Introduction to SQL Server 2005 Integration Services

Preface

Robert may be contacted through his website, http://arcanecode.com.

Objective

After completing this lab you should be able to create a basic workflow in SQL Server 2005 Integration Services (SSIS).

Scenario

The Adventure Works management team needs to see a report of Sales per Day per Product Line. Sometimes the Product Line field is Null, if so we have been instructed to treat it like an "S" Product.

Running the report from the transactional system really slows it down, and takes time for the report to generate. Therefore you have been requested to move the data to an alternate table.

In addition, sales figures from the past are often adjusted for various reasons, such as returns. Finally, sales figures do not have to be right to the minute, close of business the day before will meet the corporations needs.

And it's needed RIGHT NOW!

Exercise 1 Preliminary Work

Prior to beginning the SSIS work, tables need to be created to hold the data, both temporarily and for the final report. This exercise will setup the needed tables. In addition we will also create a new project in Visual Studio / Business Intelligence Developer Studio (BIDS) 2005 to hold our SSIS project.

Task	Detailed Steps

1. Create a Database to hold the data.	1. Open SQL Server Management Studio
	2. Right Click on Databases, select “Create a new Database”
	3. Provide a name, I’m going to call mine ArcaneCode .
	4. Set our new database as the working environment by picking it in the database environment drop down or enter the following command:
	
	`use ArcaneCode;`
	`go`

2. Create a table to hold the staging output.	`create table staging_SalesData`
	`(   OrderDate datetime`
	`  , LineTotal numeric(38,6)`
	`  , ProductLine nvarchar(2)`
	`)`

3. Create a table to stage Update rows.	`create table staging_SalesSummary`
	`(   OrderDate datetime`
	`  , LineTotal numeric(38,6)`
	`  , ProductLine nvarchar(2)`
	`)`

4. Create a table to hold the final output.	`create table Sales_Summary`
	`( OrderDate datetime`
	`  , SubTotal numeric(38,6)`
	`  , ProductLine nvarchar(2)`
	`)`

5. Install the Adventure Works 2005 Database	1. Download the AdventureWorks 2005 SP2 sample database from CodePlex and install, following their instructions. http://www.codeplex.com/MSFTDBProdSamples/Release/ProjectReleases.aspx?ReleaseId=4004
6. Create your new BIDS project.	1. Open Visual Studio 2005 *or* SQL Server 2005 Business Intelligence Developer Studio 2. Create a new project. 3. From the list of project types, select “Business Intelligence Projects”. 4. Under Installed Templates, pick Integration Services Project. 5. Give your project a meaningful name and click OK.

Exercise 2 Create the Data Sources

Your first step in the project will be to define the Data Sources; these tell SSIS where the databases are and how to talk to them.

Task	Detailed Steps
1. Create the Data Sources	1. Right click on Data Sources. 2. Pick New Data Source. 3. Click next on the opening Data Source Wizard Screen. 4. Click New. 5. Under Server Name, pick your server (mine is named N4IXT\SQLEXPRESS). 6. Under Select database name, pick AdventureWorks. 7. Click Next. 8. Enter “AdventureWorks” for the Data Source Name. Click Finish. 9. Repeat for ArcaneCode.
2. Create the Connection Managers	1. In the Package.dtsx area, right click in the Connection Managers area at the bottom. 2. Select “New Connection from Data Source”. 3. Select “Adventure Works” and click OK. 4. Repeat for “Arcane Code”.

Exercise 3 Create the first Data Flow

Now that the database connectivity has been handled, it's time to create the first data flow.

Task	Detailed Steps
1. Create the data flow.	1. Drop a Data Flow Task on the package. 2. Rename it "DFT - Move to staging." 3. Open the Data Flow Task.

2. Create the data source.	1. Drop and OLE DB Source on in the newly opened Data Flow Task. Rename to SRC – Adventure Works Sales
	2. Right click and pick EDIT.
	3. Change Data Access mode to SQL Command. Change the SQL Command text to:
	`select sh.OrderDate`
	`     , sd.LineTotal`
	`     , p.ProductLine`
	`from Sales.SalesOrderHeader sh`
	`inner join Sales.SalesOrderDetail sd`
	`        on sh.SalesOrderID = sd.SalesOrderID`
	`inner join Production.Product p`
	`        on sd.ProductID = p.ProductID`
	4. Click on Columns, just to show the columns. Click OK to close.

3. Create the data destination.	1. Add an OLE DB Destination. 2. Change name to DST – Staging Sales Data. 3. Drag the green arrow from the SRC to DST. 4. Highlight everything. Make sure the "Layout" toolbar is visible. (Right click in toolbar area, pick "Layout".) Use the left align and set all to same width buttons to clean up the layout. 5. Right click and EDIT the DST. 6. Set the connection manager to Arcane Code. 7. Set target to Table or View DO NOT PICK FAST LOAD. 8. Pick dbo.staging_SalesData from the drop down of the output. 9. Click on Mappings to show mappings. 10. Click OK.

Continued on next page…

Task	Detailed Steps
4. Test the package.	1. Go to the Solution Explorer. Right click on the Package.dtsx and pick “Set as Startup Object”. 2. Run the project. 3. Show the DataFlow area to show row output. 4. Show results in SQL Server. 5. Use STOP DEBUGGING button to stop.
5. Change the load type to fast load.	1. Now go back and change the SRC to use Table or View – fast load. 2. Re-run and watch the speed difference!

Exercise 4 Adding a SQL Script Task

There are times when the most efficient way to accomplish some task will be with direct SQL. In this exercise we’ll add a SQL task to reset our staging table.

Task	Detailed Steps
1. Add the SQL Task.	1. Above the dataflow, add a SQL Task. Name it SQL – Reset Sales Data Staging Table. 2. Edit. Change the connection to “Arcane Code”. 3. Change the SQL Statement to “truncate table dbo.staging_SalesData”. 4. Click OK.
2. Examine current number of rows.	1. Inside SQL Server Management Studio, issue this command to show current number of rows: `select count(*) from staging_SalesData`
3. Test	1. Rerun the SSIS job, then run the above query again.

Exercise 5 Expand the functionality

Now that we have a basic data flow setup, we can expand the functionality by adding some transformational logic. This will allow us to fill in missing fields and aggregate our sales totals.

Task	Detailed Steps
1. Add a new Data Flow Task.	1. Add a new DFT. Name it DFT – Move to Warehouse. 2. Drag green arrow from DFT – Move to Staging 3. Edit the DFT.
2. Add a data source.	1. Add an OLEDB Source. Name it SRC – Staging Sales Data. 2. Edit. Change Connection to Arcane Code. Change table to staging_SalesData.
3. Add a Derived Column.	1. Add a Derived Column Transformation. 2. Name it DER – Fix Null Product Lines. Edit. 3. Add a Derived Column Name of ProductLine. Set Derived column to “Replace ‘Product Line’”. 4. Set the expression to read ISNULL(ProductLine) ? “S” : ProductLine. 5. Click OK.
4. Add an Aggregate Transformation.	1. Add an Aggregate Transformation. Name it AGG – Sales Total By Day. Edit. 2. Check the Order Date for the Input Column. Verify the operation is GroupBy. 3. Add a second input by checking LineTotal. 4. Change the operation to Sum. 5. Change the Output Alias to SubTotal. 6. Add a third column by checking ProductLine. Verify GroupBy is the operation. 7. Click OK to close.
5. Add a destination.	1. Add an OLE DB Destination. Name it “DST – Warehouse” 2. Connect to Aggregate, then Edit. 3. Change Connection Manager to ArcaneCode. 4. Pick dbo.Sales_Summary for the table to output to. 5. Click Mappings. Verify the fields are mapped correctly. 6. Click OK to close.
6. Test	1. Run the SSIS and view the results in SQL Server Management Studio.

Exercise 6 Adding Lookup Logic

In most real life situations, you will want to lookup values from other tables, either to supply missing values or to see if a row already exists. This exercise will have you see if a row already exists and use the results to determine insert / update logic.

Task	Detailed Steps
1. Add a new SQL Task to reset the sales summary staging table between runs. This is a typical clean-up activity you will often need to perform on staging tables.	1. Between the two DST's add a SQL Task, SQL _ Reset Sales Summary Staging Table. Reconnect the green arrows. 2. Edit. Set connection manager to Arcane Code. 3. Set SQL Statement to "truncate table dbo.staging_SalesSummary;" Click OK.
2. Add the lookup task.	1. Edit DFT – Move to Warehouse. 2. Insert a Lookup just before DST – Warehouse. Name it "LKP – SalesSummary". 3. Connect the Green Arrow from the AGG to it.
3. Edit the lookup task.	1. Change the connection manager to ArcaneCode 2. Under use table or view, set the table to dbo.Sales_Summary 3. Click on the Columns tab. 4. On the first line of LookupColumn, pick OrderDate and Replace Order Date for the Operation. 5. On the second line pick ProductLine, and Replace ProductLine for the op. 6. Click Configure Error Output. 7. Under Lookup Output, change from Fail Component to RedirectRow. 8. Close the Lookup dialog.

Continued on next page…

Task	Detailed Steps
4. Fix/Create Destinations	1. Connect the RED arrow from the LKP to the existing DST – Warehouse. This one will now be used for Inserts. (If the lookup failed, it was because the value wasn't found, which means we need to follow the red failure line and insert the new row.) 2. Add a new destination to handle updates. Name it DST – Staging for Update. Connect GREEN arrow from LKP to it. (Note it is common to stage rows that need to be updated in a different table, then use a bulk update and remove them afterward. This is much faster than individual updates.) 3. Set the Connection Mgr to ArcaneCode, the table to dbo.staging_SalesSummary 4. On the Mappings tab, connect the SubTotal to the LineTotal Line. Click OK. 5. Return to the Control Flow.

5. Add SQL to Update from our staging update table.	1. Add a SQL Task, name it SQL – Update Summary Data
	2. Edit. Change the connection to Arcane Code.
	3. Add this text:
	
	`UPDATE dbo.Sales_Summary`
	`SET SubTotal = staging_SalesSummary.LineTotal`
	`FROM dbo.Sales_Summary AS dbo_Sales_Summary`
	`INNER JOIN staging_SalesSummary`
	`  ON dbo_Sales_Summary.OrderDate =`
	`staging_SalesSummary.OrderDate`
	`  and dbo_Sales_Summary.ProductLine =`
	`staging_SalesSummary.ProductLine`

6. Test	1. Save and run.

Exercise 7 Using Variables in your SSIS

SSIS supports the use of variables. Variables can be used to hold a variety of values, including items that are used in multiple places. In this exercise we'll use a variable to hold a SQL command that will be called from two places.

Task	Detailed Steps
1. Show the variable pane.	1. Click on SSIS, Variables or Right Click and pick Variables.
2. Add a new variable.	2. Click the "Add new Variable" button. 3. Enter "TruncateSummaryStagingSQL" for the variable name. 4. Change the variable DataType to String. 5. Enter truncate table dbo.staging_SalesSummary; for the text.
3. Replace an existing item with a variable.	1. Edit the Existing SQL – Reset Sales Summary task. 2. Change the SQL Source type to Variable. 3. Pick User::TruncateSummaryStagingSQL in the drop down. 4. Click OK to close.
4. Add a new task and use the same variable.	1. Add a new SQL Task. Name it SQL – Clean up Summary Data Staging. Edit. 2. Change the Connection to Arcane Code. 3. Change the SQL Source Type to Variable. 4. Change the Variable to "User::TruncateSummaryStagingSQL". 5. Click OK to close.
5. Test.	1. Run the package.

Exercise 8 Debugging your SSIS job

BIDS supports some basic debugging features for an SSIS package. In this exercise we will use a DataViewer to pause execution of the package and examine the data flowing through the package.

Task	Detailed Steps
1. Add a dataviewer.	1. Open the DFT – Move to Warehouse 2. On the green bar after the AGG function right click. Select DataViewers. 3. Click Add. 4. Show the Grid tab, to demo the fields. 5. Take defaults and click OK. 6. Click OK.
2. Run	1. Run and see the grid view. 2. Press the continue button to continue package execution.