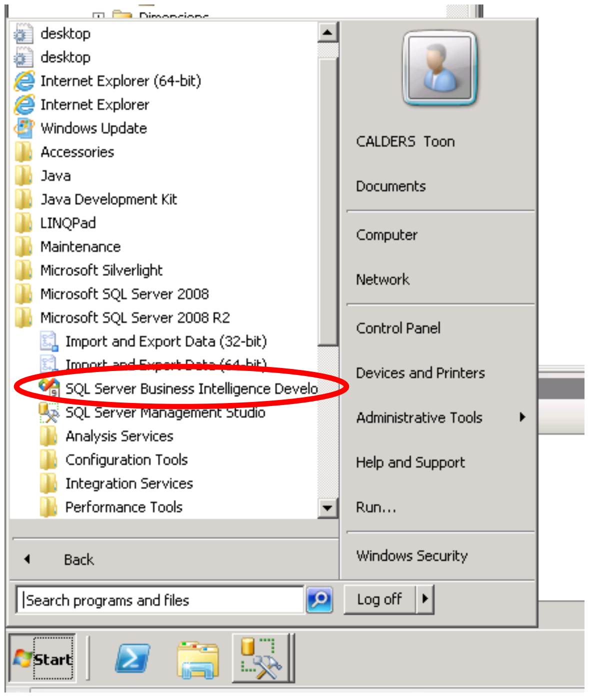
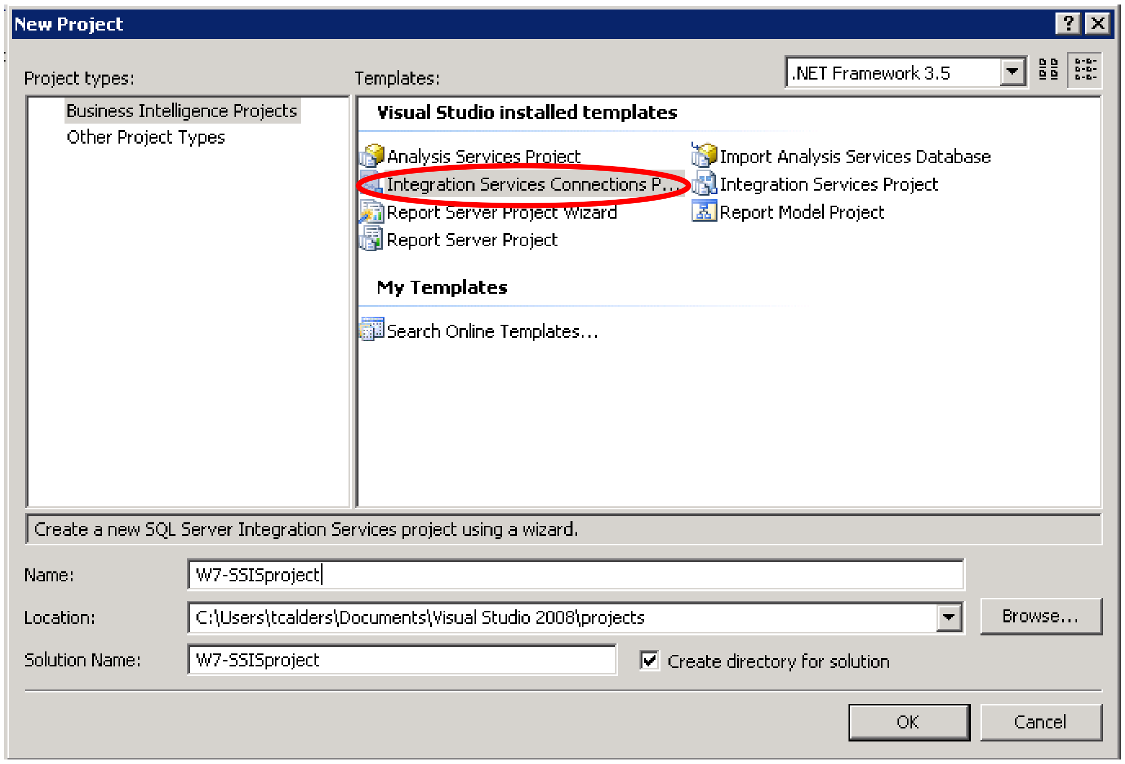
Creating your first SSIS project

1. Start Business Intelligence Development Studio



2. Start a new SSIS project


New Project
Project types:
Business Intelligence Projects
Other Project Types
Templates:
.NET Framework 3.5
Visual Studio installed templates
Analysis Services Project
Integration Services Connections P...
Report Server Project Wizard
Report Server Project
Import Analysis Services Database
Integration Services Project
Report Model Project
My Templates
Search Online Templates...
Browse...
OK
Cancel

Create a new SQL Server Integration Services project using a wizard.

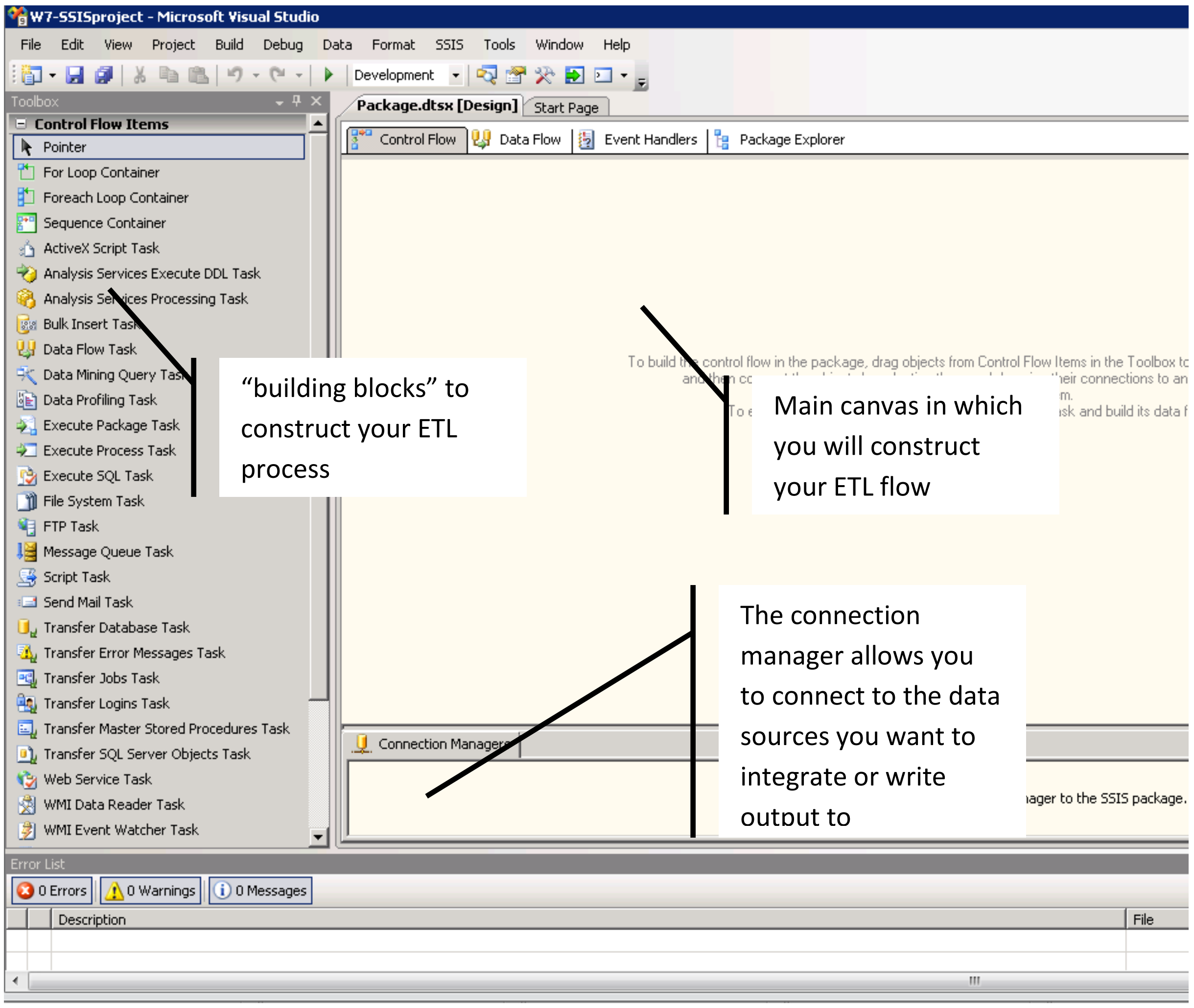
Name: W7-SSISproject

Location: C:\Users\tcalders\Documents\Visual Studio 2008\projects

Solution Name: W7-SSISproject

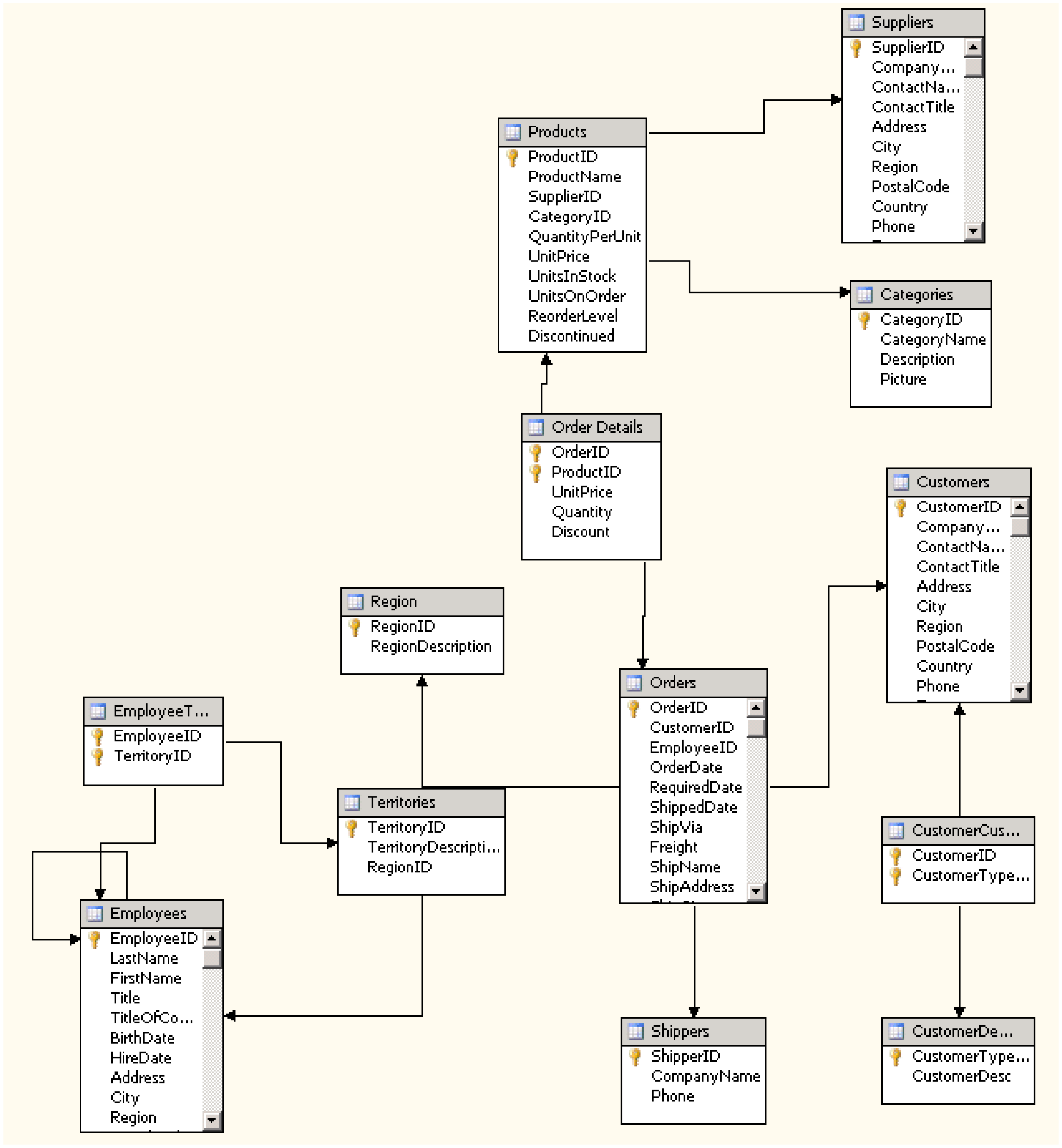
☑ Create directory for solution

3. You can use the wizard to define your data sources etc., but we won't do that, so chose cancel in the wizard. Normally you should now see the following layout (if not, double click on package.dtsx on the right-hand side):



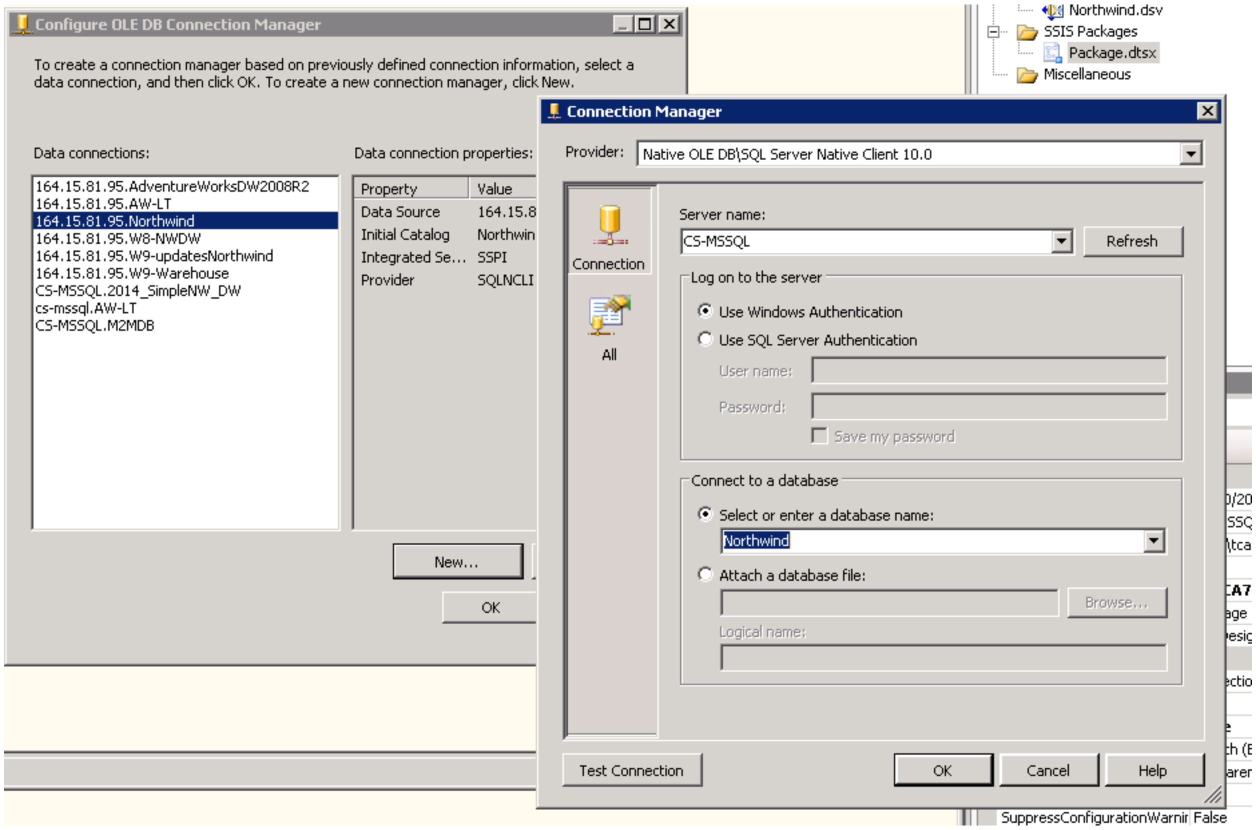
Your ETL flow will consist of multiple "data flows" and your control flow will define the order in which the data flow tasks and other simpler data manipulation tasks need to be executed. For instance, filling the dimension table "dimCustomer" will be done with one data flow, as well as filling the fact table factSales. However, it is important that the dimCustomer table is filled before the factSales. This order will be enforced in the control flow. Also tasks like creating table statements etc. will be done in the control flow. The databases and other data sources and sinks you will be using can be connected through the connection manager.

4. We will construct a very simple ETL flow that uses the Northwind database as input. The schema of this database is as follows:



Hence, we have a situation in which we can start from a clean database, and we will be using only one source database. You can think of this as the reconciled data layer in a three-tier architecture. So, we first need to make a connection to the database. Do this by right-click on “Connection Managers” (bottom) and selecting OLE DB connecting.

Navigate to the Northwind database on the CS-MSSQL database server:


Configure OLE DB Connection Manager

To create a connection manager based on previously defined connection information, select a data connection, and then click OK. To create a new connection manager, click New.

Data connections:

164.15.81.95.AdventureWorksDW2008R2
164.15.81.95.AW-LT
164.15.81.95.Northwind
164.15.81.95.W8-NWDW
164.15.81.95.W9-updatesNorthwind
164.15.81.95.W9-Warehouse
CS-MSSQL.2014_SimpleNW_DW
cs-mssql.AW-LT
CS-MSSQL.M2MDB

Data connection properties:

Property	Value
Data Source	164.15.8
Initial Catalog	Northwin
Integrated Se...	SSPI
Provider	SQLNCLI

Northwind.dsv
SSIS Packages
Package.dtsx
Miscellaneous

Connection Manager

Provider: Native OLE DB\SQL Server Native Client 10.0

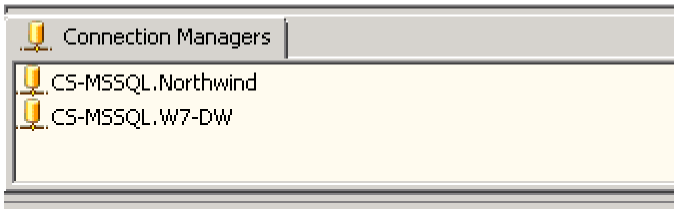
Server name: CS-MSSQL

Log on to the server
(●) Use Windows Authentication
( ) Use SQL Server Authentication
User name:
Password:
[ ] Save my password

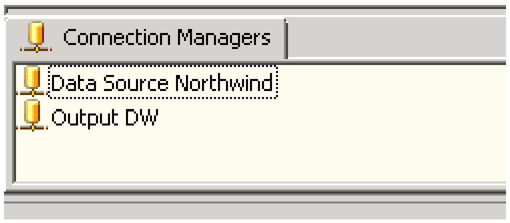
Connect to a database
(●) Select or enter a database name: Northwind
( ) Attach a database file:
Logical name:

SuppressConfigurationWarnir False

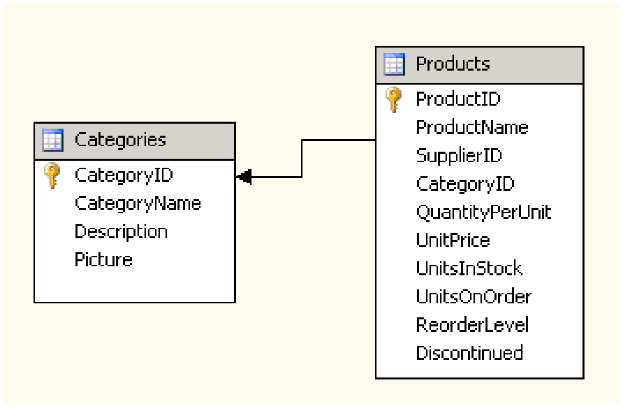
This is the connection to the input database. We also have to make a connection to our destination database. For this purpose, either create a new database using the SQL Server Management studio, or use an existing database that belongs to you (important: you need write permission and permission to create tables in this database). For this demo I have created my own new database W7-DW, and created a connection to it, in the exact same way as was done for the Northwind database:


Connection Managers
CS-MSSQL.Northwind
CS-MSSQL.W7-DW

To avoid confusions later on the datasources were renamed (right-click on name - rename):


Connection Managers
Data Source Northwind
Output DW

5. We are now ready to start creating our flow. We will start with a simple flow to create the dimension dimProduct for which we will need the following two tables:

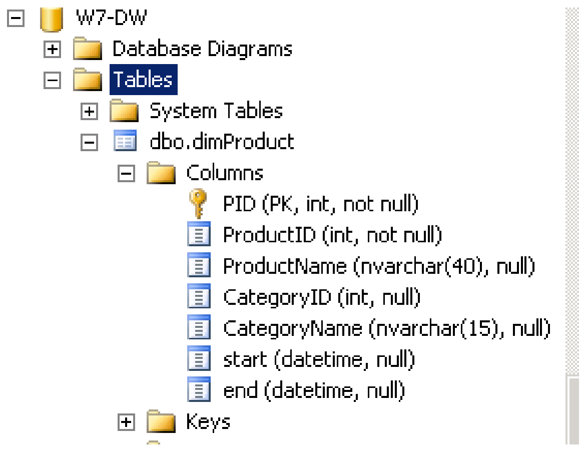


For reasons of simplicity we won't consider the supplier of a product, but only the category. The dimension table we create will contain the ProductID, ProductName, CategoryID, and CategoryName. Furthermore, we need to be able to store type-2 changes, hence we will add a surrogate key, and a start and end field.

The table to which we will be writing has to be created in the data warehouse. So, before we start defining all control flows, create the dimProduct table in the data destination with the following create table statement:

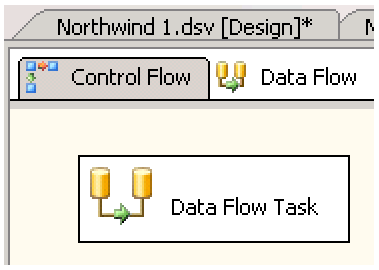
```
CREATE TABLE [dbo].[dimProduct](
      [PID] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,
      [ProductID] [int] NOT NULL,
      [ProductName] [nvarchar](40) NULL,
      [CategoryID] [int] NULL,
      [CategoryName] [nvarchar](15) NULL,
      [start] [datetime] NULL,
      [end] [datetime] NULL
)
```

So, you should see the following table structure in SQL Server Management Studio:

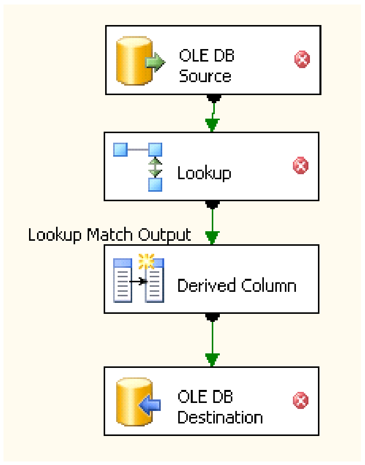


PID will be the surrogate key and by defining it as IDENTITY(1,1) it is an auto-increment number.

We are now ready to introduce a data flow task in the control flow canvas (drag and drop from the control-flow elements on the left):



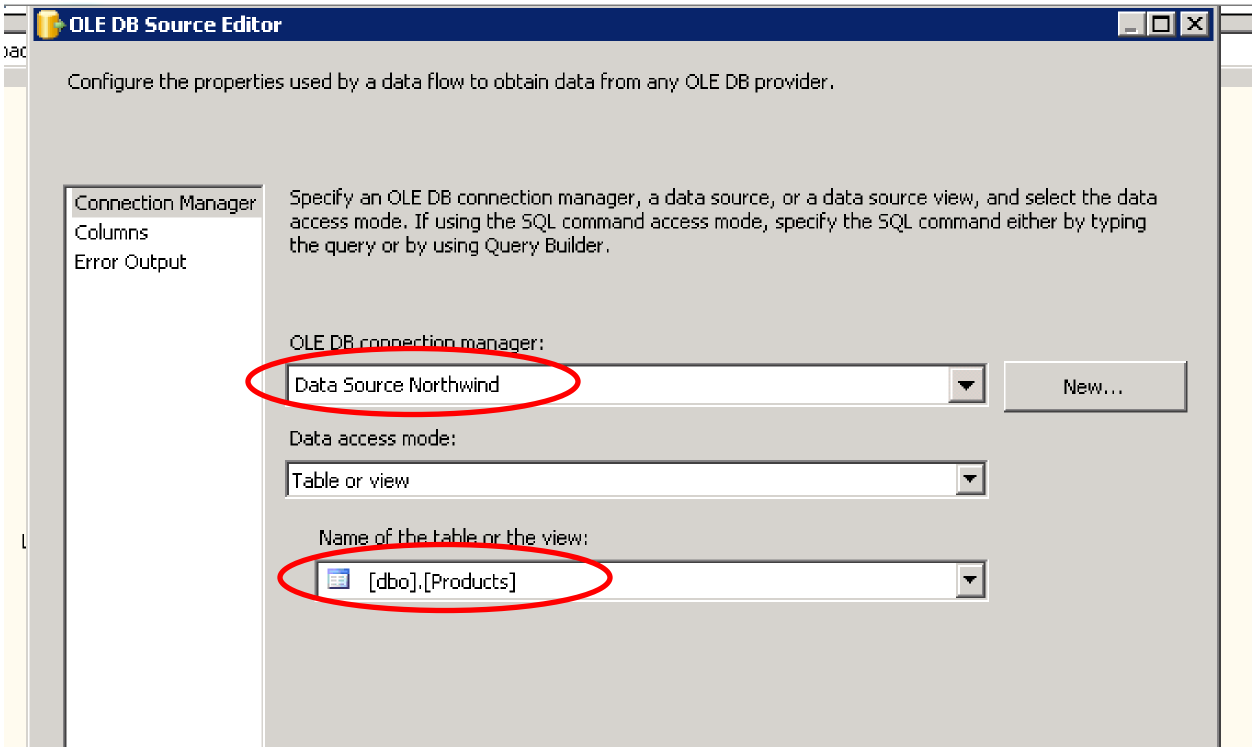
Rename the data flow task to “load dimProduct”, and open it in the data flow tab by double clicking on it. Notice that on the left the set of available elements has changed as we are now in the data flow tab. In this tab construct now the following ETL flow:



You can connect the different element by dragging the edge to the destination. This workflow expresses a pipeline where we first load a table from a source, then add attributes via a lookup, and finally write the table out to a destination table.

We will now configure the 4 elements from the top to the bottom. The first element will read the product table from our data source. The second operation is a lookup, looking up the category name in the category table with the category ID for every tuple in the product table. The derived column will introduce two now columns: one with the current date and one with NULL. These derived columns will respectively become the start and end of the introduced tuples. All configurations are shown on the next page.

The **first component**, OLE DB source will read the source table from the source database. Hence, we need to initialize it such that it will read the right table from the source database:


OLE DB Source Editor

Configure the properties used by a data flow to obtain data from any OLE DB provider.

Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder.

OLE DB connection manager: Data Source Northwind

Data access mode: Table or view

Name of the table or the view: [dbo].[Products]

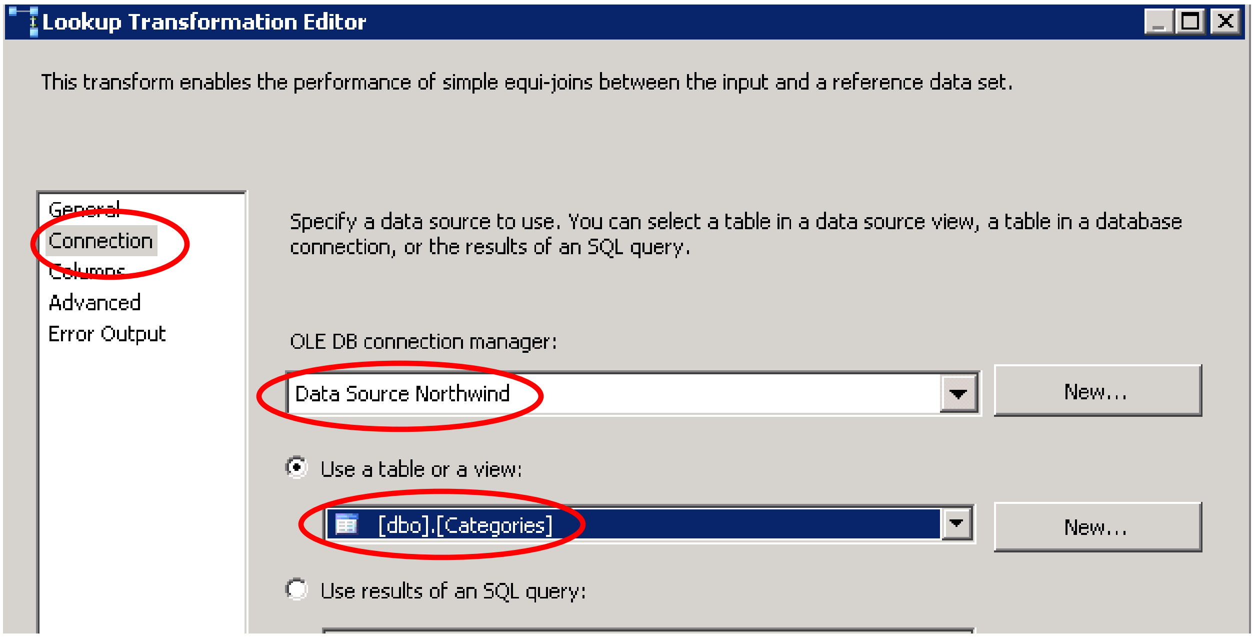
The **second component** is responsible for looking up the category code for the product. There are different ways to achieve this:

- Join the two tables in the data source and load the joined table instead of the product table. Disadvantage: burden on your source system and every time one of the tables change you will have to recompute the join
- Create a virtual view in your source table that expresses the join of these two tables. This is actually a popular method; creating virtual views in the source database to facilitate the export to the datawarehouse.
  Advantage: if the query result is much smaller than the tables on which the query is run, there may be far less data transfer between the two databases.
  Disadvantage: as the query is run on the source database this solution comes with a performance penalty at the data source side.
- Use a node that executes a query (join) on the source database and captures the result. Disadvantages are the same as previous bullet.

In this exercise we have chosen for a fourth approach:

- For every product in the product table, do a lookup for the category in the category table. There are different way to do this: multiple queries to the data source for instance (not desirable) or loading the categories table in memory (the default approach and preferable if table is small)

Hence in the lookup transformation node we need to set in which table the categories can be found:



Lookup Transformation Editor

This transform enables the performance of simple equi-joins between the input and a reference data set.

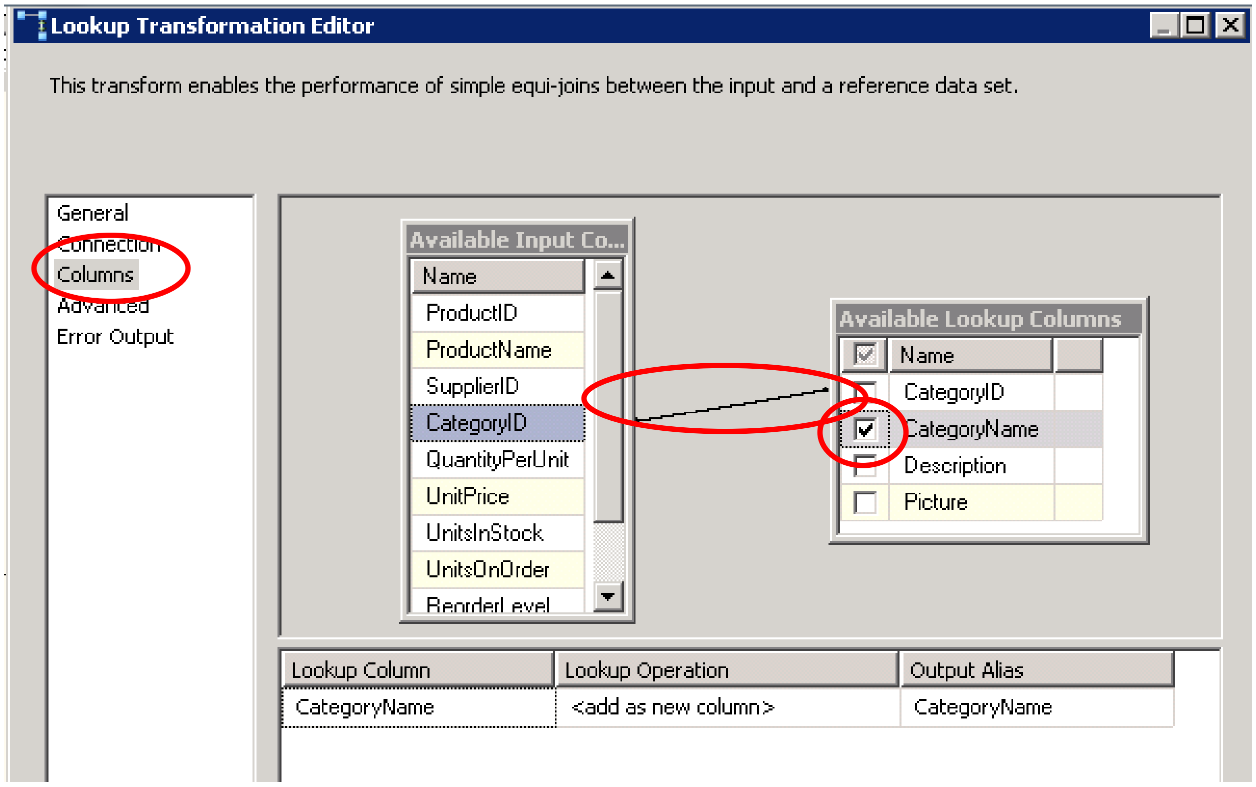
Specify a data source to use. You can select a table in a data source view, a table in a database connection, or the results of an SQL query.

OLE DB connection manager: Data Source Northwind

● Use a table or a view: [dbo].[Categories]

○ Use results of an SQL query:

And on what key we will join and which attributes to retain from the lookup table (Drag and drop to connect CategoryID of Product with CategoryID of the lookup table):

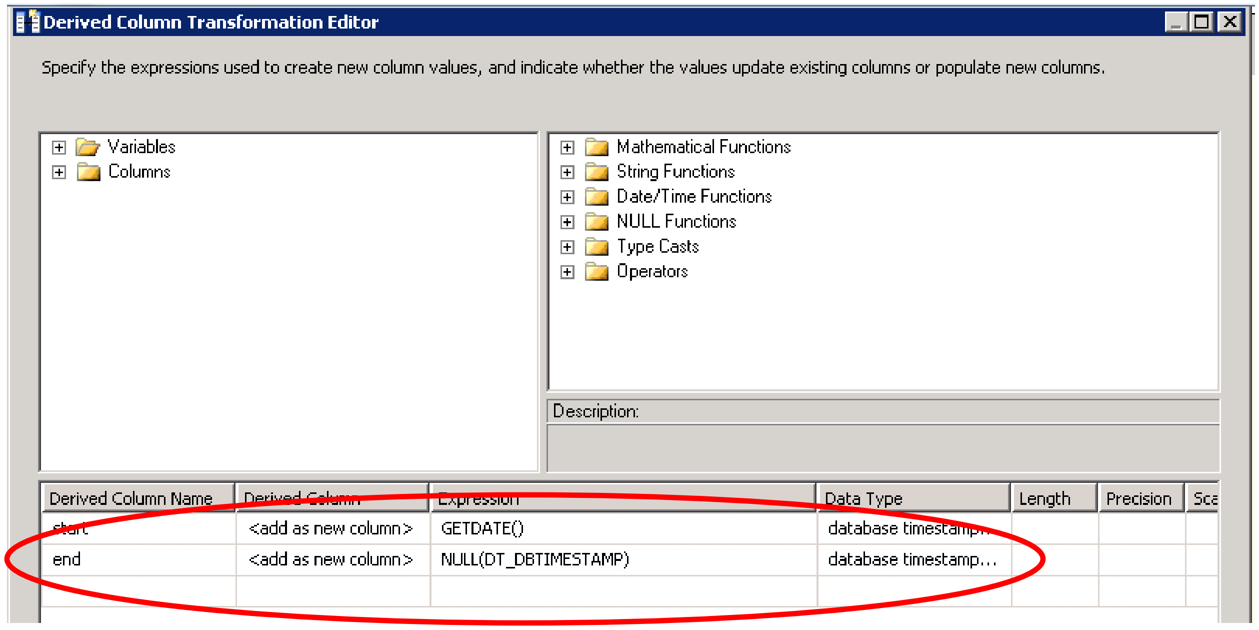


Lookup Transformation Editor

This transform enables the performance of simple equi-joins between the input and a reference data set.

Lookup Column	Lookup Operation	Output Alias
CategoryName	<add as new column>	CategoryName

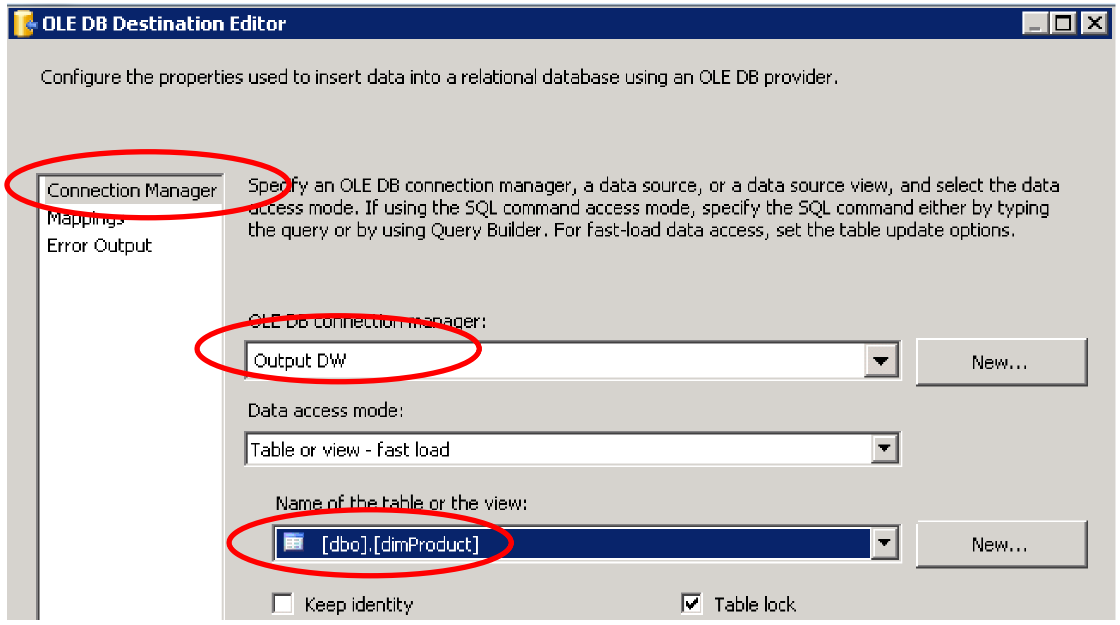
For **the third component**: in this component we introduce two new attributes start and end that take respectively values the current date and NULL. These will be the versioning attributes:


Derived Column Transformation Editor

Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

Derived Column Name	Derived Column	Expression	Data Type	Length	Precision	Sca
start	<add as new column>	GETDATE()	database timestamp...			
end	<add as new column>	NULL(DT_DBTIMESTAMP)	database timestamp...			

And finally for **the fourth component**: we need to set the destination database and table:


OLE DB Destination Editor

Configure the properties used to insert data into a relational database using an OLE DB provider.

Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder. For fast-load data access, set the table update options.

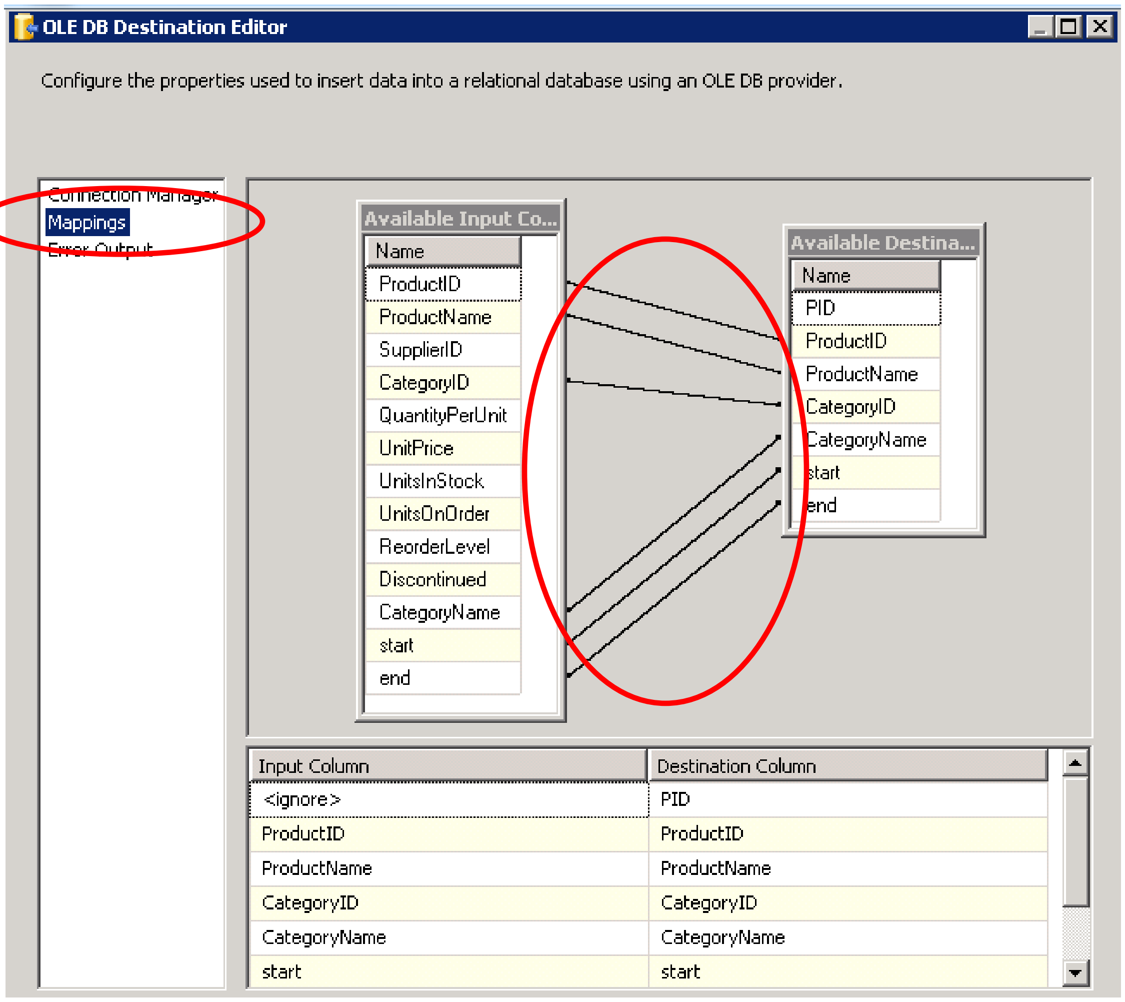
OLE DB connection manager: Output DW

Data access mode: Table or view - fast load

Name of the table or the view: [dbo].[dimProduct]

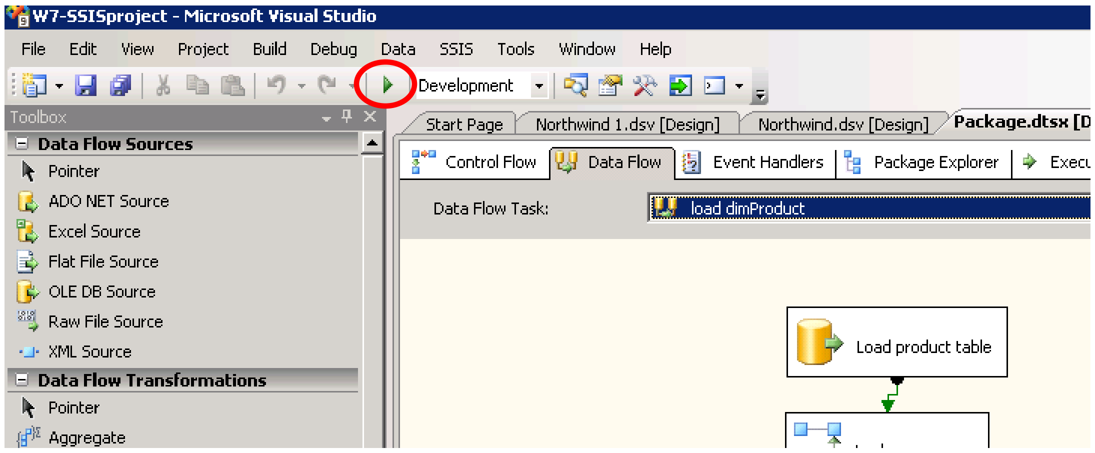
Keep identity [ ] Table lock [✓]

And set the mappings:



Input Column	Destination Column
<ignore>	PID
ProductID	ProductID
ProductName	ProductName
CategoryID	CategoryID
CategoryName	CategoryName
start	start

Now you are ready to run the package:



If the package was successfully executed you will indeed see that the table has been filled with the appropriate values.