CDC Demo

**Step 1 🡪 Start SQL Server Agent Service**

**Step 2 🡪 Create a database CDC\_Demo\_DB and enable it for CDC**

UseMaster

go

CreateDatabaseCDC\_Demo\_DB

go

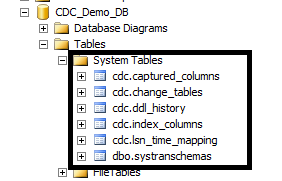
UseCDC\_Demo\_DB

go

EXECsys.sp\_cdc\_enable\_db

GO

Notice the system tables getting created inside that database after executing the last command of **EXEC sys.sp\_cdc\_enable\_db**

****

**Step 3.1 🡪 Create a table students inside this database with primary key, add 2 rows in it and enable it for CDC**

UseCDC\_Demo\_DB

go

createtablestudent

(rollnointeger Not Null,

snamevarchar(30),

marksint)

go

Altertablestudent

AddconstraintPK\_RollnoPrimaryKey(Rollno)

go

InsertIntoStudent(Rollno,Sname,Marks)Values (1,'Smith',90)

InsertIntoStudent(Rollno, Sname,Marks)Values

(2, 'Martin',78)

go

**Step 3.2 🡪 Enable cdc at the table level**

EXECsys.sp\_cdc\_enable\_table

@source\_schema=N'dbo',

@source\_name=N'Student',

@role\_name=N'cdc\_admin',

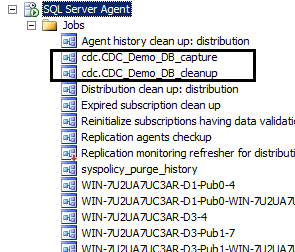
@supports\_net\_changes=1

GO

**Job 'cdc.CDC\_Demo\_DB\_capture' started successfully.**

**Job 'cdc.CDC\_Demo\_DB\_cleanup' started successfully.**

Notice that two new jobs have been created in SQL Server Job list:



**Step 4 🡪 Create the destination table:**

UseCDC\_Demo\_DB

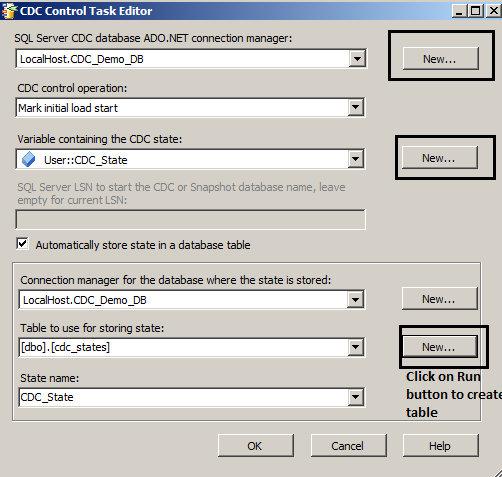
go

Select\*intoStudent\_DestinationfromStudent

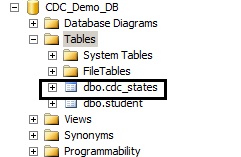
where1=2

**Step 5 🡪 SSIS Project**

1. Create a new SSIS Project. Name it as **CDC\_Project**
2. Rename the existing package as **Initial\_Load**
3. \*\*Place **CDC Control Task** in the Control Flow Designer
4. Name the CDC task as **CDC Control Task - Initial Start**



Notice in SSMS a new table gets created:



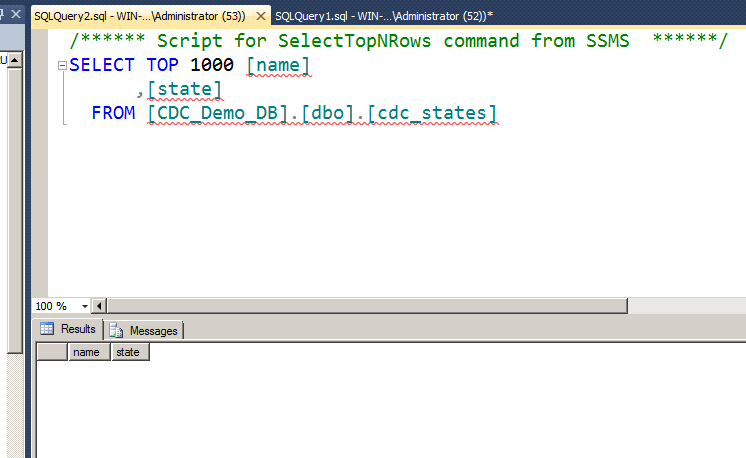
Right now, this table is empty:

Run the following command:

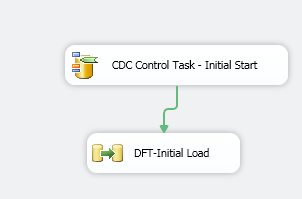
SELECT TOP (1000) [name]

,[state]

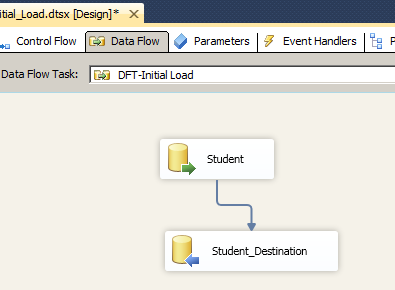
FROM [CDC\_Demo\_DB].[dbo].[cdc\_states]



1. Put a **DFT** below the CDC task and link it to it. Rename the DFT as **DFT-Initial Load**

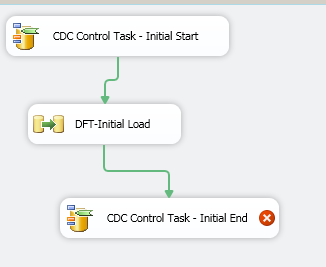


1. The DFT details:

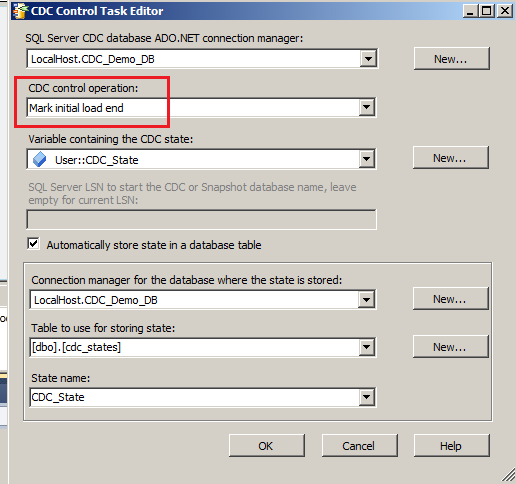


*Note 🡪 Keep Identity option is made on for the destination*

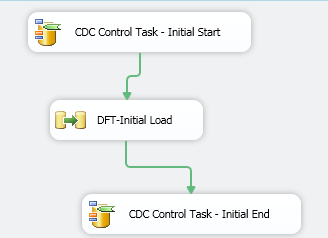
1. Now put one more CDC Control Task. Rename it as **CDC Control Task – Initial End**
2. Link it to the above DFT task.

****

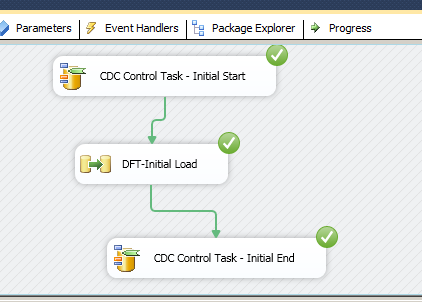
1. Open its editor window & do the following changes:



Final look of the Control Flow will be as shown next –



1. Save & Run the package



1. Stop Debugging
2. Cross check in SSMS:

UseCDC\_Demo\_DB

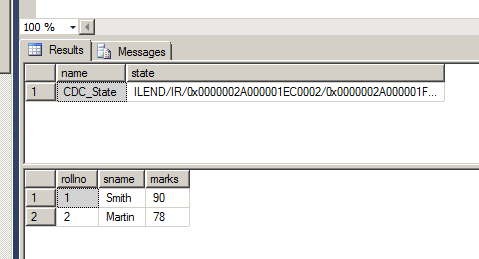
go

select\*fromcdc\_states

go

select\*fromstudent\_destination

go



**Explanation of first package:**

After you enable CDC on the tables you want to track, you use an SSIS package to do a one-time load of your existing data into your table. The first task is a CDC Control task, which is new in SSIS 2012. In this case, you configure the task to specify the CDC state for the initial load. The CDC state is stored within an SSIS variable, and the variable's value is persisted to a table in the target database.

**By persisting this value, you can source only the unprocessed changes on subsequent runs.**

Phase 2: Creating the Incremental Load Package

1. Create the staging tables in SSMS:

UseCDC\_Demo\_DB

go

SELECTTOP0\*INTOstaging\_UPDATES

FROMstudent

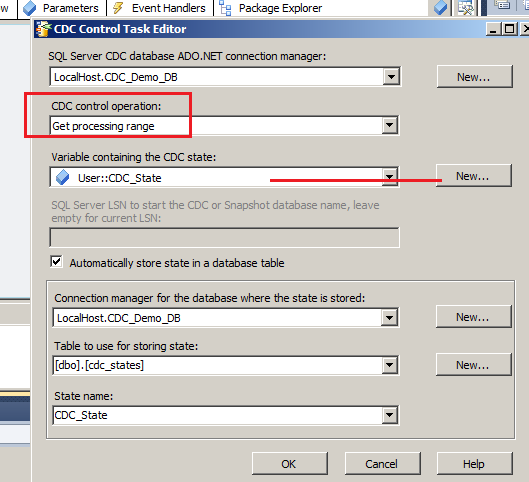
go

SELECTTOP0\*INTOstaging\_DELETES

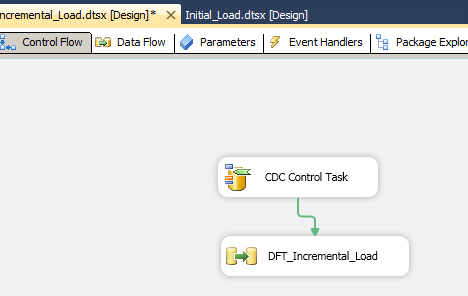
FROMStudent

Go

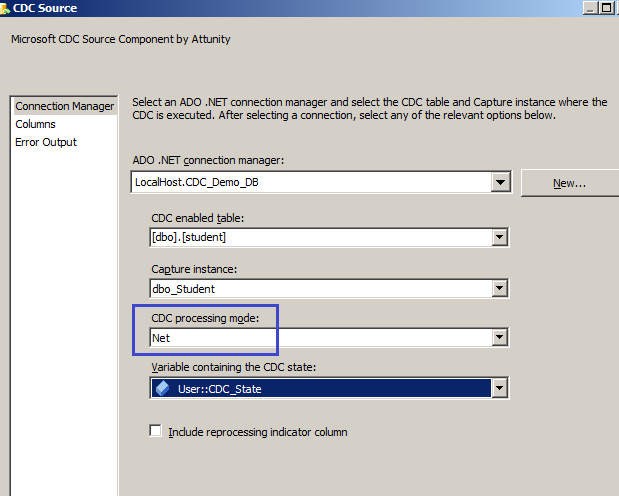
1. Create a **new package** in the same project with name as **Incremental\_Load**
2. Put CDC Control Task in it



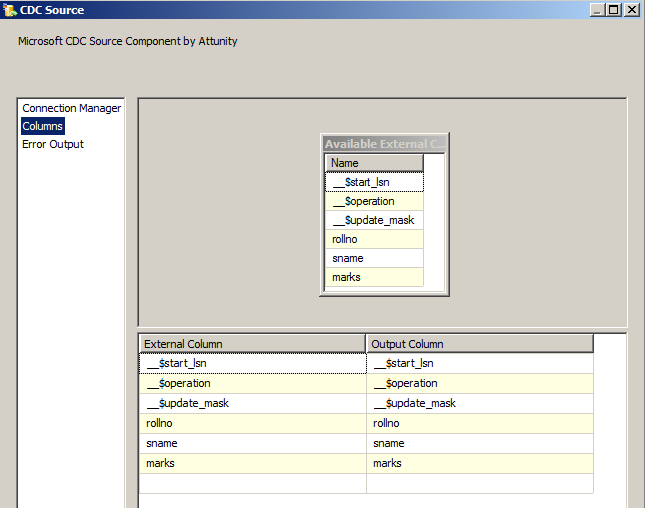
1. Put **DFT** below CDC Control task and link it. Rename it as **DFT\_Incremental\_Load**



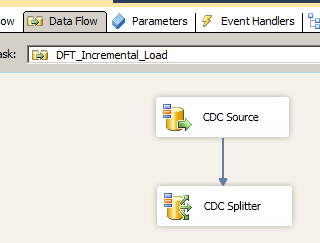
1. \*\* Put **CDC Source** inside DFT



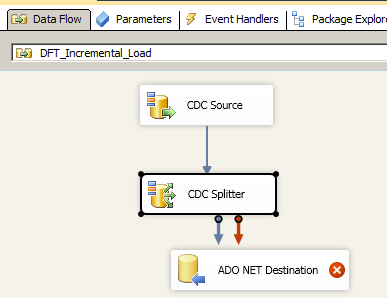
Notice Columns:



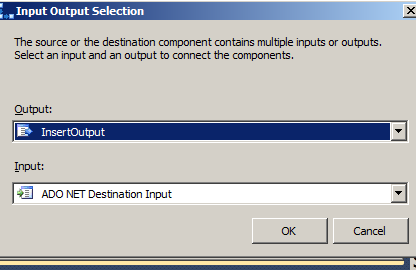
1. \*\* From Other Transformations put **CDC Splitter.** Link it with the above CDC Source.



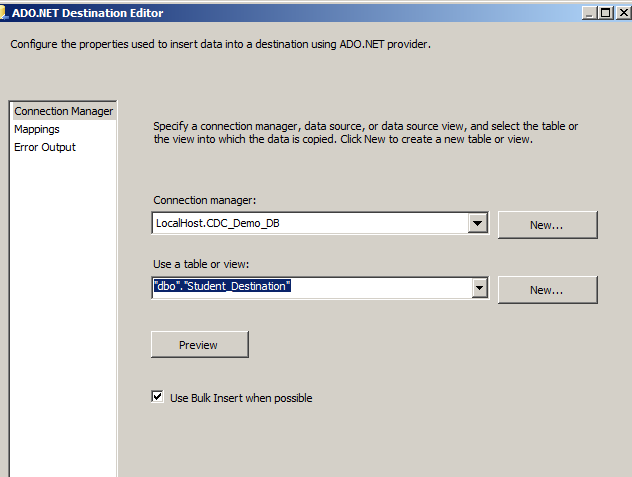
1. \*\* From Other Destinations put **ADO Net Destination**.



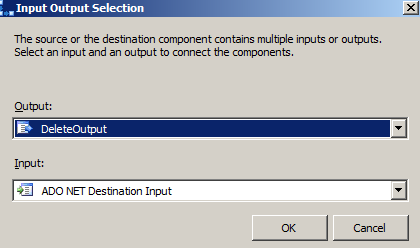
1. Link it with the CDC Splitter.
2. In the Input Output Selection window select **InsertOutput** from Output: drop down.



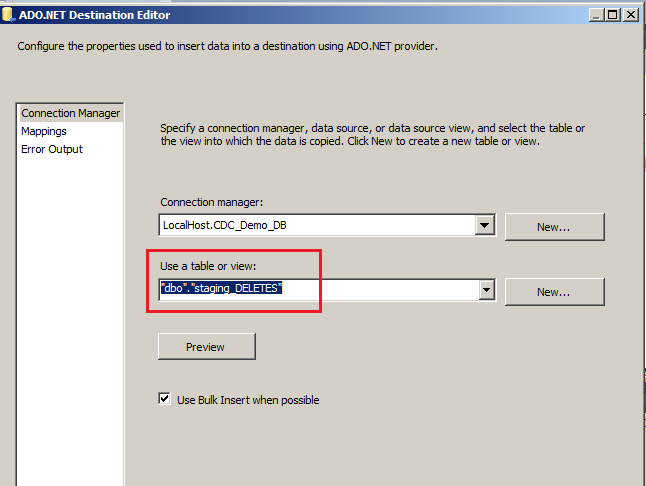
1. Open ADO.Net Destination Editor



1. Activate its **Mappings** of columns.
2. Put 2nd ADO Net Destination.
3. Link it with CDC Splitter.
4. In the Input Output Selection window select **DeleteOutput** from the Output list:

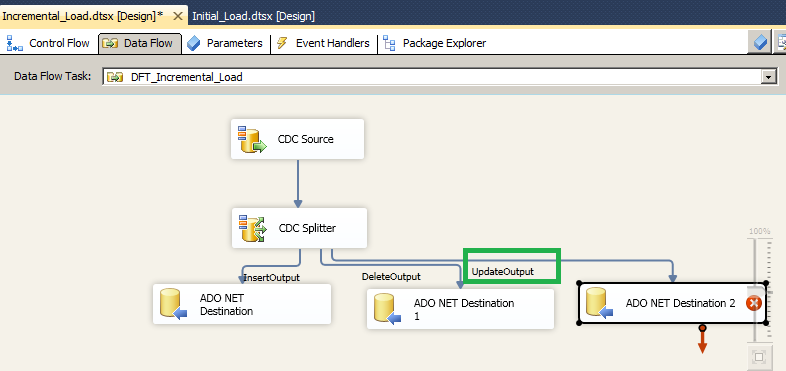


1. Open the editor of this 2nd ADO.Net Destination.

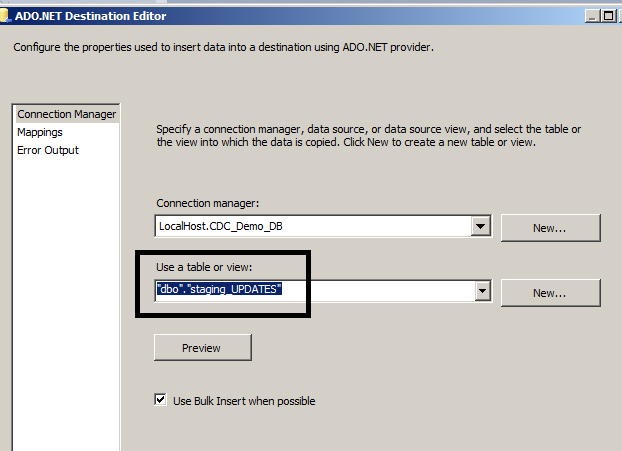


Note 🡪 Staging Delete table is selected.

1. Activate its **Mappings** of columns.
2. Put 3rd ADO.Net Destination
3. Link it with CDC Splitter.
4. \*\* Note 🡪 By default UpdateOutput selection (which is the remaining one) gets to this destination.



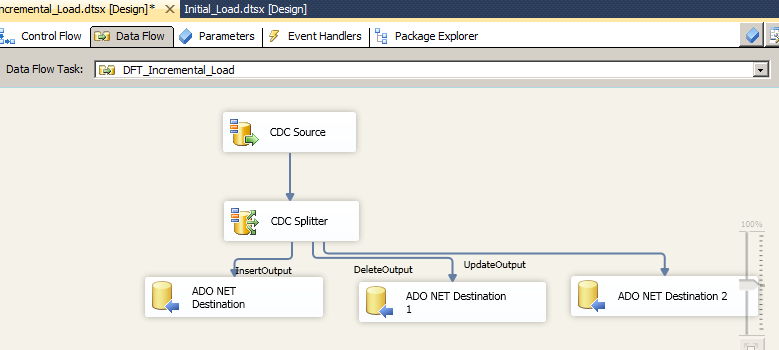
1. Open the editor of this 3rd Destination.



Note 🡪 The staging update table is selected.

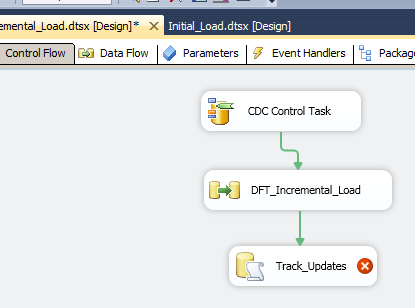
1. Activate its **Mappings** of columns.

So far, the Data flow has the following look:

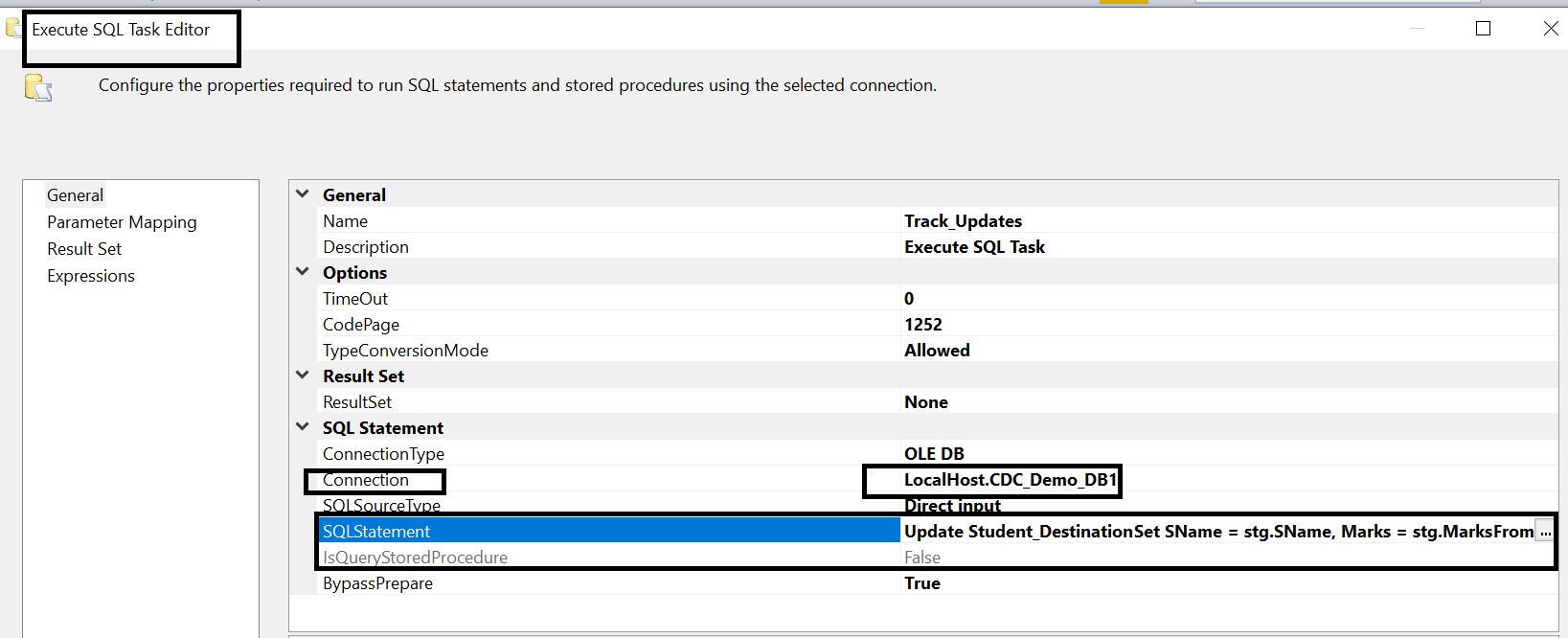


1. Put EST below the DFT for refreshing updates from staging table into Destination table.

Rename the EST as **Track\_Updates**



Provide the Connection Manager details.



1. SQL Statement of this EST will be:

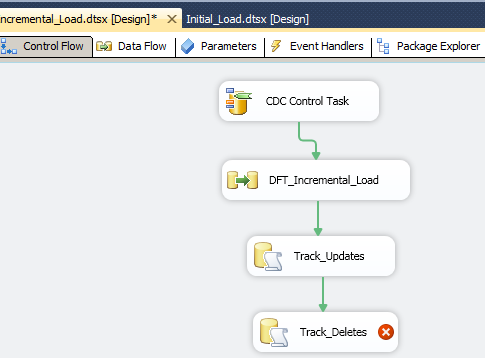
**Update Student\_Destination**

**Set SName = stg.SName, Marks = stg.Marks**

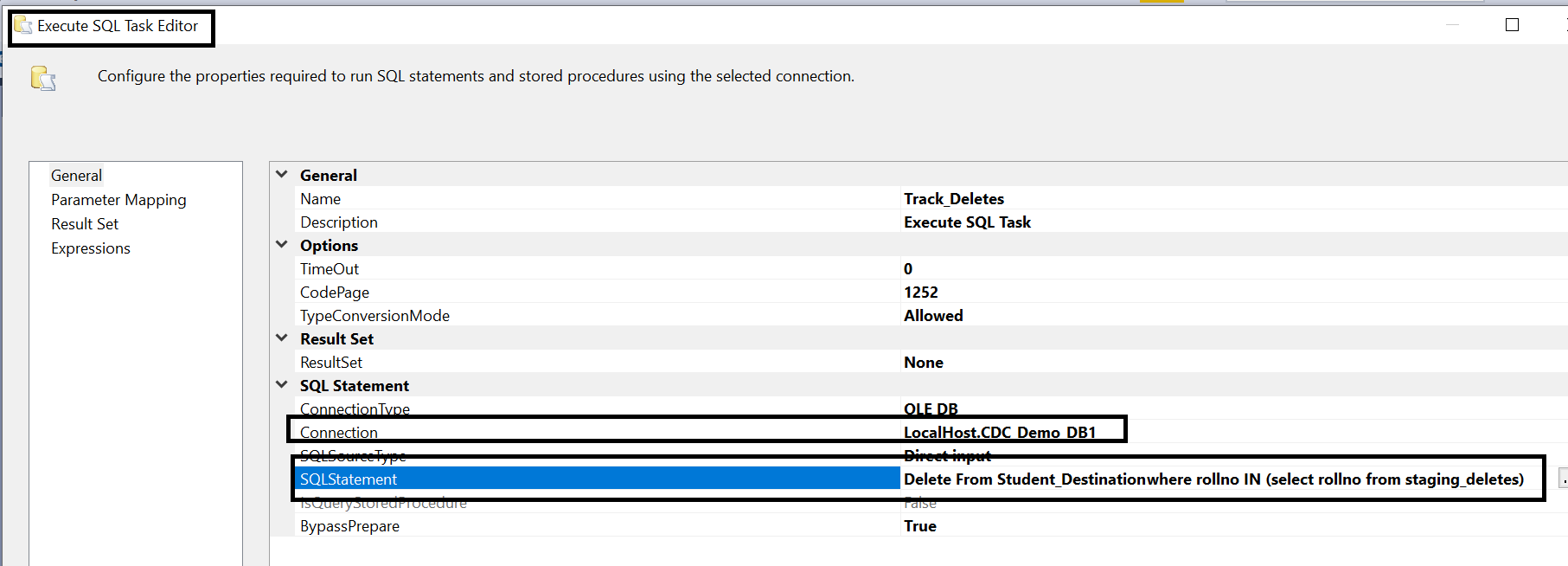
**From Student\_Destination, Staging\_Updates stg**

**where Student\_Destination.Rollno = stg.Rollno**

1. Put another EST. Rename it as **Track\_Deletes**



Provide the Connection Manager details.

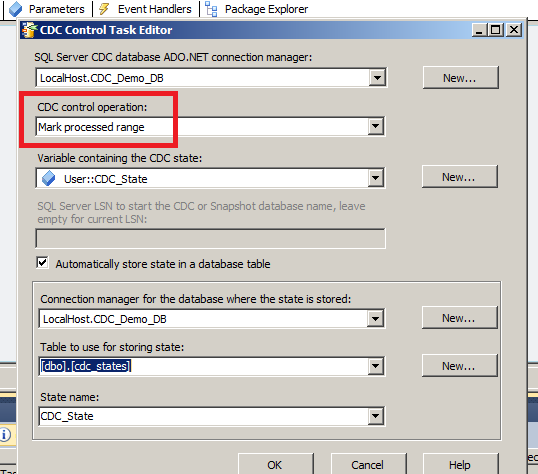


1. SQL Statement of this 2nd EST will be:

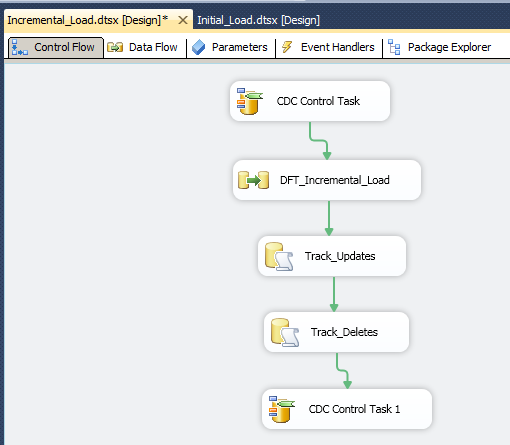
**Delete From Student\_Destination**

**where rollno IN (select rollno from staging\_deletes)**

1. Now put **CDC Control task** after the 2nd EST of Track\_Deletes
2. Link it with Track\_Deletes EST.
3. Open its editor



The final look of the Control Flow will be as shown below:



1. **Save the project.**
2. Let us check the source table, destination table, staging tables and cdc\_states table in SSMS

UseCDC\_Demo\_DB

go

Select\*FromStudent

go

Select\*fromStudent\_Destination

go

Select\*fromStaging\_Updates

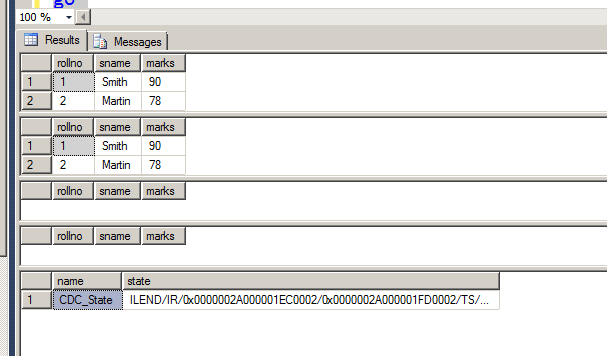
go

Select\*fromStaging\_Deletes

go

Select\*fromcdc\_states

go

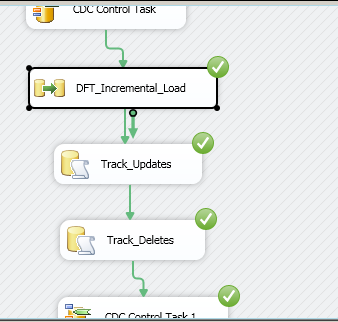


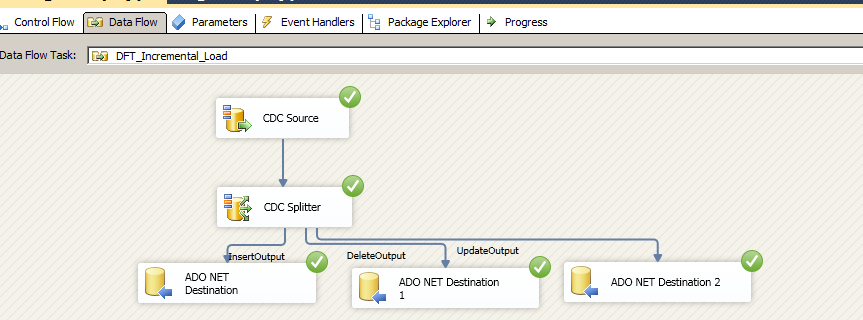
Note:

* The source & destination tables are having same records.
* Staging tables are empty
* CDC\_States table was earlier populated.

1. Run the 2nd package, i.e. Incremental\_Load

There is no effect:





Stop debugging.

**Explanation of second package:**

A second SSIS package is used to perform the incremental load. It consists of five tasks.

The **first task** is a **CDC Control task** that's used to get the processing range.

The processing range isn't a time range but rather a range of LSNs that contain the changes to be loaded. In the initial load package, the maximum LSN was written into the CDC\_State variable and then persisted in the cdc\_states table. This marks where the CDC Source task needs to start reading changes. When the CDC Control task runs, the end LSN is obtained using the same logic as the sys.fn\_cdc\_get\_max\_lsn function. The end LSN value is stored in the CDC state table and used for the CDC source in the Data Flow task.

The **second task** in the incremental load package is a **Data Flow task** that loads the staging tables. It uses a CDC source as the first component. An ADO.NET connection to the source database must be used with the CDC source.

After the connection is selected, you can add a CDC-enabled table to the source. To get only net changes, you must select Net from the CDC processing mode drop-down list. The Net option is only available if you used the supports\_net\_changes = 1 parameter when running the sys.sp\_cdc\_enable\_table stored procedure to enable CDC on the table. The CDC source pulls the data in the same manner as the sys.fn\_cdc\_get\_net\_changes\_<capture\_instance> function.

The from\_lsn and to\_lsn values are parsed out of the CDCState value and used to determine the range of changes that will be pulled from the cdc.Sales\_SalesOrderHeader\_CT table. (CDCState is a string value that's used throughout the package.)

The data retrieved by the CDC source is fed into a CDC splitter so that inserts, deletes, and updates are handled separately.

The **third task** is **CDC splitter** is a customized Conditional Split task that uses the \_$operation field to send different operations to different outputs. In this example, each of the splitter's outputs is sent to a separate ADO.NET data destination. If you use a different type of data destination (e.g., OLE DB, ODBC) you might run into data typing problems because the CDC source is required to use an ADO.NET connection manager.

There's no configuration needed for the CDC splitter, because it automatically creates three separate outputs for the different operations (insert, update, and delete). Each of the data flow destinations points to the corresponding staging table. In addition to the source data, there are three fields available from the CDC source: \_\_$start\_lsn, \_\_$operation, and \_\_$update\_mask. You could record any of these columns for historical purposes, but for this example, let's map the \_$operation column from the CDC source to the ChangeType column in the staging tables (1=delete, 2 = insert, and 4= update). Once the staging tables are loaded, the Data Flow task is complete.

The CDC splitter directly is adding new records in the destination table, but updated & deleted records from source are added in the staging table.

The **next 2 tasks (fourth & fifth)** are **Execute SQL Tasks**, for updating and deleting records of destination table by referring data of staging tables.

To finish the process, **a last task (sixth task)** i.e. **CDC Control task** marks the LSNs that have been processed. You configure this task by selecting the *Mark processed range* option in the *CDC control operation* drop-down list. This updates the persisted CDC state in the cdc\_states table so that when the package runs the next time, the starting LSN can be calculated.

The last task in the incremental load package is an Execute SQL task. It runs the following code to mark the staging records as processed in the ODS target table:

UPDATE Stage\_SalesOrderHeader\_Insert SET ProcessedToODS = 1

1. **Crosscheck Update:** Change a record from the source table.

Let us change Smith’s marks from 90 to 95.

UseCDC\_Demo\_DB

go

UpdateStudent

setmarks=95

whererollno=1-- Smith

go

Confirm the status in both the tables:

UseCDC\_Demo\_DB

go

Select\*FromStudent

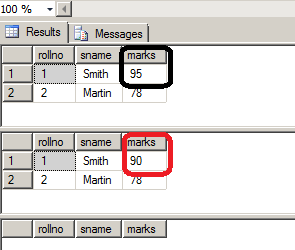
go

Select\*fromStudent\_Destination

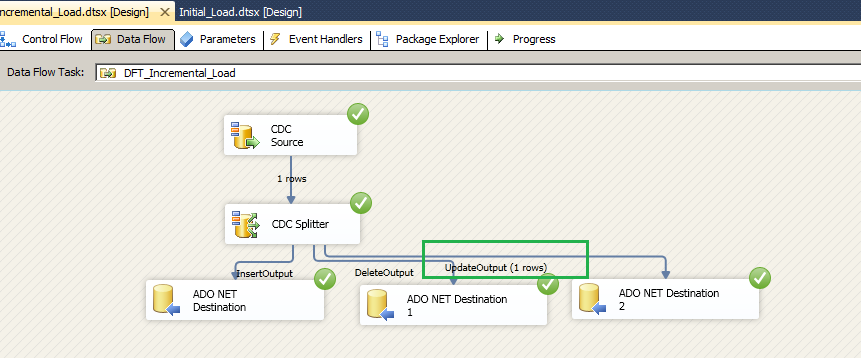
go

Select\*fromStaging\_Updates

go



1. Run the Incremental\_Load Package again:



Note 🡪 In the Data Flow Tab 1 row updated remark comes for the UpdateOutput

1. Stop Degugging:
2. Confirm changes in SSMS:

UseCDC\_Demo\_DB

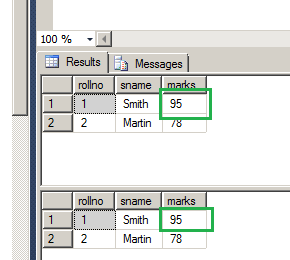
go

Select\*FromStudent

go

Select\*fromStudent\_Destination

go



1. **Cross check Insert:**
2. In SSMS add one row in Student table:

InsertintoStudent (Rollno, Sname,Marks)

Values(3, 'King',99)

Go

Check both tables before running the package:

UseCDC\_Demo\_DB

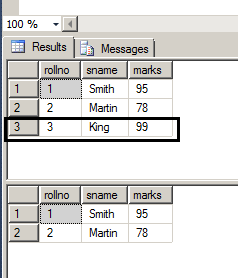
go

Select\*FromStudent

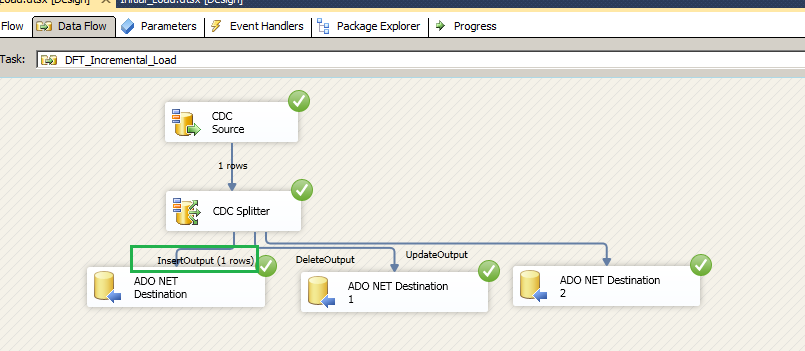
go

Select\*fromStudent\_Destination

go



1. Run the Incremental load package.



Note 🡪 The remark 1 row for InsertOutput

Stop debugging.

1. Confirm the status in SSMS

UseCDC\_Demo\_DB

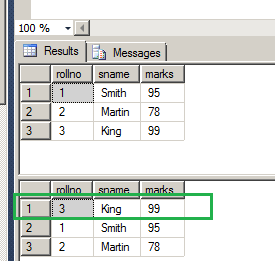
go

Select\*FromStudent

go

Select\*fromStudent\_Destination

go



1. **Crosscheck Delete**
2. In SSMS delete the record of Martin, .i.e. rollno 2

DeleteFromStudent

whererollno=2

go

1. Confirm the status in SSMS

UseCDC\_Demo\_DB

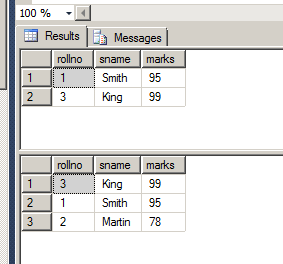
go

Select\*FromStudent

go

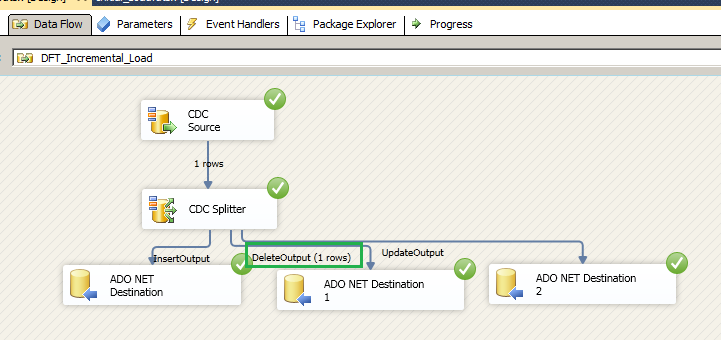
Select\*fromStudent\_Destination

go



1. Again run the Incremental Load package.

Notice one row in DeleteOutput of DFT



Stop debugging.

1. Confirm the status in SSMS

UseCDC\_Demo\_DB

go

Select\*FromStudent

go

Select\*fromStudent\_Destination

go

