**Importing SQL Server Data Using SSIS - Which Option is Fastest**

|  |  |
| --- | --- |
| MSSQLTips author Daniel Calbimonte | By: [Daniel Calbimonte](http://www.mssqltips.com/sqlserverauthor/83/daniel-calbimonte/)   |   [Read Comments (27)](http://www.mssqltips.com/sqlservertip/2684/importing-sql-server-data-using-ssis--which-option-is-fastest/?utm_source=dailynewsletter&utm_medium=email&utm_content=headline&utm_campaign=20140702#comments)   |   Related Tips: [More](http://www.mssqltips.com/sql_server_business_intelligence_tips.asp) > [Integration Services Development](http://www.mssqltips.com/sql-server-tip-category/17/integration-services-development/) |

**Problem**

This article is useful for SSIS developers who do not know which tasks are best to use in an SSIS projects. The main problem is that at the end of development if performance is slow then you will need to rebuild the project and change components. This article shows various ways of importing data and shows which types of components perform best within SSIS. The contest will be between the following components:

* ODBC Tasks
* ADO NET Tasks
* OLEDB Task
* SQL Server Destination
* T-SQL Tasks

**Solution**

I created different SSIS packages in order to test performance. In this demo I used SSIS 2012 and the database [Adventureworks 2012](http://msftdbprodsamples.codeplex.com/releases/view/55330).

In this demo I am going to import the table [AdventureWorks2012].[Sales].[SalesOrderDetail] to the test2 database which is on the same instance of SQL Server. SalesOrderDetails is the table with more rows in AdventureWorks2012.

In order to create the database test2 and the destination table dbo.OrderDetails, use this T-SQL code:

CREATE DATABASE Test2

GO

USE Test2

GO

CREATE TABLE [dbo].[OrderDetails](

[SalesOrderID] [int] NULL,

[SalesOrderDetailID] [int] NULL,

[CarrierTrackingNumber] [nvarchar](25) NULL,

[OrderQty] [smallint] NULL,

[ProductID] [int] NULL,

[SpecialOfferID] [int] NULL,

[UnitPrice] [money] NULL,

[UnitPriceDiscount] [money] NULL,

[LineTotal] [numeric](38, 6) NULL,

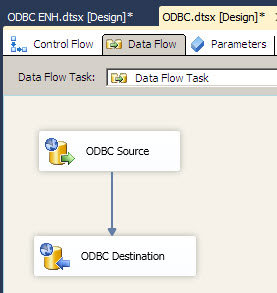
[rowguid] [uniqueidentifier] NULL,

[ModifiedDate] [datetime] NULL

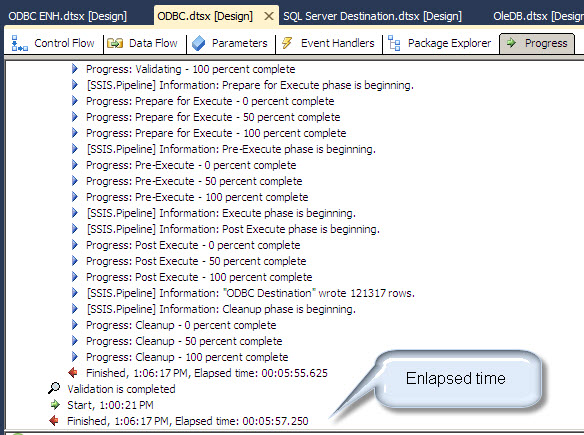
) ON [PRIMARY]

**Test 1 - ODBC Tasks**

The first example will use ODBC Source and ODBC Destination as shown below:



When we run the package we notice the average time is **5 minutes 57 seconds** to import the rows:

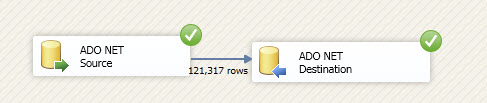


**Test 2 - ADO NET Tasks**

As noticed, ODBC is pretty slow. Let's try another approach. We are going to truncate the destination table first:

TRUNCATE TABLE test2.dbo.OrderDetails

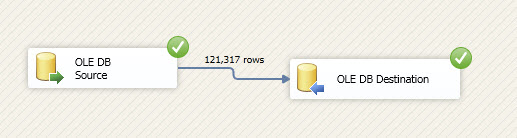
Let's try ADO tasks to import the same data and verify if these components are faster:



The average elapsed time in my testing was **11 seconds.** This is much better.

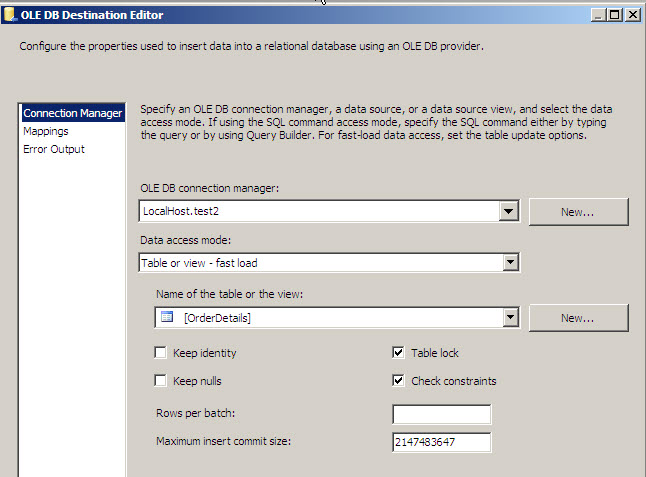
**Test 3 - OLEDB Tasks**

This time we are going to import the same data using the OLEDB Tasks. Again we will truncate the table in the test2 database first.

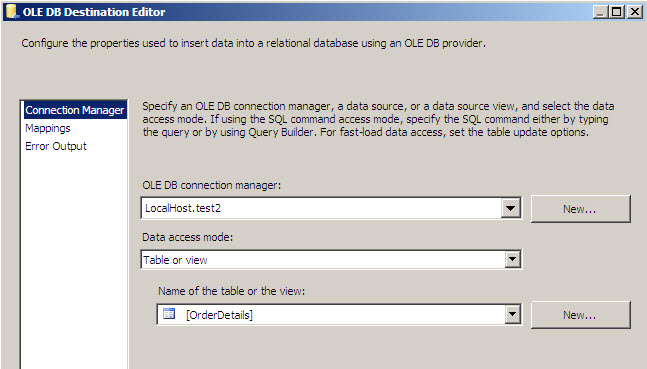


The average elapsed time is **5 seconds.**

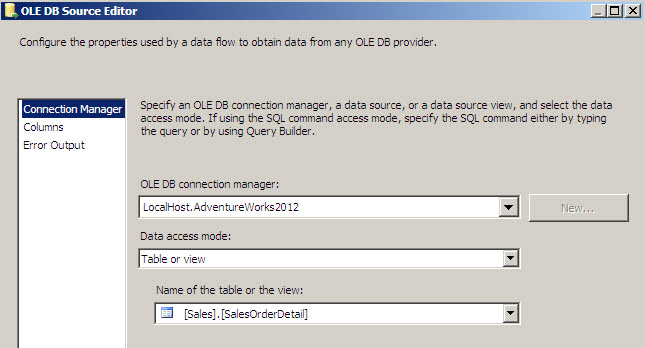
Note that I am using the "fast load" option with the "Table Lock" option in the OLE DB Destination Task:



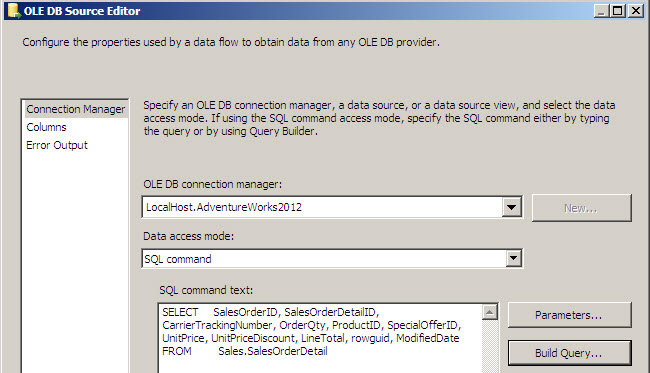
If we do not use the fast load option, the average elapsed time was **2 minutes and 21 seconds:**



OK. The fast load option really improves performance. I will return to that configuration. What about the OLE DB Source. By default I am using the option "Table or view" in the OLE DB Source as shown below:



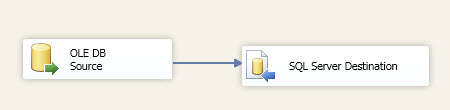
Let's use a "SQL Command" instead as shown below.



The average elapsed time is **2.85 seconds**.

**Test 4 - SQL Server Destination**

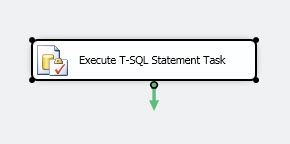
Now, let's try to use the SQL Destination as the destination instead of OLE DB Destination:



The average elapsed time is **2.5 seconds.** At this point it is the best option.

**Test 5 - Execute T-SQL Task**

Finally, some people think that the best option is to use the Execute T-SQL Task:



I am using a simple insert statement to import data from one source to another:

USE [TEST2]

GO

INSERT INTO [dbo].[OrderDetails]

([SalesOrderID]

,[CarrierTrackingNumber]

,[OrderQty]

,[ProductID]

,[SpecialOfferID]

,[UnitPrice]

,[UnitPriceDiscount]

,[rowguid]

,[ModifiedDate])

SELECT

[SalesOrderID]

,[CarrierTrackingNumber]

,[OrderQty]

,[ProductID]

,[SpecialOfferID]

,[UnitPrice]

,[UnitPriceDiscount]

,[rowguid]

,[ModifiedDate]

FROM [AdventureWorks2012].[Sales].[SalesOrderDetail]

The average elapsed time is **1.8 seconds!**

Finally I've been told that if the query runs inside a stored procedure it is even faster:

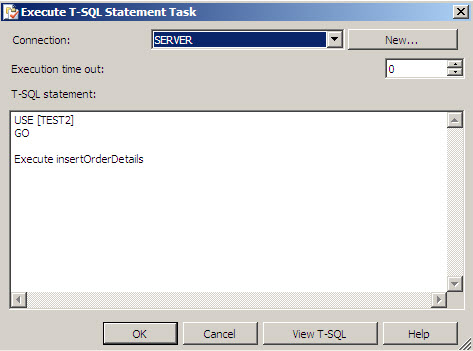
Let's create a stored procedure:

CREATE PROCEDURE insertOrderDetails   
as

INSERT INTO [dbo].[OrderDetails]  
 ([SalesOrderID]  
 ,[CarrierTrackingNumber]  
 ,[OrderQty]  
 ,[ProductID]  
 ,[SpecialOfferID]  
 ,[UnitPrice]  
 ,[UnitPriceDiscount]  
 ,[rowguid]  
 ,[ModifiedDate])

SELECT   
 [SalesOrderID]  
 ,[CarrierTrackingNumber]  
 ,[OrderQty]  
 ,[ProductID]  
 ,[SpecialOfferID]  
 ,[UnitPrice]  
 ,[UnitPriceDiscount]  
 ,[rowguid]  
 ,[ModifiedDate]  
FROM [AdventureWorks2012].[Sales].[SalesOrderDetail]

After creating the stored procedure we are going to call it in the Execute T-SQL Task:



The average elapsed time is **2.12 seconds**. The stored procedures does not improve performance.

**Summary**

Let's review the table with the results:

|  |  |
| --- | --- |
| **Position** | **Time in seconds** |
| ODBC Tasks | 357 |
| ADO Tasks | 11 |
| OLEDB | 2.85 |
| OLEDB, SQL destination | 2.5 |
| T-SQL | 1.8 |

The winner is: **T-SQL**

You may think the morale of the story is to use the Execute T-SQL Task instead of other SSIS tasks.  In this example we were importing data on the same instance, but this will not always be the case.

So the morale of the story is that there are many alternatives when creating a SSIS project and we have to carefully study the alternatives in different scenarios. There are great SSIS tools and we do not always use the best options. With each new version of SSIS new tasks are added and performance may be improved with existing tasks. The main changes in SSIS for SQL 2008 and 2012 are related to performance improvements.

**Next Steps**