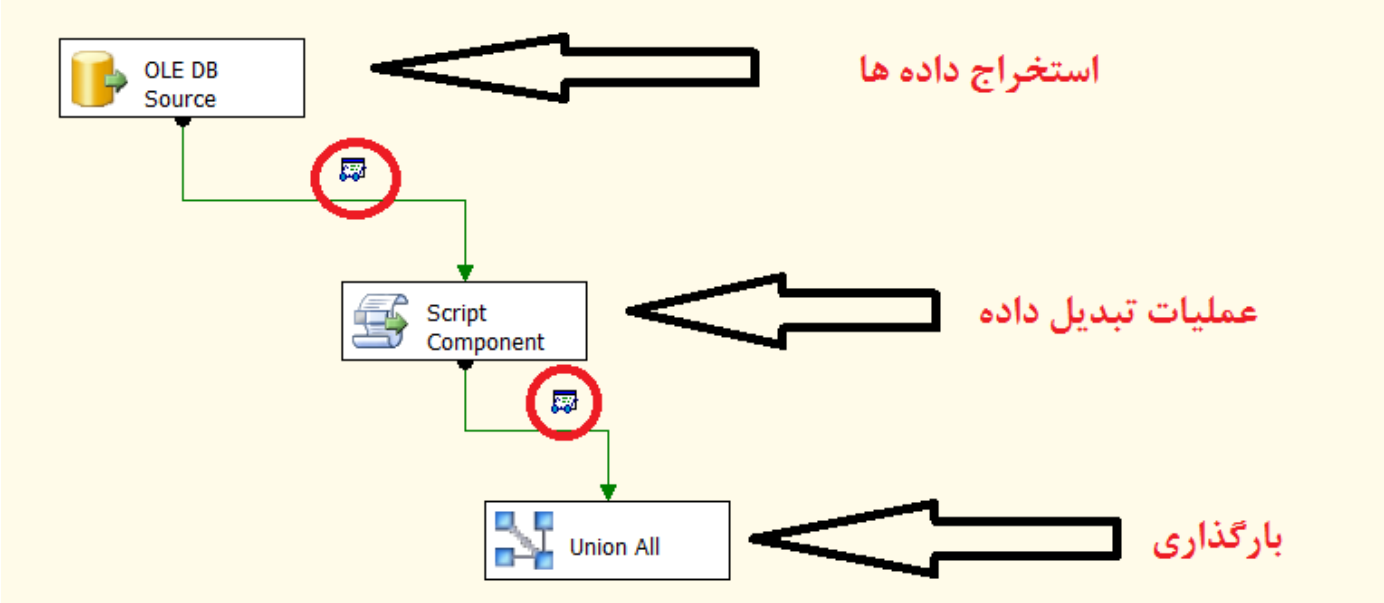
عنوان: **تبدیل تاریخ میلادی به شمسی در SSIS به کمک سی شارپ**
نویسنده: محمد باقر سیف اللهی
تاریخ: ۱۳۹۱/۰۵/۳۱ ۲۲:۲
آدرس: www.dotnettips.info
برچسب‌ها: C#, SSIS

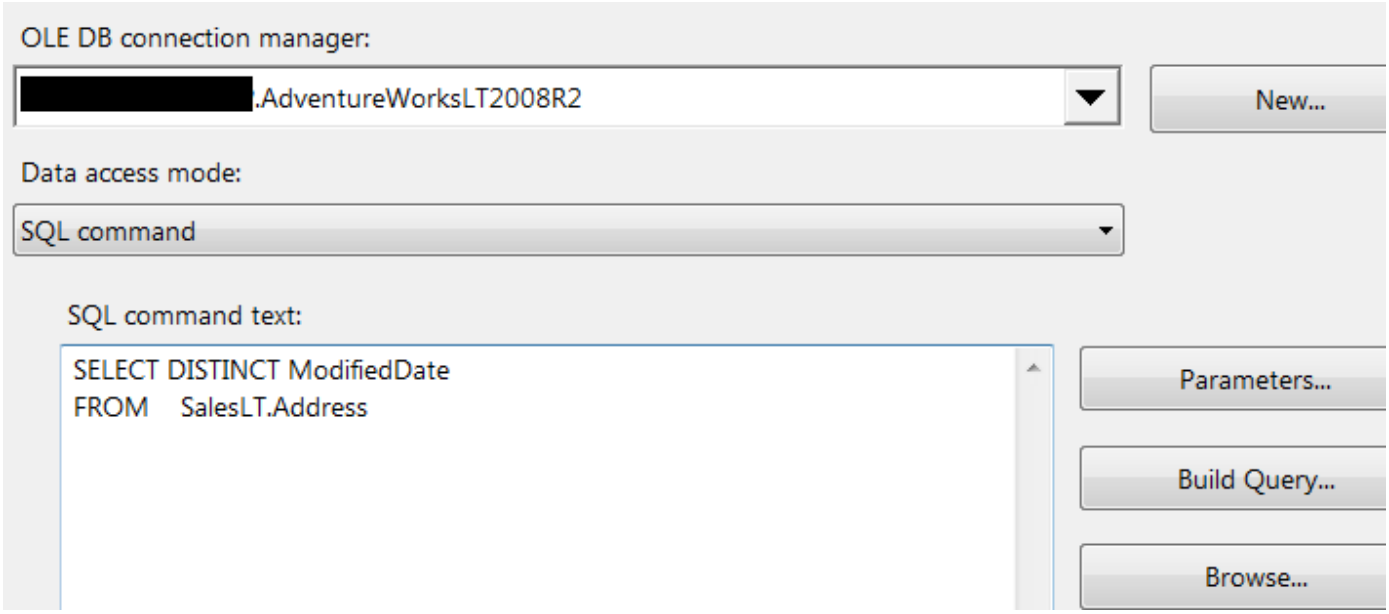
برای تبدیل تاریخ میلادی به تاریخ شمسی در packageهای SSIS می‌توان از زبان سی شارپ استفاده کرد . بدین طریق می‌توان در طی عملیات ETL و هنگام transform کردن داده‌ها ، عملیات تبدیل از میلادی به شمسی را انجام داد . عملیات تبدیل داده در این مثال به کمک Script Component انجام می‌شود.

برای این کار از داده‌های موجود در پایگاه داده **[AdventureWorksLT2008R2].[SalesLT].[Address]** استفاده می‌کنم .
برای این کار یک package تعریف می‌کنم و برای استخراج داده‌ها یک OLEDB تعریف می‌کنم . برای تبدیل داده‌ها از Script Component و برای گرفتن خروجی آزمایشی از union استفاده می‌کنم :

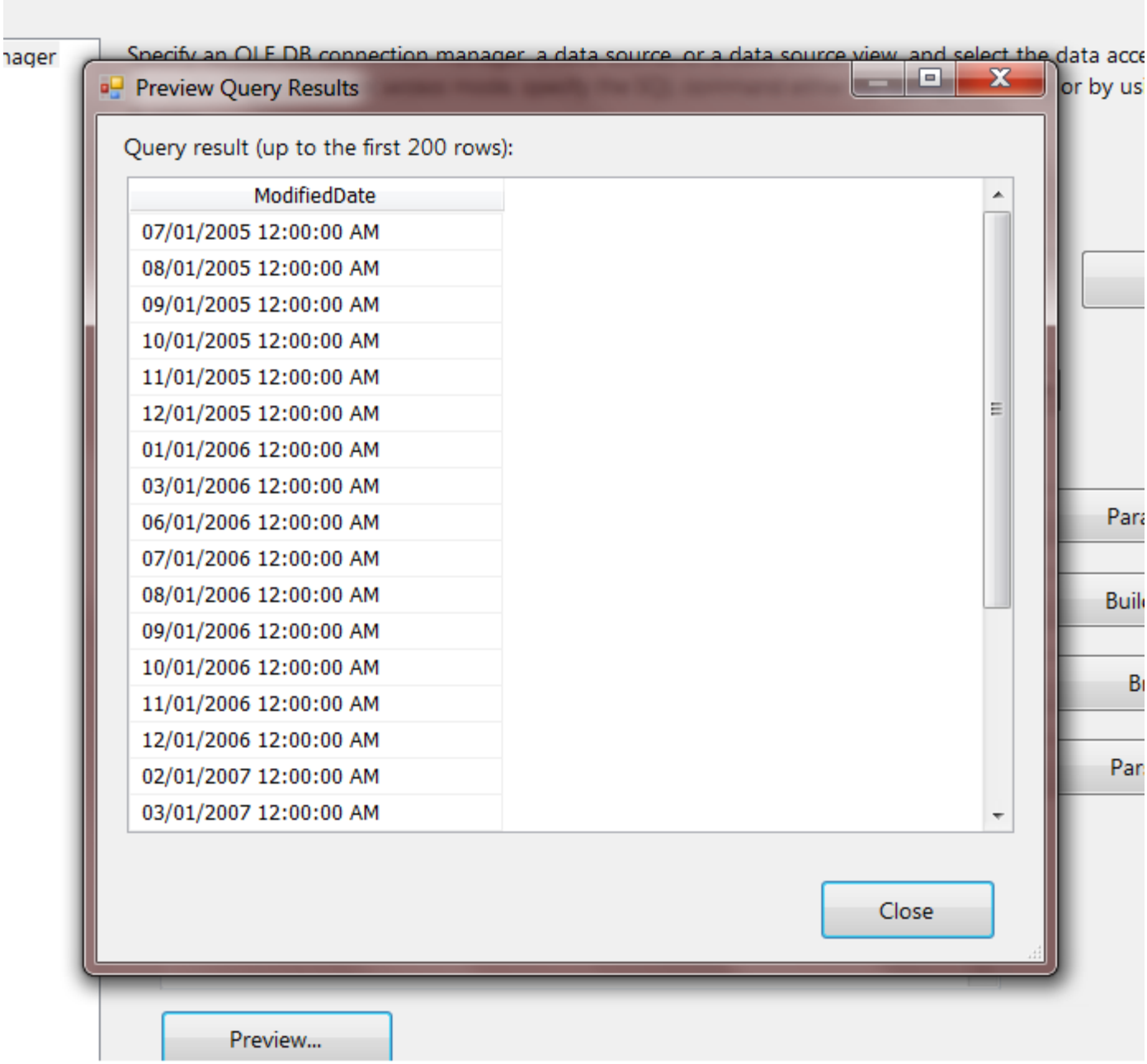


دایره‌های قرمز رنگ مشخص شده در تصویر ، بیانگر data viewerهای تعریف شده است.

در تنظیمات dataSource مانند زیر عمل می‌کنیم :
دستور زیر فقط برای نمایش یک نمونه از کارکرد عملیات مورد نظر در این مثال می‌باشد

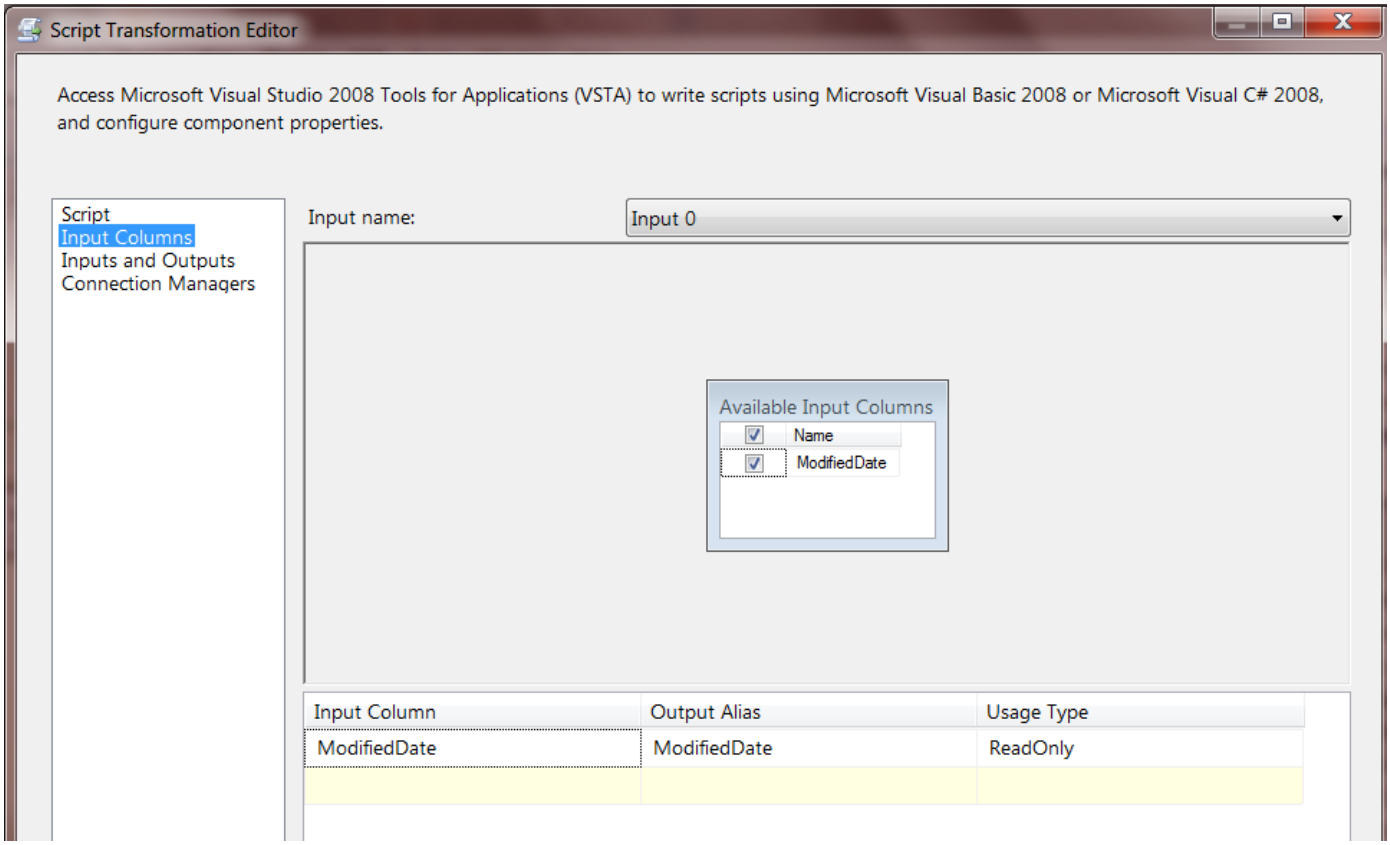


نمونه خروجی مانند زیر می‌باشد :

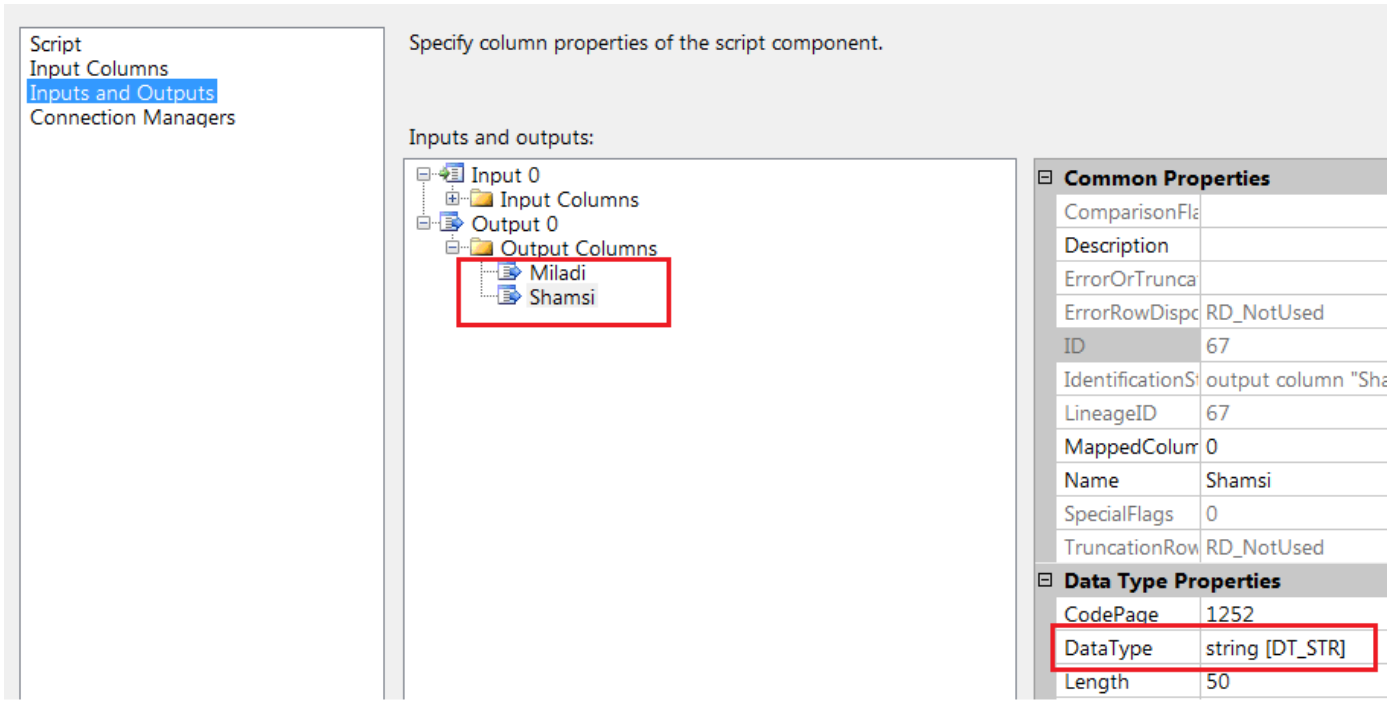


سپس برای تنظیمات Script Component به ترتیب زیر عمل می‌کنیم :

در قسمت Input column ستون هایی را که به عنوان پارامتر می‌توانیم با آنها کار کنیم ، تعریف می‌کنیم : (دقت شود که تغییر نام متغیرها ، در کدها اعمال می‌شوند .)

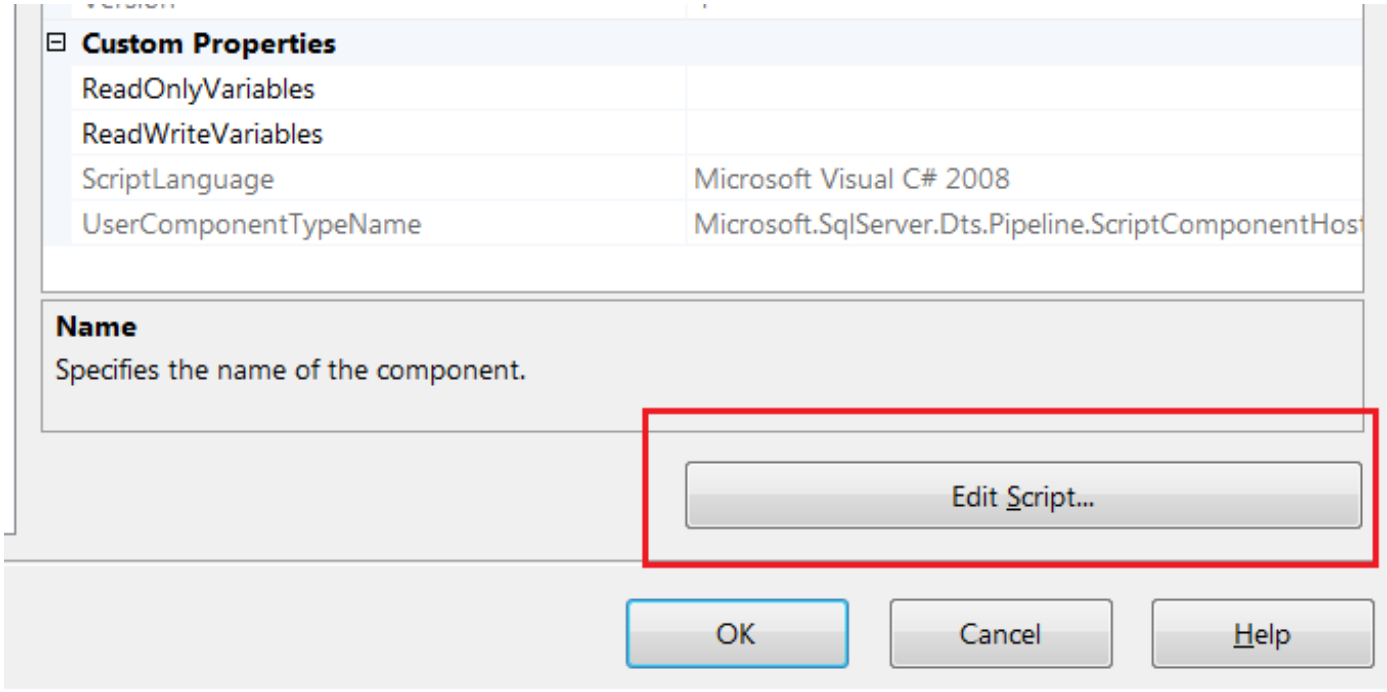


حال می‌خواهم ستون‌های تاریخ میلادی و شمسی را برای خروجی تعریف کنم :

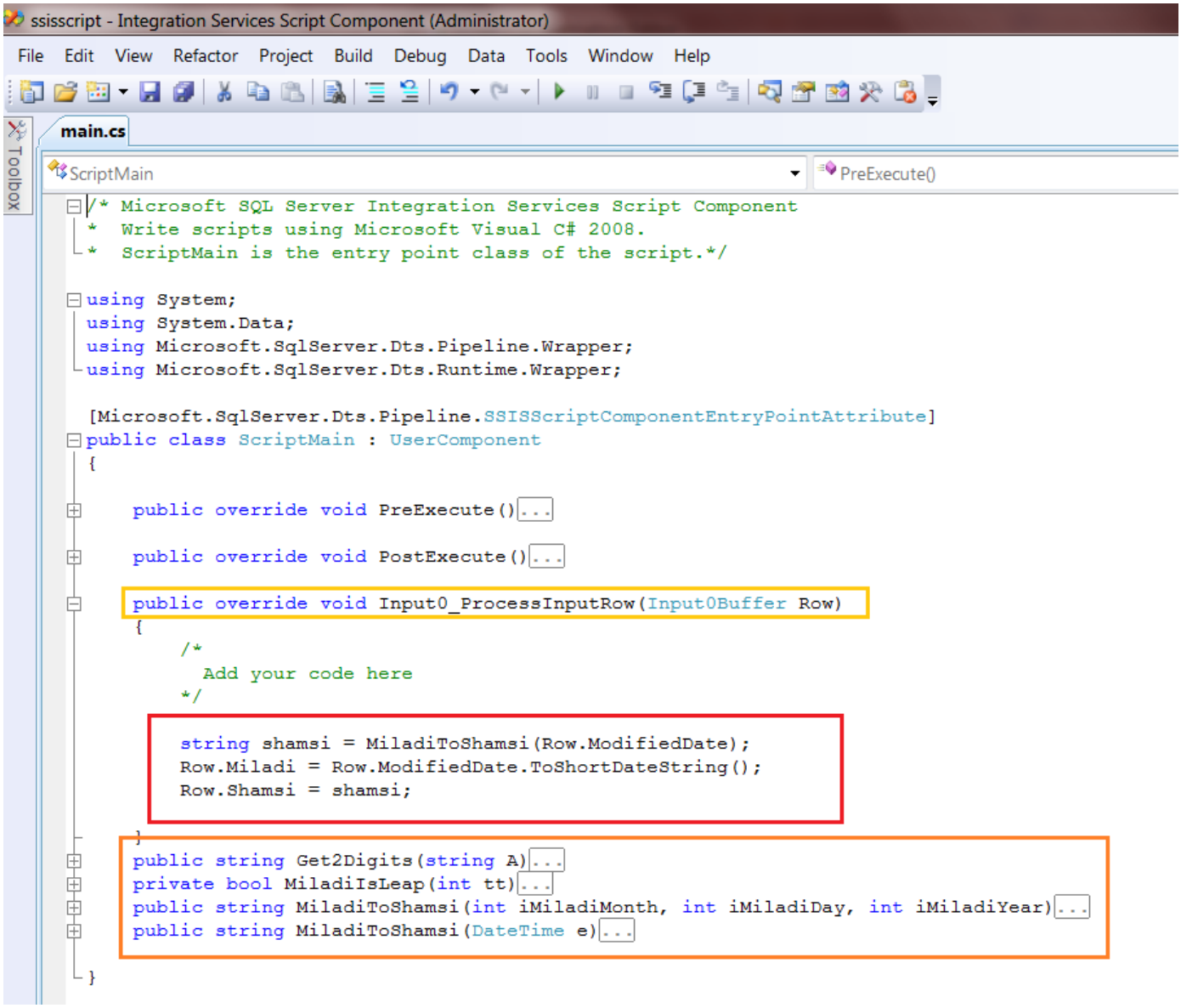


همانطور که مشاهده می‌کنید ، نوع داده ای برای خروجی را رشته تعریف کردم.

سپس به قسمت script بر می‌گردیم (سمت چپ پنجره) و روی Edit Script کلیک می‌کنیم : (در صورتی که تمایل به کد نویسی با VB را دارید در همین قسمت می‌توانید آن را تنظیم کنید)

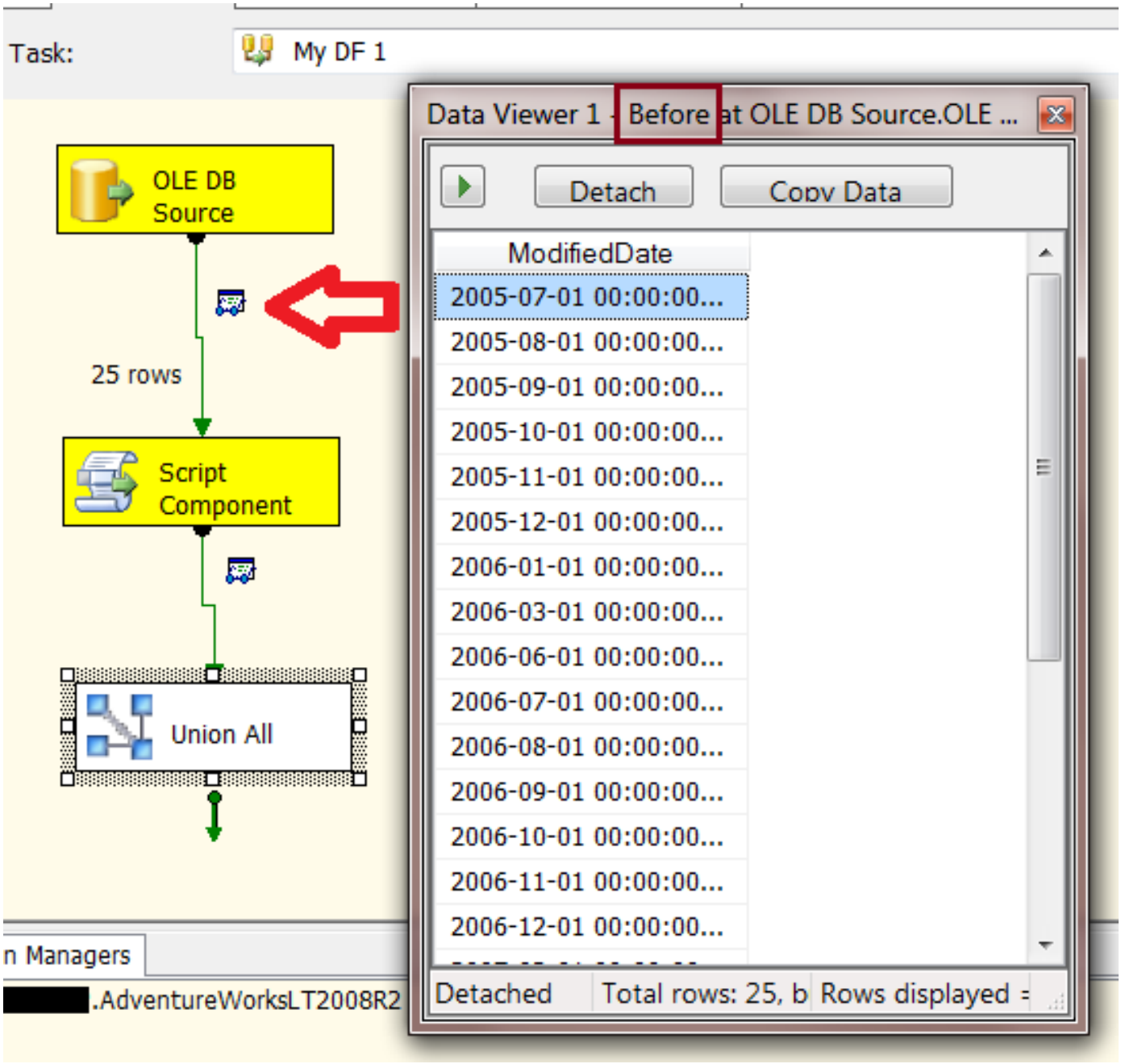


پنجره ای مانند زیر برای ویرایش کدها ، باز می‌شود

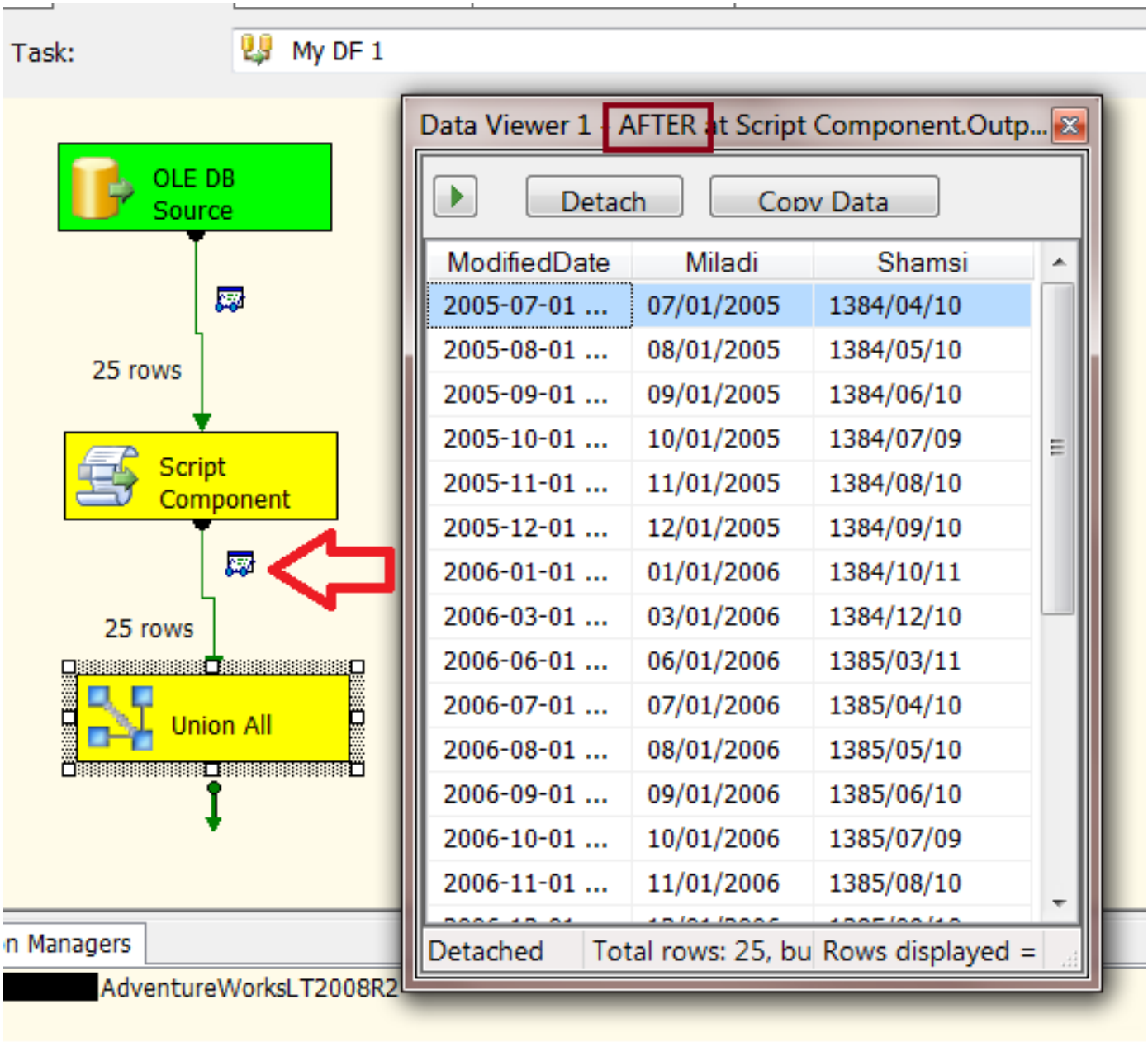


همانطور که مشاهده می‌کنید درون این کلاس 4 متد تبدیل تاریخ را پیاده کردم و از آنها در متد input0_processInputRow استفاده کردم . این کار نیازی به پیاده سازی حلقه ندارد و به راحتی می‌توان آنها را روی سطرهای دلخواه تعریف کرد . خروجی نمایش داده شده در data viewerها مانند زیر می‌باشند :

قبل از اعمال تبدیلات :



بعد از اعمال تبدیلات :


Task:
My DF 1
OLE DB Source
25 rows
Script Component
25 rows
Union All
Data Viewer 1 - AFTER at Script Component.Outp...
Detach
Copy Data
ModifiedDate
Miladi
Shamsi
2005-07-01 ... 07/01/2005 1384/04/10
2005-08-01 ... 08/01/2005 1384/05/10
2005-09-01 ... 09/01/2005 1384/06/10
2005-10-01 ... 10/01/2005 1384/07/09
2005-11-01 ... 11/01/2005 1384/08/10
2005-12-01 ... 12/01/2005 1384/09/10
2006-01-01 ... 01/01/2006 1384/10/11
2006-03-01 ... 03/01/2006 1384/12/10
2006-06-01 ... 06/01/2006 1385/03/11
2006-07-01 ... 07/01/2006 1385/04/10
2006-08-01 ... 08/01/2006 1385/05/10
2006-09-01 ... 09/01/2006 1385/06/10
2006-10-01 ... 10/01/2006 1385/07/09
2006-11-01 ... 11/01/2006 1385/08/10
Detached
Total rows: 25, bu
Rows displayed =
n Managers
AdventureWorksLT2008R2

موفق باشید

نظرات خوانندگان

نویسنده: پژمان
تاریخ: ۱۳۹۱/۰۵/۳۱ ۲۲:۱۰

سلام. ممنون از شما.

کاربرد ssis در محیط کاری برای شما بیشتر در چه مواردی است؟

نویسنده: محمد باقر سیف اللهی
تاریخ: ۱۳۹۱/۰۶/۰۱ ۷:۴۴

سلام ... SSIS کاربردهای زیادی داره مخصوصا وقتی که با تکنولوژی‌های دیگه ترکیب بشه .... اما نمونه ای که خودم با اون برخورد داشتم ، برای انتقال داده از پایگاه داده‌های قدیمی با فرمت هایی مثل Access به پایگاه داده‌های جدید در قالب SQL بود که خیلی مفید بود و سرعت اجرای عملیاتش فوق العاده است . موفق باشید

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۶/۰۱ ۱۳:۶

ما نیز با SSIS زیاد سر و کار داریم. چون از زبانی با نام Natural استفاده می‌کنیم که برای به اشتراک گذاری داده‌ها فقط می‌تونه با فایل‌های متنی ساده کار کنه. SSIS به ما کمک می‌کنه تا داده‌های ایجاد شده رو به SQL Server منتقل کنیم.

نویسنده: علیرضا
تاریخ: ۱۳۹۱/۰۸/۲۷ ۱۴:۴۹

/http://learnbi.ir/1391/08/22/post-14

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۲۷ ۱۸:۴۶

امکان به اشتراک گذاشتن مطالب برای کلیه اعضای سایت وجود دارد:
http://www.dotnettips.info/DailyLinks
لطفا برای بعد از این قسمت ویژه استفاده کنید.

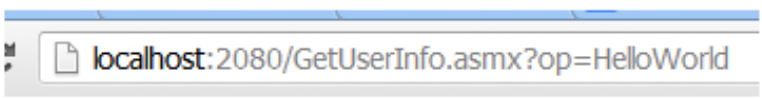
عنوان: **استخراج داده های وب سرویس توسط SSIS**
نویسنده: محمد باقر سیف اللهی
تاریخ: ۱۳۹۱/۱۰/۲۲ ۱۲:۵۵
آدرس: www.dotnettips.info
گروه‌ها: SSIS, Web-Service

یکی از راه‌های انتقال اطلاعات بین زیر سیستم ها، استفاده از وب سرویس‌ها است . در این پست نحوه استخراج اطلاعات از وب سرویس و تبدیل آنها به یک فایل XML را به کمک packageهای SSIS توضیح می‌دهم .
در قدم اول نیاز به یک سرویس داریم که برای نمونه سرویس زیر را طراحی و پیاده سازی می‌کنیم :
متدی به نام HelloWorld در کلاس GetUserInfo که خروجی آن آرایه ای از کلاس UserInfo است.

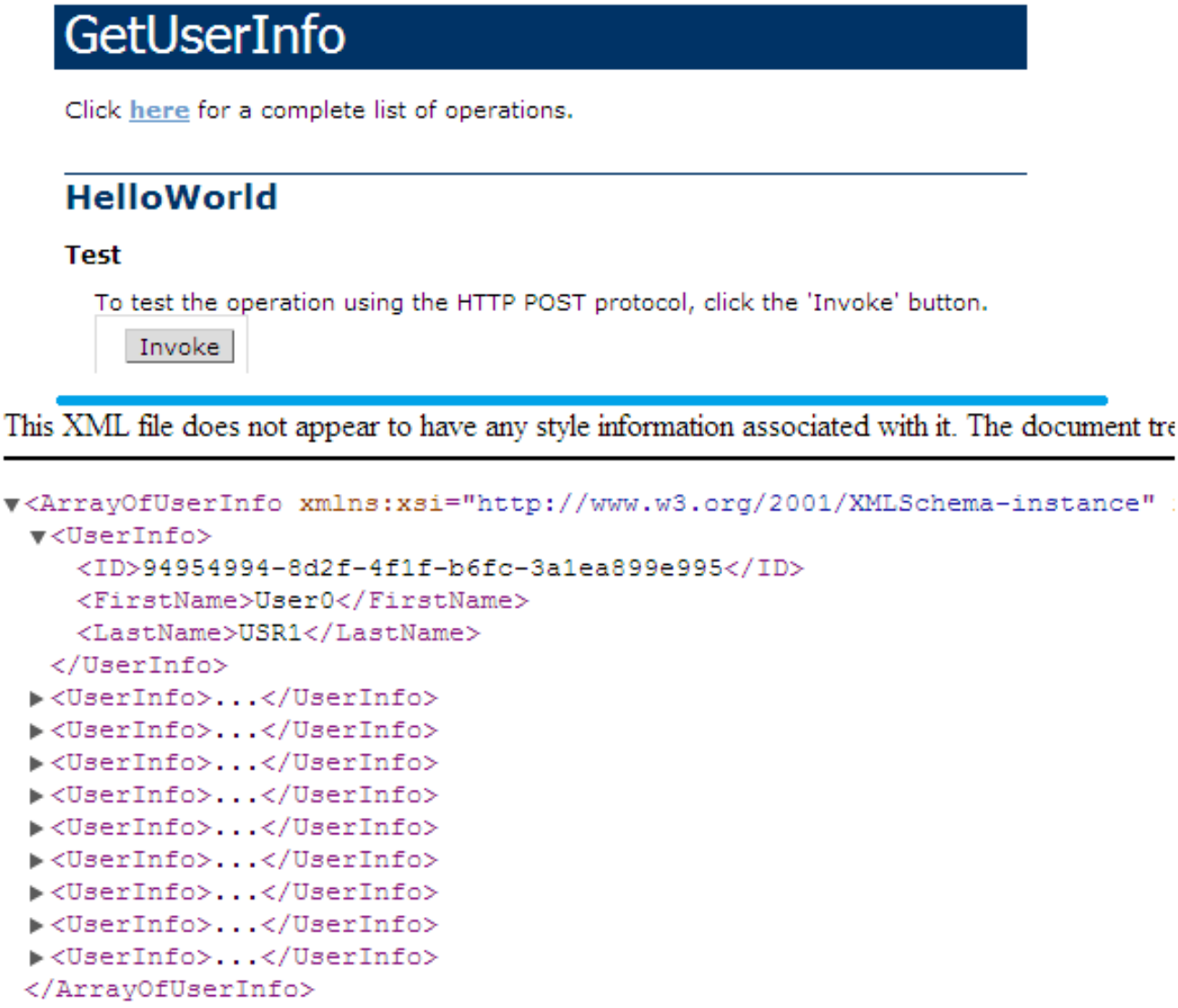
GetUserInfo.asmx.cs

MBS.MySrv.GetUserInfo — HelloWorld()

```
public class GetUserInfo : System.Web.Services.WebService
{

    [WebMethod]
    public UserInfo[] HelloWorld()
    {
        List<UserInfo> data = new List<UserInfo> ();
        for (int i = 0; i < 10; i++)
        {
            data.Add(new UserInfo() {ID = Guid.NewGuid().ToString() ,
                FirstName = "User"+i.ToString() ,
                LastName = "USR"+(i+1).ToString() });
        }
        return data.ToArray();
    }
}
public class UserInfo
{
    string _ID;
    public string ID...

    string _FirstName;
    public string FirstName...

    string _LastName;
    public string LastName...


}
}
```



و اجرای سرویس



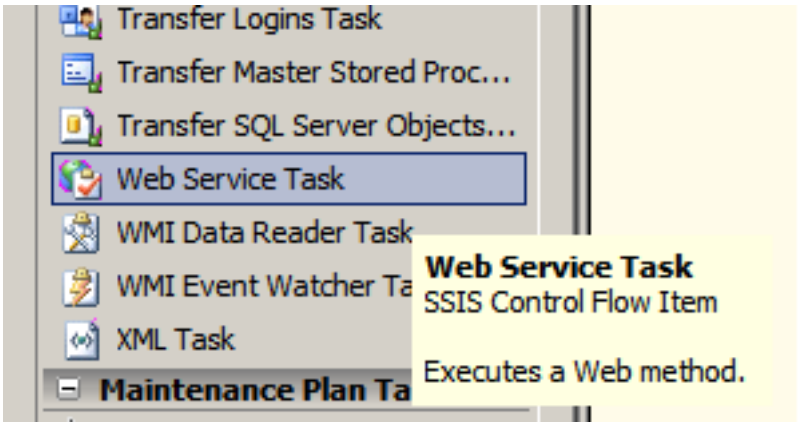
و ساختار سرویس که SSIS به این ساختار نیاز دارد .

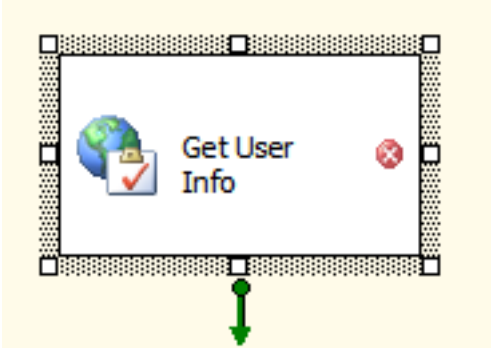
localhost:2080/GetUserInfo.asmx?wsdl

This XML file does not appear to have any style information associated with it. The document tree is shown below.

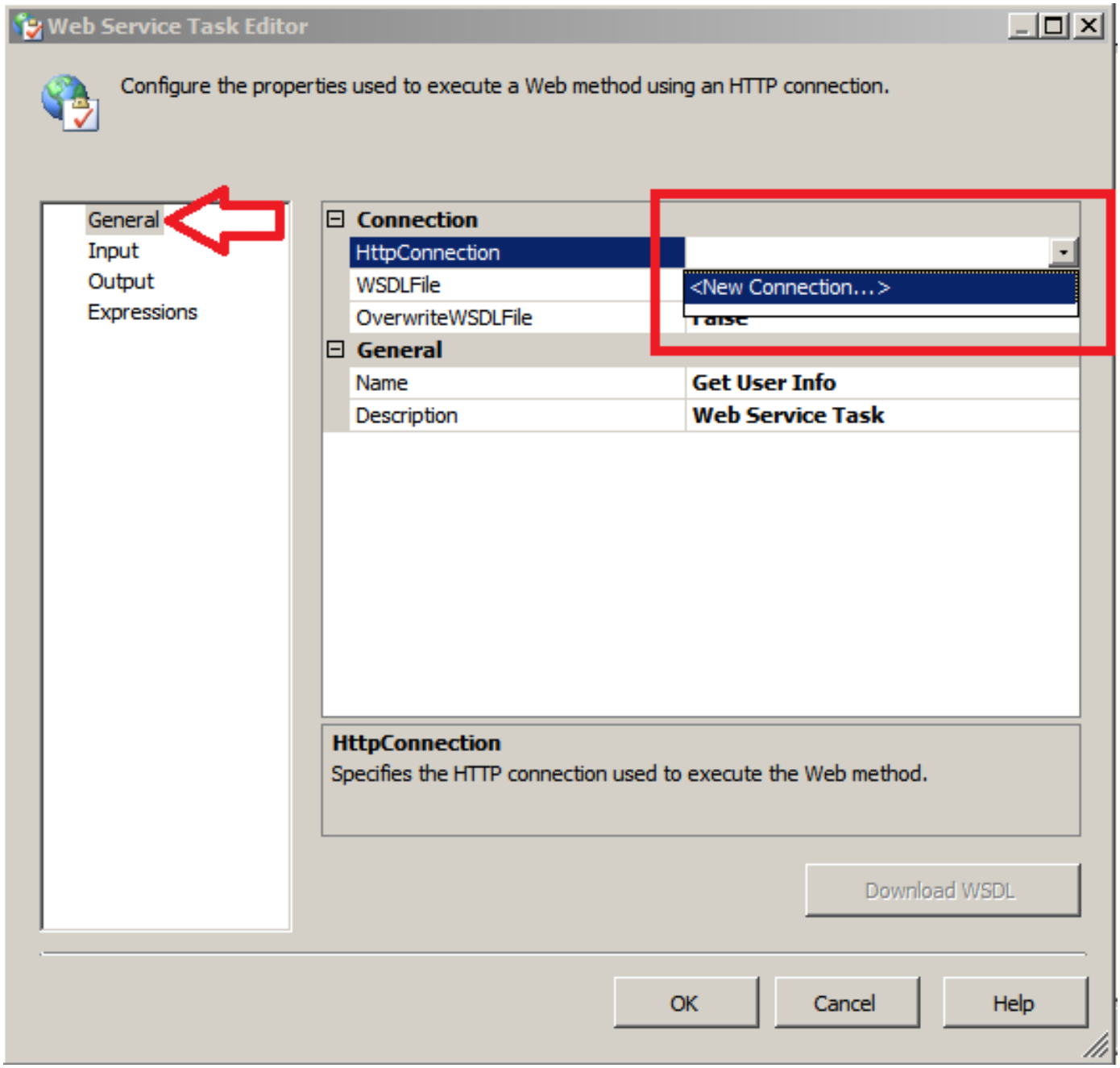
```
▼<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tm="http://m
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xm
 xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/wsd
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://tempuri.org/">
  ▼<wsdl:types>
    ▼<s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
      ▶<s:element name="HelloWorld">...</s:element>
      ▶<s:element name="HelloWorldResponse">...</s:element>
      ▶<s:complexType name="ArrayOfUserInfo">...</s:complexType>
      ▼<s:complexType name="UserInfo">
        ▼<s:sequence>
           <s:element minOccurs="0" maxOccurs="1" name="ID" type="s:string"/>
           <s:element minOccurs="0" maxOccurs="1" name="FirstName" type="s:string"/>
           <s:element minOccurs="0" maxOccurs="1" name="LastName" type="s:string"/>
         </s:sequence>
       </s:complexType>
     </s:schema>
   </wsdl:types>
  ▶<wsdl:message name="HelloWorldSoapIn">...</wsdl:message>
  ▶<wsdl:message name="HelloWorldSoapOut">...</wsdl:message>
  ▶<wsdl:portType name="GetUserInfoSoap">...</wsdl:portType>
  ▶<wsdl:binding name="GetUserInfoSoap" type="tns:GetUserInfoSoap">...</wsdl:binding>
  ▶<wsdl:binding name="GetUserInfoSoap12" type="tns:GetUserInfoSoap">...</wsdl:binding>
  ▶<wsdl:service name="GetUserInfo">...</wsdl:service>
 </wsdl:definitions>
```

حال از محیط BIDS یک پروژه SSIS ایجاد می‌کنیم (برای آشنایی با BIDS به این پست مراجعه کنید) و یک Web Service Task به آن اضافه می‌کنیم :

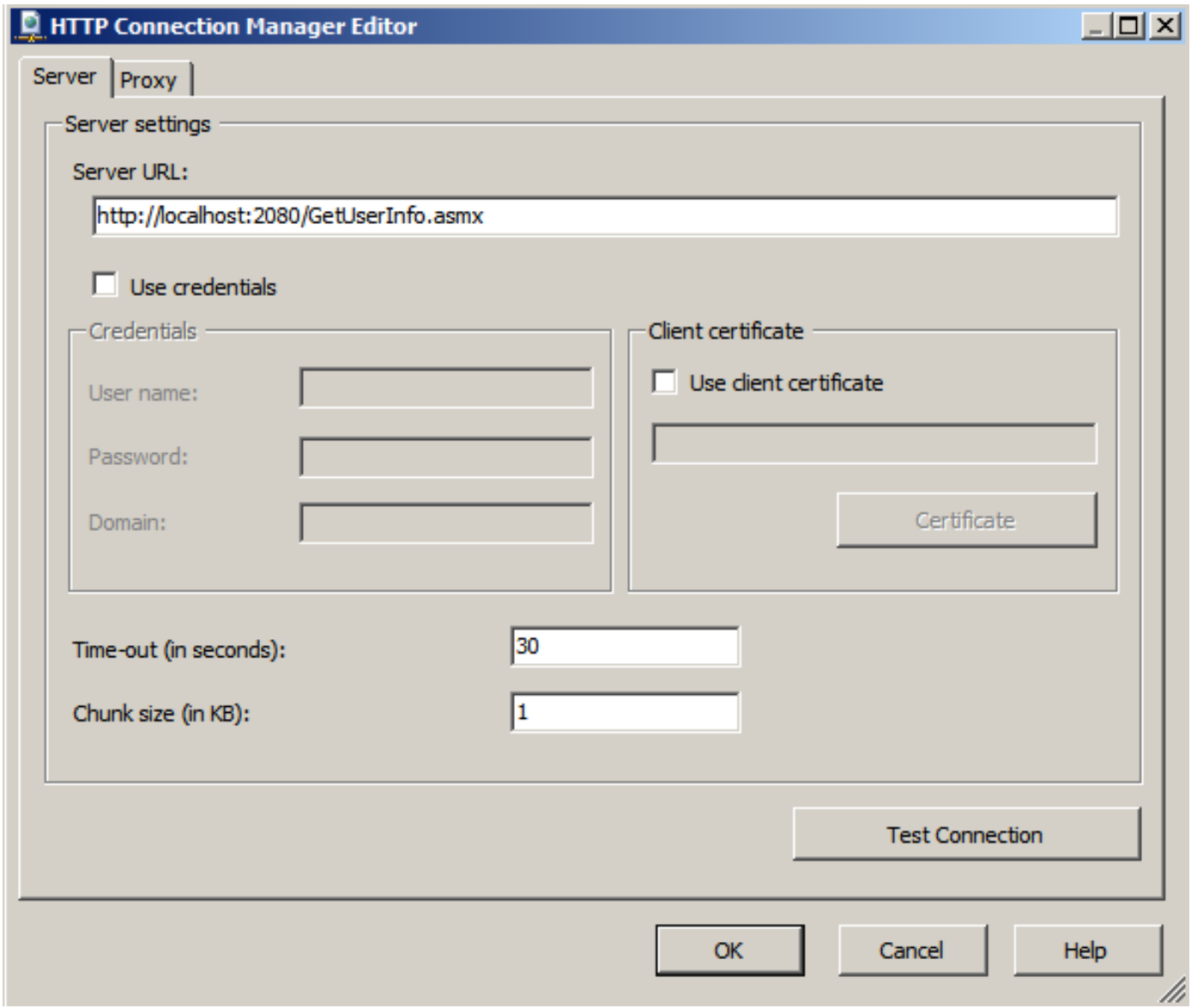




سپس روی task دوبار کلیک کنید تا پنجره تنظیمات آن باز شود :

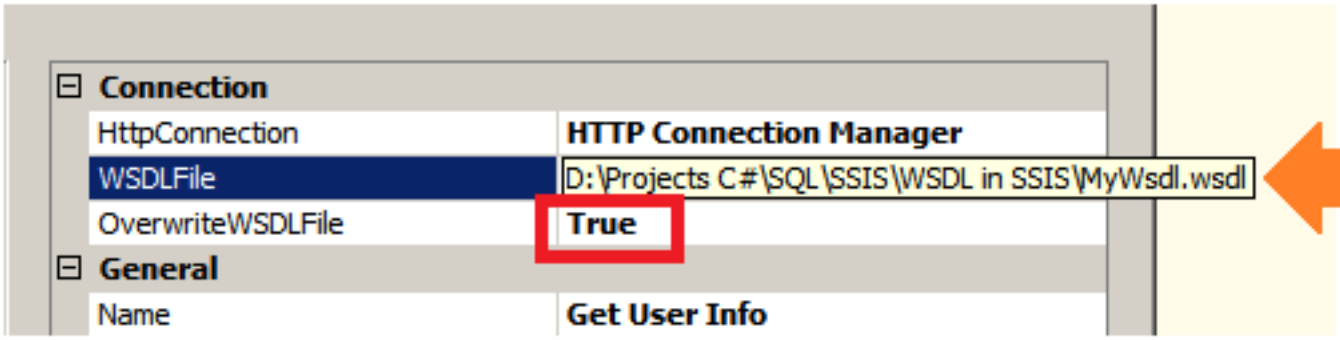


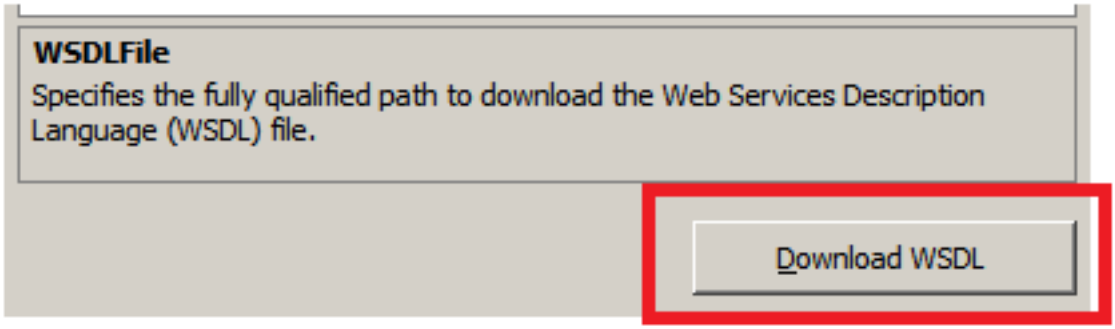
حال باید یک دیتا سورس کانکشن که همان سرویس ما می‌باشد تعریف کنیم :

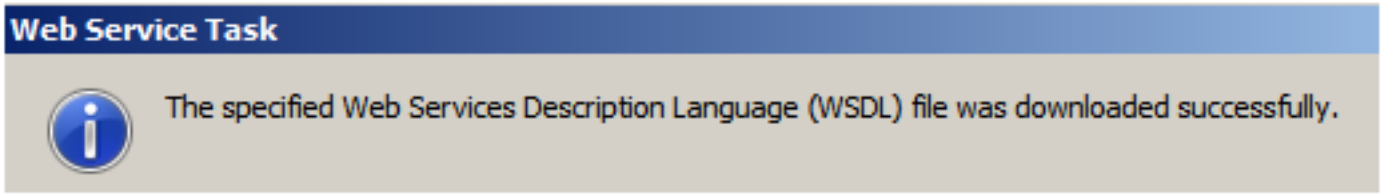


دکمه تست را فشار دهید تا تاییدیه معتبر بودن سرویس را دریافت کنید ، OK را فشار دهید .

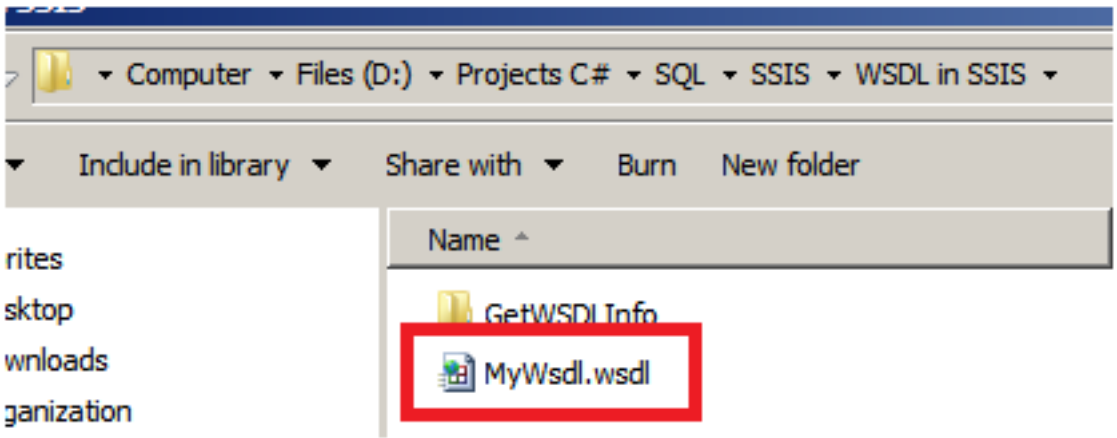
در قدم بعد SSIS به قالب فایل xml که در خروجی WSDL است نیاز دارد . اگر این فایل را دارید آدرس آن را باید در قسمت WSDL FIle وارد کنید در غیر این صورت (مانند شرایط فعلی این مثال) باید این فایل را از سرویس خود دانلود کنید . نکته اینکه در صورتی که این فایل را ندارید ، گزینه overwriteWSDLFile را True کنید و مسیر یک فایلی که در سیستم شما موجود نیست را در آدرس WSDLFile وارد کنید .



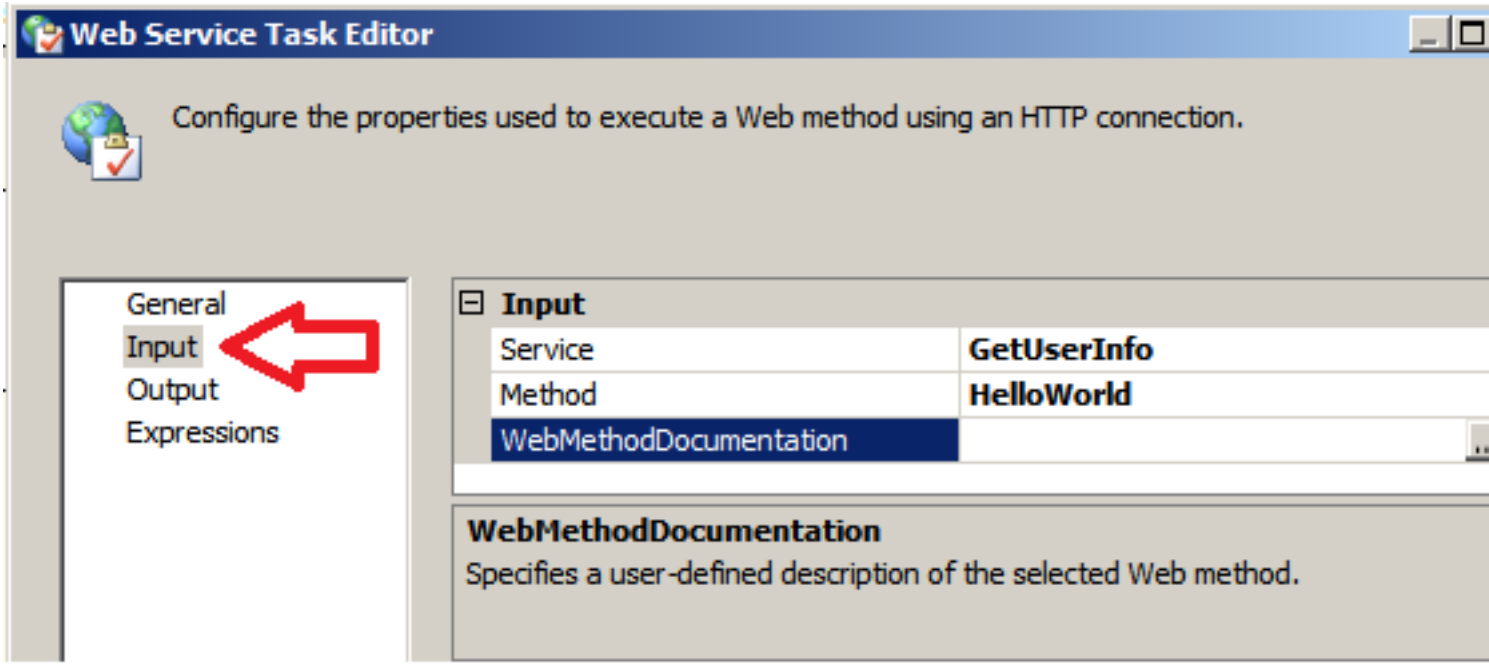




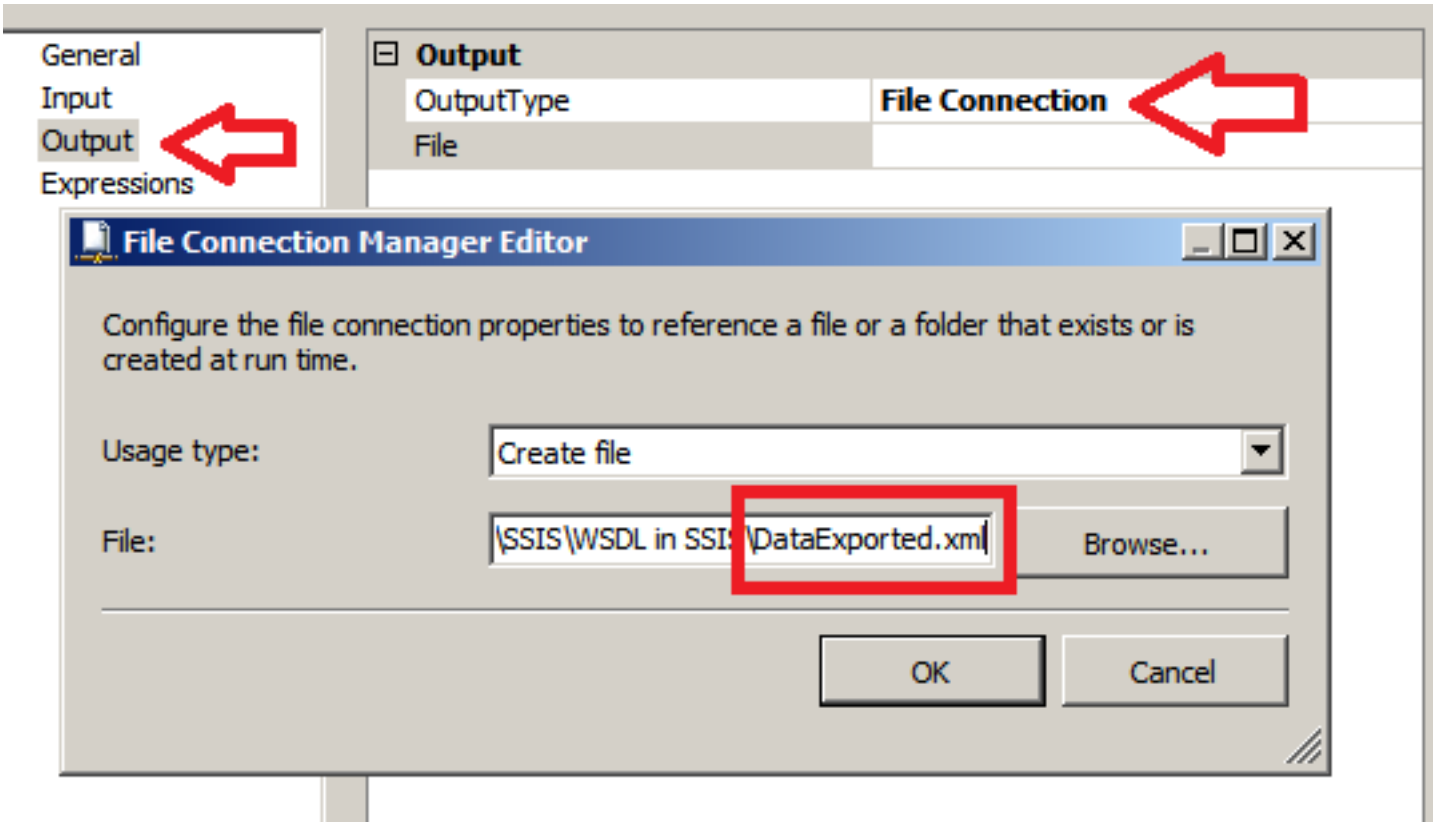
حال شما فایل مورد نظر را دارید :



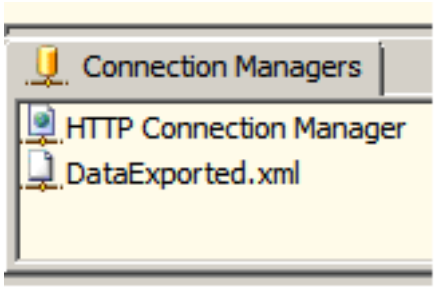
اگر به تب input بازگردید می‌توانید ادامه تنظیمات را انجام دهید :



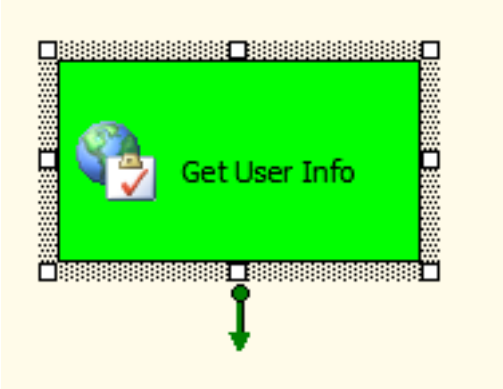
حال باید خروجی مورد نظر از این سرویس را تعریف کنیم . دو نوع خروجی برای ما امکان پذیر است . یکی انتقال اطلاعات به یک فایل (مناسب برای مواردی که نیاز به داده‌های offline دارید) و یکی منتقل کردن آنها به متغیرهای خود SSIS Package برای استفاده در کام‌های بعدی flow (برای مواردی که نیاز به انجام تغییرات روی داده‌های Online دارید) .



پس از انجام این تنظیمات باید کانکشن هایی مطابق زیر داشته باشید :



F5 را فشار دهید تا عملیات شروع شود :



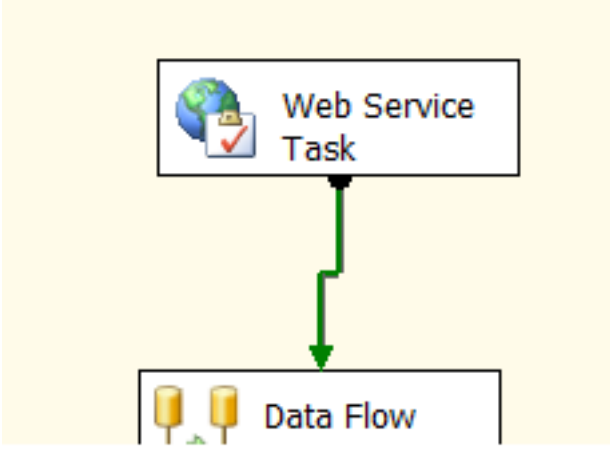
و خروجی :


```
<?xml version="1.0" encoding="utf-16" ?>
- <ArrayOfUserInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
  - <UserInfo>
      <ID xmlns="http://tempuri.org/">0f7f5abb-292a-4b46-ba2a-3651d9715397</ID>
      <FirstName xmlns="http://tempuri.org/">User0</FirstName>
      <LastName xmlns="http://tempuri.org/">USR1</LastName>
    </UserInfo>
  + <UserInfo>
  + <UserInfo>
  + <UserInfo>
  + <UserInfo>
  + <UserInfo>
  + <UserInfo>
  + <UserInfo>
  + <UserInfo>
  - <UserInfo>
      <ID xmlns="http://tempuri.org/">5501dc85-c199-4531-bcdd-cc8e61cb970d</ID>
      <FirstName xmlns="http://tempuri.org/">User9</FirstName>
      <LastName xmlns="http://tempuri.org/">USR10</LastName>
    </UserInfo>
</ArrayOfUserInfo>
```

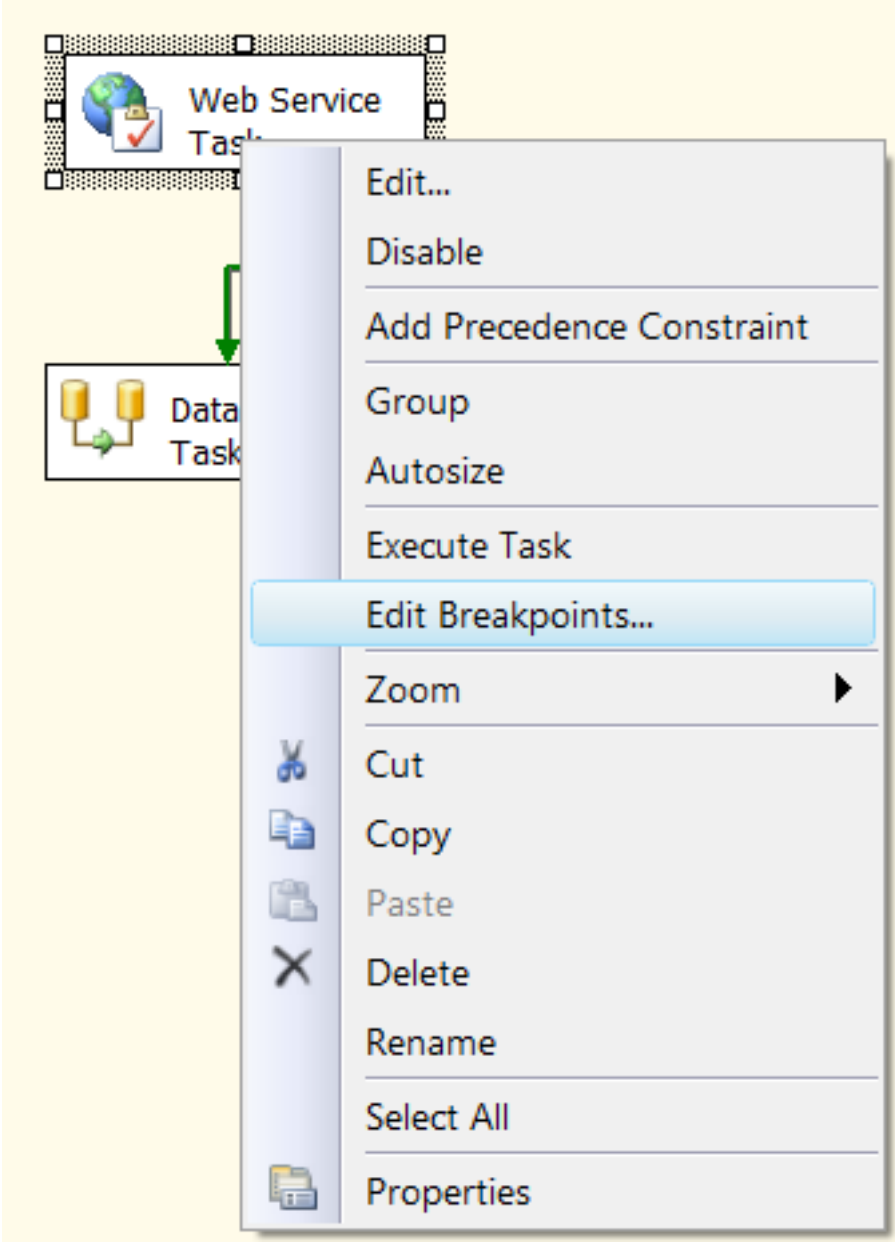
موفق باشید

عنوان: **نحوه Debug کردن پکیج های SSIS و استفاده از Breakpoint در آنها**
نویسنده: محمد باقر سیف اللهی
تاریخ: ۱۳۹۲/۰۲/۳۰ ۲۲:۰
آدرس: www.dotnettips.info
برچسب‌ها: Tricks, SSIS

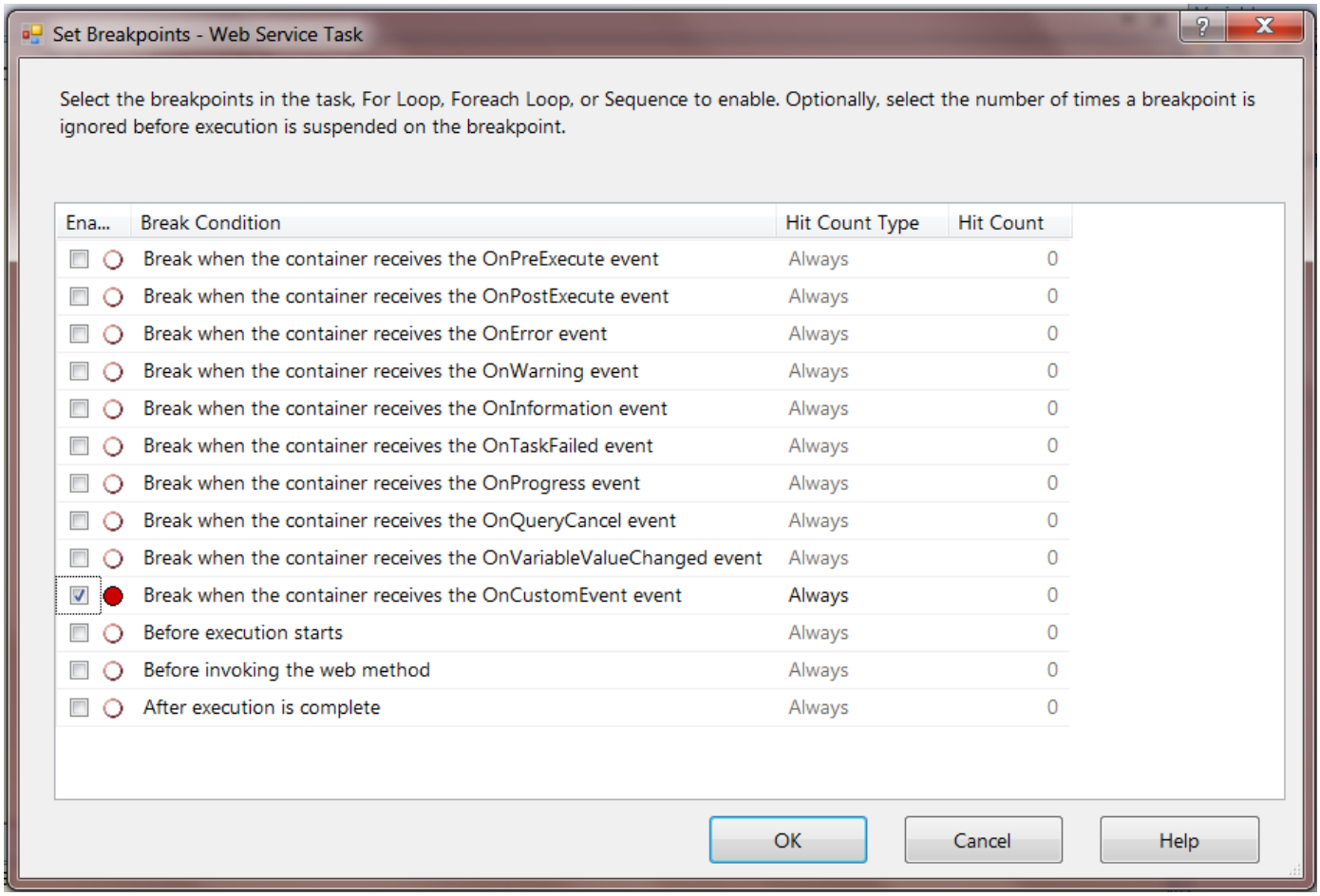
فرض کنید می‌خواهیم اطلاعات یک وب سرویس را داخل یک متغیر در package ریخته و پس از مقدار دهی ، مقدار آن متغیر آن را مشاهده کنیم . (برای اطلاع از کار با وب سرویس به اینجا مراجعه کنید) .



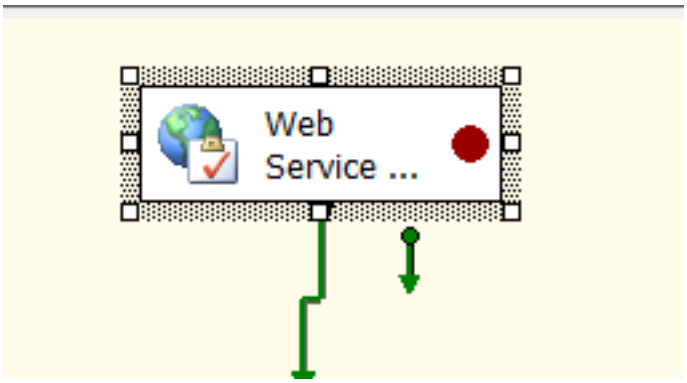
برای این کار روی کنترل کلیک سمت راست کرده و گزینه Edit Breakpoints را انتخاب می‌کنیم :



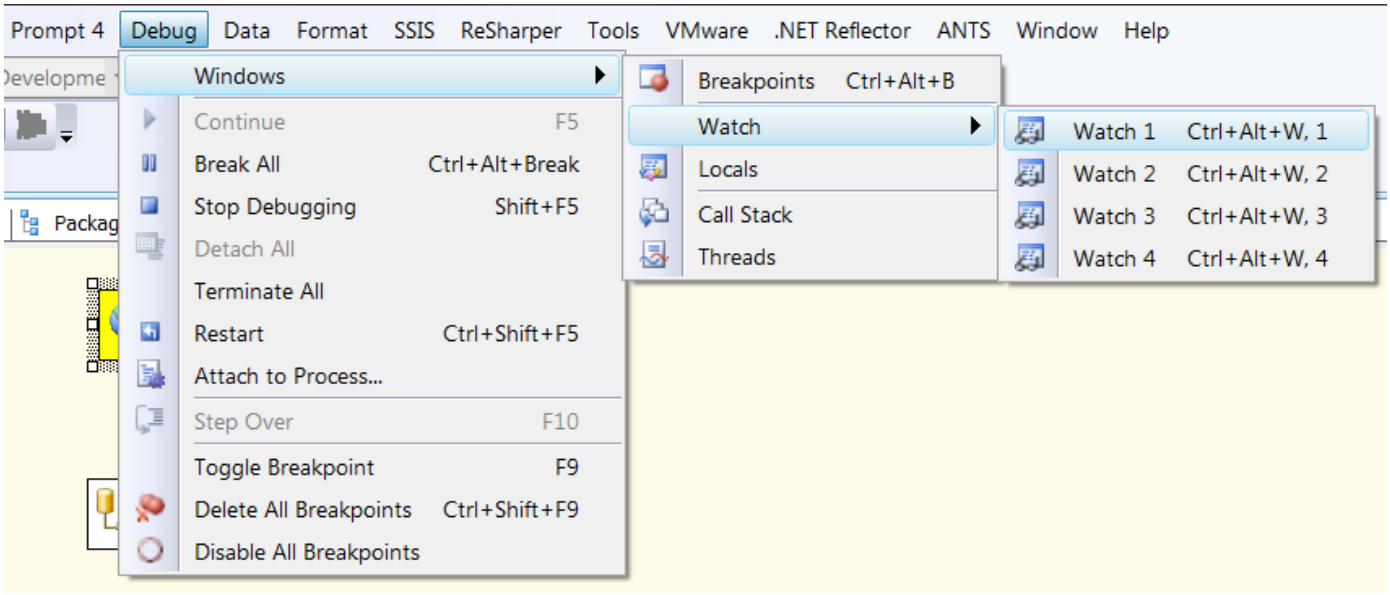
در پنجره Set Breakpoints گزینه هایی که می‌خواهیم در آنها break انجام شود را انتخاب می‌کنیم . لازم به ذکر است که این موارد بسته به کنترل‌های مختلف تا حدود کمی با هم فرق می‌کنند

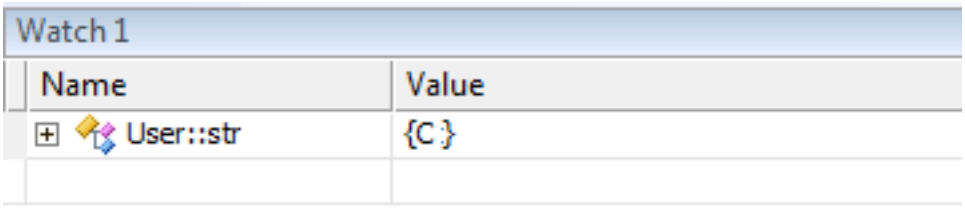


پس از انتخاب گزینه یا گزینه‌های مورد نظر و بستن پنجره یک آیکون کنار کنترل نمایش داده می‌شود .



اکنون می‌توانید با باز کردن پنجره watch پس از آغاز شدن debug به مشاهده مقادیر مورد نظر بپردازید . در پنجره watch می‌توانید نام متغیر را وارد کنید تا برای شما نمایش داده شود.


Prompt 4
Debug
Data
Format
SSIS
ReSharper
Tools
VMware
.NET Reflector
ANTS
Window
Help
Windows
Continue F5
Break All Ctrl+Alt+Break
Stop Debugging Shift+F5
Detach All
Terminate All
Restart Ctrl+Shift+F5
Attach to Process...
Step Over F10
Toggle Breakpoint F9
Delete All Breakpoints Ctrl+Shift+F9
Disable All Breakpoints
Breakpoints Ctrl+Alt+B
Watch
Locals
Call Stack
Threads
Watch 1 Ctrl+Alt+W, 1
Watch 2 Ctrl+Alt+W, 2
Watch 3 Ctrl+Alt+W, 3
Watch 4 Ctrl+Alt+W, 4

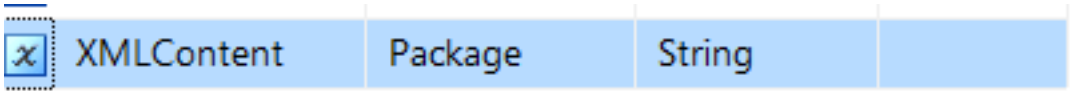

Watch 1
Name
Value
User::str
{C}

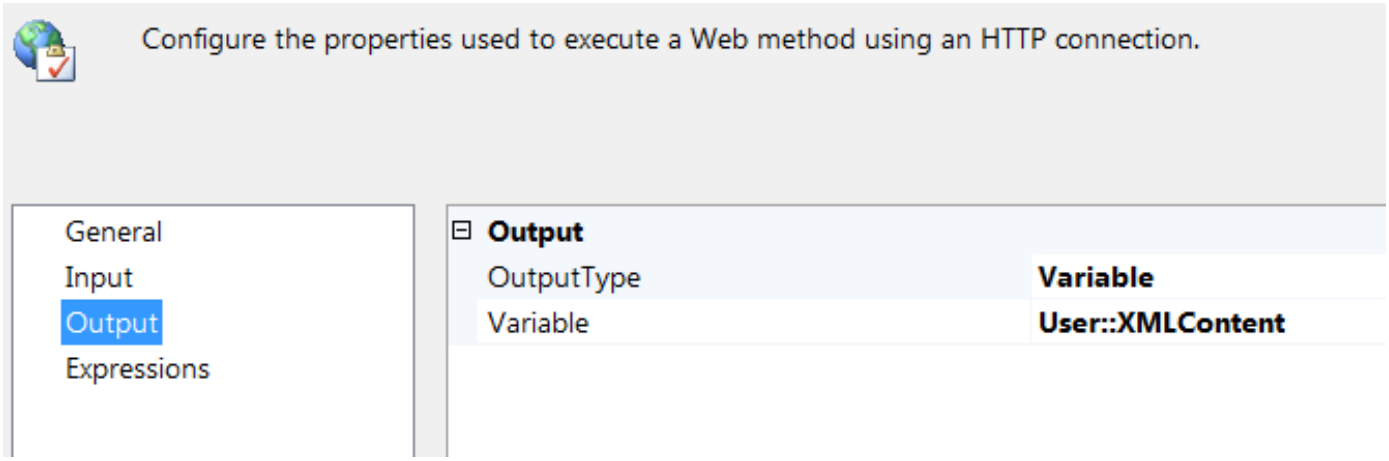
موفق باشید

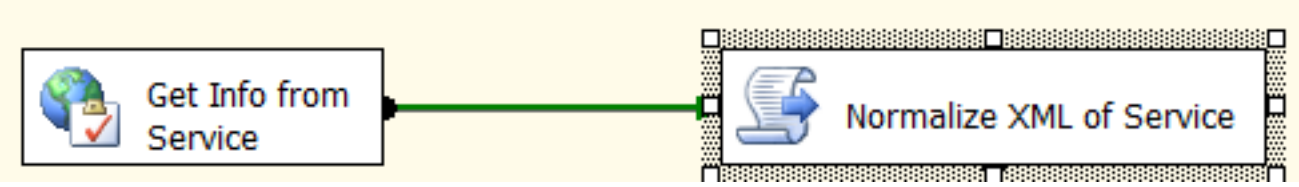
عنوان: **نحوه انتقال اطلاعات استخراج شده از وب سرویس به SQL Server به کمک SSIS**
نویسنده: محمد باقر سیف اللهی
تاریخ: ۱۳۹۲/۰۲/۳۰ ۲۲:۱۵
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, Web Service, SSIS, data mining

ممکن است در مواقعی نیاز به اطلاعات استخراج شده از وب سرویسی داشته باشید که در همان مقطع زمانی به آن دسترسی ندارید . مسلما برای این منظور باید آن اطلاعات را ذخیره کرده تا در صورت نیاز بتوان به آنها رجوع کرد . یکی از راه حل‌ها ذخیره آن در پایگاه داده (در اینجا Sql Server) است که در این پست به کمک امکانات BIDS در پکیج‌های SSIS و کوئری‌های SQL این مشکل را برطرف میکنیم. برای مشاهده نحوه استخراج اطلاعات از وب سرویس به اینجا مراجعه کنید .

تنها تفاوتی که در این پست در کار با سرویس با پست اشاره شده در بالا وجود دارد ذخیره اطلاعات استخراج شده است که در آن پست در یک فایل xml ذخیره شدند ولی در اینجا ما نیاز داریم تا اطلاعات را در یک متغیر با حوزه کاری Package ذخیره کنیم (به این معنی که مختص به همان flow نباشد و در تمام پکیج دیده شود)



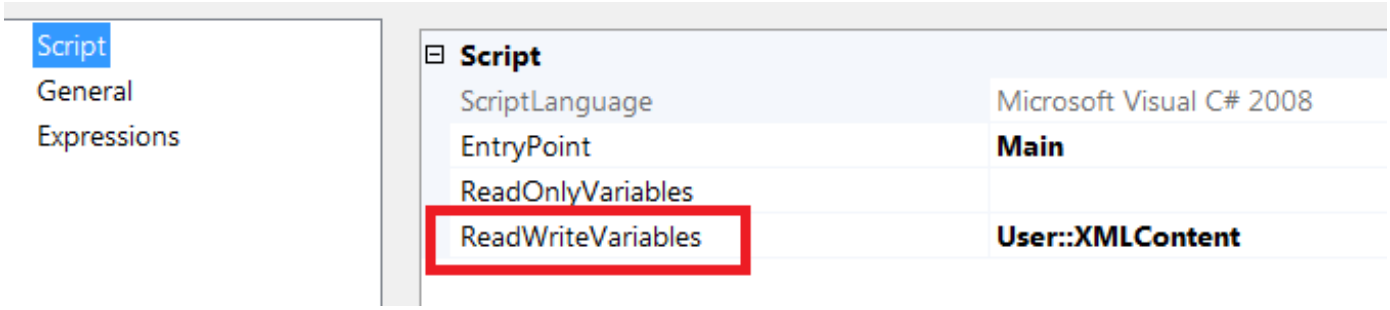




به دلیل اینکه هدر xml خروجی از سرویس دارای چندین namespace هست هنگام کار با آنها به مشکل خواهیم خورد. (هم هنگام کار با xmlها task و هم هنگام کار با xml در sql)

```
▼<ArrayOfUserInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

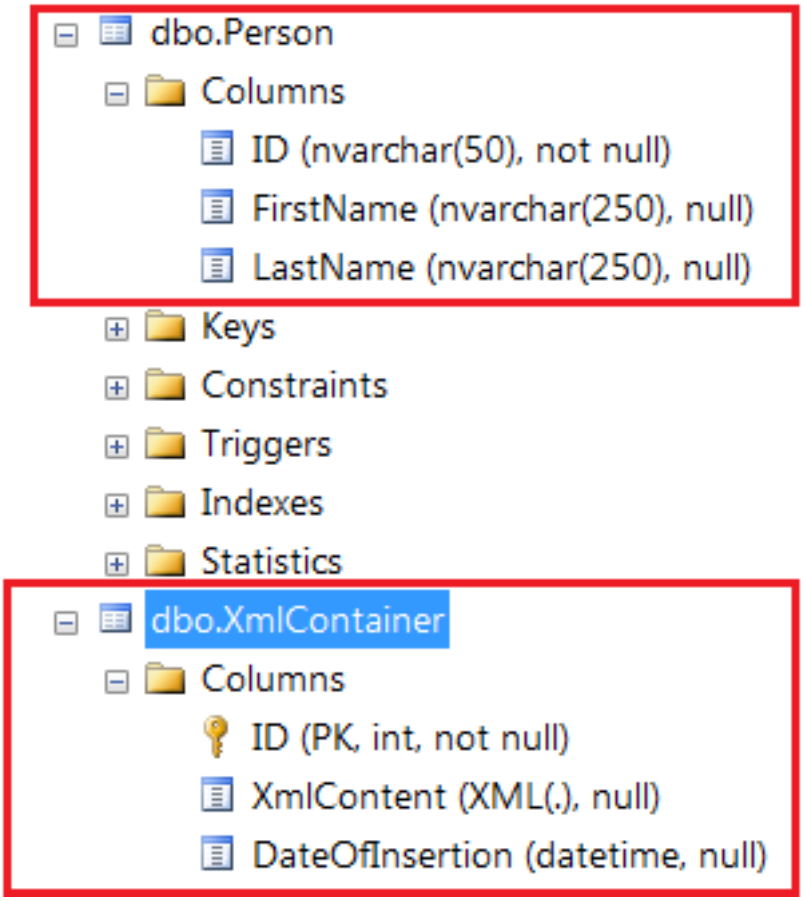
به همین دلیل باید این قسمت از محتوا را حذف کرد . برای همین پس از گرفتن اطلاعات از سرویس آن را به کمک یک Script task حذف می‌کنیم




```
public void Main()
{
    //xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://tempuri.org/"
    object o = "null";
    byte[] emptyBytes = new byte[0];
    try
    {
        Dts.VariableDispenser.LockForRead("User::XMLContent");
        Variable xmlc = Dts.Variables["User::XMLContent"];
        o = xmlc.Value;
        if (o != null)
        {
            string s = o.ToString();
            s = s.Replace("xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"", "");
            s = s.Replace("xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"", "");
            s = s.Replace("xmlns=\"http://tempuri.org/\"", "");
            Dts.VariableDispenser.Reset();
            Dts.VariableDispenser.LockForWrite("User::XMLContent");
            Dts.Variables["User::XMLContent"].Value = s;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error SSIS : " + Environment.NewLine + Environment.NewLine + ex.Message, "Error on Package...");
        Dts.TaskResult = (int)ScriptResults.Failure;
    }
    finally
    {
        Dts.VariableDispenser.Reset();
    }

    Dts.Log("nameSpace Removed Successfully...", 0, emptyBytes);
    Dts.TaskResult = (int)ScriptResults.Success;
}
```

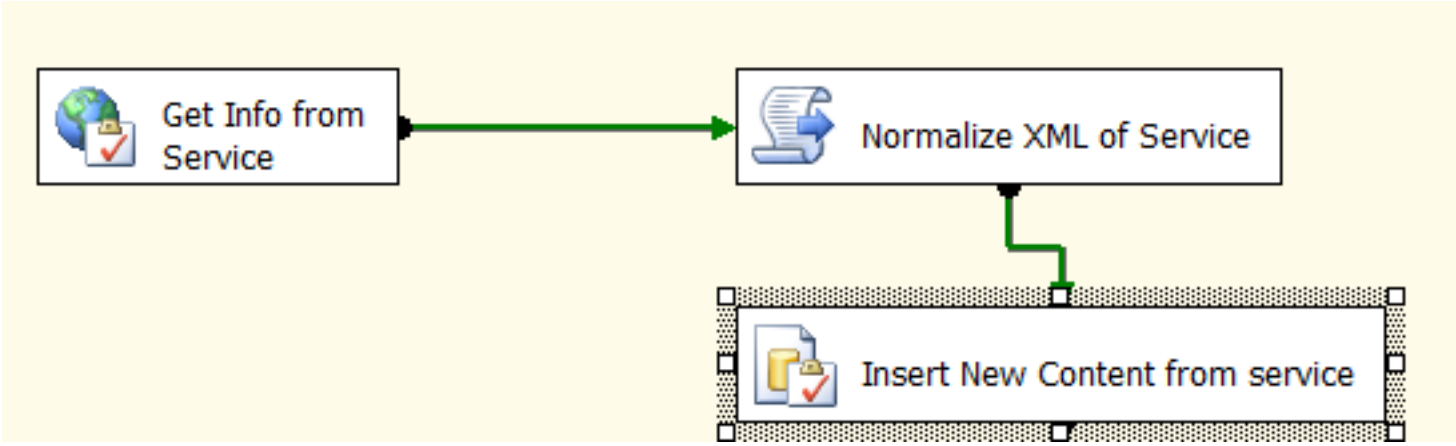
در این مرحله اطلاعات استخراج شده را باید در SQL درج کنیم . برای همین ساختاری که باید اطلاعات را در SQL نگه دارد را در دیتابیس ایجاد می‌کنیم :

جدول person برای نگهداری اطلاعات سرویس و XmlContainer برای نگهداری xmlهای سرویس .(برای داشتن History)

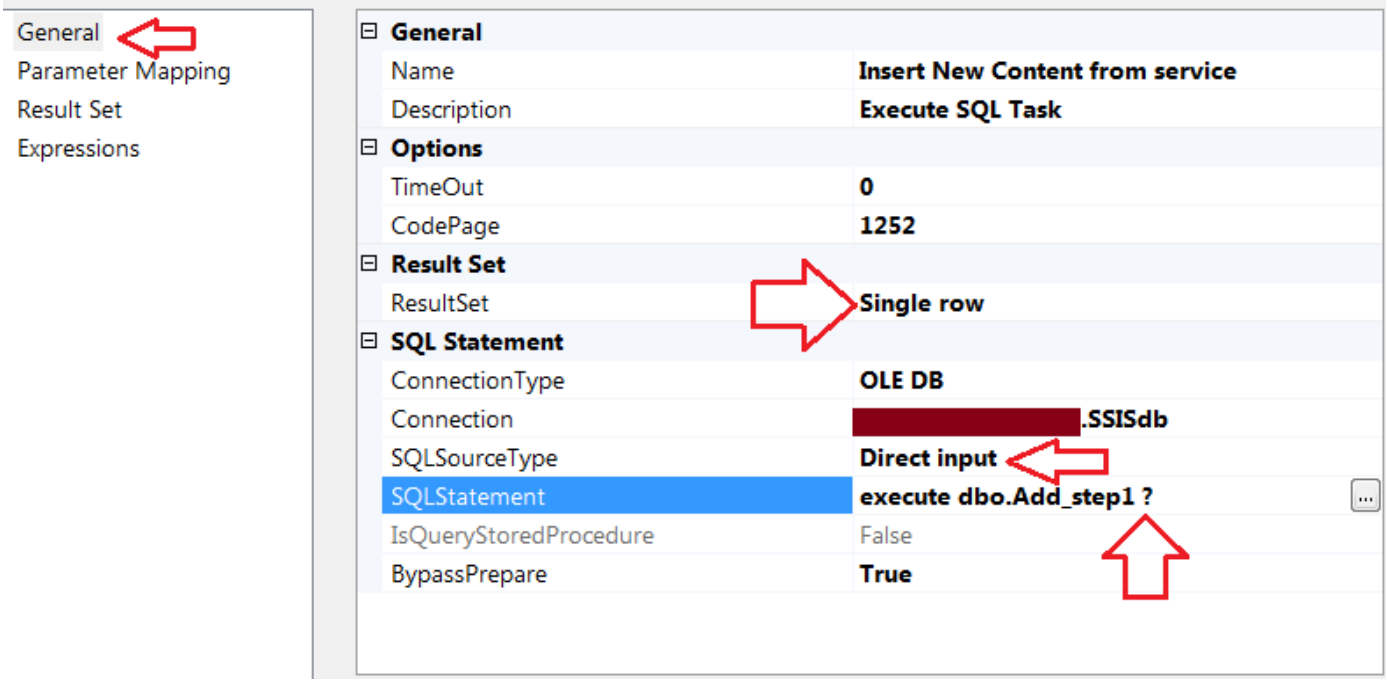


برای درج هم از SP استفاده می‌کنیم :

```
ALTER PROCEDURE [dbo].[Add_step1]

    @xml XML
AS
BEGIN
    SET NOCOUNT ON;

INSERT INTO dbo.XmlContainer
values( @xml, GETDATE())

DECLARE @id INT
SET @id = SCOPE_IDENTITY() ;

SELECT  @id AS Result


END
```

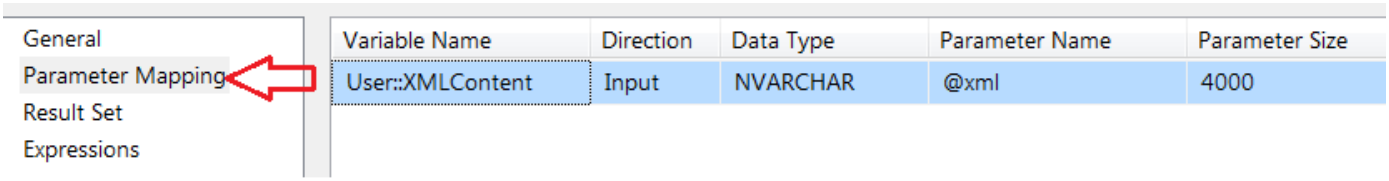


و پیکربندی SQL Task :

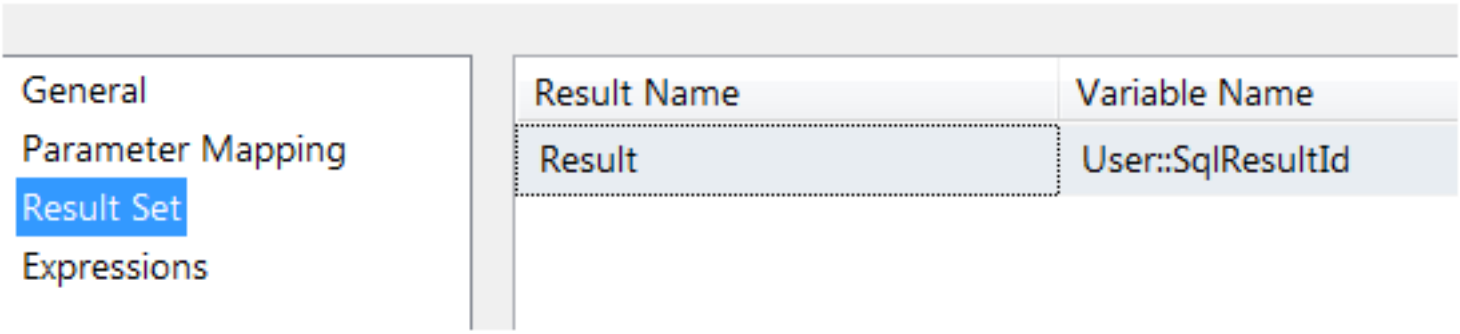


نکته اول ایجاد کانکشن به پایگاه داده است که در اینجا از نمایش جزییات آن صرف نظر شده است . نکته دیگری پارامتر SP است که چون یک پارامتر دارد یک علامت سوال قرار میگیرد . اگر چند پارامتر بود به صورت علامت‌های سوال با ویرگول از هم جدا می‌شوند . ?,?,?,?

سپس در بخش parameter mapping به ترتیب مقدار دهی می‌شوند :



و مقدار خروجی SP که شناسه آیتم درج شده است را در یک متغیر نگهداری می‌کنیم :



برای قدم بعد می‌خواهیم اطلاعات موجود در XML استخراج شده در پایگاه داده را در جدول مربوطه ذخیره کنیم . برای این کار این SP را می‌نویسیم :

```
ALTER PROCEDURE [dbo].[transform_Step2]
 @id int
AS
BEGIN

    SET NOCOUNT ON;
    DECLARE @trans CHAR(50),@xml XML
    SET @trans = 'MyTrans'
    BEGIN TRY
    BEGIN TRANSACTION @trans

    DELETE FROM Person -- empty table
    SELECT @xml = xc.XmlContent FROM XmlContainer xc
    WHERE xc.ID = @id
    DECLARE  @dt DATETIME , @hoc int;
    EXEC sp_xml_preparedocument @hoc OUTPUT, @xml
    INSERT INTO Person (ID,FirstName,LastName)
    SELECT ID, FirstName, LastName
    FROM OPENXML(@hoc, 'ArrayOfUserInfo/UserInfo')
    WITH
    (
    ID [nvarchar](50) 'ID',
    FirstName [nvarchar](250) 'FirstName',
    LastName [nvarchar](250) 'LastName'
    )
    EXEC sp_xml_removedocument @hoc -- empty cache
    IF (@@ERROR <> 0 )
    BEGIN
        ROLLBACK TRANSACTION @trans
    END
    ELSE
    BEGIN
    COMMIT TRANSACTION @trans
     SELECT 'OK' AS  'ERROR'
    END

    END TRY
    begin CATCH
        ROLLBACK TRANSACTION @trans
        SELECT  ERROR_MESSAGE()  AS  'ERROR'
    END CATCH
END
```

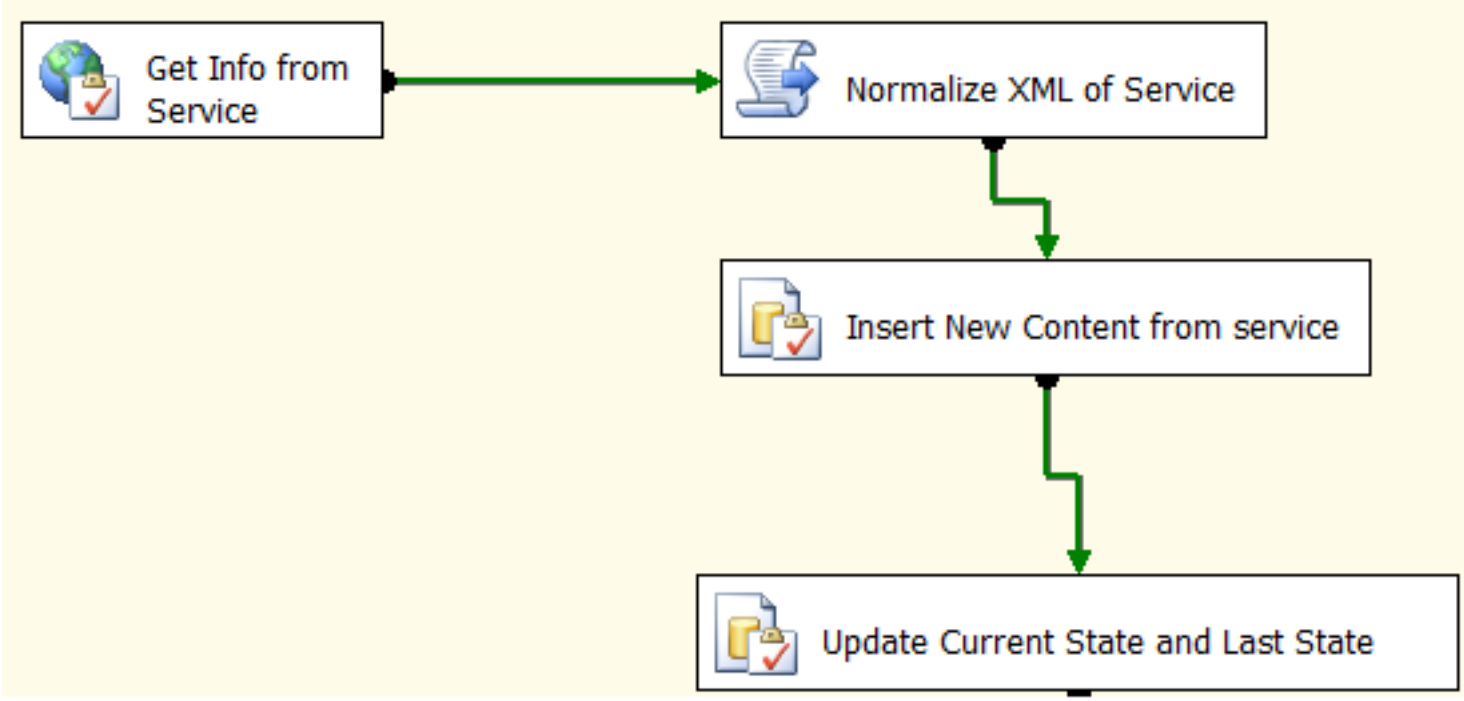
نکات مهم موارد زیر هستند :

1 - استفاده از OpenXml برای parse کردن xml

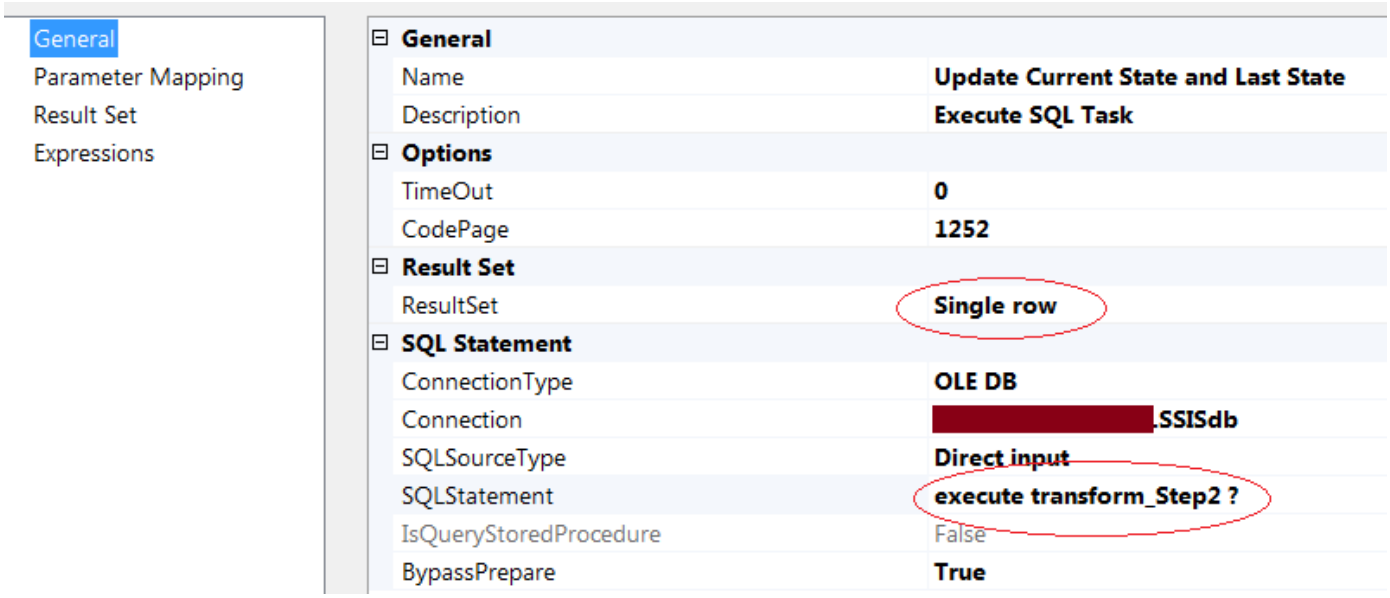
2 - استفاده از sp سیستمی sp_xml_preparedocument و sp_xml_removedocument (بیشتر)

3- اطلاعات شناسه و نام و نام خانوادگی innerهای نودهای xml هستند . اگر این موارد به صورت attribute باشند باید قبل از نام آنها علامت @ قرار بگیرند.

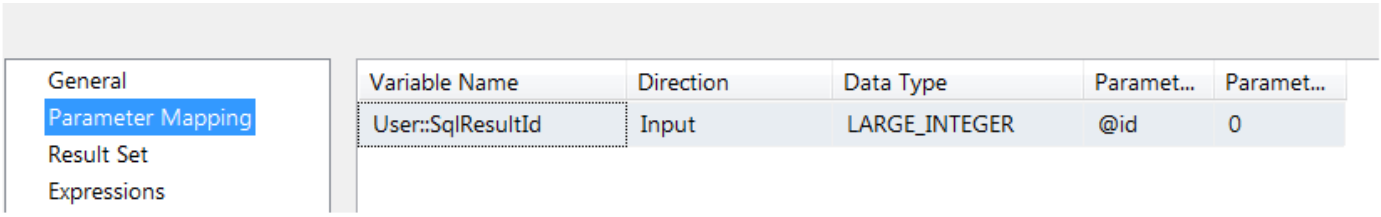
پس از ایجاد این Sp باید آن را در package فراخوانی کنیم :



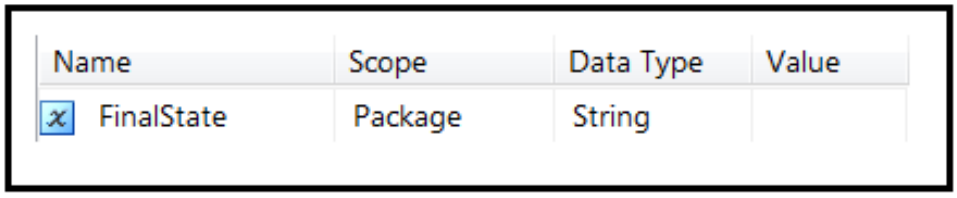
و پیکربندی این SQL Task :

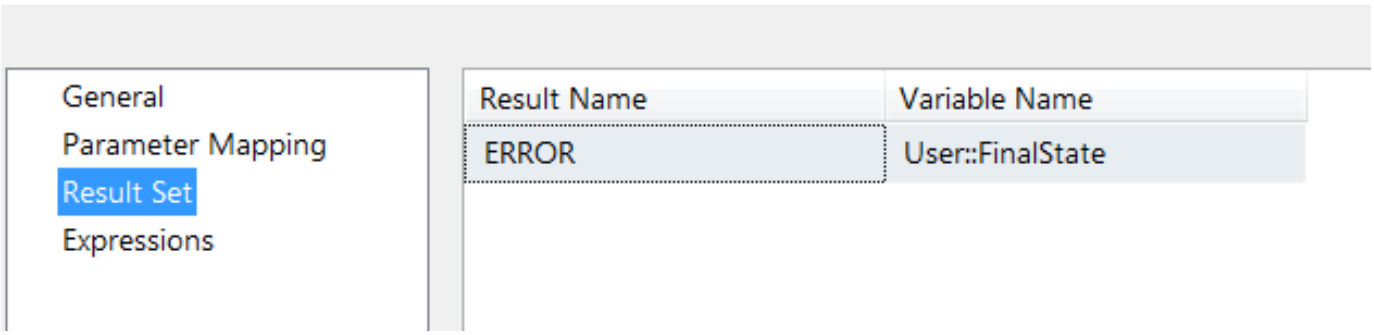


و پارامترهای این SP :

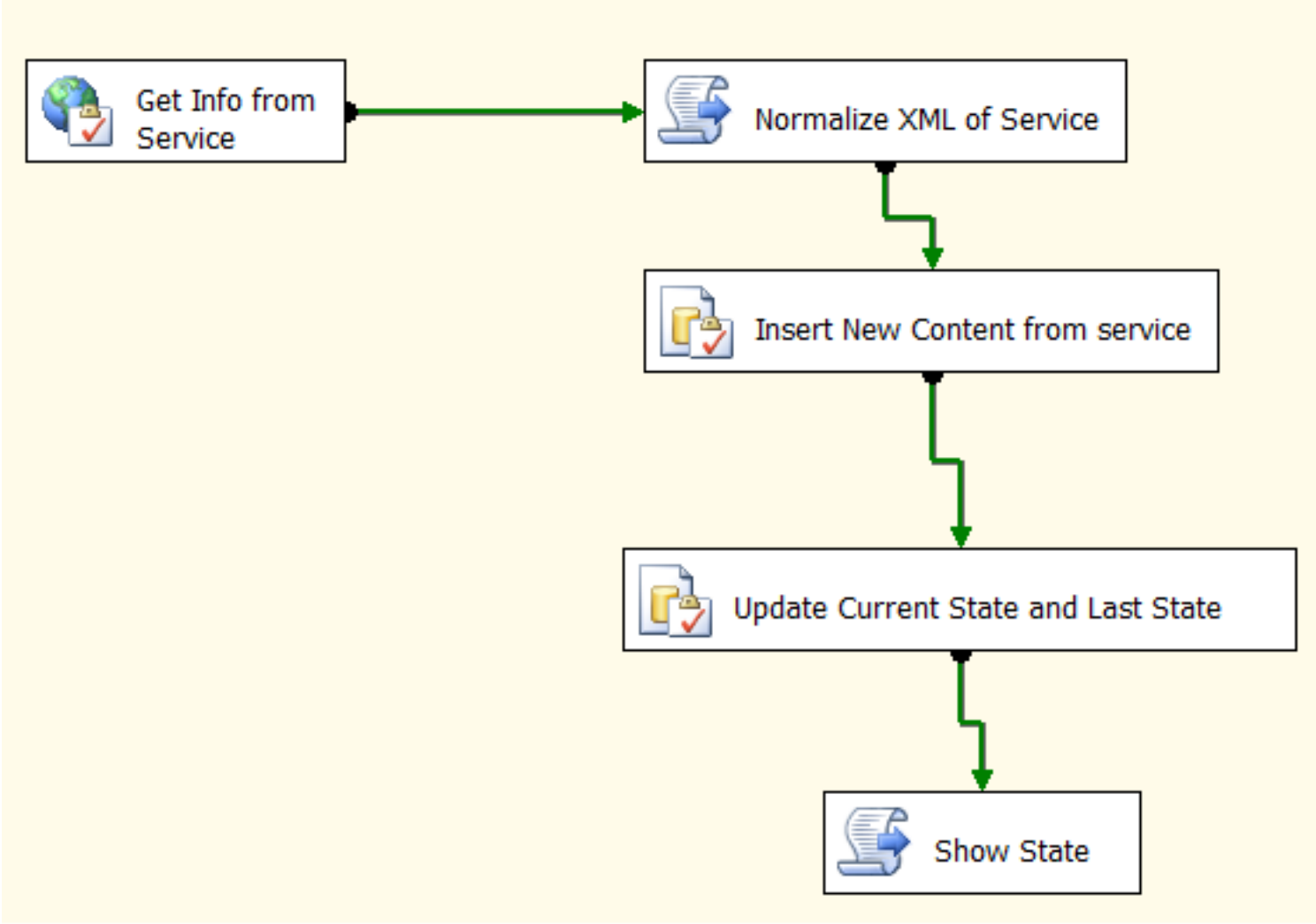


و ذخیره نتیجه تراکنش در متغیری در پکیج :

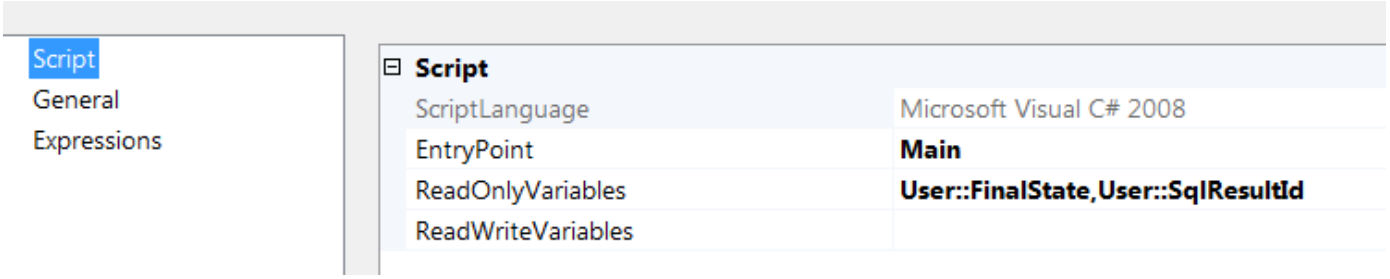




و اکنون پکیج ما ظاهری شبیه به این مورد خواهد داشت :

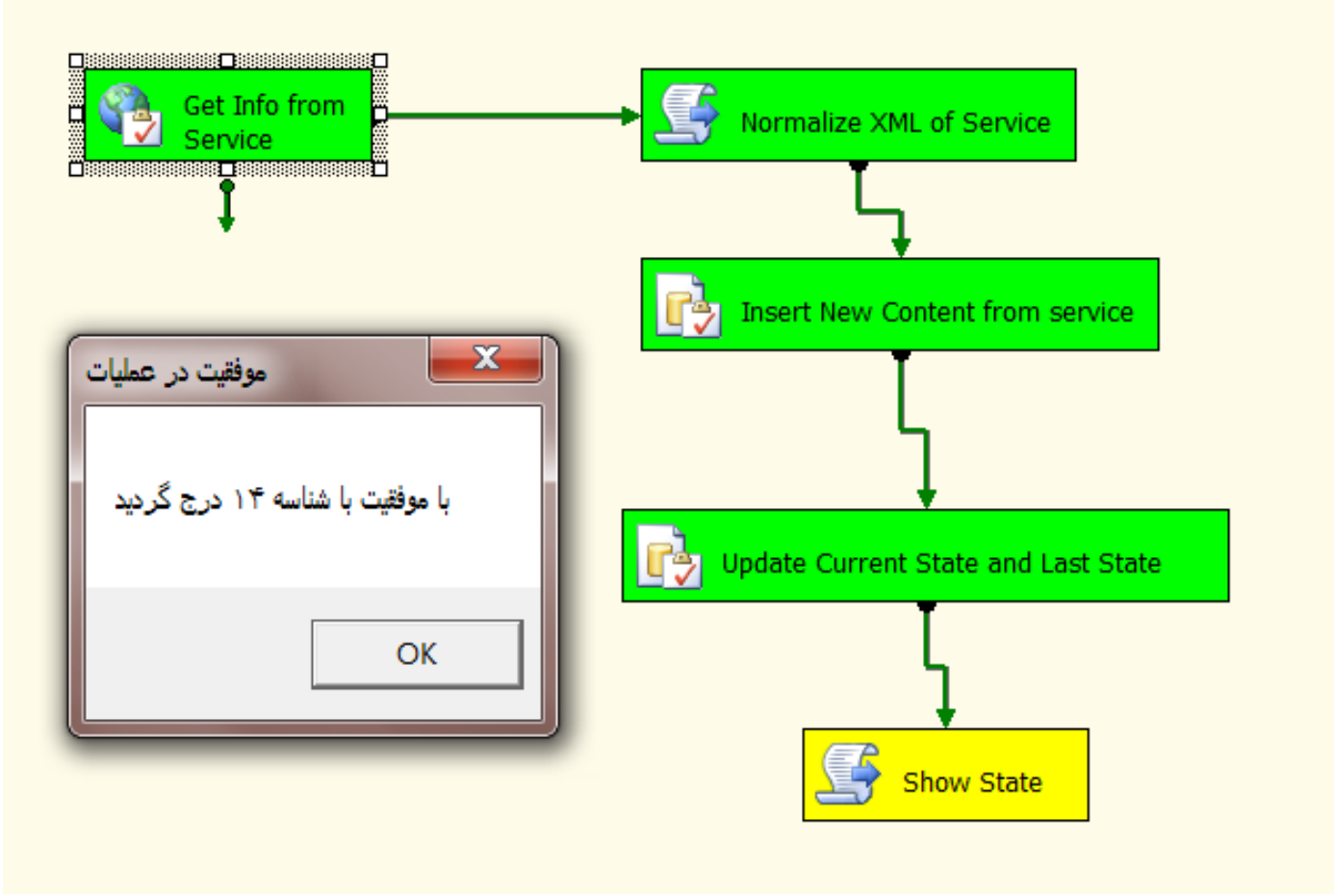


در نهایت به عنوان یک facility می‌توانیم وضعیت تراکنش را به کاربر نمایش دهیم (به کمک Script Task) :



```
public void Main()
{

    const string _id = "User::SqlResultId";
    const string _err = "User::FinalState";

    Dts.VariableDispenser.LockForRead(_id);
    Dts.VariableDispenser.LockForRead(_err);


    string id = Convert.ToString( Dts.Variables[_id].Value);
    string error = Convert.ToString(Dts.Variables[_err].Value);

    string msg = "با موفقیت با شناسه "+id+" درج گردید";
    Dts.VariableDispenser.Reset();

    if (error.Equals("OK"))
    {
        MessageBox.Show(msg,"موفقیت در عملیات");
        Dts.TaskResult = (int)ScriptResults.Success;
    }
    else
    {
        MessageBox.Show("Error : "+Environment.NewLine+error, "خطا در انجام عملیات");
        Dts.TaskResult = (int)ScriptResults.Failure;
    }


    // TODO: Add your code here
}
```

و تمام ...



Results | Messages

	ID	XmlContent	DateOfInsertion
1	7	<ArrayOfUserInfo><UserInfo><ID>bd36d158-f2bd-40d...	2013-05-16 21:36:44.84
2	8	<ArrayOfUserInfo><UserInfo><ID>5848557b-6482-4d8...	2013-05-16 21:44:21.49
3	9	<ArrayOfUserInfo><UserInfo><ID>5f465898-bce8-4153...	2013-05-16 21:45:02.61
4	10	<ArrayOfUserInfo><UserInfo><ID>0836a55b-3d8f-4ce...	2013-05-16 21:53:19.83
5	11	<ArrayOfUserInfo><UserInfo><ID>73670cd5-1d09-4a9...	2013-05-16 22:35:52.70
6	12	<ArrayOfUserInfo><UserInfo><ID>316e06a0-4063-479...	2013-05-16 22:36:30.89
7	13	<ArrayOfUserInfo><UserInfo><ID>c56ab672-9415-418...	2013-05-16 22:38:56.86
8	14	<ArrayOfUserInfo><UserInfo><ID>c991b4ee-a389-427...	2013-05-17 10:55:12.27

موفق باشید

نظرات خوانندگان

نویسنده: Amir
تاریخ: ۱۳۹۲/۱۰/۱۸ ۰:۴۴

با سلام وسپاس از مطالب مفیدتون

سوالی دارم ممنون میشم اگه جواب بدید

برای انتقال داده‌های هشت سرور که از نوع پارادوکس به SQL Server 2012 هستند آیا به غیر از روش SSIS راه دیگری وجود دارد ؟ ناگفته نماند که داده‌ها حجم قابل توجهی دارند و با روش SSIS سرعت شبکه را کاهش می‌دهد.

نکته دیگر، داده‌ها به صورت کامل به SQL Server منتقل نمیشوند و بعضی جداول منتقل شده یا خالی هستند یا جداول تکراری داریم در SQL Server که در نتیجه باعث قابل اطمینان نبودن Data Warehouse میشود.

با سپاس

نویسنده: محمد باقر سیف اللهی
تاریخ: ۱۳۹۲/۱۰/۱۸ ۱۲:۲۹

سلام
اجازه بدید اول به این نکته اشاره کنم که SSIS روش نیست بلکه یک ابزاره. روش مد نظر برای Migration داده ، ETL نام داره (Extract , Transform , Load) نه SSIS.
حالا برای انجام عمل ETL ابزارهای دیگری هم میشه پیدا کرد مثل informatica یا DB2 Infosphere که می‌تونید تعدادی از اون‌های رو اینجا ببینید
در مورد مشکلاتی که در باره داده‌ها یا شبکه می‌فرمایید هم باید یک plan مناسب برای این مهاجرت داده ای ایجاد کنید . جهت اطلاع عرض کنم که این پروسه ممکنه گاها تا یک ماه طول بکشه بسته به پلنی که طراحی شده و شرایط و محدودیت هایی که موجوده. اطلاعات بیشتر رو می‌تونید از اینجا کسب کنید
موفق باشید

نویسنده: Amiir
تاریخ: ۱۳۹۲/۱۰/۱۸ ۱۳:۴۵

ممنون از راهنمایتون

عنوان: **اجرای SSIS Package از طریق برنامه کاربردی**
نویسنده: محمد رجبی
تاریخ: ۱۳۹۳/۰۲/۰۷ ۱۷:۵
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, SSIS

مقدمه

در اکثر موارد در یک Landscape عملیاتی، چنانچه به تجمیع و انتقال داده‌ها از بانک‌های اطلاعاتی مختلف نیاز باشد، از SSIS Package اختصار (SQL Server Integration Service) استفاده می‌شود و معمولاً با تعریف یک Job در سطح SQL Server به اجرای Package در زمانهای مشخص می‌پردازند. چنانچه در موقعیتی لازم باشد که از طریق برنامه کاربردی توسعه یافته، به اجرای Package مبادرت ورزیده شود و البته نخواهیم Job تعریف شده را از طریق کد برنامه، اجرا کنیم و در واقع این امکان را داشته باشیم که همانند یک رویه ذخیره شده تعریف شده در سطح بانک اطلاعاتی به اجرای عمل فوق بپردازیم، یک راه حل می‌تواند تعریف یک CLR Stored Procedures باشد. در این مقاله به بررسی این موضوع پرداخته می‌شود، در ابتدا لازم است به بیان تئوری موضوع پرداخته شود (قسمت‌های 1 الی 5) در ادامه به ذکر پیاده سازی روش پیشنهادی پرداخته می‌شود.

1- اجرای Integration Service Package جهت اجرای یک Package از ابزارهای زیر می‌توان استفاده کرد:

- ابزار خط فرمان command-line dtexec.exe
- ابزار اجرائی پکیج dtexecui.exe
- استفاده از SQL Server Agent job

توجه: همچنین یک Package را در زمان طراحی در BIDS (Business Intelligence Development Studio) می‌توان اجرا نمود.

2- استفاده از dtexec جهت اجرای Package با استفاده از ابزار dtexec می‌توان Packageهای ذخیره شده در فایل سیستم، یک SQL Instance و یا Packageهای ذخیره شده در Integration Service را اجرا نمود.

توجه: در سیستم عامل‌های 64 بیتی، ابزار dtexec موجود در Integration Service با نسخه 64 بیتی نصب می‌شود. چنانچه بایست Packageهای معینی را در حالت 32 بیتی اجرا کنید، لازم است ابزار dtexec نسخه 32 بیتی نصب شود. ابزار dtexec دستیابی به تمامی ویژگی‌های پیکربندی و اجرای Package از قبیل اتصالات، مشخصات(Properties)، متغیرها، logging و شاخص‌های پردازشی را فراهم می‌کند.
توجه: زمانی که از نسخه‌ی ابزار dtexec که با SQL Server 2008 ارائه شده استفاده می‌کنید برای اجرای یک SSIS Package نسخه 2005، Integration Service به صورت موقت Package را به نسخه 2008 ارتقا می‌دهد، اما نمی‌توان از ابزار dtexec برای ذخیره این تغییرات استفاده کرد.

2-1- ملاحظات نصب dtexec روی سیستم‌های 64 بیتی به صورت پیش فرض، یک سیستم عامل 64 بیتی که هر دو نسخه 64 بیتی و 32 بیتی ابزار خط فرمان Integration Service را دارد، نسخه 32 بیتی نصب شده را در خط فرمان اجرا خواهد کرد. نسخه 32 بیتی بدین دلیل اجرا می‌شود که در متغیر محیطی (Path environment variable) Path مسیر directory نسخه 32 بیتی قرار گرفته است.به طور معمول:

```
(<drive>:\Program Files(x86)\Microsoft SQL Server\100\DTS\Binn)
```

توجه: اگر از SQL Server Agent برای اجرای Package استفاده می‌کنید، SQL Server Agent به طور خودکار از ابزار نسخه 64 بیتی استفاده می‌کند. SQL Server Agent از Registry و نه از متغیر محیطی Path استفاده می‌کند. برای اطمینان از اینکه نسخه 64 بیتی این ابزار را در خط فرمان اجرا می‌کنید، directory را به directory ای تغییر دهید که شامل نسخه 64 بیتی این ابزار است(<drive>:\Program Files\Microsoft SQL Server\100\DTS\Binn) و ابزار را از این مسیر اجرا کنید و یا برای همیشه مسیر قرار گرفته در متغیر محیطی path را با مسیری که نسخه 64 بیتی قرار دارد، جایگزین کنید.

2-2- تفسیر کدهای خروجی هنگامی که یک Package اجرا می‌شود، dtexec یک کد خروجی (Return Code) بر می‌گرداند:

مقدار	توصیف
0	Package با موفقیت اجرا شده است.
1	Package با خطا مواجه شده است.
3	Package در حال اجرا توسط کاربر لغو شده است.
4	Package پیدا نشده است.
5	Package بارگذاری نشده است.
6	ابزار با یک خطای نحوی یا خطای معنایی در خط فرمان برخورد کرده است.

2-3- قوانین نحوی dtexec تمامی گزینه‌ها (Options) باید با یک علامت Slash (/) و یا Minus (-) شروع شوند.
یک آرگومان باید در یک quotation mark محصور شود چنانچه شامل یک فاصله خالی باشد.
گزینه‌ها و آرگومان‌ها بجز رمزعبور حساس به حروف کوچک و بزرگ نیستند.

Syntax 2-3-1-

```
dtexec /option [value] [/option [value]]…
```

Parameters 2-3-2- نکته: در Integration Service، ابزار خط فرمان dtsrun که برایDTS (Data Transformation Service)های نسخه SQL Server 2000 استفاده می‌شد، با ابزار خط فرمان dtexec جایگزین شده است.

- تعدادی از گزینه‌های خط فرمان dtsrun به طور مستقیم در dtexec معادل دارند برای مثال نام Server و نام Package.
- تعدادی از گزینه‌های dtsrun به طور مستقیم در dtexec معادل ندارند.
- تعدادی گزینه‌های خط فرمان جدید dtsexec وجود دارد که در ویژگی‌های جدید Integration Service پشتیبانی می‌شود.

2-3-3- مثال 1) به منظور اجرای یک SSIS Package که در SQL Server ذخیره شده است، با استفاده از Windows Authentication :

```
dtexec /sq <Package Name> /ser <Server Name>
```

2) به منظور اجرای یک SSIS Package که در پوشه File System در SSIS Package Store ذخیره شده است :

```
dtexec /dts “\File System\<Package File Name>”
```

3) به منظور اجرای یک SSIS Package که در سیستم فایل ذخیره شده است و مشخص کردن گزینه logging:

```
dtexec /f “c:\<Package File Name>” /l “DTS.LogProviderTextFile; <Log File Name>”
```

4) به منظور اجرای یک SSIS Package که در SQL Server ذخیره شده با استفاده از SQL Server Authentication برای نمونه(user:ssis;pwd:ssis@ssis)و رمز (123)Package:

```
dtexec  /server “<Server Name>”  /sql “<Package Name>”  / user “ssis” /Password “ssis@ssis” /De “123”
```

3- تنظیمات سطح حفاظتی یک Package به منظور حفاظت از داده‌ها در Packageهای Integration Service می‌توانید یک سطح

حفاظتی (protection level) را تنظیم کنید که به حفاظت از داده‌های صرفاً حساس یا تمامی داده‌های یک Package کمک نماید. به علاوه می‌توانید این داده‌ها را با یک Password یا یک User Key رمزگذاری نمائید یا به رمزگذاری داده‌ها در بانک اطلاعاتی اعتماد کنید. همچنین سطح حفاظتی که برای یک Package استفاده می‌کنید، الزاماً ایستا (static) نیست و در طول چرخه حیات یک Package می‌تواند تغییر کند. اغلب سطح حفاظتی در طول توسعه یا به محض (deploy) استقرار Package تنظیم می‌شود.
توجه: علاوه بر سطوح حفاظتی که توصیف شد، Packageها در بانک اطلاعاتی msdb ذخیره می‌شوند که همچنین می‌توانند توسط نقش‌های ثابت در سطح بانک اطلاعاتی (fixed database-level roles) حفاظت شوند. Integration Service شامل 3 نقش ثابت بانک اطلاعاتی برای نسبت دادن مجوزها به Package است که عبارتند از db_ssisltduser، db_ssisadmin و db_ssisoperator

3-1- درک سطوح حفاظتی در یک Package اطلاعات زیر به عنوان حساس تعریف می‌شوند:
• بخش password در یک connection string. گرچه، اگر گزینه ای را که همه چیز را رمزگذاری کند، انتخاب کنید تمامی connection string حساس در نظر گرفته می‌شود.
• گره‌های task-generated XML که برچسب (tagged) هایی حساس هستند.
• هر متغییری که به عنوان حساس نشان گذاری شود.

3-1-1- Do not save sensitive هنگامی که Package ذخیره می‌شود از ذخیره مقادیر ویژگی‌های حساس در Package جلوگیری می‌کند. این سطح حفاظتی رمزگذاری نمی‌کند اما در عوض از ذخیره شدن ویژگی هایی که حساس نشان گذاری شده اند به همراه Package جلوگیری می‌کند.

3-1-2- Encrypt all with password به منظور رمزگذاری تمامی Package از یک Password استفاده می‌شود. Package توسط Password ای رمزگذاری می‌شود که کاربر هنگامی که Package را ایجاد یا Export می‌کند، ارائه می‌دهد. به منظور باز کردن Package در SSIS Designer یا اجرای Package توسط ابزار خط فرمان dtexec کاربر بایست رمز Package را ارائه نماید. بدون رمز کاربر قادر به دستیابی و اجرای Package نیست.

3-1-3- Encrypt all with user key به منظور رمزگذاری تمامی Package از یک کلید که مبتنی بر Profile کاربر جاری می‌باشد، استفاده می‌شود. تنها کاربری که Package را ایجاد یا Export می‌کند، می‌تواند Package را در SSIS Designer باز کند و یا Package را توسط ابزار خط فرمان dtexec اجرا کند.

3-1-4- Encrypt sensitive with password به منظور رمزگذاری تنها مقادیر ویژگی‌های حساس در Package از یک Password استفاده می‌شود. برای رمزگذاری از DPAPI استفاده می‌شود. داده‌های حساس به عنوان بخشی از Package ذخیره می‌شوند اما آن داده‌ها با استفاده از Password رمزگذاری می‌شوند. به منظور باز نمودن Package در SSIS Designer کاربر باید رمز Package را ارائه دهد. اگر رمز ارائه نشود، Package بدون داده‌های حساس باز می‌شود و کاربر باید مقادیر جدیدی برای داده‌های حساس فراهم کند. اگر کاربر سعی نماید Package را بدون ارائه رمز اجرا کند، اجرای Package با خطا مواجه می‌شود.

3-1-5- Encrypt sensitive with user key به منظور رمزگذاری تنها مقادیر ویژگی‌های حساس در Package از یک کلید که مبتنی بر Profile کاربر جاری می‌باشد، استفاده می‌شود. تنها کاربری که از همان Profile استفاده می‌کند، Package را می‌تواند بارگذاری (load) کند. اگر کاربر متفاوتی Package را باز نماید، اطلاعات حساس با مقادیر پوچی جایگزین می‌شود و کاربر باید مقادیر جدیدی برای داده‌های حساس فراهم کند. اگر کاربر سعی نماید Package را بدون ارائه رمز اجرا کند، اجرای Package با خطا مواجه می‌شود. برای رمزگذاری از DPAPI استفاده می‌شود.

3-1-6- Rely on server storage for encryption (ServerStorage) با استفاده از نقش‌های بانک اطلاعاتی، SQL Server تمامی Package را حفاظت می‌کند. این گزینه تنها زمانی پشتیبانی می‌شود که Package در بانک اطلاعاتی msdb ذخیره شده است.

4- استفاده از نقش‌های Integration Service برای کنترل کردن دسترسی به SSIS، Package شامل 3 نقش ثابت در سطح بانک اطلاعاتی است. نقش‌ها می‌توانند تنها روی Package هایی که در بانک اطلاعاتی msdb ذخیره شده اند، بکار روند. با استفاده از SSMS می‌توانید نقش‌ها را به Packageها نسبت دهید، این انتساب نقش‌ها در بانک اطلاعاتی msdb ذخیره می‌شود.

Role	Read action	Write action
db_ssisadmin **or** **sysadmin**	Enumerate own packages Enumerate all packages View own packages View all packages Execute own packages Execute all packages Export own packages Export all packages Execute all packages in SQL Server Agent	Import packages Delete own packages Delete all packages Change own package roles Change all package roles * به نکته رجوع شود
db_ssisltduser	Enumerate own packages Enumerate all packages View own packages Execute own packages Export own packages	Import packages Delete own packages Change own package roles
db_ssisoperator	Enumerate all packages View all packages Execute all packages Export all packages Execute all packages in SQL Server Agent	None
Windows administrators	View execution details of all running packages	Stop all currently running packages

*** نکته:** اعضای نقش‌های db_ssisadmin و dc_admin ممکن است قادر باشند مجوزهای خودشان را تا سطح sysadmin ارتقا دهند براساس این ترفیع مجوز امکان اصلاح و اجرای Packageها از طریق SQL Server Agent میسر می‌شود. برای محافظت در برابر این ارتقا، با استفاده از یک (account) حساب Proxy با دسترسی محدود، Job هایی که این Packageها را اجرا می‌کنند، پیکربندی شوند یا تنها اعضای نقش sysadmin به نقش‌های db_ssisadmin و dc_admin افزوده شوند.

همچنین جدول sysssispackages در بانک اطلاعاتی msdb شامل Package هایی است که در SQL Server ذخیره می‌شوند. این جدول شامل ستون هایی که اطلاعاتی درباره نقش هایی که به Packageها نسبت داده شده است، می‌باشد.
به صورت پیش فرض، مجوزهای نقش‌های ثابت بانک اطلاعاتی db_ssisadmin و db_ssisoperator و شناسه منحصر به فرد کاربری (unique security identifier) که Package را ایجاد کرده برای خواندن Package بکار می‌رود، و مجوزهای نقش db_ssisadmin و شناسه منحصر به فرد کاربری که Package را ایجاد کرده برای نوشتن Package به کار می‌رود. یک User باید عضو نقش db_ssisadmin و db_ssisltduser یا db_ssisoperator برای داشتن دسترسی خواندن Package باشد. یک User باید عضو نقش db_ssisadmin برای داشتن دسترسی نوشتن Package باشد.

5- اتصال به صورت Remote به Integration Service زمانی که یک کاربر بدون داشتن دسترسی کافی تلاش کند به یک Integration Service به صورت Remote متصل شود، با پیغام خطای "Access is denied" مواجه می‌شود. برای اجتناب از این پیغام خطا می‌توان تضمین کرد که کاربر مجوز مورد نیاز DCOM را دارد.
به منظور پیکربندی کردن دسترسی کاربر به صورت Remote به سرویس Integration مراحل زیر را دنبال کنید:

- Component Service را باز نمایید (در Run عبارت dcomcnfg را تایپ کنید).

- گره Component Service را باز کنید، گره Computer و سپس My Computer را باز نمایید و روی DCOM Config کلیک نمایید.
- گره DCOM Config را باز کنید و از لیست برنامه هایی که می‌توانند پیکربندی شوند MsDtsServer را انتخاب کنید.
- روی Properties برنامه MsDtsServer رفته و قسمت Security را انتخاب کنید.
- در قسمت Lunch and Activation Permissions، مورد Customize را انتخاب و سپس روی Edit کلیک نمایید تا پنجره Lunch Permission باز شود.
- در پنجره Lunch Permission، کاربران را اضافه و یا حذف کنید و مجوزهای مناسب را به کاربران یا گروه‌های مناسب نسبت دهید. مجوزهای موجود عبارتند از Local Lunch، Remote Lunch، Local Activation و Remote Activation .
- در قسمت Access Permission مراحل فوق را به منظور نسبت دادن مجوزهای مناسب به کاربران یا گروه‌های مناسب انجام دهید.
- سرویس Integration را Restart کنید.

مجوز دسترسی Lunch به منظور شروع و خاتمه سرویس، اعطا یا رد می‌شود و مجوز دسترسیActivation به منظور متصل شدن به سرویس، اعطا (grant) یا رد (deny) می‌شود.

6- پیاده سازی در ابتدا به ایجاد یک CLR Stored Procedures پرداخته می‌شود نام اسمبلی ساخته شده به این نام RunningPackage.dll می‌باشد و حاوی کد زیر است:

```
Partial Public Class StoredProcedures
    '-----------------------------------------------
    'exec dbo.Spc_NtDtexec 'Package','ssis','ssis@ssis','1234512345'
    '-----------------------------------------------
    <Microsoft.SqlServer.Server.SqlProcedure()> _
    Public Shared Sub Spc_NtDtexec(ByVal PackageName As String, _
                                   ByVal UserName As String, _
                                   ByVal Password As String, _
                                   ByVal Decrypt As String)
        Dim p As New System.Diagnostics.Process()
        p.StartInfo.FileName = "C:\Program Files\Microsoft SQL Server\100\DTS\Binn\DTExec.exe"
        p.StartInfo.RedirectStandardOutput = True
        p.StartInfo.Arguments = "/sql " & PackageName & " /User " & UserName & " /Password " & Password
& " /De " & Decrypt
        p.StartInfo.UseShellExecute = False
        p.Start()
        p.WaitForExit()
        Dim output As String
        output = p.StandardOutput.ReadToEnd()
        Microsoft.SqlServer.Server.SqlContext.Pipe.Send(output)
    End Sub
End Class
```

در حقیقت توسط این رویه به اجرای برنامه dtexec.exe و ارسال پارامترهای مورد نیاز جهت اجرا پرداخته می‌شود. با توجه به توضیحات تئوری بیان شده، سطح حفاظتی Package ایجاد شده Encrypt all with password توصیه می‌شود که رمز مذکور در قالب یکی از پارامتر ارسالی به رویه ساخته شده موسوم به Spc_NtDtexec ارسال می‌گردد.

در قدم بعدی نیاز به Register کردن dll ساخته شده در سطح بانک اطلاعاتی SQL Server است، این گام‌ها پس از اتصال به SQL Server Management Studio به شرح زیر است:

1- فعال کردن CLR در سرویس SQL Server

```
SP_CONFIGURE 'clr enabled',1
GO
RECONFIGURE
```

2- فعال کردن ویژگی TRUSTWORTHY در بانک اطلاعاتی مورد نظر

```
ALTER DATABASE <Database Name> SET TRUSTWORTHY ON
GO
RECONFIGURE
```

3- ایجاد Assembly و Stored Procedure در بانک اطلاعاتی مورد نظر

Assembly ساخته شده با نام RunningPacakge.dll در ریشه :C کپی شود. بعد از ثبت نمودن این Assembly لزومی به وجود آن نمی‌باشد.

```
USE <Database Name>
GO
CREATE ASSEMBLY [RunningPackage]
AUTHORIZATION [dbo]
FROM 'C:\RunningPackage.dll'
WITH PERMISSION_SET = UNSAFE
Go
CREATE PROCEDURE [dbo].[Spc_NtDtexec]
@PackageName [nvarchar](50),
@UserName [nvarchar](50),
@Password [nvarchar](50),
@Decrypt [nvarchar](50)
WITH EXECUTE AS CALLER
AS
EXTERNAL NAME [RunningPackage].[RunningPackage.StoredProcedures].[Spc_NtDtexec]
GO
```

توجه: Application User برنامه بایست دسترسی اجرای رویه ذخیره شده Spc_NtDtexec را در بانک اطلاعاتی مورد نظر داشته باشد همچنین بایست عضو نقش db_ssisoperator در بانک اطلاعاتی msdb باشد.(منظور از Application User، لاگین است که در Connection string برنامه قرار داده اید.)

در برنامه کاربردی تان کافی است متدی به شکل زیر ایجاد و با توجه به نیازتان در برنامه به فراخوانی آن و اجرای Package بپردازید.

```
Private Sub ExecutePackage()
        Dim oSqlConnection As SqlClient.SqlConnection
        Dim oSqlCommand As SqlClient.SqlCommand
        Dim strCnt As String = String.Empty
        strCnt = "Data Source=" & txtServer.Text & ";User ID=" & txtUsername.Text & ";Password=" &
txtPassword.Text & ";Initial Catalog=" & cmbDatabaseName.SelectedValue.ToString() & ";"
        Try
            oSqlConnection = New SqlClient.SqlConnection(strCnt)
            oSqlCommand = New SqlClient.SqlCommand
            With oSqlCommand
                .Connection = oSqlConnection
                .CommandType = System.Data.CommandType.StoredProcedure
                .CommandText = "dbo.Spc_NtDtexec"
                .Parameters.Clear()
                .Parameters.Add("@PackageName", System.Data.SqlDbType.VarChar, 50)
                .Parameters.Add("@UserName", System.Data.SqlDbType.VarChar, 50)
                .Parameters.Add("@Password", System.Data.SqlDbType.VarChar, 50)
                .Parameters.Add("@Decrypt", System.Data.SqlDbType.VarChar, 50)
                .Parameters("@PackageName").Value = txtPackageName.Text.Trim()
                .Parameters("@UserName").Value = txtUsername.Text.Trim()
                .Parameters("@Password").Value = txtPassword.Text.Trim()
                .Parameters("@Decrypt").Value = txtDecrypt.Text.Trim()
            End With
            If (oSqlCommand.Connection.State <> System.Data.ConnectionState.Open) Then
                oSqlCommand.Connection.Open()
                oSqlCommand.ExecuteNonQuery()
                System.Windows.Forms.MessageBox.Show("Success")
            End If
            If (oSqlCommand.Connection.State = System.Data.ConnectionState.Open) Then
                oSqlCommand.Connection.Close()
            End If
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error")
        End Try
    End Sub 'ExecutePackage
```

عنوان: **بررسی بارگذاری داده ها در انبار های داده و معرفی الگوهای بکار رفته در آن**
نویسنده: محمد رجبی
تاریخ: ۱۳۹۳/۰۴/۱۸ ۱۷:۳۵
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, SSIS, data warehouse, ETL

مقدمه در لینکی که چندی پیش به اشتراک گذاشته بودم؛ به مطلبی تحت این عنوان اشاره شده بود: "آیا از KPI باید به انباره داده و هوش تجاری رسید؟" (بر گرفته از وبلاگ آقای جام سحر) که در آن به موانع پیش روی انجام پروژه‌های BI در ایران پرداخته شده است.

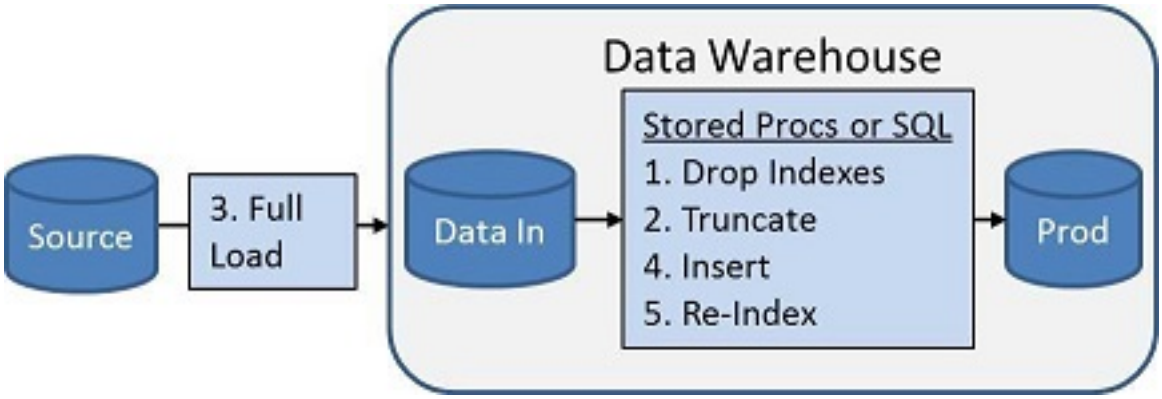
این مقاله بر گرفته از فصل سوم یکی از White Paperهای ماکروسافت با عنوان Microsoft EDW Architecture, Guidance and Deployment Best Practices می‌باشد. که به شرح عملیات Loading در فاز ETL می‌پردازد. از آنجا که به منظور پیاده سازی این نوع پروژه‌ها معمولاً در ایران برون سپاری صورت می‌گیرد و مدیران شرکت‌ها بیشتر درگیر سیستم‌های OLTP هستند و مجری پروژه (شرکت پیمانکار) معمولاً کوتاهترین مسیر را جهت انجام پروژه انتخاب می‌کند(و امروزه نیک میدانیم که "انتخاب مسیرهای کوتاه در زمان کم می‌تواند به پیچیدگی‌های بسیار جدی در دراز مدت منجر شود!") و همچنین از آنجا که متاسفانه به دلیل عدم ثبات مدیریت در ایران معمولاً "مدیریت برای تحویل پروژه تحت فشار است و نه برای مسائل پشتیبانی " و مسائل دیگری از این دست؛ چنانچه در تحویل گیری محصول به درستی تست نرم افزار صورت نگیرد، در نظر گرفتن موارد زیر:

```
Verification: Are we building the product right? ~ Software correctly implements a specific function
```

```
Validation: Are we building the right product? ~   Software is traceable to customer requirements
```

پروژه با شکست مواجه می‌شود و انتظارات مدیران بهره بردار را برآورده نمی‌کند. به هر روی در این مقاله به ترجمه مطالب زیر پرداخته می‌شود، توصیه میکنم در صورتی که با خواندن متن انگلیسی مشکلی ندارید، اصل مقاله مذکور خوانده شود.

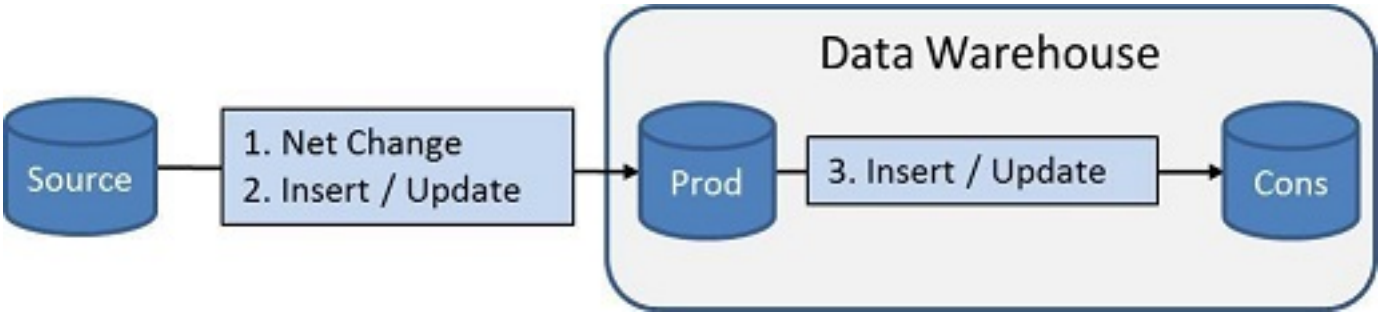
1- Full Load vs Incremental Load نسل‌های اولیه DW (اختصار Data Warehouse) به شکل Full Loads پیاده سازی می‌شدند، به این طریق که هر بار عملیات بارگذاری صورت می‌گرفت، DW از نو دوباره ساخته می‌شد. شکل زیر مراحل مختلف انجام شده در این روش را نمایش می‌دهد:



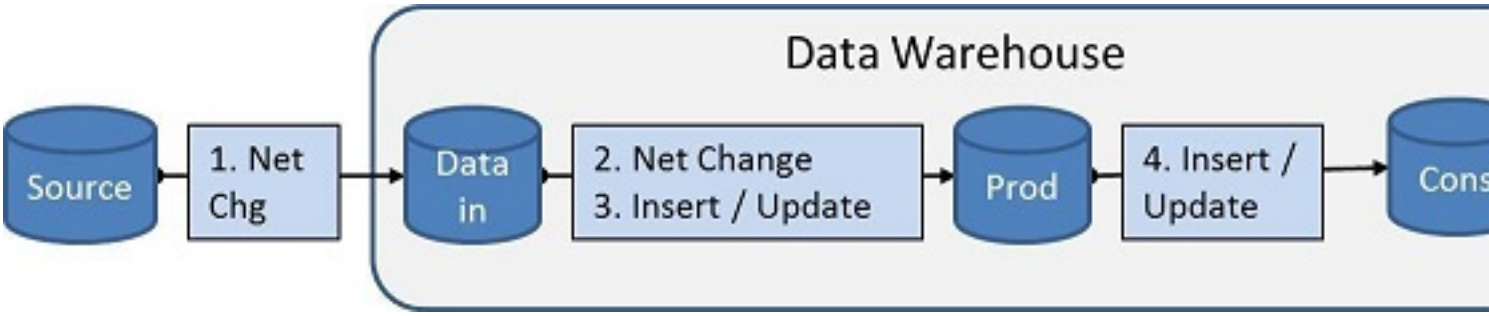
پروسه Full Load شامل مراحل زیر بود:

Drop Indexes: از آنجا که Indexها زمان بارگذاری را افزایش می‌دادند، این عمل صورت می‌پذیرفت.
Truncate Tables: تمامی رکوردهای موجود در جداول حذف می‌شدند.
Bulk Copy
Load Data
Post Process: شامل عملیاتی نظیر شاخص گذاری روی داده هایی است که اخیراً بارگذاری شده اند و....

روی هم رفته Full Load مسئله ای مشکل ساز بود، زیرا نیاز به زمانی برای بارگذاری مجدد داده‌ها داشت و مسئله‌ی مهم‌تر نداشتن امکان دستیابی به گزارشاتی تاریخچه ای با ماهیت زمان برای مشتریان کسب وکار بود. به این دلیل که همواره یک کپی از آخرین داده‌های موجود در سیستم عملیاتی درون DW قرار می‌گرفت؛ که با بکارگیری Full Load اغلب قادر به ارائه‌ی این نوع از گزارشات نبودیم، بدین ترتیب سازمان‌ها به نسل دوم روی آورند که در این دیدگاه از مفهوم Incremental Load استفاده می‌شود. اشکال زیر مراحلی که در این روش انجام می‌شود را نمایان می‌سازد:



Incremental Load with an Extract In area



Incremental Load without an Extract In area

مراحل Incremental Load شامل:

بارگذاری تغییرات نسبت به آخرین فرآیند بارگذاری انجام شده
درج / بروزرسانی تغییرات درون Production area
درج / بروزرسانی Consumption area نسبت به Production area

تفاوت‌های اصلی میان Full Load و Incremental Load در این است که در Incremental Load:

نیازی به پردازش‌های اضافی جهت حذف شاخص ها، پاک کردن تمامی رکوردهای جداول و ساخت مجدد شاخص‌ها نیست.
البته نیاز به رویه ای جهت شناسایی تغییرات می‌باشد.
و همچنین نیاز به بروزرسانی بعلاوه درج رکوردهای جدید نیز می‌باشد.

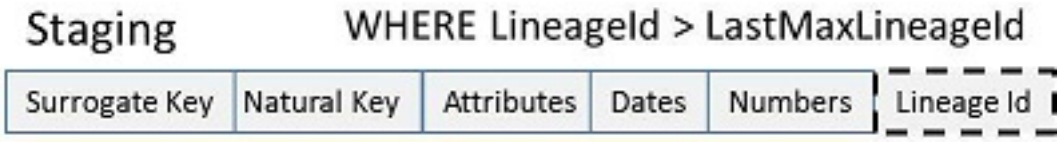
ترکیب این عوامل برای ساخت Incremental Load کارآمد تر، منجر به پیچیده‌تر شدن پیاده سازی و نگهداری آن نیز می‌شود.

2- Detecting Net Changes
فرآیند لود افزایشی ETL، بایست قادر به شناسائی رکوردهای تغییریافته در مبداء باشد، که این عمل با استفاده از هر یک از تکنیک‌های Push یا Pull انجام می‌شود.

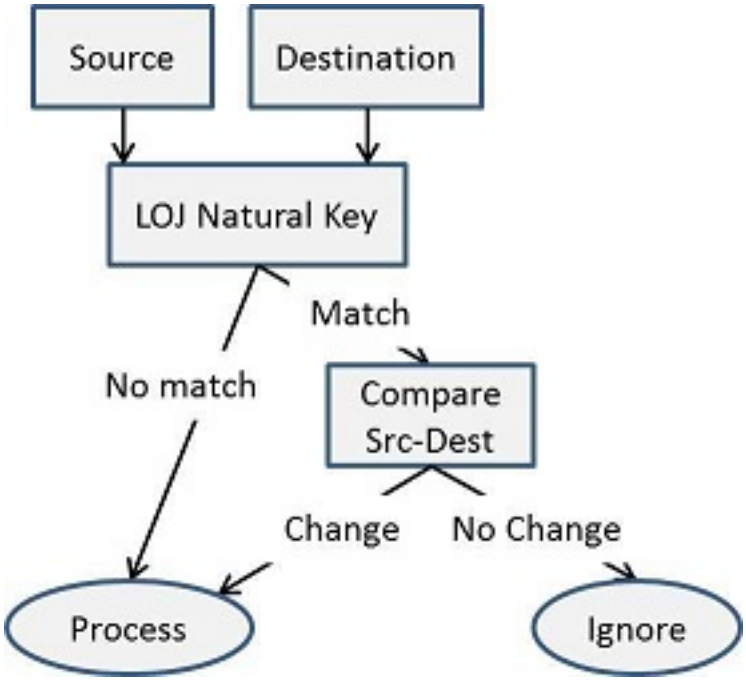
در تکنیک Pull، فرآیند ETL رکوردهای تغییریافته در مبداء را انتخاب می‌کند:
ایده‌آل وجود داشتن یک ستون Last Changed در سیستم مبداء است؛ که از آن می‌توان جهت انتخاب رکوردهای تغییر یافته استفاده نمود.
چنانچه ستون Last Changed وجود نداشته باشد، تمامی رکوردهای مبداء باید با رکوردهای مقصد مقایسه شود.
در تکنیک Push، مبداء تغییرات را شناسائی می‌کند و آنها را به سمت مقصد Push می‌کند؛ این درخواست می‌تواند توسط فرآیند ETL انجام شود.
از آنجایی که پردازش ETL معمولاً در زمان هایی که Peak کاری وجود ندارد، اجرا می‌شود، استفاده از مکانیسم Pull برای شناسایی تغییرات نسبت به مکانسیم Push ارجحیت دارد.

2-1- Pulling Net Changes – Last Change Column بیشتر جداول در سیستم‌های مبداء حاوی ستون هایی هستند که زمان ایجاد و یا اصلاح رکوردها را ثبت می‌کنند. در نوع دیگری از سیستم‌های مبداء ستونی با مقدار عددی وجود دارد، که هر زمان رکوردی تغییر یافت به آن ستون مقداری اضافه می‌شود. هر دوی این تکنیک‌ها به فرآیند ETL اجازه می‌دهند، بطور کارآمدی رکوردهای تغییریافته را انتخاب کند. (با مقایسه، بیشترین مقدار قرار گرفته در آن ستون؛ که در طول آخرین اجرای فرآیند ETL بدست آمده است).
نمونه ای از جداول سیستم مبداء که دارای تغییرات زمانی است در شکل زیر نمایش داده می‌شود.

همچنین شکل زیر نشان می‌دهد، چگونه یک مقدار عددی می‌تواند به منظور انتخاب رکوردهای تغییریافته استفاده شود.



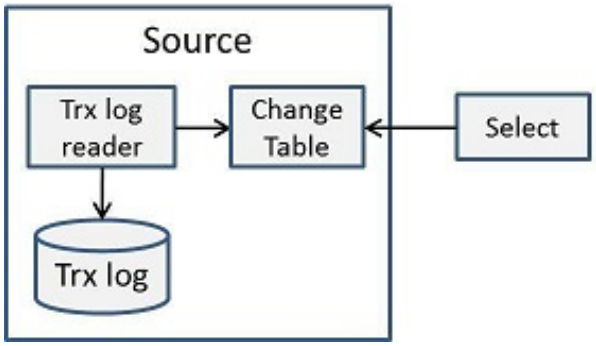
2-2- Pulling Net Changes – No Last Change Column شکل زیر گردش فرآیند را هنگامی که ستون Last Change وجود ندارد؛ نمایش می‌دهد.

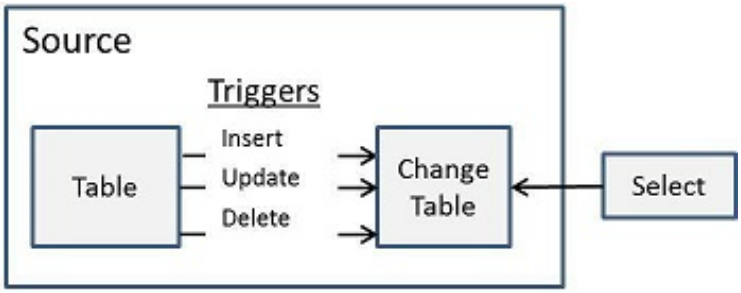


این گردش فرآیند شامل:

Join میان مبداء و مقصد با استفاده از یک دستور Left Outer Join است.
تمامی رکوردهای مبداء که در مقصد وجود ندارند، پردازش می‌شوند.
زمانی که رکوردی در مقصد وجود داشته باشد مقادیر داده‌های مبداء و مقصد مقایسه می‌شوند.
تمامی رکوردهای مبداء که تغییر یافته اند پردازش می‌شوند.
از آنجایی که تمامی رکوردها پردازش می‌شوند، این روش بویژه برای جداول حجیم؛ روش کارآمدی نیست.

2-3- Pushing Net Changes دو متد متداول Push وجود دارد که در تصویر زیر نمایش داده شده است.





تفاوت این دو روش به شرح زیر است:

در سناریو اول (شکل سمت چپ)؛ بانک اطلاعاتی رابطه ای سیستم مبداء Transaction Log را مرتب مانیتور می‌کند تا تغییرات را شناسائی کرده و در ادامه تمامی این تغییرات را در جدولی در مقصد درج می‌کند.
در سناریو دوم؛ توسعه دهندگان Trigger هایی ایجاد می‌کنند تا هر زمان که رکوردی تغییر یافت، تغییرات در جدولی که در مقصد وجود دارد درج گردد.

مسئله ای که در هر دو مورد وجود دارد Load اضافه ای است؛ که روی سیستم مبداء وجود دارد و می‌تواند Performance سیستم‌های OLTP را تحت تاثیر قرار دهد. به هر روی سناریو نخست معمولاً کاراتر از سناریویی است که از Trigger استفاده می‌کند.

3- ETL Patterns

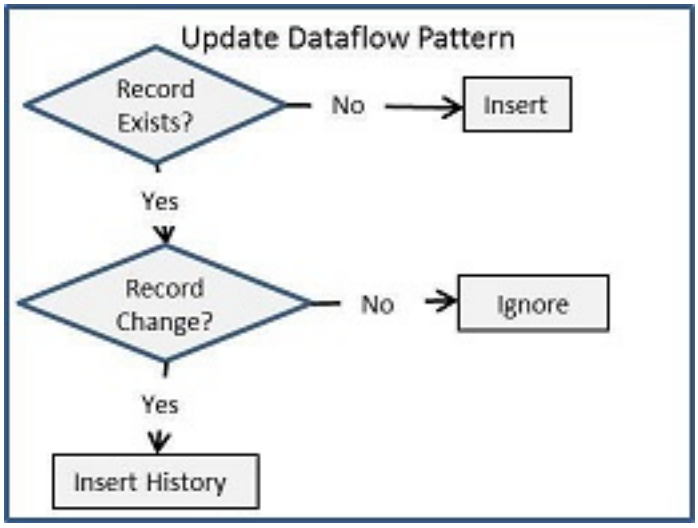
پس از شناسائی رکوردهایی که در مبداء تغییر یافته اند، نیاز داریم تا این تغییرات در مقصد اعمال شود. در این قسمت به معرفی الگوهایی که برای اعمال این تغییرات وجود دارد می‌پردازیم.

3-1- Destination load Patterns

تشخیص چگونگی اضافه نمودن تغییرات در مقصد تابع دو عامل زیر است:

آیا رکورد هم اینک در مقصد وجود دارد؟
الگوی استفاده شده برای جدول مقصد به کدام شکل است؟ (Update یا Versioned Insert)

فلوچارت زیر نشان می‌دهد، به چه شکل جداول مقصد متاثر از چگونگی پردازش رکوردهای مبداء قرار دارند. توجه داشته باشید که عمل بررسی بطور جداگانه و در یک لحظه صورت می‌گیرد.



Versioned Insert Pattern 3-2-

Kimball Type II Slowly Changing Dimension نمونه ای از الگوی Versioned Insert است؛ که در آن نمونه ای از یک موجودیت دارای ورژن‌های متعددی است. مطابق تصویر زیر؛ این الگو به ستون‌های اضافه ای نیاز دارند که وضعیت نمونه ای از یک رکورد را نمایش دهد.

Surrogate Key	Natural Key	Data...	Start Date	End Date	Record Status	Version #

این ستون‌ها به شرح زیر هستند:

Start Date: زمانی که وضعیت آن نمونه از رکورد فعال می‌شود.
End Date: زمانی که وضعیت آن نمونه از رکورد غیر فعال می‌شود.
Record Status: وضعیت‌های یک رکورد را نشان می‌دهد، که حداقل به شکل Active یا Inactive است.
Version #: این ستون که اختیاری می‌باشد، ورژن آن نمونه از رکورد را ثبت می‌کند.

برای مثال شکل زیر؛ بیانگر وضعیت اولیه رکوردی در این الگو است:

Surrogate Key	Natural Key	Data...	Start Date	End Date	Status	Ver #
100	AB549		2001-02-05	NULL	A	1

فرض کنید که این رکورد در تاریخ March 2 , 2010 در سیستم مبداء تغییر می‌کند. فرآیند ETL این تغییر را شناسائی می‌کند و همانند تصویر زیر؛ به شکل نمونه ای ثانویه از این رکورد، اقدام به درج آن می‌کند.

Surrogate Key	Natural Key	Data...	Start Date	End Date	Status	Ver #
100	AB549		2001-02-05	2010-03-02	I	1
537	AB549		2010-03-02	NULL	A	2

توجه داشته باشید زمانی که رکورد دوم در جدول درج می‌شود، به منظور بازتاب این تغییر؛ رکورد اول به شکل زیر بروزرسانی می‌گردد:

End Date: تا این زمان وضعیت این رکورد فعال بوده است.
Record Status:که Active به Inactive تغییر پیدا می‌کند.

در برخی از پیاده سازی‌های DW عمدتاً از الگوی Versioned Insert استفاده می‌شود و هرگز از الگوی Update استفاده نمی‌شود. مزیت این استراتژی در این است که تمامی تاریخچه تغییرات ردیابی و ثبت می‌شود. به هر روی غالباً هزینه ثبت کردن این تغییرات منجر به ایجاد نسخه‌های زیادی از تغییرات می‌شود. تیم DW برای مواردی که تغییرات متاثر از گزارشات تاریخچه ای نیستند، می‌توانند الگوی Update را در نظر گیرند.

Update Pattern 3-3-

الگوی Update روی رکورد موجود، تغییرات سیستم مبداء را بروزرسانی می‌کند. مزیت این روش در این است که همواره یک رکورد وجود دارد و در نتیجه باعث ایجاد Queryهای کارآمدتر می‌شود. تصویر زیر بیانگر ستون هایی است که برای پشتیبانی از الگوی Update بایست ایجاد کرد.

Surrogate Key	Natural Key	Data...	Record Status	Version #

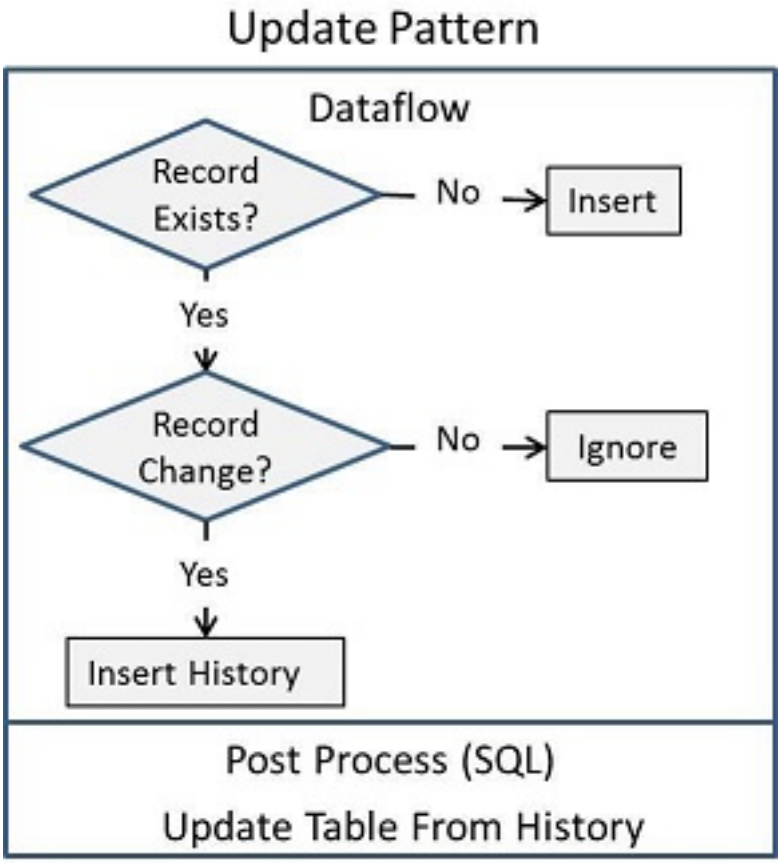
این ستون‌ها به شرح زیر هستند:

Record Status: وضعیت‌های یک رکورد را نشان می‌دهد که حداقل به شکل Active یا Inactive است.
Version #: این ستون که اختیاری می‌باشد، ورژن آن نمونه از رکورد را ثبت می‌کند.

موارد اصلی الگوی Update عبارتند از:

تاریخ ثبت نمی‌شود. ابزاری ارزشمند برای نظارت بر داده ها، تغییرات تاریخی است و زمانی که ممیزی داده رخ می‌دهد؛ می‌تواند مفید واقع شود.
بروزرسانی‌ها یک الگوی مبتنی بر مجموعه هستند. استفاده از بروزرسانی هر بار یک رکورد در ابزار ETL خیلی کارآمد (موجه) نیست.

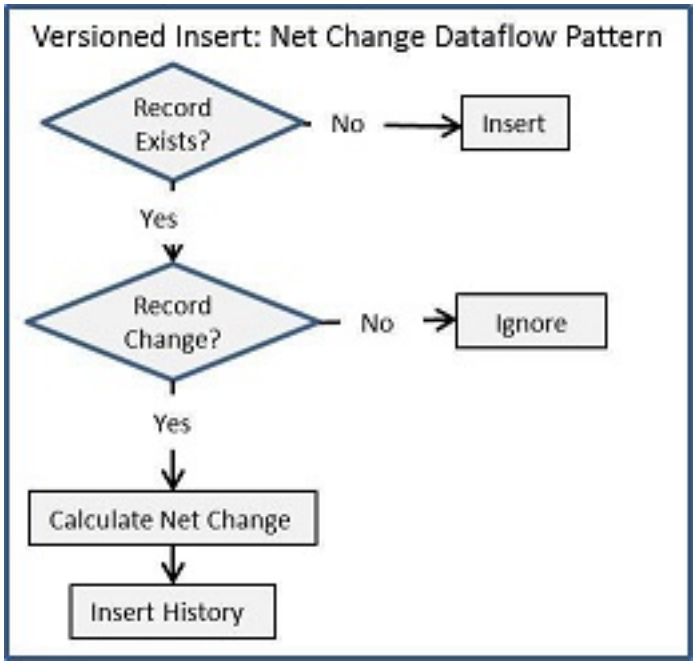
یک روش دیگر برای در نظر گرفتن موارد فوق؛ اضافه کردن یک جدول برای درج ورژن‌ها به الگوی Update است که در شکل زیر نشان داده شده است.



اضافه نمودن یک جدول تاریخچه، که تمامی تغییرات سیستم مبداء را ثبت می‌کند؛ نظارت و ممیزی داده‌ها را نیز فراهم می‌کند و همچنین بروزرسانی‌های کارآمد مبتنی بر مجموعه را برای جداول DW به ارمغان می‌آورد.

3-4- Versioned Insert: Net Changes

این الگو غالباً در جداول حجیم Fact که بروزرسانی آنها پر هزینه است استفاده می‌شود. شکل زیر منطق استفاده شده در این الگو را نشان می‌دهد.



توجه داشته باشید در این الگو:
مقادیر مالی و عددی محاسبه شده؛ به عنوان یک Net Change از نمونه قبلی رکورد در جدول Fact ذخیره می‌شود.
هیچ گونه فعالیت Post Processing صورت نمی‌گیرد (از قبیل بروزرسانی جداول Fact پس از کامل شدن Data Flow). هدف استفاده از این الگو اجتناب از بروزرسانی روی جداول بسیار حجیم می‌باشد.
عدم بروزرسانی و همچنین اندازه جدول Fact زمینه ای را فراهم می‌کند که منطق شناسائی رکوردهای تغییریافته پیچیده تر می‌شود. این پیچیدگی از آنجا ناشی می‌شود که نیاز به مقایسه رکوردهای جدول Fact آتی با جدول Fact موجود می‌باشد.

4- Data Integration Best Practices

هم اکنون پس از آشنایی با مفاهیم و الگوهای توزیع داده‌ها به ارائه تعدادی نمونه می‌پردازیم؛ که بتوان این ایده‌ها و الگوها را در عمل پوشش داد.

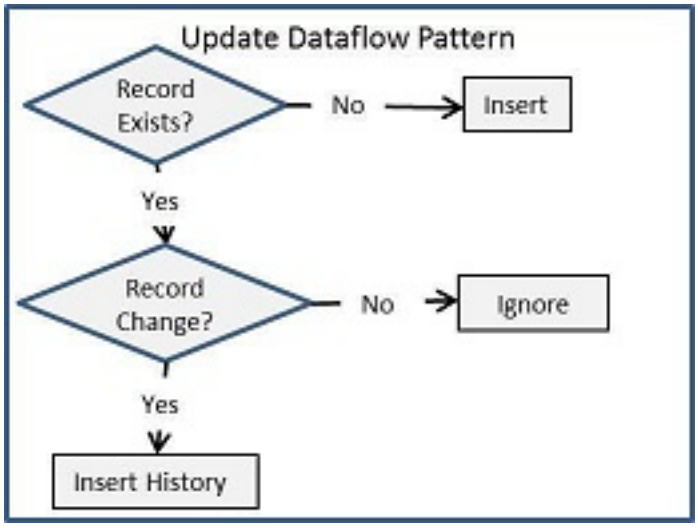
4-1- Basic Data Flow Patterns

هر یک از الگوهای Update Pattern و Versioned Insert Pattern می‌توانند برای انواعی از جداول بکار روند که معروفترین آن‌ها توسط Kimball ساخته شده اند.

Slowly Changing Dimension Type I (SCD I): از Update Pattern استفاده می‌کند.
Slowly Changing Dimension Type II (SCD II): از Versioned Insert Pattern استفاده می‌کند.
Fact Table: نوع الگویی که استفاده می‌کند به نوع جدول Fact ای که Load خواهد شد بستگی دارد.

4-1-1- Update Pattern

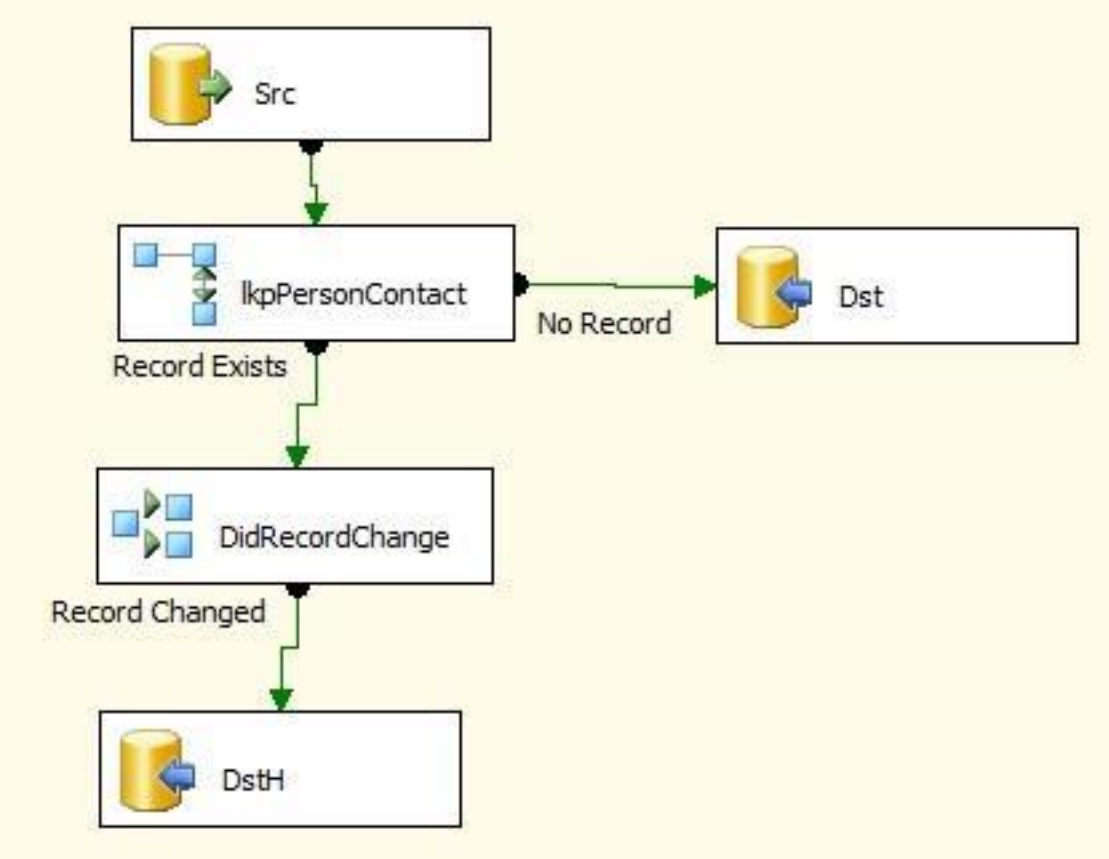
مطابق تصویر زیر جدولی که تنها حاوی ورژن فعلی رکورد هاست؛ از Update Dataflow Pattern استفاده می‌کند.



مواردی که در مورد این گردش کاری باید در نظر داشت به شرح زیر است:

این Data Flow فقط سطرهایی را به یک مقصد اضافه خواهد کرد. SSIS دارای گزینه "Table or view fast load" می‌باشد که بارگذاری‌های انبوه و سریع را پشتیبانی می‌کند.
درون یک Data Flow بروزرسانی رکوردها را می‌توان با استفاده از تبدیل OLE DB Command انجام داد. توجه داشته باشید خروجی‌های این تبدیل در یک دستور Update به ازای هر رکورد بکار می‌رود؛ مفهوم بروزرسانی انبوه در این Data Flow وجود ندارد. بدین ترتیب الگوی فعلی ارائه شده؛ تنها رکوردها را درج می‌کند و هرگز در این Data Flow رکوردها Update نمی‌شوند.
هر جدول دارای یک جدول تاریخچه است که برای ذخیره همه فعالیت‌های مرتبط با آن بکار می‌رود. یک رکورد در جدول تاریخچه زمانی درج خواهد شد؛ که رکورد مبداء در مقصد وجود داشته باشد ولی دارای مقداری متفاوت باشد.
راه دیگر فرستادن تغییرات رکوردها به یک جدول کاری است که پس از پایان یافتن فرآیند Update ، خالی (Truncate) می‌شود.
مزیت نگهداری تمامی رکوردها در یک جدول تاریخچه؛ ایجاد یک دنباله ممیزی است که می‌تواند برای نظارت بر داده‌ها به منظور نمایان ساختن موارد مطرح شده توسط مصرف کننده‌های کسب و کار استفاده شود.
گزینه‌های متفاوتی برای تشخیص تغییرات رکوردها وجود دارد که در ادامه به شرح آنها می‌پردازیم.

شکل زیر نمایش دهنده چگونگی پیاده سازی Update Dataflow Pattern در یک SSIS می‌باشد:



این SSIS شامل عناصر زیر است:

Destination table lookup:

به منظور تشخیص اینکه رکورد در جدول مقصد وجود دارد از "lkpPersonContact" استفاده می‌کنیم.

Change detection logic:

با استفاده از "DidRecordChange" مبداء و مقصد مقایسه می‌شوند. اگر تفاوتی بین مبداء و مقصد وجود نداشت؛ رکورد نادیده گرفته می‌شود. چنانچه بین مبداء و مقصد تفاوت وجود داشت؛ رکورد در جدول تاریخچه درج خواهد شد.

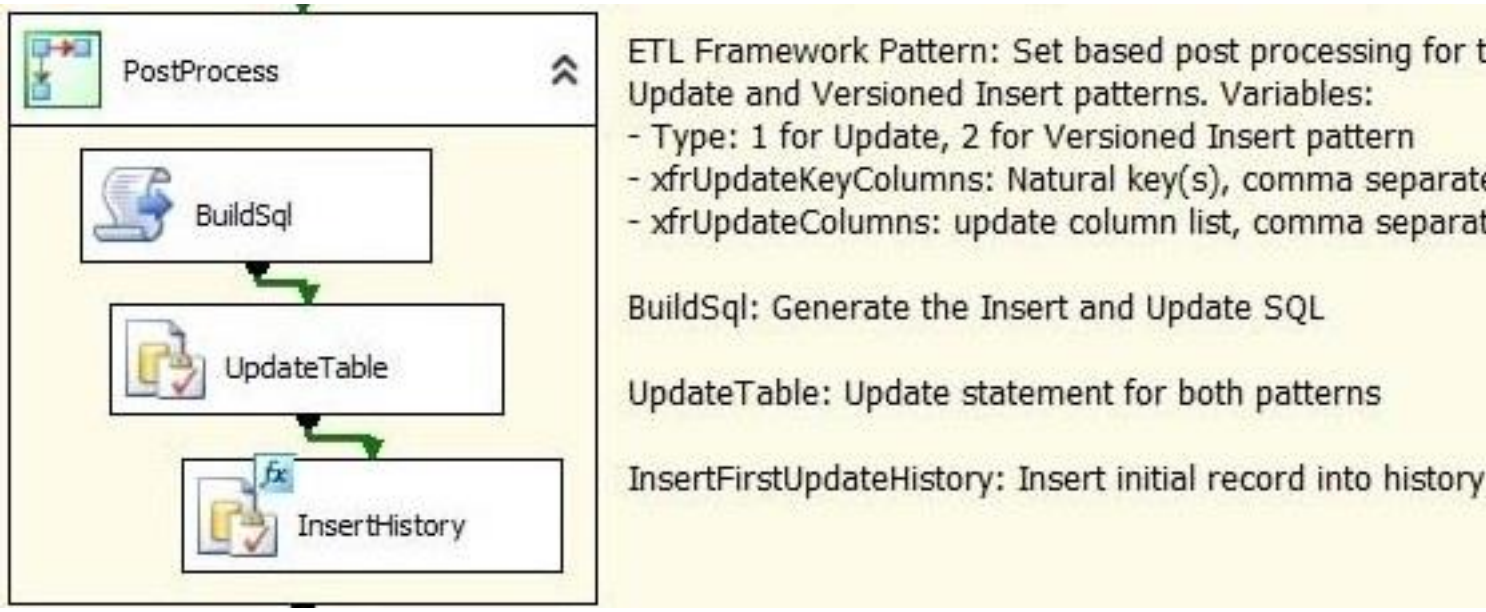
Detection Inserts:

رکوردها در جدول مقصد درج خواهند شد در صورتیکه در آن وجود نداشته باشند.

Destination History Inserts:

رکوردها در جدول تاریخچه مقصد درج خواهند شد، در صورتیکه (در مقصد) وجود داشته باشند.

پس از اتمام Data Flow یک روال Post-processing مسئولیت بروزرسانی رکوردهای جدول اصلی و رکوردهای ذخیره شده در جدول تاریخچه را بر عهده دارد که می‌تواند مطابق تصویر زیر با استفاده از یک Execute Process Task پیاده سازی شود.



PostProcess مسئولیت اجرای تمامی فعالیت‌های زیر را در این الگو برعهده دارد که شامل:

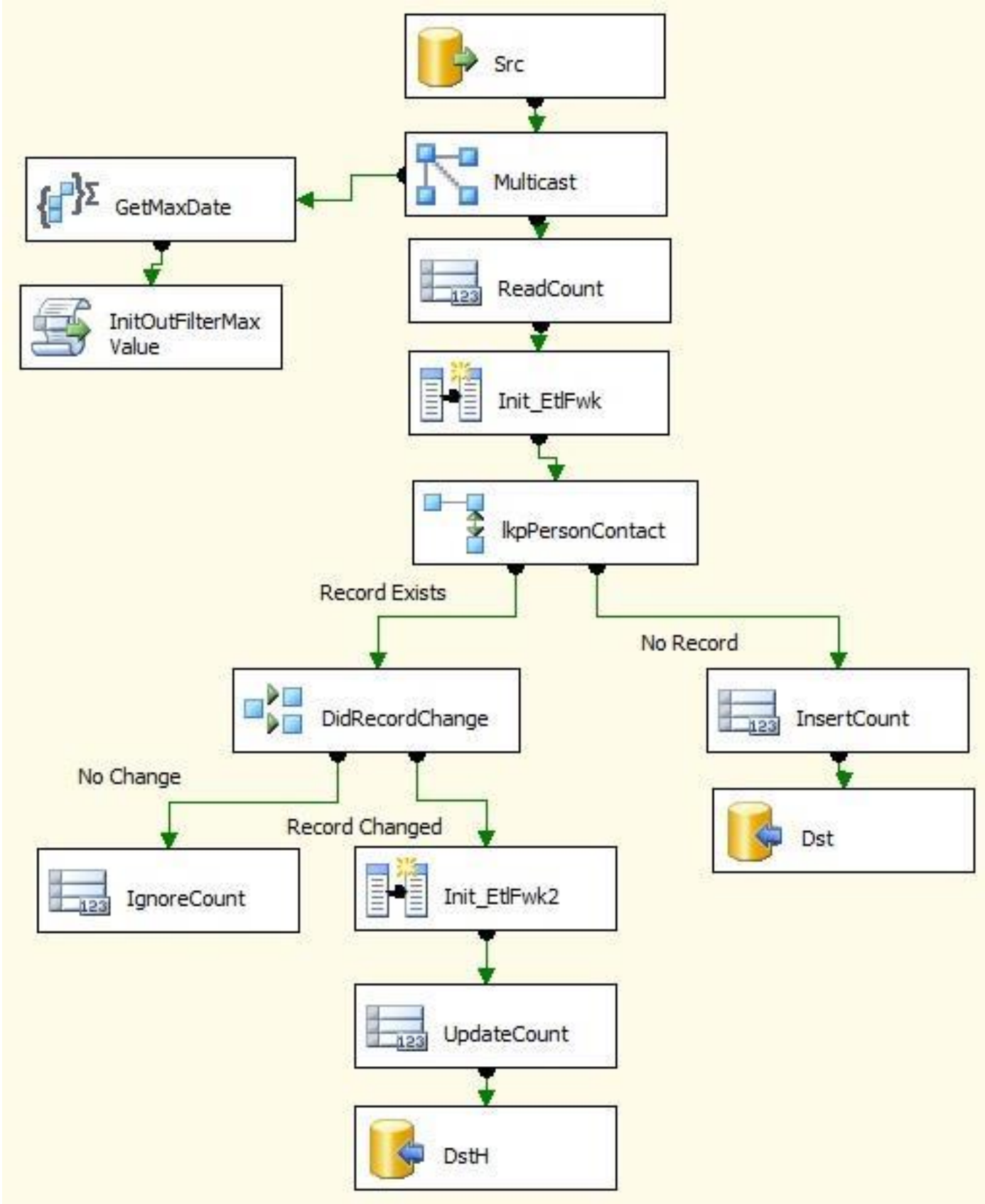
بروزرسانی رکوردهای جداول با استفاده از رکوردهای درج شده در جدول تاریخچه.
درج تمامی رکوردهای جدید (نسخه اولیه و در درون جدول تاریخچه). کلید اصلی جداولی که ستون آنها IDENTITY است مقدار نامشخصی دارد؛ تا زمانی که درج صورت گیرد، این به معنای آن است که پیش از انتقال آنها به جدول تاریخچه نیاز است منتظر درج شدن آنها باشیم.

Update Pattern – ETL Framework 4-1-2-

تصویر زیر بیانگر انجام این عملیات با استفاده از ابزارهای ETL است.
در نگاه نخستین ممکن است Data Flow از نوع اصلی خود پیچیده‌تر به نظر آید؛ که در واقع این گونه نیز هست، زیرا در فاز توسعه بیشتر Frameworkها جهت پیاده سازی به یک زمان اضافه‌تری نیاز دارند. به هر روی این زمان جهت اجتناب از هزینه روزانه تطبیق داده‌ها گرفته خواهد شد.
مزایای حاصل شده از افزودن این منطق اضافی عبارت است از:

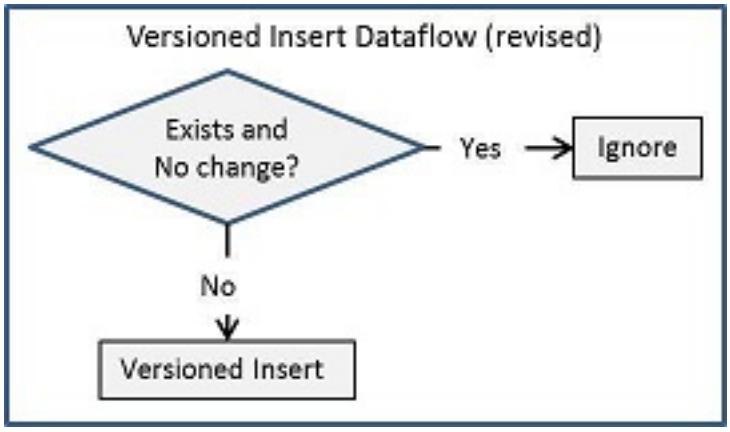
پشتیبانی از ستون هایی که کارهای ممیزی و نظارت بر داده‌ها را آسانتر می‌کنند.
تعداد سطرها شاخص مناسبی است که می‌تواند بهبود آن Data Flow خاص را فراهم کند. ناظر اطلاعات با استفاده از تعداد رکوردها می‌تواند ناهنجاری‌ها را شناسائی کند.

بهره برداران ETL و ناظران اطلاعات می‌توانند با استفاده از خلاصه تعداد رکوردها درک بیشتری درباره فعالیت‌های آن کسب کنند. پس از آنکه تعداد رکوردها، مشکوک به نظر آمد؛ تحقیقات بیشتری می‌تواند اتفاق افتد. (با عمیق‌تر شدن در جزئیات گزارشات)



Versioned Insert Pattern 4-1-3-

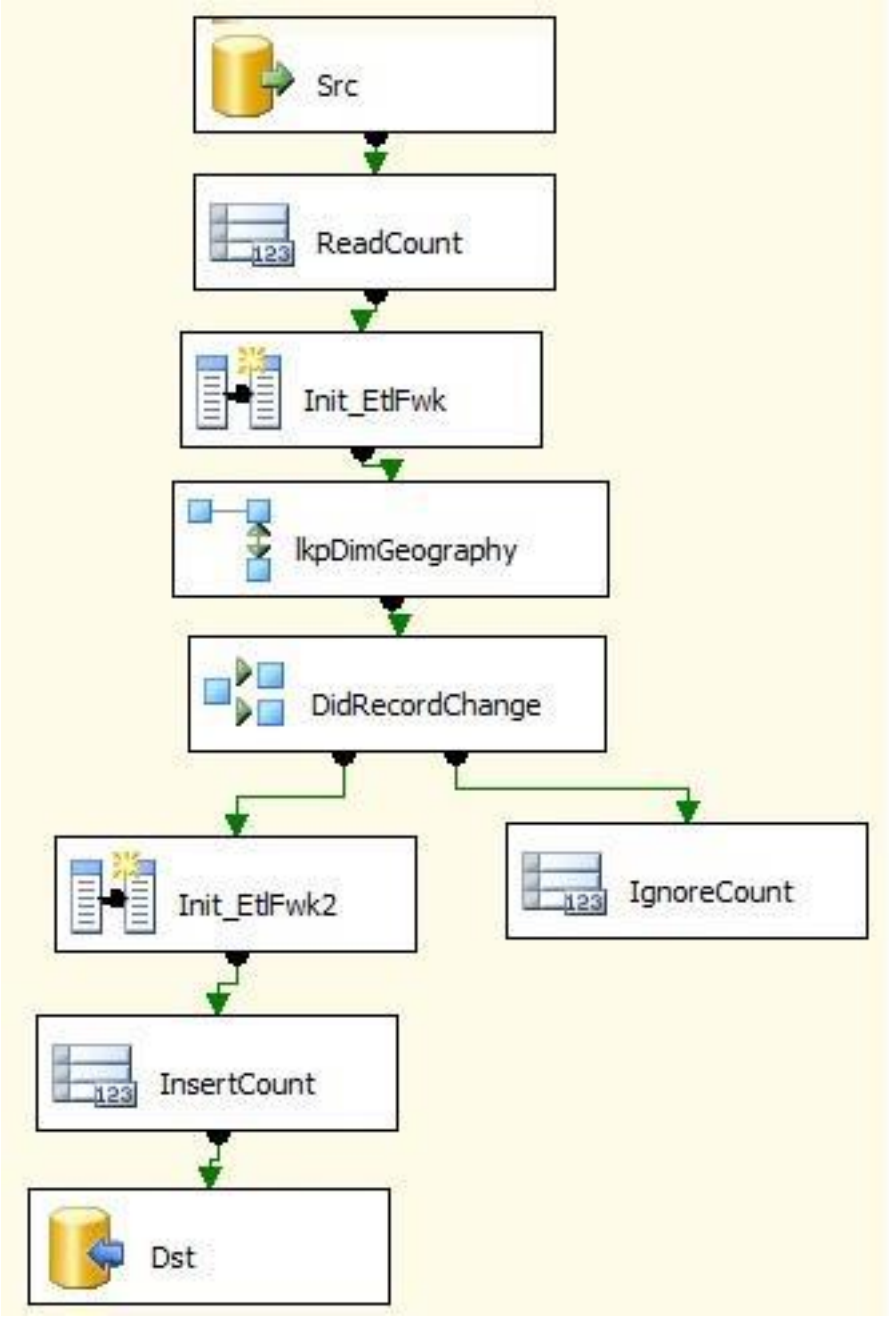
جدولی که به صورت Versioned Insert پر شده است می‌تواند از Versioned Insert Dataflow Pattern استفاده کند. همانند شکل زیر که گردش کار در آن برای کارآئی بیشتر بازنگری شده است.



توجه داشته باشید Data Flow در این روش شامل:

تمامی رکوردهای جدید و تغییر یافته در جدول Versioned Insert قرار می‌گیرند.
این روش دارای Data Flow ساده‌تری نسبت به الگوی Update می‌باشد.

شکل زیر SSIS versioned insert data flow pattern را نشان می‌دهد:



تعدادی نکته در Data Flow فوق وجود دارد که عبارتند از:

در شیء "lkpDimGeography" گزینه "Redirect rows to no match output" با مقدار "Ignore Failures" تنظیم شده است.
شیء "DidRecordChange" بررسی می‌کند چنانچه ستون‌های مبداء و مقصد یکسان باشند، آیا کلید اصلی جدول مقصد Not Null است. اگر این عبارت True ارزیابی شود، رکورد نادیده گرفته می‌شود.
منطق شناسائی تغییرات دربردارنده تغییرات ستون داده ای در مبداء نمی‌باشد.
ستون و تعداد رکوردها مشابه با Data Flow قبلی (ETL Framework) می‌باشد.

4-1-4- Update vs. Versioned Insert

الگوی Versioned Insert نسبت الگوی Update دارای پیاده سازی ساده‌تر و فعالیت‌های I/O کمتری است. از منظر دیگر، جدولی که از الگوی Update استفاده می‌کند، دارای تعداد رکوردهای کمتری است که می‌تواند به معنای Performance بهتر نیز تعبیر شود. ممکن است سوالی مطرح شود، اینکه چرا برای انجام کار به جدول تاریخچه نیاز است؛ این جدول را که نمی‌توان Truncate نمود،

پس چرا به منظور بروزرسانی از جدول اصلی استفاده می‌شود؟ پاسخ این پرسش در این است که جدول تاریخچه، ناظر اطلاعات و ممیزین داده را قادر می‌سازد، تغییرات در طول زمان را پیگیری نمایند.

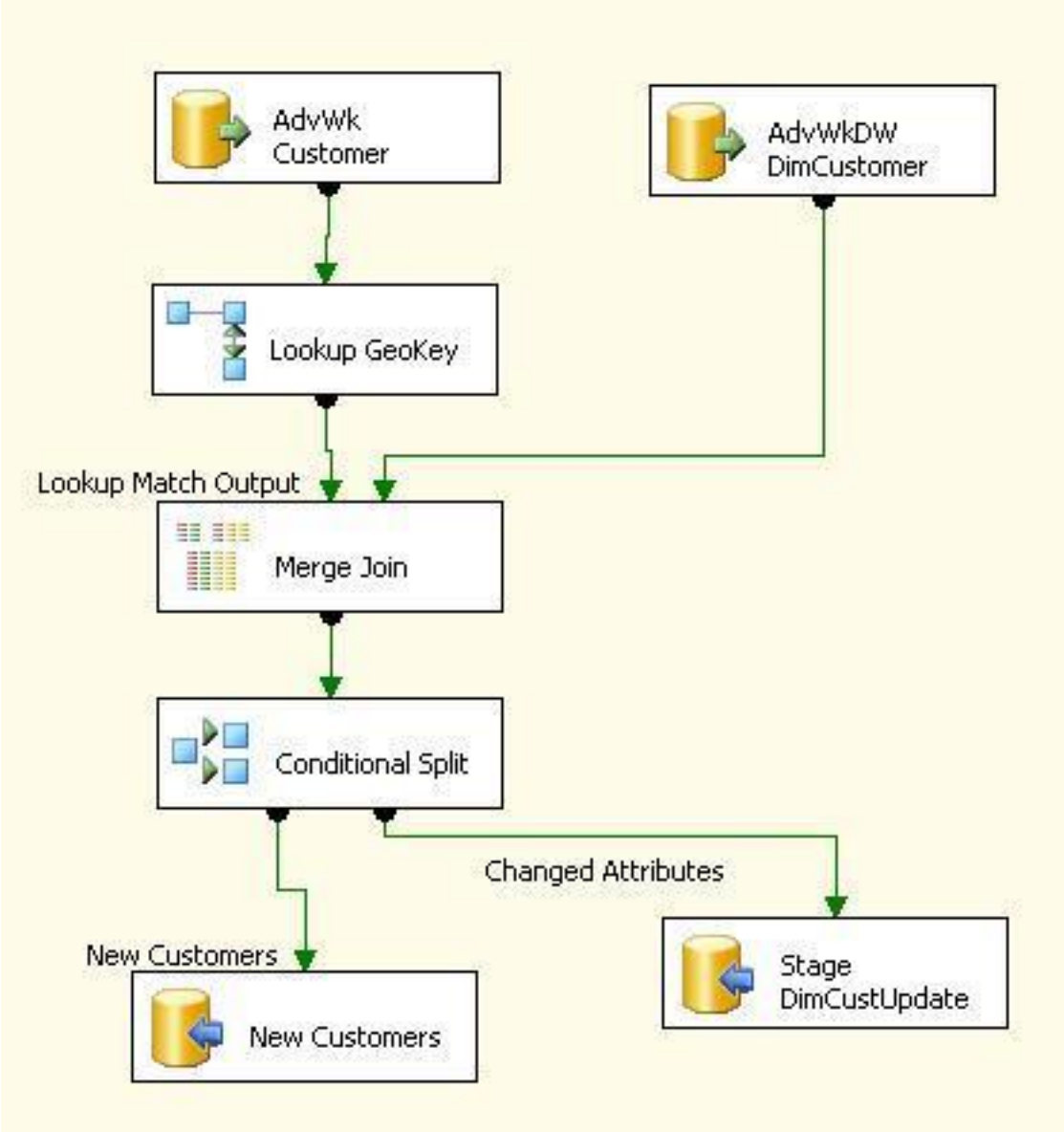
4-2- Dimension Patterns

بروزرسانی Dimension موارد زیر را شامل می‌شود:

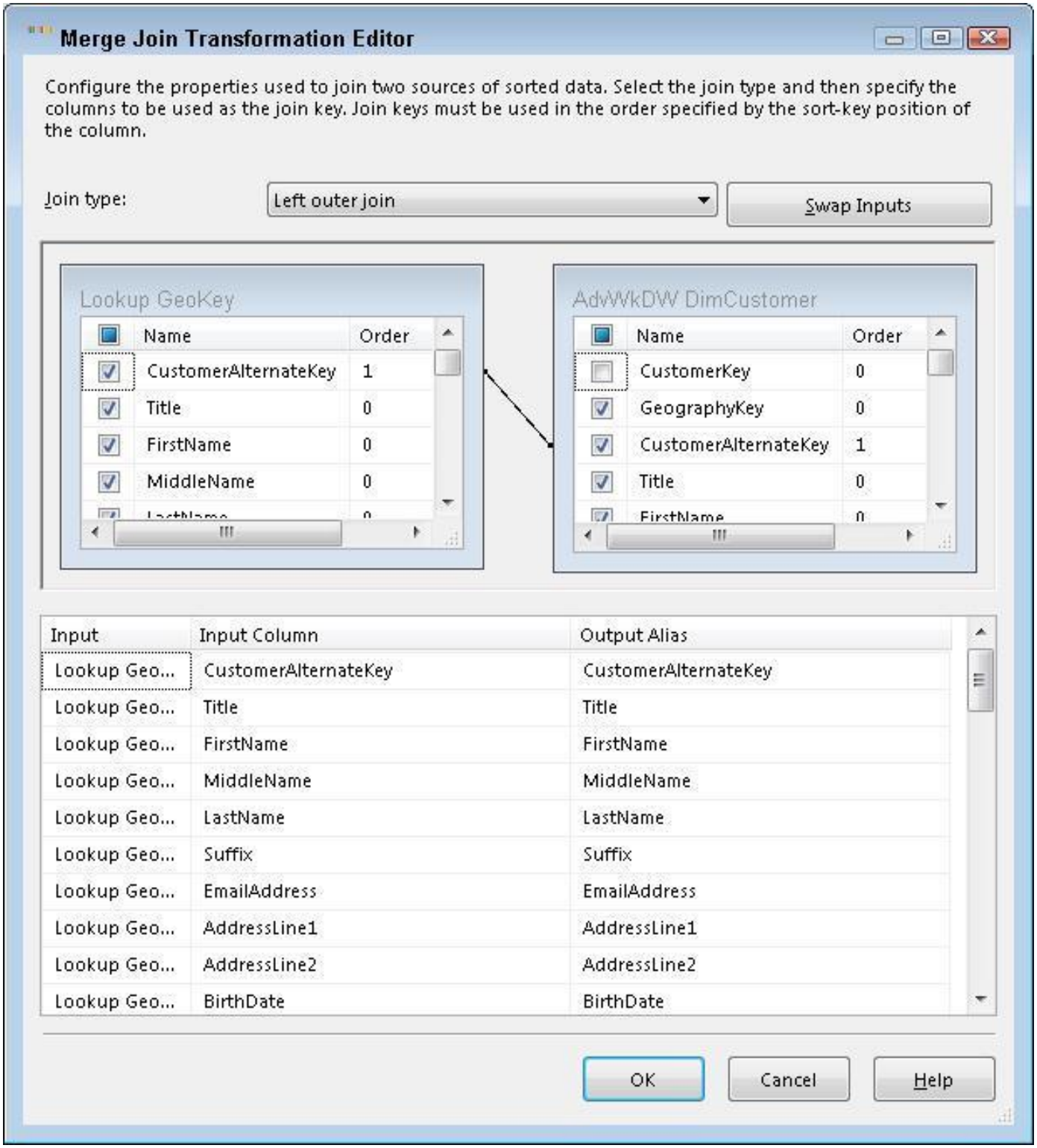
پیگیری تاریخچه
انجام بروزرسانی
تشخیص رکوردهای جدید
مدیریت surrogate keys

چنانچه با یک Dimension کوچک مواجه هستید (با مقدار هزاران رکورد یا کمتر، که با صدها هزار رکورد یا بیشتر ضدیت دارد)، می‌توانید از تبدیل "Slowly Changing Dimension" که بصورت Built-in در SSIS موجود است، استفاده نمائید. به هر روی با آنکه این تبدیل چندین ویژگی محدودکننده Performance دارد، اغلب کارآمدتر از پروسسه هایی که توسط خودتان ایجاد می‌شود. در واقع فرآیند بارگذاری در جداول Dimension با مقایسه داده‌ها بین مبداء و مقصد انجام می‌شود. به طور معمول مقایسه روی یک ورژن جدید و یا مجموعه ای از سطرهای جدید یک جدول با مجموعه داده‌های موجود در جدول متناظرش صورت می‌گیرد. پس از تشخیص چگونگی تغییر در داده ها، یک سری عملیات درج و بروزرسانی انجام می‌شود. شکل زیر نمونه ای از پردازش سریع در Dimension را نمایش می‌دهد؛ که شامل مراحل اساسی زیر است:

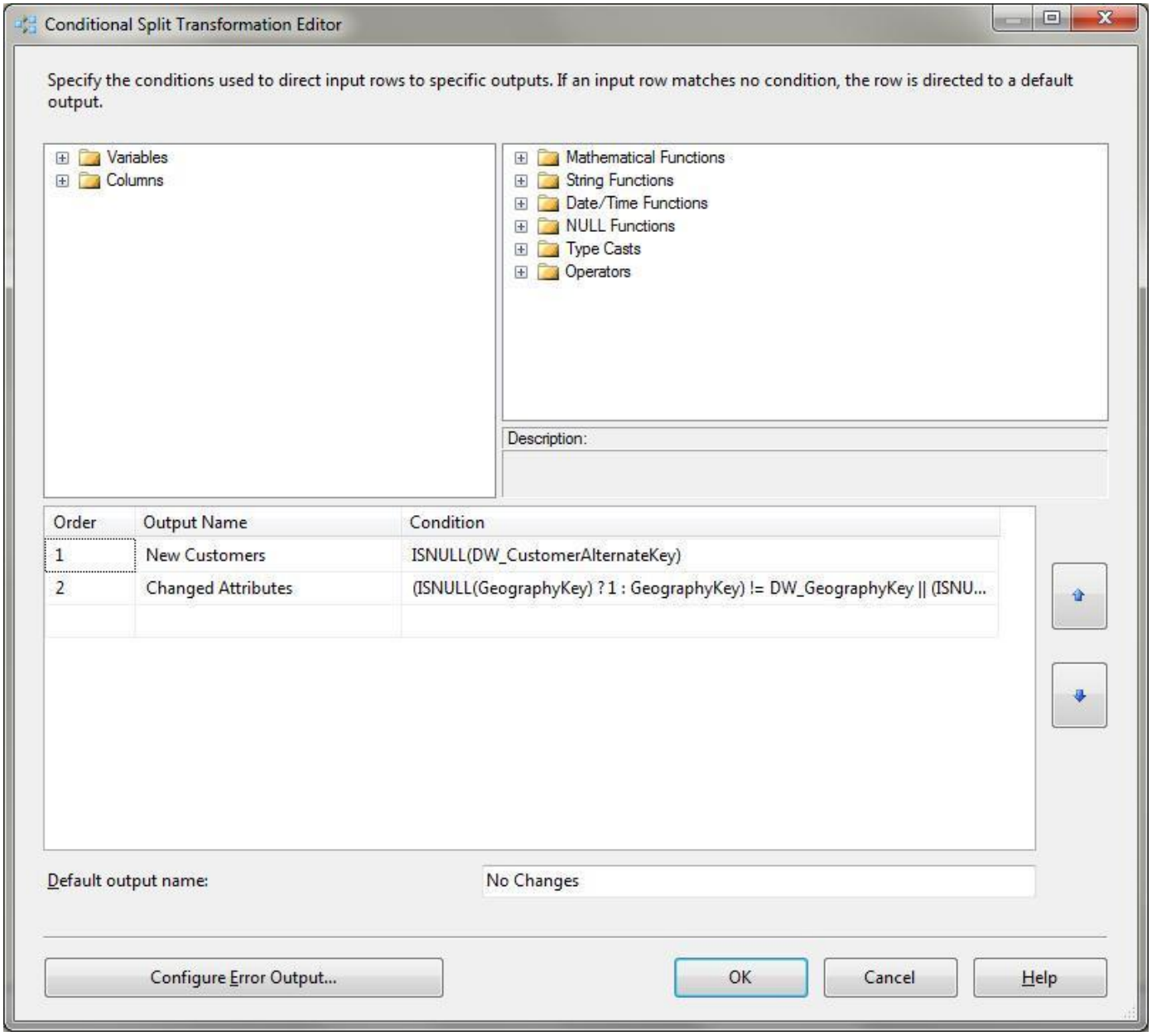
منبع فوقانی سمت چپ، رکوردها را در یک SSIS از یک سیستم مبداء (یا یک سیستم میانی) به شکل Pull دریافت می‌کند. منبع فوقانی سمت راست، داده‌ها را از خود جدول Dimension به شکل Pull دریافت می‌کند.
با استفاده از Merge Join رکوردها از طریق Source Key شان مقایسه می‌شوند. (در شکل بعدی جزئیات این مقایسه نمایش داده شده است.)
با استفاده از یک Conditional Spilt داده‌ها ارزیابی می‌شوند؛ سطرها یا مستقیماً در جدول Dimension درج می‌شوند (منبع تحتانی سمت چپ) و یا در یک جدول عملیاتی (منبع تحتانی سمت راست) جهت انجام بروزرسانی درج می‌شوند.
در گام پایانی (که نمایش داده نشده) مجموعه ای از بروزرسانی بین جدول عملیاتی و جدول Dimension صورت می‌گیرد.



با Merge Join ارتباطی بین رکوردهای مبداء و رکوردهای مقصد برقرار می‌شود. (در این مثال "CustomerAlternateKey"). هنگامی که از این دیدگاه استفاده می‌کنید، خاطر جمع شوید که نوع Join با مقدار "Left outer join" تنظیم شده است؛ بدین ترتیب قادر هستید تا رکوردهای جدید را از مبداء تشخیص دهید؛ از آنجا که هنوز در جدول Dimension قرار نگرفته اند.



گام پایانی به منظور تشخیص اینکه آیا رکورد، جدید یا تغییر یافته است (یا بلاتکلیف است)، مقایسه داده هاست. شکل زیر نمایش می‌دهد چگونه این ارزیابی با استفاده از تبدیل "Conditional Spilt" صورت می‌گیرد.



Conditional Spilt مستقیماً با استفاده از یک Adapter تعریف شده روی مقصد یا یک جدول کاری بروزرسانی که از یک Adapter تعریف شده روی مقصد استفاده می‌کند؛ توسط مجموعه دستور Update زیر، رکوردها را در جدول Dimension قرار می‌دهد. دستور Update زیر مستقیماً با استفاده از روش Join روی جدول Dimension و جدول کاری، مجموعه ای را بصورت انبوه بروزرسانی می‌کند.

```
UPDATE AdventureWorksDW2008R2.dbo.DimCustomer
    SET AddressLine1 = stgDimCustomerUpdates.AddressLine1
    , AddressLine2 = stgDimCustomerUpdates.AddressLine2
    , BirthDate = stgDimCustomerUpdates.BirthDate
    , CommuteDistance = stgDimCustomerUpdates.CommuteDistance
    , DateFirstPurchase = stgDimCustomerUpdates.DateFirstPurchase
    , EmailAddress = stgDimCustomerUpdates.EmailAddress
    , EnglishEducation = stgDimCustomerUpdates.EnglishEducation
    , EnglishOccupation = stgDimCustomerUpdates.EnglishOccupation
    , FirstName = stgDimCustomerUpdates.FirstName
    , Gender = stgDimCustomerUpdates.Gender
    , GeographyKey = stgDimCustomerUpdates.GeographyKey
    , HouseOwnerFlag = stgDimCustomerUpdates.HouseOwnerFlag
    , LastName = stgDimCustomerUpdates.LastName
    , MaritalStatus = stgDimCustomerUpdates.MaritalStatus
    , MiddleName = stgDimCustomerUpdates.MiddleName
```

```
    , NumberCarsOwned = stgDimCustomerUpdates.NumberCarsOwned
    , NumberChildrenAtHome = stgDimCustomerUpdates.NumberChildrenAtHome
    , Phone = stgDimCustomerUpdates.Phone
    , Suffix = stgDimCustomerUpdates.Suffix
    , Title = stgDimCustomerUpdates.Title
    , TotalChildren = stgDimCustomerUpdates.TotalChildren
FROM AdventureWorksDW2008.dbo.DimCustomer DimCustomer
  INNER JOIN dbo.stgDimCustomerUpdates ON
DimCustomer.CustomerAlternateKey = stgDimCustomerUpdates.CustomerAlternateKey
```

4-3- Fact Table Patterns

جداول Fact به پردازش‌های منحصر به فردی نیازمند هستند، نخست به کلیدهای Surrogate جدول Dimension نیاز دارند تا Measureهای محاسبه شدنی را بدست آورند. این اعمال از طریق تبدیلات Lookup، Merge Join و Derived Column صورت می‌گیرد. با بروزرسانی ها، تفاضل رکوردها و یا Snapshot بیشتر این فرآیندهای دشوار انجام می‌شوند.

4-3-1- Inserts

روی اغلب جداول Fact عمل درج صورت می‌گیرد؛ که کار متداولی در جدول Fact می‌باشد. شاید ساده‌ترین کار که در فرآیند ساخت ETL صورت می‌گیرد، عملیات درج روی تنها تعدادی از جدول Fact می‌باشد. درج کردن در صورت لزوم بارگذاری انبوه داده ها، مدیریت شاخص‌ها و مدیریت پارتیشن‌ها را شامل می‌شود.

4-3-2- Updates

بروزرسانی روی جداول Fact معمولاً به یکی از سه طریق زیر انجام می‌گیرد:

از طریق یک تغییر یا بروزرسانی رکورد
از طریق یک دستور Insert خنثی کننده (Via an Insert of a compensating transaction)
با استفاده از یک SQL MERGE

در موردی که تغییرات با فرکانس کمی روی جدول Fact صورت می‌گیرد و یا فرآیند بروزرسانی قابل مدیریت است؛ ساده‌ترین روش انجام یک دستور Update روی جدول Fact می‌باشد. نکته مهمی که هنگام انجام بروزرسانی باید به خاطر داشته باشید، استفاده از روش بروزرسانی مبتنی بر مجموعه است؛ به همان طریق که در قسمت الگوهای Dimension ذکر آن رفت.
در طریقی دیگر (درج compensating) می‌توان اقدام به درج رکورد تغییر یافته نمود، تا ترجیحاً بروزرسانی روی آن صورت گیرد. این استراتژی به سادگی داده‌های جدول Fact میان سیستم مبداء و مقصد را که تغییر یافته اند، به صورت یک رکورد جدید درج خواهد کرد. تصویر زیر مثالی از اجرای موارد فوق را نمایش می‌دهد.

Source ID	Measure Value
12345	80

Source data

Source ID	Measure Value
12345	100

Current fact table data

Source ID	Current Measure Value
12345	100
12345	**- 20**

New fact table data

در آخرین روش از یک دستور SQL MERGE استفاده می‌شود که در آن با استفاده از ادغام و مقایسه، تمامی داده‌های جدید و تغییر یافته جدول Fact، درج و یا بروزرسانی می‌شوند. نمونه ای از استفاده دستور Merge به شرح زیر است:

```
MERGE dbo.FactSalesQuota AS T
USING SSIS_PDS.dbo.stgFactSalesQuota AS S
ON T.EmployeeKey = S.EmployeeKey
AND T.DateKey = S.DateKey
WHEN MATCHED AND BY target
THEN INSERT(EmployeeKey, DateKey, CalendarYear, CalendarQuarter, SalesAmountQuota)
VALUES(S.EmployeeKey, S.DateKey, S.CalendarYear, S.CalendarQuarter, S.SalesAmountQuota)
WHEN MATCHED AND T.SalesAmountQuota != S.SalesAmountQuota
THEN UPDATE SET T.SalesAmountQuota = S.SalesAmountQuota
;
```

اشکال این روش Performance است؛ گرچه این دستور به سادگی عملیات درج و بروزرسانی را انجام می‌دهد ولی به صورت سطر به سطر عملیات انجام می‌شود (در هر زمان یک سطر). در موقعیت هایی که با مقدار زیادی داده مواجه هستید، اغلب بهتر است به صورت انبوه عملیات درج و به صورت مجموعه عملیات بروزرسانی انجام گیرد.

4-3-3- Managing Inferred Members

زمانیکه یک ارجاع در جدول Fact به یک عضو Dimension که هنوز بارگذاری نشده‌است بوجود آید؛ یک Inferred Member تعبیر می‌شود. به سه طریق می‌توان این Inferred Memberها را مدیریت نمود:

رکوردهای جدول Fact پیش از درج اسکن شوند؛ ایجاد هر Inferred Member در Dimension و سپس بارگذاری رکوردها در جدول Fact

در طول عملیات بارگذاری روی Fact؛ هر رکورد مفقوده شده به یک جدول موقتی ارسال شود، رکوردهای مفقوده شده به Dimension اضافه شود، در ادامه مجدداً آن رکوردهای Fact در جدول Fact بارگذاری شوند.

در یک Data Flow زمانی که یک رکورد مفقود شده، بلاتکلیف تعبیر می‌شود؛ آن زمان یک رکورد به Dimension اضافه شود و

Surrogate Key بدست آمده را برگردانیم؛ سپس Dimension بارگذاری شود.

شکل زیر این موارد را نمایش می‌دهد: