінити NULL значення на «A» скористаємося компонентом «Derived Column» в який направимо потік з «Union All»:



Подвійним клацанням зайдемо в редактор «Derived Column» і налаштуємо його наступним чином:

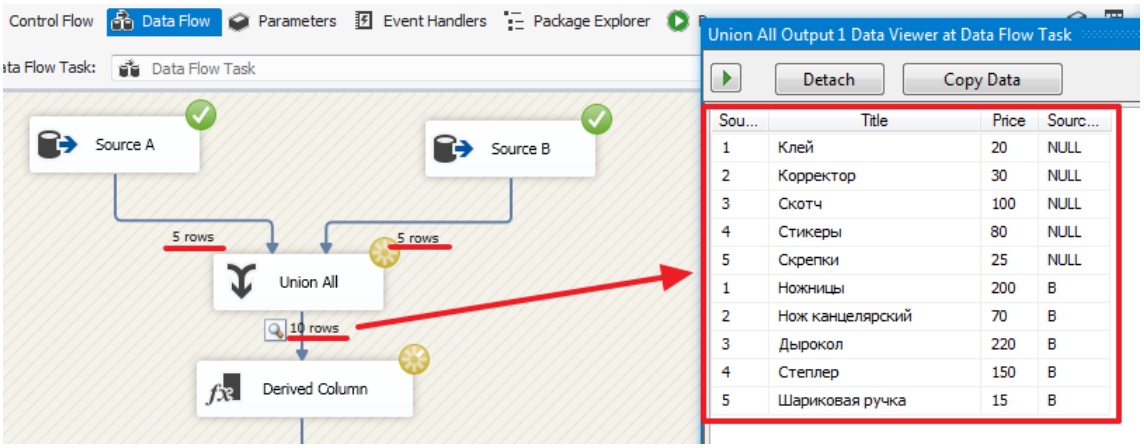


Зробимо наступне (миша в допомогу):

Вкажемо в «Derived Column» значення «Replace 'SourceID'» - це буде означати що ми на виході замінюємо стару колонку SourceID на нову;Перетягнемо в область «Expression» функцію REPLACENULL;Перетягнемо на місце першого аргументу функції REPLACENULL колонку SourceID

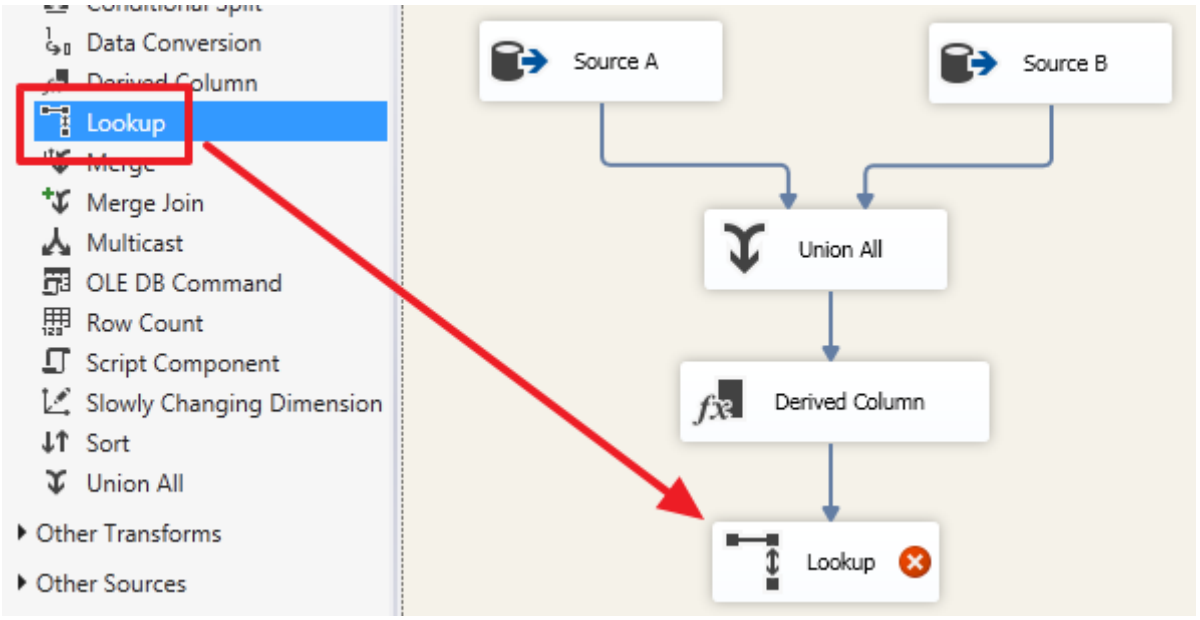
В якості другого аргументу пропишемо константу «A».Для того щоб зрозуміти, що сталося з даними після проходження «Union All» зробіть «Enable Data Viewer» для стрілки, що йде від «Union All» до «Derived Column»:

Тепер при запуску пакета на виконання ви зможете побачити набір, який вийшов в результаті:

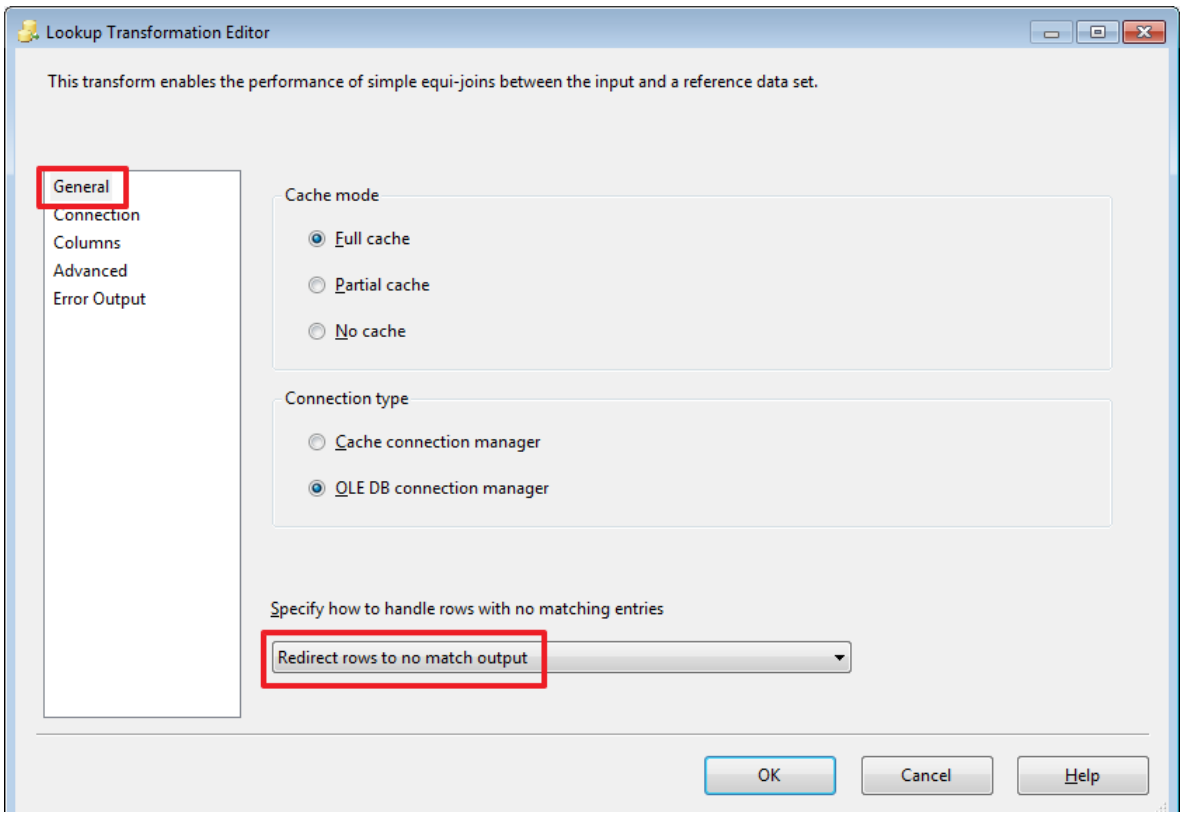


Тут видно, що на цьому етапі (до Derived Column) в колонці SourceID для рядків першого набору стоять значення NULL.

Для того щоб визначити чи була додана раніше запис в базу DemoSSIS_Target скористаємося компонентом Lookup:



Двічі клацнувши по ньому налаштуємо даний елемент:



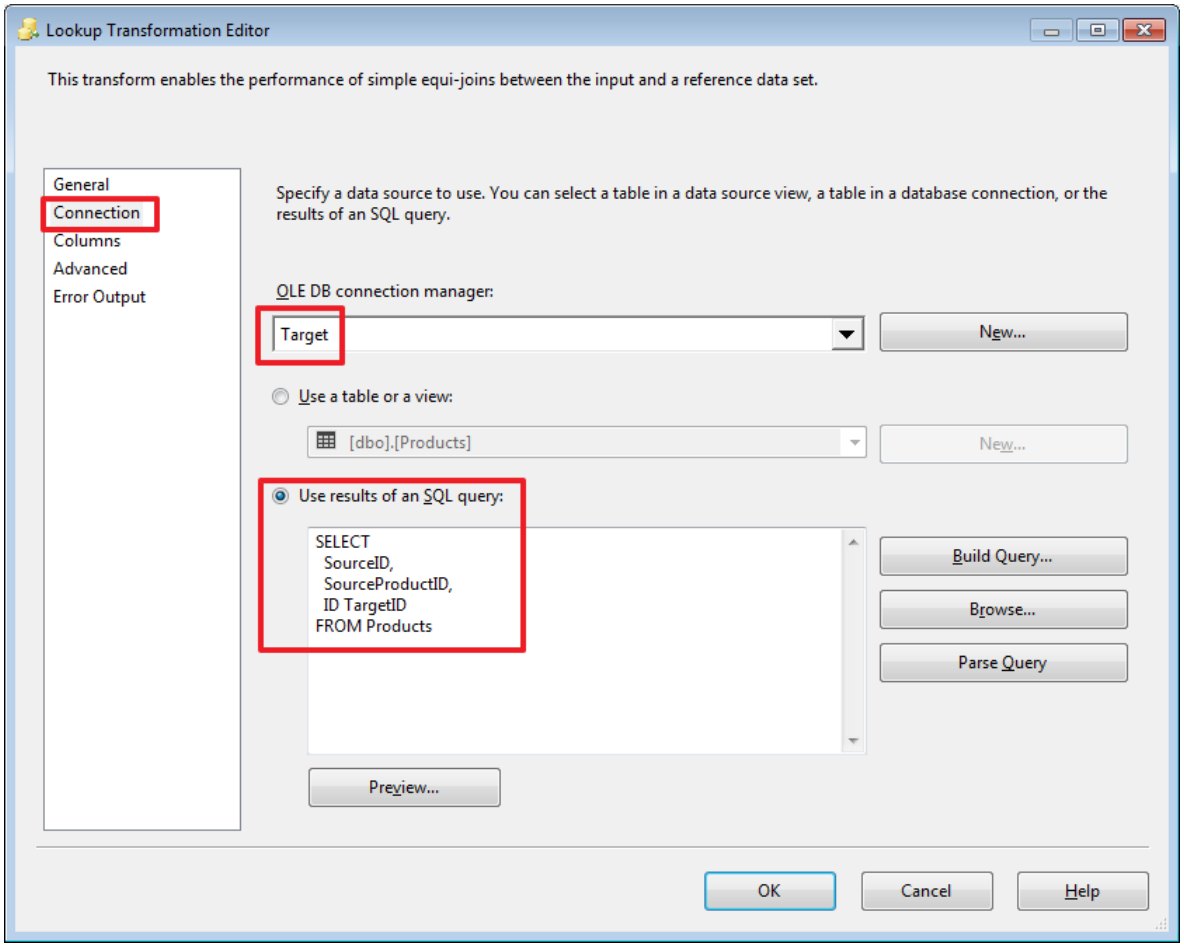
Тут ми скажемо, що ті рядки, для яких не знайдено відповідність, ми будемо перенаправляти в потік «no match output». У цьому випадку на виході ми отримаємо 2 набору «Lookup Match Output» і «Lookup No Match Output».

Наприклад, якщо виставити значення «Ignore failure», то в рядках, для яких не знайшлося зіставлення в поле TargetID (див. Нижче) буде записано значення NULL і всі рядки будуть повернуті через один набір «Lookup Match Output».

«Full cache» говорить про те, що набір, який буде використовуватися в якості довідника одним SQL запитом (см.на наступній вкладці) буде повністю завантажений в пам'ять і рядки будуть зіставлятися вже з кешу без повторних звернень до SQL Server.

Якщо ж вибрати «Partial cache» або «No cache», то на вкладці Advanced можна буде прописати запит з параметрами, який буде виконуватися для зіставлення кожного рядка вхідного набору. Для інтересу можна погратися з цією властивістю і через SQL Server Profiler подивитися які будуть формуватися запити при виконанні пакета.

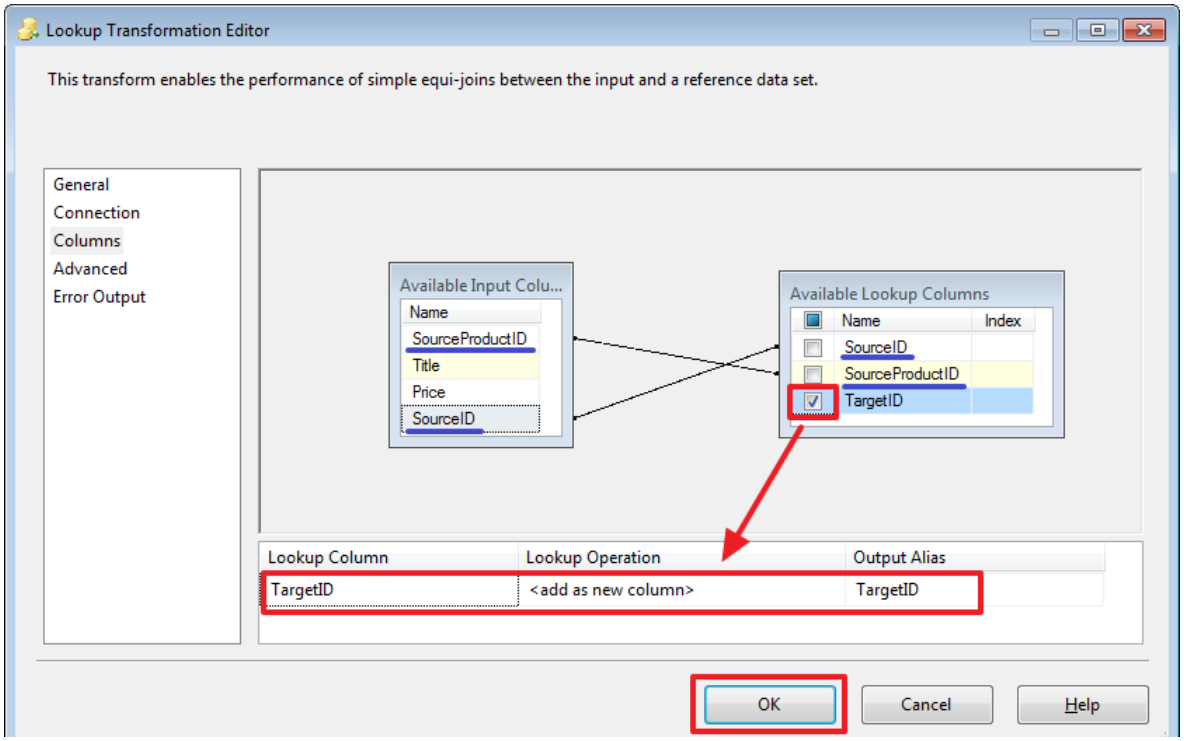
На наступній вкладці нам потрібно визначити набір, який буде виступати в ролі довідника:



Я прописав тут запит:

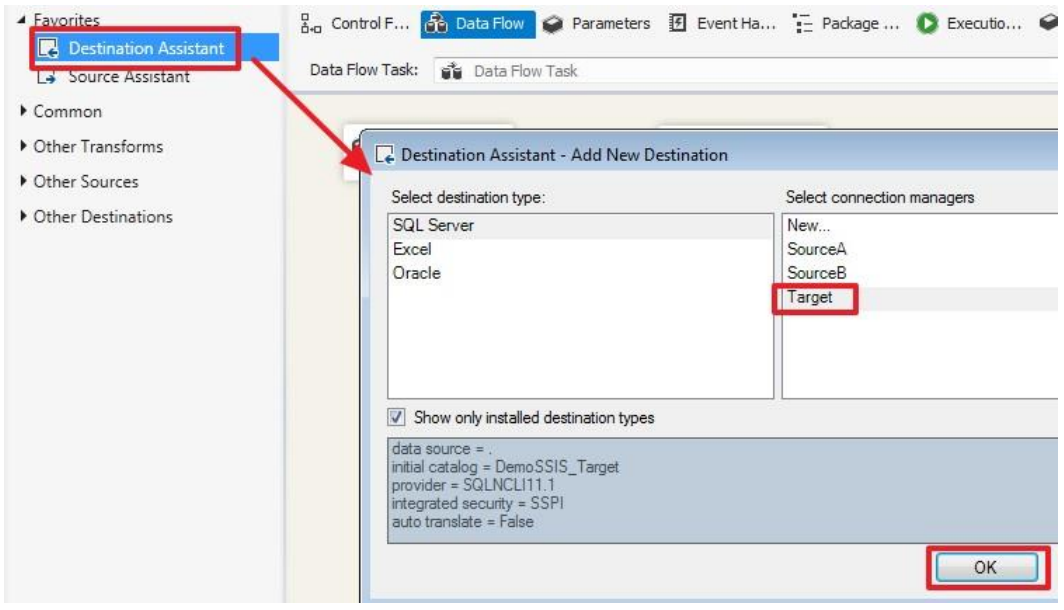
```
SELECT
SourceID,
SourceProductID,
IDTargetID
FROMProducts
```

На наступній вкладці потрібно вказати по яких полях робиться пошук рядка в довіднику і які колонки з довідника потрібно додати в вихідний набір (якщо це потрібно):

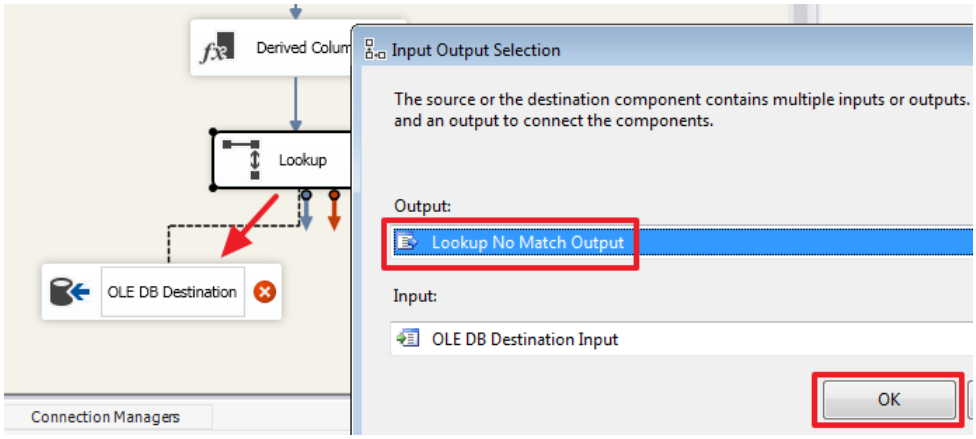


Для визначення зв'язку потрібно за допомогою миші перетягнути поле SourceProductID на SourceProductID і поле SourceID на SourceID.

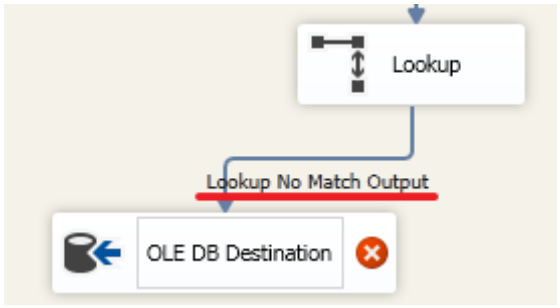
Додамо компонент «Destination Assistant» для вставки записів з потоку «Lookup No Match Output»:



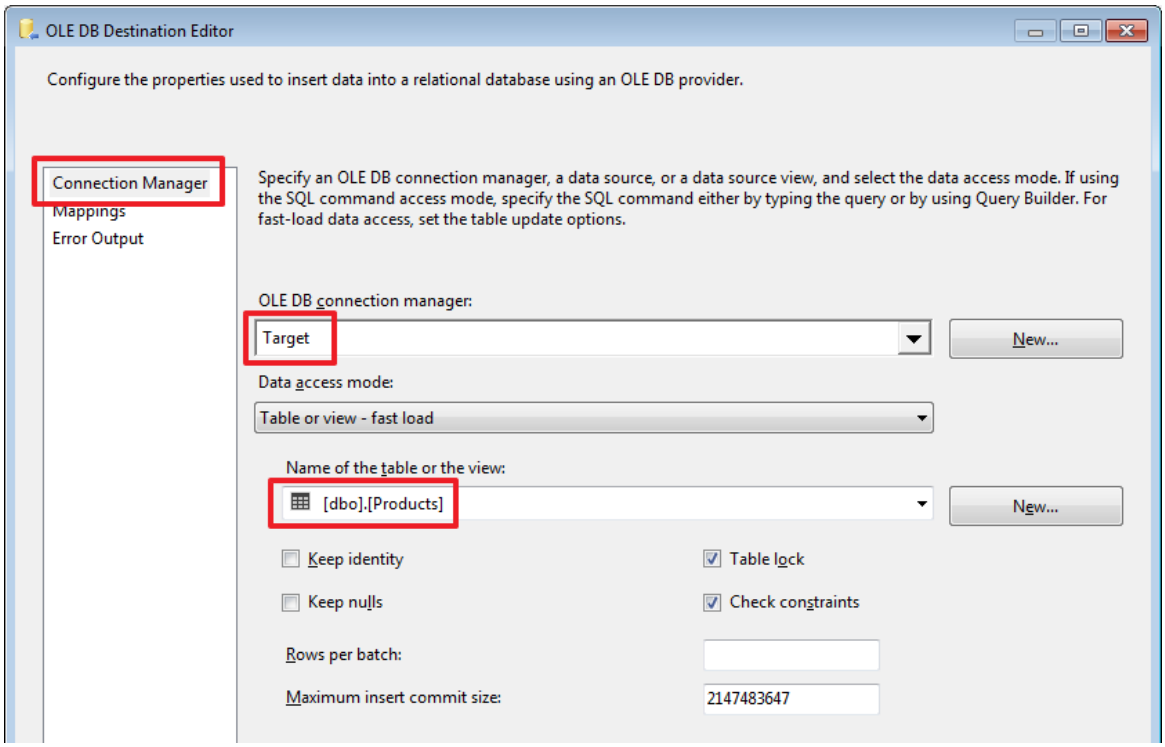
Перетягнемо синю стрілку з «Lookup» на «OLE DB Destination» і в діалоговому вікні виберемо потік «Lookup No Match Output»:

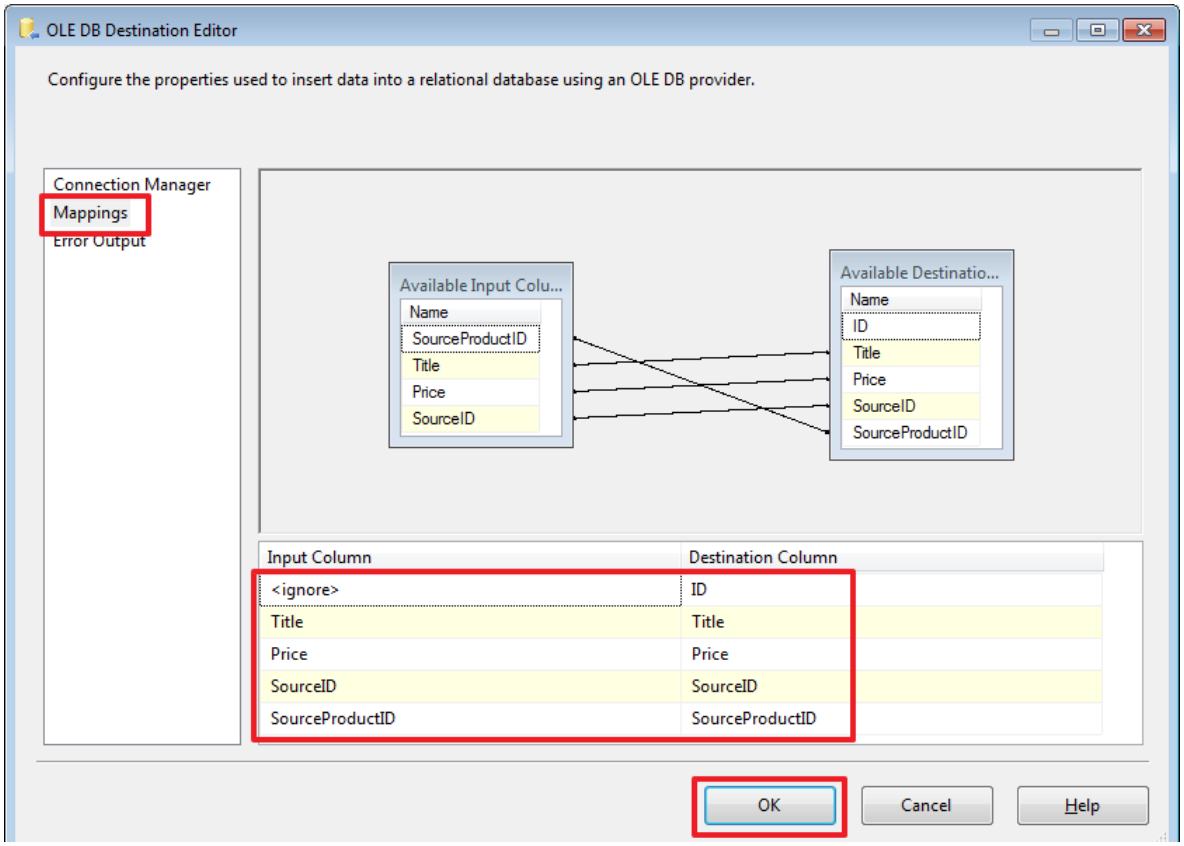


У підсумку ми отримаємо наступне:



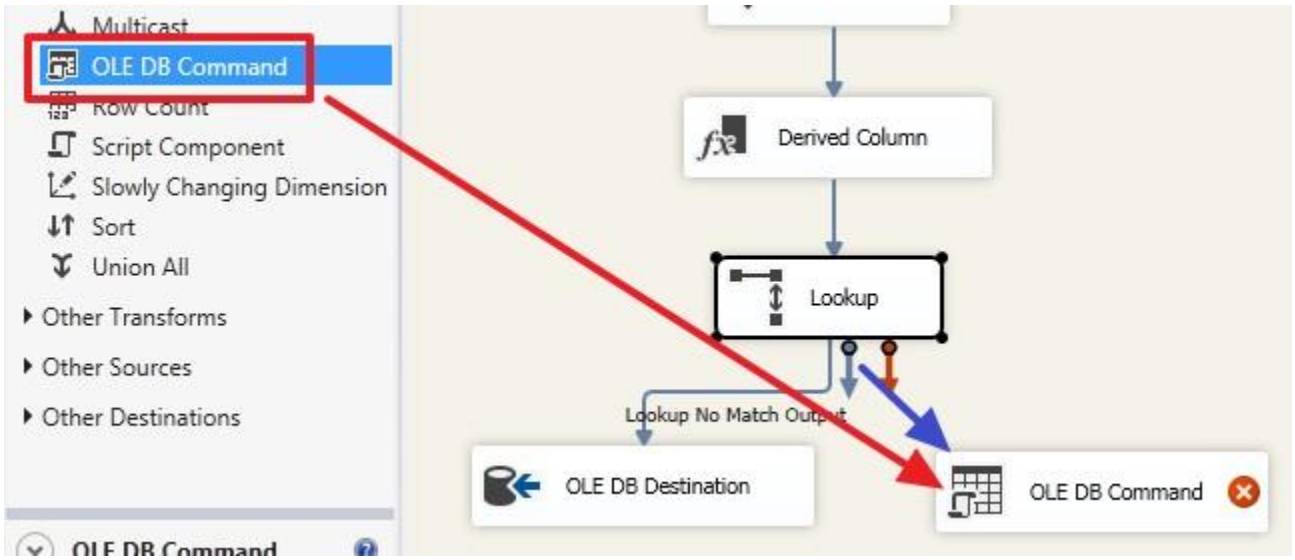
Двічі клацнувши по «OLE DB Destination» налаштуємо його:



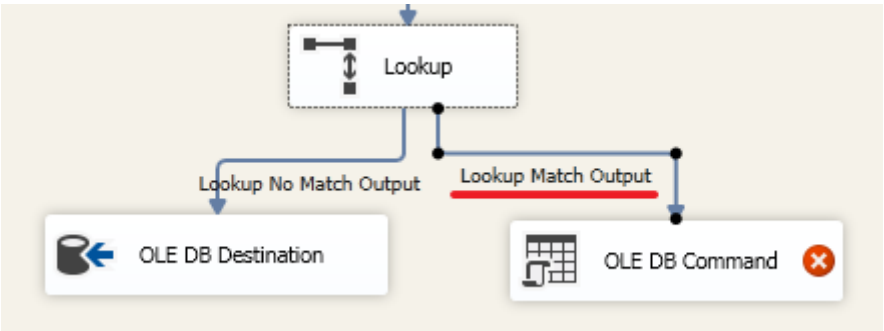


Обробку вставки нових записів ми зробили.

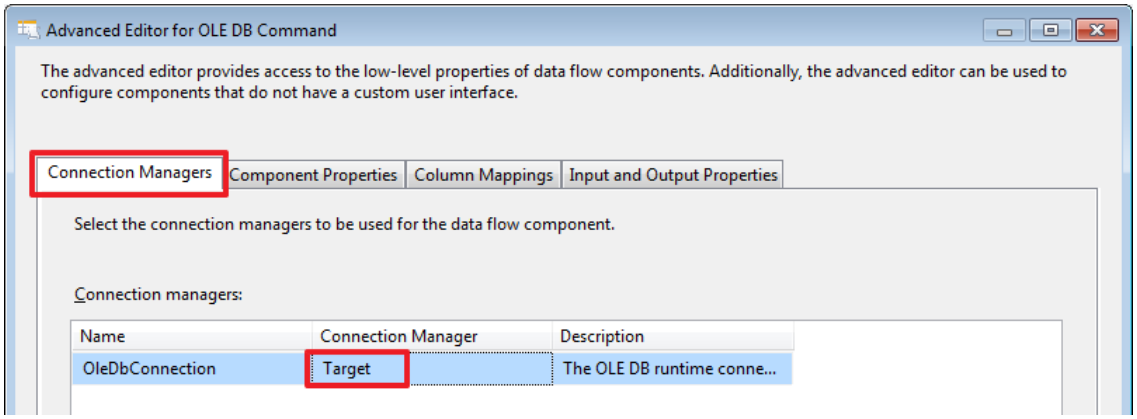
Тепер для поновлення раніше вставлених записів скористаємося компонентом «OLE DB Command» і перенесемо на нього синю стрілку від Lookup:

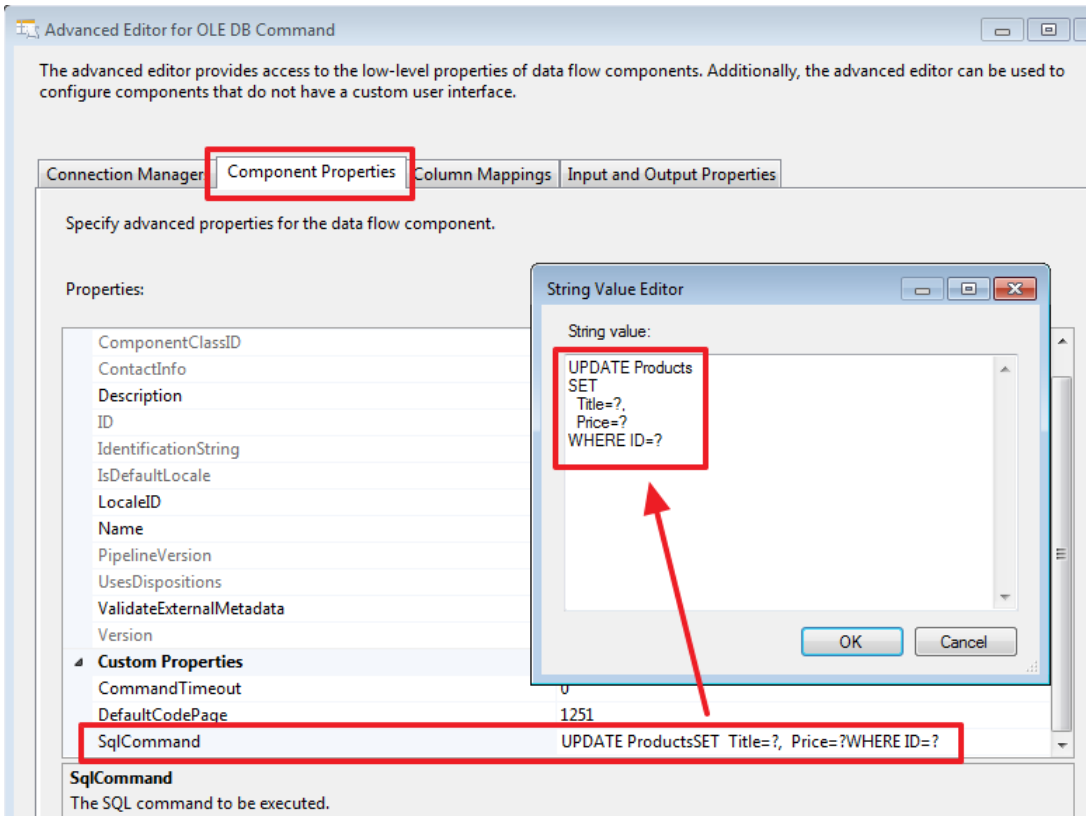


В цей компонент автоматично буде направлений потік «Lookup Match Output», тому що потік «Lookup No Match Output» ми вже вибрали раніше:



Двічі клацнемо на «OLE DB Command» і налаштуємо його:



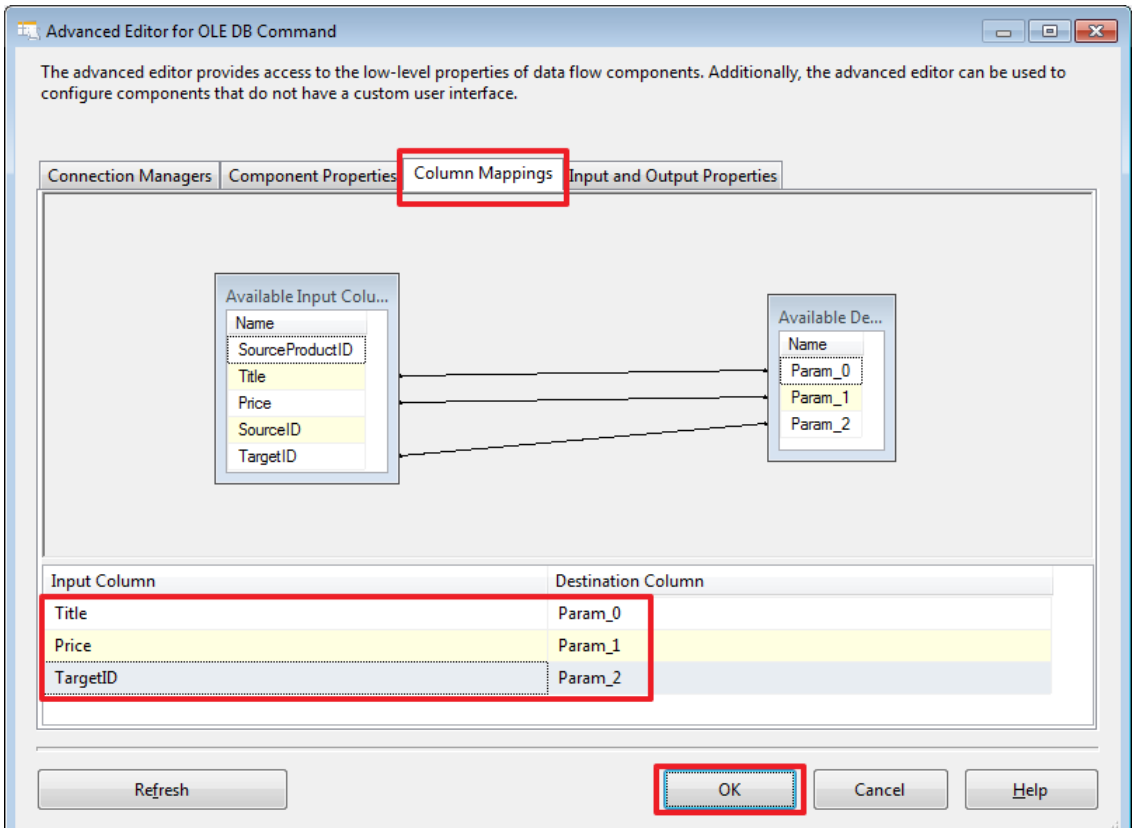


Пропишемо наступний запит на оновлення:

```
UPDATE Products
SET
```

```
 Title=?,
 Price=?
WHEREID=?
```

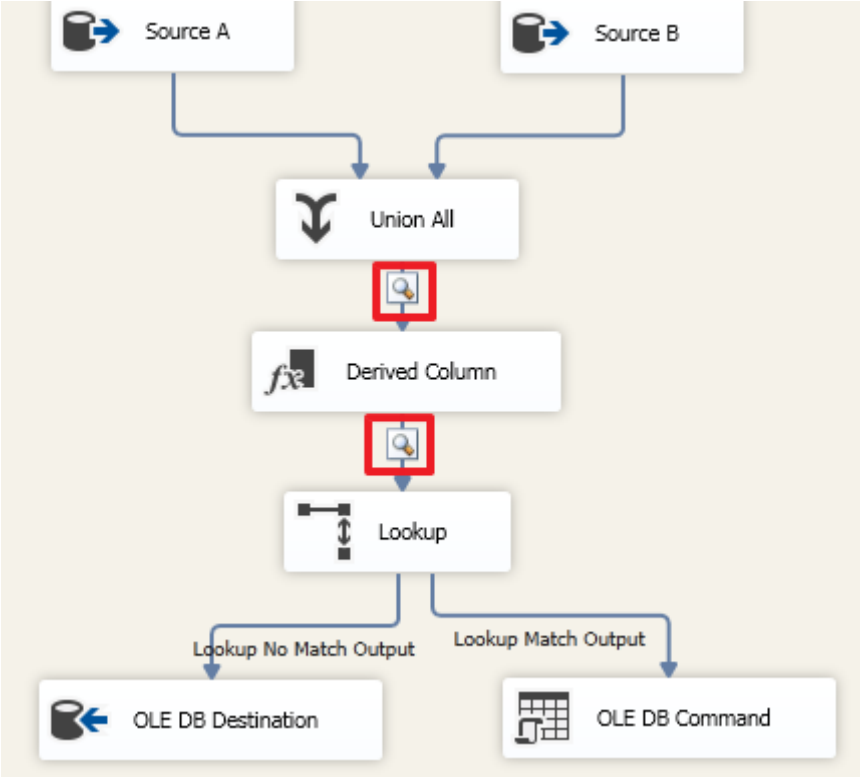
На наступній вкладці вкажемо яким чином будуть задаватися параметри на підставі даних рядків вхідного набору «Lookup Match Output»:



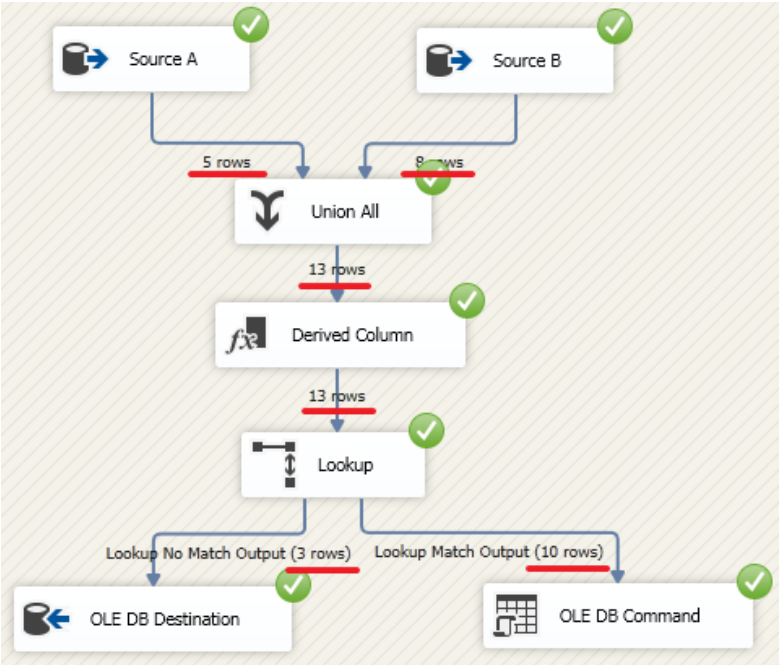
Через SSMS додамо нових продуктів в базу DemoSSIS_SourceB:

```
USE DemoSSIS_SourceB
GO

-- додамо нових товарів
SET IDENTITY_INSERT Products ON

INSERT Products(ID,Title,Price)VALUES
(6,N'Точилка',NULL),
(7,N'Ластик',NULL),
(8,N'Карандашпростой',NULL)

SET IDENTITY_INSERT Products OFF
GO
```

Для того щоб відстежити як змінювалися дані, ви можете, перед запуском пакета на виконання, в необхідних місцях зробити «Enable Data Viewer»:



Запустимо пакет на виконання:



У підсумку ми повинні побачити, що 3 рядки було вставлено за допомогою компонента «OLE DB Destination» і 10 рядків оновлено за допомогою компонента «OLE DB Command».

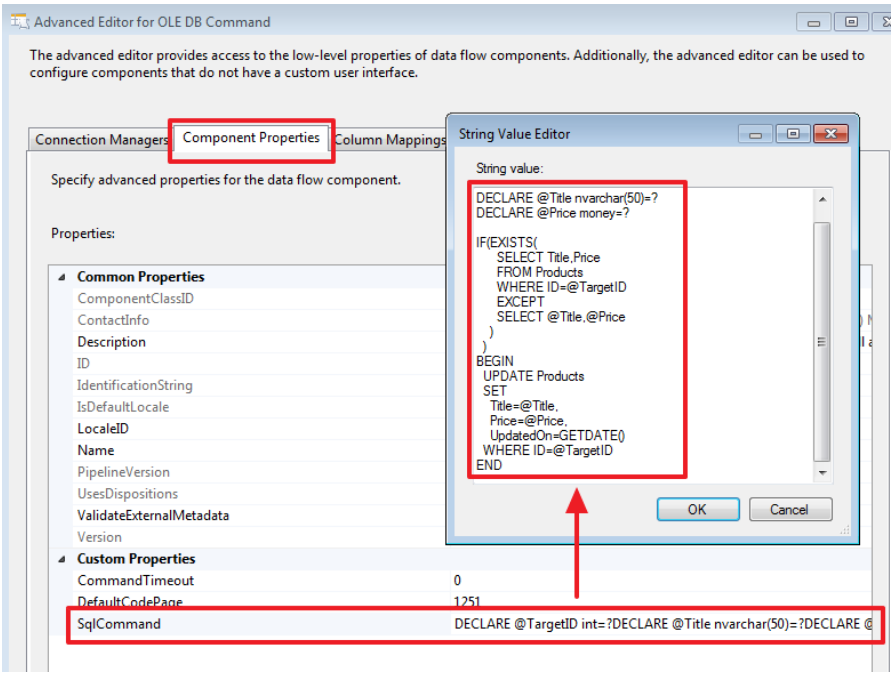
Запит прописаний в «OLE DB Command» виконався для кожного рядка вхідного набору, тобто в даному прикладі 10 разів.

У «OLE DB Command» можна прописати більш складну логіку на TSQL, наприклад, зробити перевірку, чи були змінені Title або Price, і робити оновлення відповідного рядка тільки якщо якесь із значень відрізняється.

Для наочності додамо нову колонку в таблицю Products в базі DemoSSIS_Target:

```
USE DemoSSIS_Target
GO

ALTERTABLE Products ADD UpdatedOn datetime
GO
```

Давайте тепер пропишемо наступну команду:



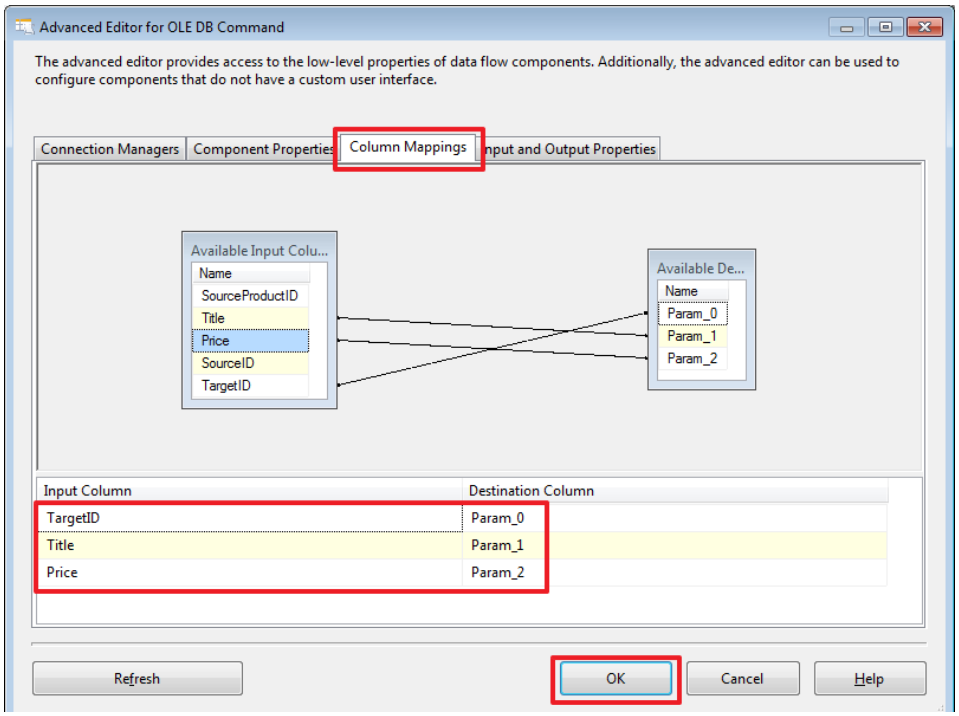
Текст команди:

```
DECLARE @TargetID int=?
DECLARE @Title nvarchar(50)=?
DECLARE @Price money=?

IF(EXISTS(
SELECTTitle,Price
FROM Products
WHEREID=@TargetID
EXCEPT
```

```
SELECT @Title,@Price
    )
  )
BEGIN
UPDATE Products
SET
    Title=@Title,
    Price=@Price,
    UpdatedOn=GETDATE()
WHEREID=@TargetID
END
```

Так само можна було б все це оформити у вигляді процедури, що, а тут прописати її через виклик «EXEC ProcName?,?,?». Тут, думаю, кому як зручніше, мені часом зручніше, щоб все було прописано в одному місці, тобто в SSIS-проект. Але якщо використовувати процедуру, то теж отримуємо свої зручності, в цьому випадку можна, було б просто змінити процедуру і уникнути переробки та повторного розгортання SSIS-проекту.

Після чого перевизначити прив'язку параметрів згідно їх черговості в тексті команди:

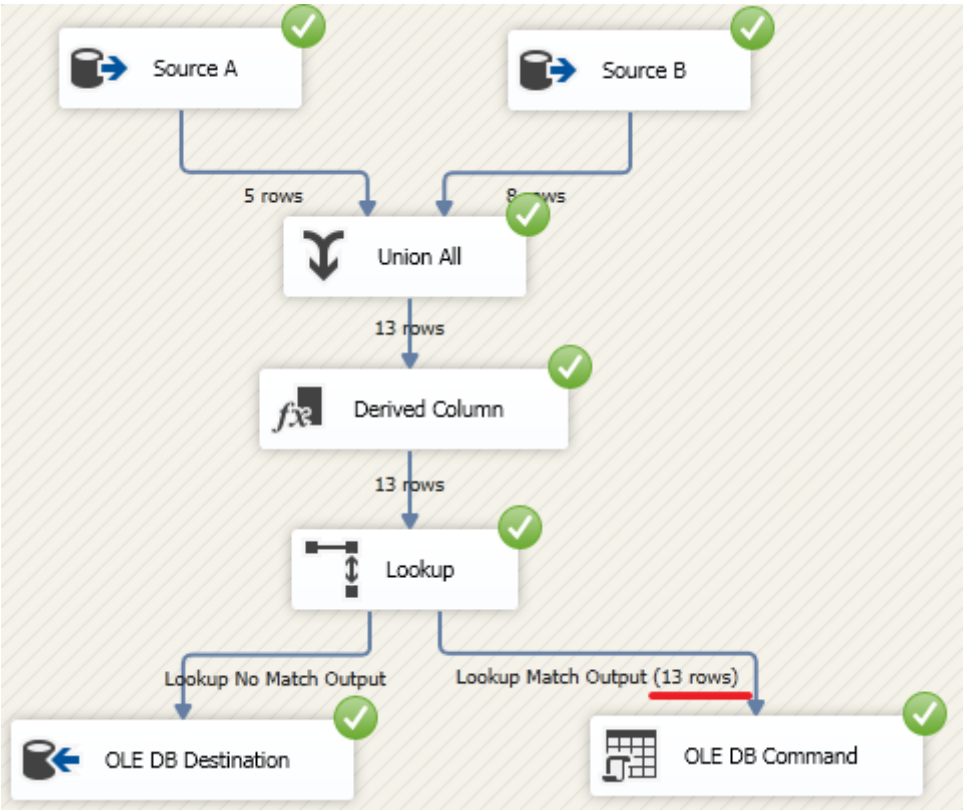


Зробимо в базі DemoSSIS_SourceA оновлення:

```
USE DemoSSIS_SourceA
```

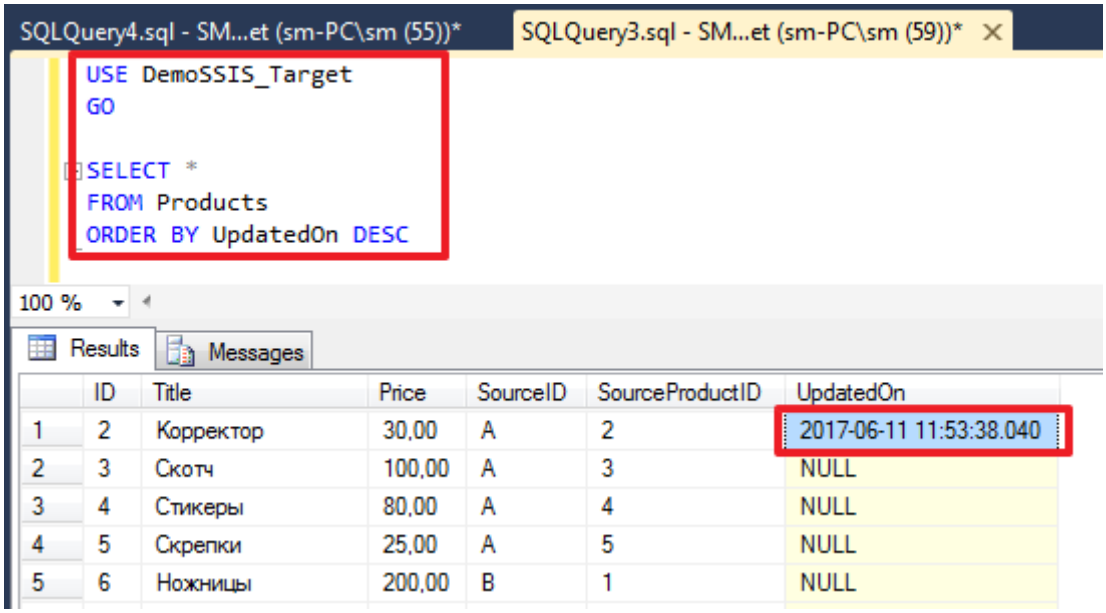
```
GO

UPDATE Products
SET
  Price=30
WHEREID=2-- Коректор
```

І знову запустимо проект на виконання. В результаті після чергового запуску пакета на виконання, UPDATE повинен буде виконатися тільки 1 раз, тільки для цього запису.



Після виконання пакету перевіримо це за допомогою запиту:

```
USE DemoSSIS_Target
GO

SELECT *
FROM Products
ORDERBY UpdatedOn DESC
```

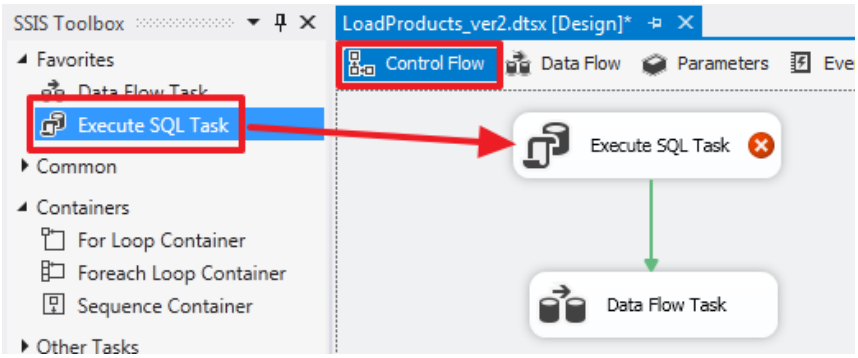


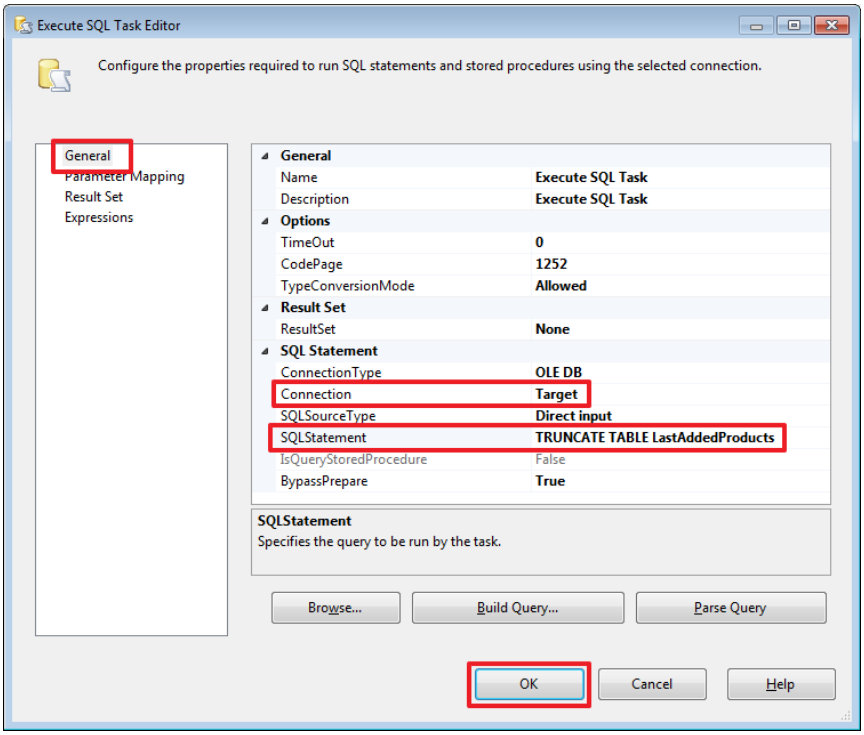
В рамках даної частини розглянемо ще компонент «Multicast». Даний компонент дозволяє отримати з одного потоку кілька. Це може бути корисно, коли одні і ті ж дані необхідно записати в два або більше різних місць - тобто входить один набір, а виходить стільки його копій скільки нам потрібно, і з кожною копією цього набору ми можемо робити що захочемо.

Для прикладу створимо в базі DemoSSIS_Target ще одну таблицю LastAddedProducts:

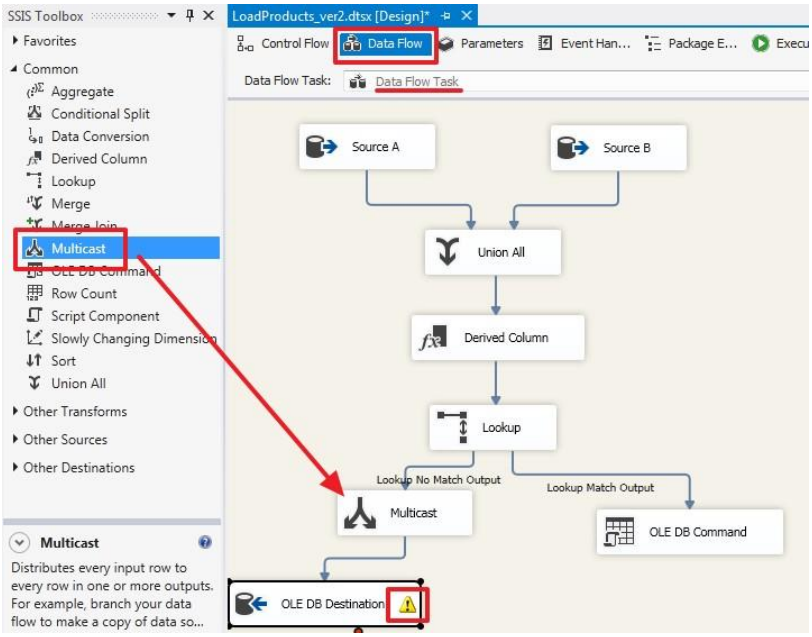
```
USE DemoSSIS_Target
GO

CREATETABLELastAddedProducts(
SourceID char(1) NOTNULL, -- використовується для ідентифікації джерела
SourceProductID intNOTNULL, -- ID в джерелі
 Title nvarchar(50) NOTNULL,
 Price money,
CONSTRAINT PK_LastAddedProducts PRIMARY
KEY(SourceID,SourceProductID),
CONSTRAINT CK_LastAddedProducts_SourceID CHECK(SourceID IN('A','B'))
)
GO
```

Для очищення цієї таблиці додамо в область «Control Flow» компонент «Execute SQL Task» і пропишемо в ньому команду «TRUNCATE TABLE LastAddedProducts»:

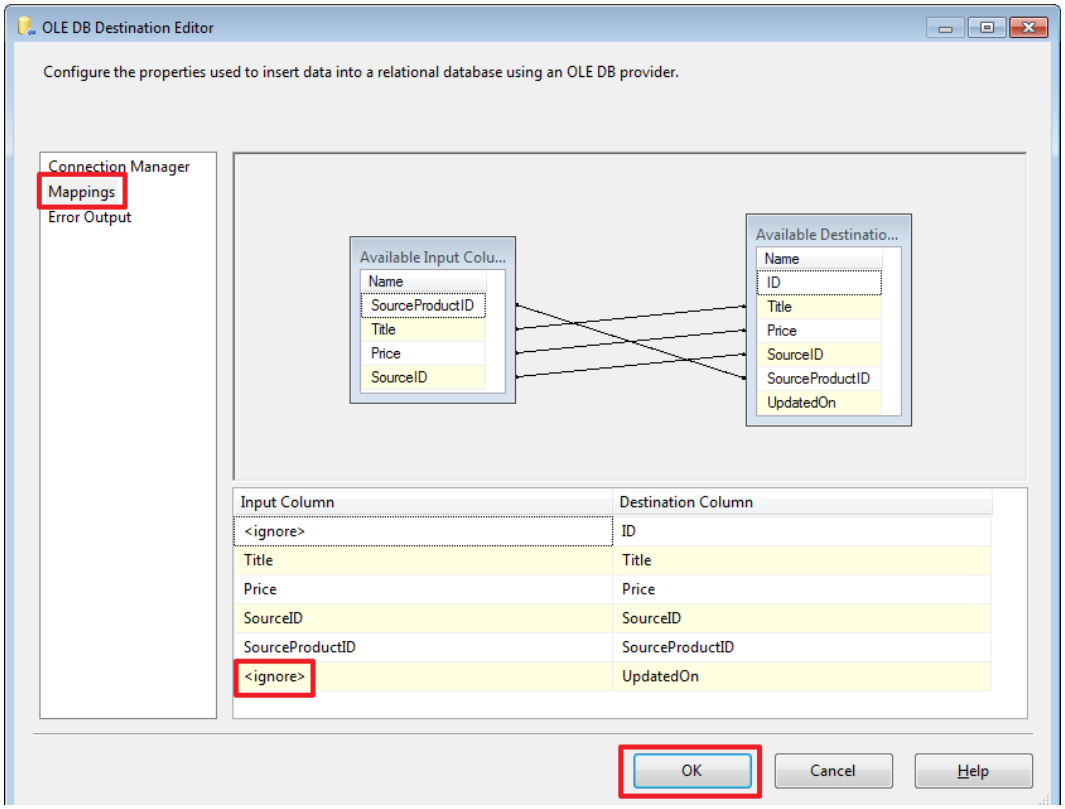




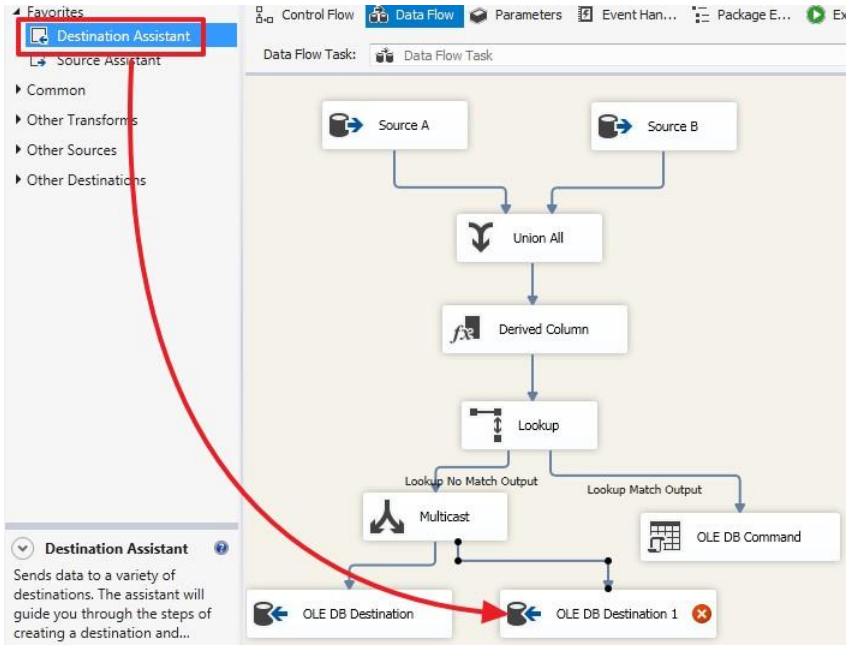
Перейдемо в область «Data Flow» компонента «Data Flow Task» і додамо компонент наступним чином:



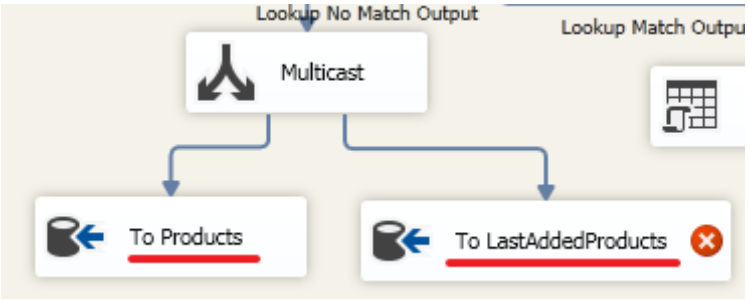
Зверніть увагу на жовтий знак оклику - це сталося через те, що ми додали колонку UpdatedOn і не прив'язали її. Зайдемо в елемент «OLE DB Destination», перейдемо на вкладку Mappings залишимо для колонки UpdatedOn як вхідного поля Ignore і натиснемо OK



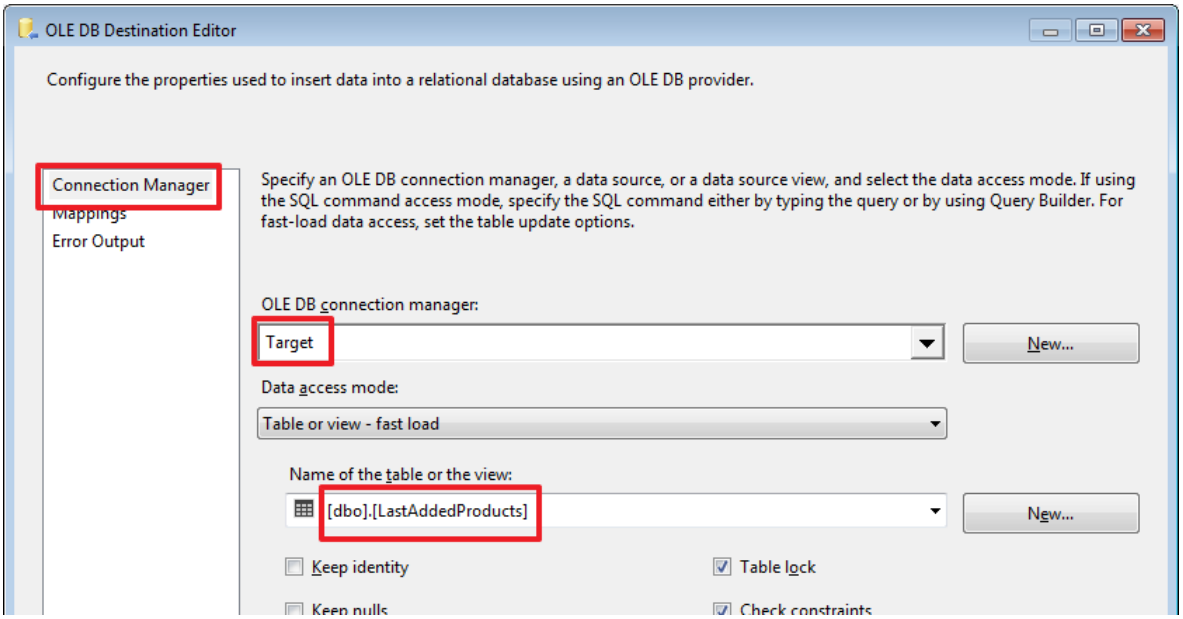
Створимо ще один елемент «OLE DB Destination» і перетягнемо на нього другу синю стрілку від елемента Multicast:

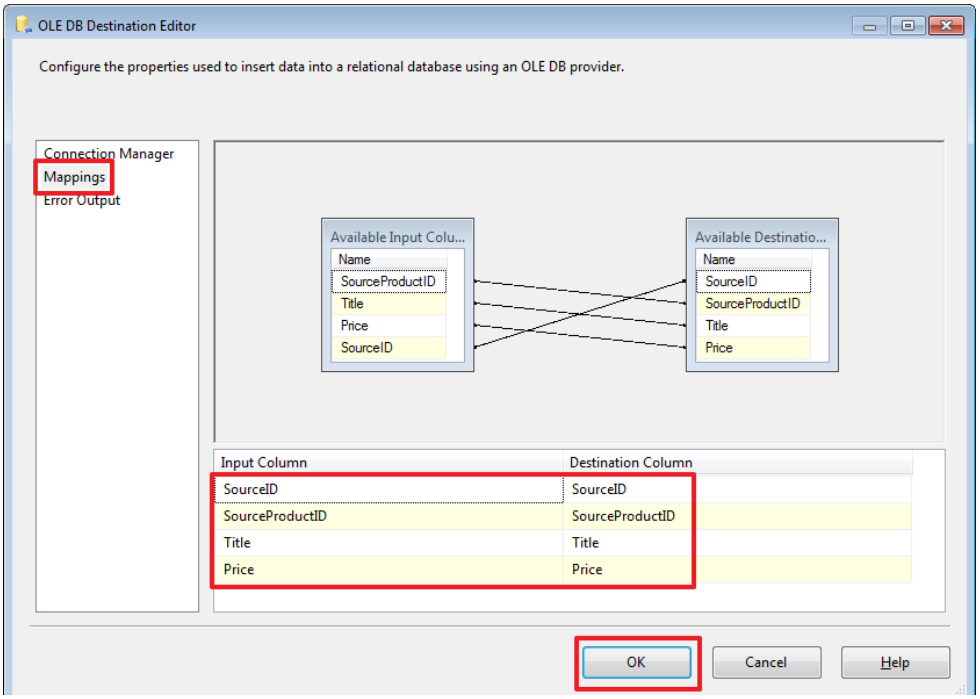


Перейменуємо для наочності:



Налаштуємо «To LastAddedProducts»:

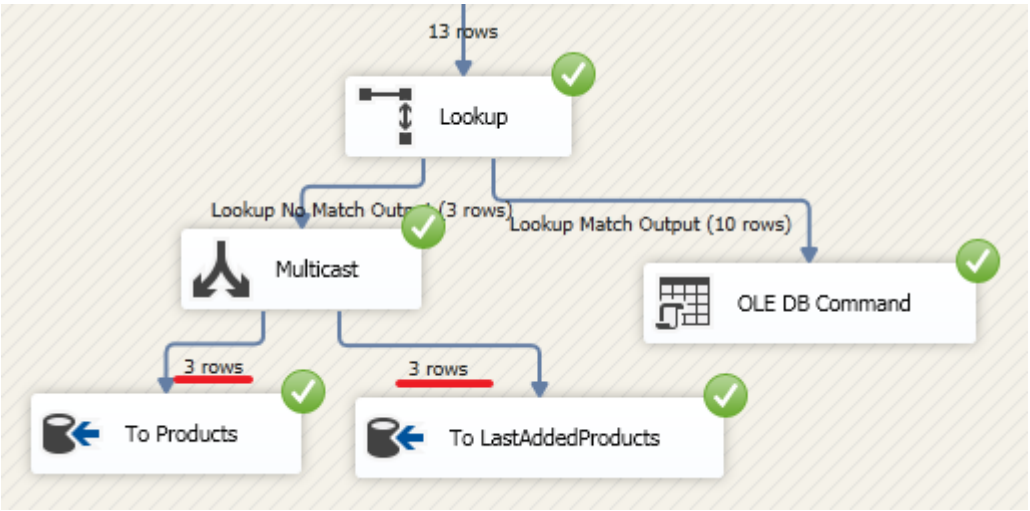




Видалимо через SSMS три останні вставлені записи:

```
USE DemoSSIS_Target
GO

DELETE Products
WHERE SourceID='B'
AND SourceProductID>=6
```

І запустимо пакет на виконання:



В результаті додавання відбулося в 2 таблиці - Products і LastAddedProducts.

SSIS досить цікавий інструмент, який на мій погляд не завадить мати всвоєму арсеналі, так як в деяких випадках він може сильно спростити процес інтеграції. Але звичайно бувають ситуації, коли все зваживши, розумніше написати інтеграцію вдаючись до інших способів, наприклад, використовувати Linked Servers і писати процедури на чистому TSQL або писати свою утиліту на якомусь іншому мовою програмування із застосуванням всієї потужності ООП і т.п.

Завдання на самостійну роботу

https://habrahabr.ru/post/330840/