

White Paper



Tutorial: Change Data Capture (CDC)

Virendra Wadekar and Phaneendra Babu Subnivis

Introduction

This tutorial shows how to enable Change Data Capture (CDC) on database and table level in SQL Server 2008. We will begin with a general overview of CDC concepts. Next, we will do a step-by-step walkthrough of enabling the CDC on the database level and table level. We will look at the CDC functions available to extract the changed information from the table. After this, the tutorial will move further in advanced concepts and explain the usage of CDC in SQL Server Integration Services (SSIS) Package. This package shows how the data can be extracted from OLTP databases and loaded incrementally in data ware house. This tutorial is intended to showcase CDC capability and it's usage in ETL (Extract – Transform – Load) scenarios.

For more information, Contact askus@infosys.com

Prerequisites

To understand and execute this tutorial you should be familiar with:

- Basic SQL programming
- Knowledge of SSIS Package
- Understanding of using the Business Intelligence Development Studio (BIDS)

The technologies required to execute this tutorial are:

- SQL Server 2008
- BIDS
- SQL Agent service should be running

Overview of CDC

This section introduces the CDC feature in SQL Server 2008, which helps in capturing the changes on the source database tables. CDC keeps track of inserts and updates and makes this changed data available in the changed tables. It further provides CDC functions which can be used to query those changed tables and track the modifications.

In earlier versions of SQL Server, loading the data marts, data warehouse (DW) was not so easy. The source table keeps changing over time and the DW and data marts based on this source DB need to be refreshed. The earlier approaches of timestamp columns, triggers or complex join queries increase the complexity in this process and have an impact on performance.

The Change Data Capture feature (CDC) in SQL Server 2008 offers effective solution to the above problems. This is a generic database engine feature. It captures the insert, update and delete activities on the database tables and makes those changes available in the relational format. This tutorial demonstrates the usage of the CDC feature along with SSIS package to load the destination database in incremental fashion.

The Figure 1: CDC components show the different components involved in the CDC. The “OrderInfo” is a table that needs to be tracked for the changes. When the CDC is enabled on this table, a new table called “OrderInfo change” table is created. A capture job monitors the Database log and captures the change information and puts it in the change table. The clean up job is created by CDC to clean the change table after a certain interval. There are some CDC functions provided to query the change tables to get the changed data, which is shown as CDC APIs in the snapshot. CDC metadata tables are created to check the CDC configuration on various databases and tables. All the objects are created under a schema called CDC.

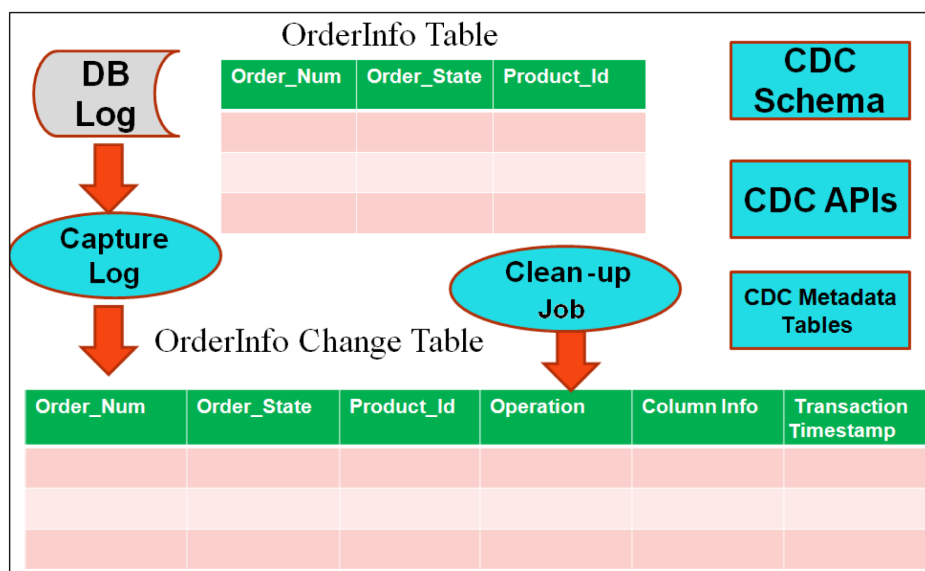


Figure 1: CDC components

Usage scenario

CDC can be useful in the following scenarios:

- Incremental DW loading
- Building the history data
- Synchronizing databases in required application scenarios

Advantages

- Avoids the complexity of writing triggers, time stamp columns and complex join queries
- Has better performance than the earlier mechanism as it does not directly work with source tables
- Works asynchronously on the logs to capture the changes in the tracking table.
- Provides an automatic way of purging the tracking tables

What you should also know

- It gives historical information on the changed table and hence storage requirements is more
- Same as replication, you cannot use Truncate Option on the table on which CDC is enabled
- Additional maintenance for the capture, clean up jobs on the change tracking table

Tutorial

Following steps illustrate how to implement CDC feature in SQL Server 2008:

Step 1: CDC Configuration

1. Create a database called “CDCFROMSSISPOCSRC” using the code snippet below:

```
USE MASTER
IF EXISTS (SELECT NAME FROM SYS.Databases WHERE Name = 'CDCFROMSSISPOCSRC')
BEGIN
    --If Exists Drop the Database
    DROP DATABASE CDCFROMSSISPOCSRC
END
GO
--Create the Database Reuired to run the Tutorial
Create DataBase CDCFROMSSISPOCSRC
GO
```

2. Create the table required for this tutorial using the code snippet below:

```
USE CDCFROMSSISPOCSRC

CREATE TABLE [dbo].[Product](
    [ProductId] [int] IDENTITY(1,1) NOT NULL,
    [ProductName] [nvarchar](250) NOT NULL,
    CONSTRAINT [PK_Product] PRIMARY KEY CLUSTERED (
        [ProductId] ASC )
GO
```

3. Enable the CDC on the database and the “Product” table using the code snippet below. As mentioned earlier, we are using CDC metadata tables here. Start the SQL agent service before running the script below.

```
--Enable the CDC on the source Database i.e. CDCFROMSSISPOCSRC
--Just to show following statement uses the metadata for checking whether CDC is
--Already enabled on this database.

DECLARE @isCDCEnabled int
select @isCDCEnabled = is_cdc_enabled from sys.databases
    WHERE name = 'CDCFROMSSISPOCSRC'
print @isCDCEnabled
IF @isCDCEnabled <> 1
```

```
BEGIN
    EXEC sys.sp_cdc_enable_db
END

GO

--Once the CDC is enabled at the database level enable it at table level.
--Lets enable it on the Product table.

EXECUTE sys.sp_cdc_enable_table
    @source_schema = N'dbo'
    , @source_name = N'Product'
    , @role_name = N'cdc_Admin',
    @capture_instance=N'dbo_Product',
    @supports_net_changes = 1

GO
```

After following the above mentioned steps, your Management Studio should show all the CDC objects as shown in the attached snapshot Figure 2: Management Studio.

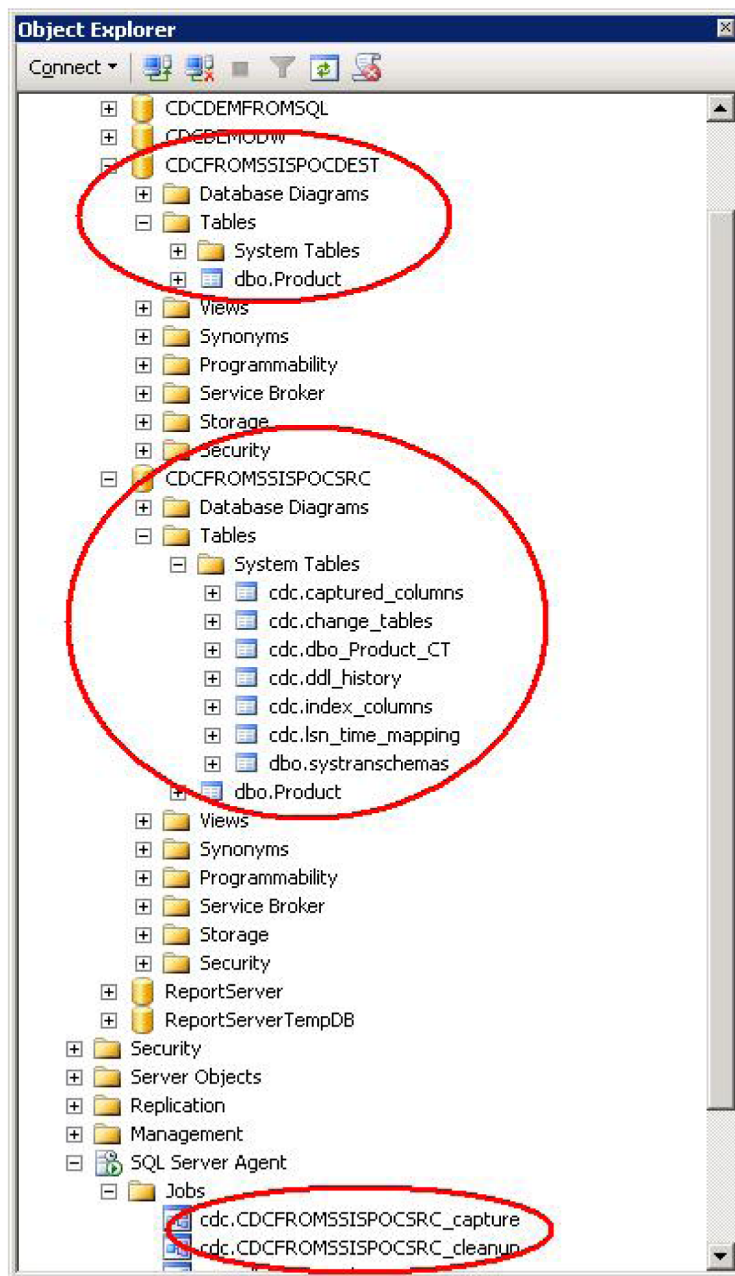


Figure 2: Management Studio

Step 2: Creating data in the “Product” Table

Use the following code snippet to create the data in the “Product” table:

```
INSERT INTO dbo.Product Values('EVF 2005 Elevator');  
INSERT INTO dbo.Product Values('EVF 2005 Elevator 1');  
INSERT INTO dbo.Product Values('EVF 2005 Elevator 2');
```

```

INSERT INTO dbo.Product Values('EVF 2005 Elevator 3');
INSERT INTO dbo.Product Values('EVF 2005 Elevator 4');
INSERT INTO dbo.Product Values('EVF 2005 Elevator 5');
INSERT INTO dbo.Product Values('EVF 2005 Elevator 6');
INSERT INTO dbo.Product Values('EVF 2005 Elevator 7');
INSERT INTO dbo.Product Values('EVF 2005 Elevator 8');
INSERT INTO dbo.Product Values('EVF 2005 Elevator 9');
INSERT INTO dbo.Product Values('EVF 2005 Elevator 10');

UPDATE dbo.Product SET ProductName = ProductName + '1 st Modification'
WHERE ProductId > 2 AND ProductId < 5;

DELETE FROM dbo.Product WHERE ProductId > 9

```

Step 3: Using CDC functions to extract the change table data

Use the following code snippet to get hold on the changed data. The code sets the @starttime to yesterday and the @endtime to Getdate(). This logic is put just to ensure that the Capture job is run and the data is available in the change table. Using the time we are getting the LSN (Log Sequence number) and passing it to the CDC functions. There are two functions which give you net changes and all changes. In step 4, actual snapshots of the function's output are attached. The CDC functions for getting the LSN have different parameters. Please look at the BOL for the same.

```

DECLARE @from_lsn BINARY(10)
DECLARE @to_lsn BINARY(10)
DECLARE @isnull int = 0
DECLARE @starttime DateTime
DECLARE @endtime DateTime

SET @starttime = DATEADD(dd,-1,Getdate())
SET @endtime = Getdate()

```

```

SET @from_Isn = sys.fn_cdc_map_time_to_Isn('smallest greater than',
@starttime)
SET @to_Isn = sys.fn_cdc_map_time_to_Isn('largest less than or equal',
@endtime)
IF @from_Isn IS NOT NULL AND @to_Isn IS NOT NULL
BEGIN

SELECT * FROM
cdc.fn_cdc_get_net_changes_dbo_product(@from_Isn,@to_Isn,'all')

--SELECT * FROM cdc.fn_cdc_get_all_changes_dbo_product ( @from_Isn ,
@to_Isn , 'all' )
END

```

Step 4: Verifying the changed data

The “fn_cdc_get_net_changes_dbo_product function” returns the data as shown below.

__\$start_Isn	__\$operation	ProductId	ProductName
0x0000001C000000830014	2	1	EVF 2005 Elevator
0x0000001C000000870004	2	2	EVF 2005 Elevator 1
0x0000001C000000910006	2	3	EVF 2005 Elevator 21 st Modification
0x0000001C000000910006	2	4	EVF 2005 Elevator 31 st Modification
0x0000001C0000008A0004	2	5	EVF 2005 Elevator 4
0x0000001C0000008B0004	2	6	EVF 2005 Elevator 5
0x0000001C0000008C0004	2	7	EVF 2005 Elevator 6
0x0000001C0000008D0004	2	8	EVF 2005 Elevator 7
0x0000001C0000008E0004	2	9	EVF 2005 Elevator 8

The operation column displays whether it was insert, update or delete, value 2 being for insert. Since we have asked for only the net changes we get the net effect as insert for all rows, even though we have updated and deleted some records.

The “fn_cdc_get_all_changes_dbo_product function” returns the data as shown below:

__\$start_lsn	__\$seqval	__\$operation	ProductId	ProductName
0x0000001C000000830014	0x0000001C000000830013	2	1	EVF 2005 Elevator
0x0000001C000000870004	0x0000001C000000870003	2	2	EVF 2005 Elevator 1
0x0000001C000000880004	0x0000001C000000880003	2	3	EVF 2005 Elevator 2
0x0000001C000000890004	0x0000001C000000890003	2	4	EVF 2005 Elevator 3
0x0000001C0000008A0004	0x0000001C0000008A0003	2	5	EVF 2005 Elevator 4
0x0000001C0000008B0004	0x0000001C0000008B0003	2	6	EVF 2005 Elevator 5
0x0000001C0000008C0004	0x0000001C0000008C0003	2	7	EVF 2005 Elevator 6
0x0000001C0000008D0004	0x0000001C0000008D0003	2	8	EVF 2005 Elevator 7
0x0000001C0000008E0004	0x0000001C0000008E0003	2	9	EVF 2005 Elevator 8
0x0000001C0000008F0004	0x0000001C0000008F0003	2	10	EVF 2005 Elevator 9
0x0000001C000000900004	0x0000001C000000900003	2	11	EVF 2005 Elevator 10
0x0000001C000000910006	0x0000001C000000910002	4	3	EVF 2005 Elevator 21 st Modification
0x0000001C000000910006	0x0000001C000000910004	4	4	EVF 2005 Elevator 31 st Modification
0x0000001C000000930006	0x0000001C000000930002	1	10	EVF 2005 Elevator 9
0x0000001C000000930006	0x0000001C000000930005	1	11	EVF 2005 Elevator 10

If you closely look at the ProductId 3 and 4 in the table, you will notice that it is appearing two times in the list with operation 2 and 4 as we had first inserted the record and then updated the same. So you can get both the old and new value. The product id 10 and 11 are listed with operation as 1, which is delete done on these two rows. One more column returned by both the above functions is update mask and is not shown in the above table. It gives information on the columns updated. You can look at books online to get more information on this.

Step 5: SSIS Package Creation

The SSIS package needs to have four major tasks as described below:

1. “Execute SQL task” to extract the time window between which the change table data is to be extracted. In this, the start time is set to the previous run end time and the end time is set to the next duration for which you want to extract the data
2. A “for loop container” can be built to check whether the changed data is available in the changed table for the specific period. Run through the loop until the changed data is available
3. Once the changed data is available use the “Data Flow Task” to extract the information using a “conditional split transformation”
4. Use “Execute SQL task” to set the start and end time for the next run

This tutorial explains only the “Data Flow Task” as the other tasks are pretty trivial.

Step 6: Data Flow Task

1. In the Data Flow task, call a stored procedure using the “OLEDB Source” task. The stored procedure can be created as say usp_GetChangedData ?,?. The start time and end time variables can be mapped to the ?,? Parameters. The stored procedure can be created using the code snippet given in Step 3
2. Then define the “Conditional Split” task and check the Operation column in the conditions. Depending on the conditions divert the data flow to either Insert, Update or Delete.
3. The insert, update and delete can be handled using the OLEDB command.
4. The configured Data Flow task should look as attached in the Figure 3: DataFlow Task.

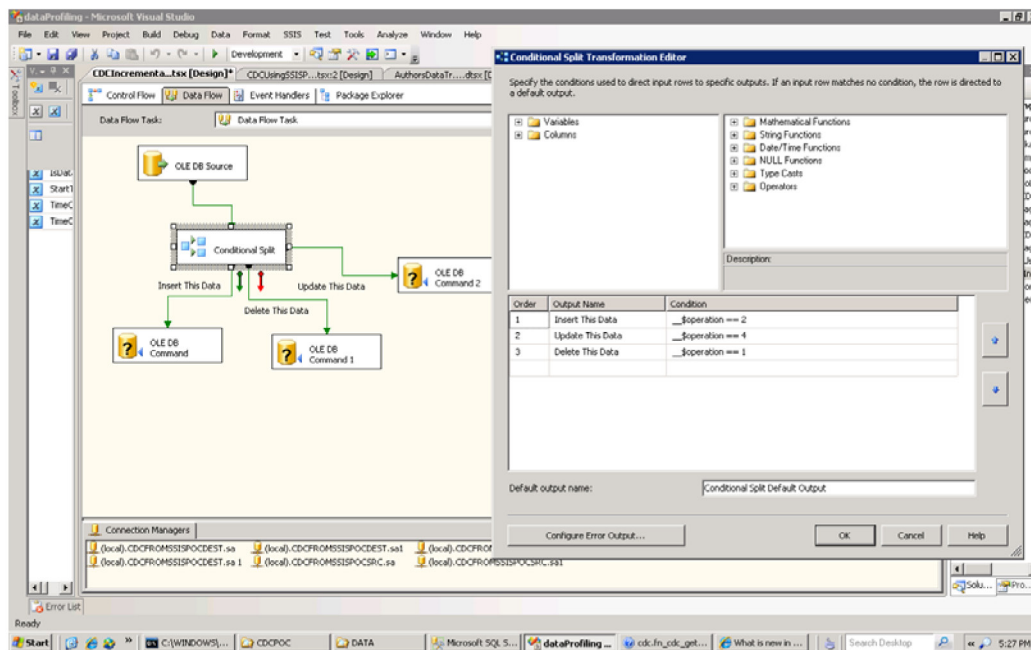


Figure 3: DataFlow Task

5. Once this is done, the package can be tested and scheduled using SQL agent job. The schedule of the job could be defined as per the requirement.

Summary

As mentioned earlier, you can use CDC in place of complex mechanisms like Triggers, Join Queries, Time Stamp columns to load the data ware house incrementally.



For more information, contact askus@infosys.com

About Infosys

Many of the world's most successful organizations rely on Infosys to deliver measurable business value. Infosys provides business consulting, technology, engineering and outsourcing services to help clients in over 30 countries build tomorrow's enterprise.

For more information about Infosys (NASDAQ:INFY), visit www.infosys.com.