Steve Stedman
Freelance SQL Server Consultant
http://stevestedman.com

TSQL - Change Data Capture

You have a need to keep track of all insert, update and delete actions on a table, or multiple tables. As you consider solutions, you might think about using a trigger, however triggers have their own baggage. You consider using the OUTPUT clause to log to a changes table, but then realize that the output clause cant be enforced.

Then the SQL Server feature called Change Data Capture comes into play. CDC is a SQL Server Feature that monitors the transaction log, looking for changes to specific tables, when the changes are discovered, they are then written into a Change Table that can then be queried to find out what was changed and when it was changed.

Sample Code

Lets take a look, to start with, I create a database called [DemoCDC] to use for the demo. That database contains a single table called [History] that you may recognize from the Week 3 Database Corruption Challenge. Following that are a few insert statements to just start with some data in the table.

Now to check what is in the table...

Six rows with an id of 1 to 6, all with the same dateAdded.

Now lets set up the Change Data Capture

Which shows the following output:

At this point, Change Data Capture is now enabled on the [dbo].[People] table. This can be confirmed by checking the [is_tracked_by_cdc] column in the sys.tables view.

The value of 1 in the [is_tracked_by_cdc] indicates that we have configured things correctly on the People table.

Once CDC is enabled, you will also notice several additional tables in the [cdc] schema in the System Tables folder:

At this point, lets take a look at the [cdc].[dbo_People_CT] table which is the change tracking table that goes along with the People table.

The first 5 columns (shown in red) are the same for all tracking tables, the remaining columns (marked in blue) vary for each table being tracked and will match the columns in the source table.

At this point, there is nothing in the [cdc].[dbo_People_CT] table. Lets make a few changes.

You can now see that the [People] table has 6 people, but the row with an id of 2 is now gone, and id 5 has been renamed to Quinn, and row 7 contains Alex. Hopefully that is exactly what you would have expected.

Now lets look at the contents of the [cdc].[dbo_People_CT] table.

The column called __$operation can have one of the following values

- 1 = Delete Statement
- 2 = Insert Statement
- 3 = Value before Update Statement
- 4 = Value after Update Statement

The first row with a operation value of 2 indicates that an insert statement was run, followed by the operation of a 1 for the delete statement. The final two rows with an operation of 3 and 4 show that the id 5 previously had the value of Mary, which was then changed to Quinn.

Performance

Unlike a trigger that occurs when an action is performed, the CDC utilizes the transaction log, and therefore adds no additional load at the time the change occurs. A moment later there will be some additional work as the log file is scanned, and then as the insert occurs into the [dbo_People_CT] table. This does add load to the SQL Server, but no additional load during the transaction that inserts to the [People] table in this example.

Turning Off CDC

Before we turn off CDC, we can first check to see if it is enabled with the following SQL Statement:

Which shows that CDC is enabled on the [People] table.

Now to turn it off.

To disable, or turn off everything that we have done so far, just run the following TSQL:

Availability

SQL Server 2008, 2008R2, 2012, 2014 or newer - only on the Enterprise, Developer or Evaluation Editions.

Summary

At this point, you are ready to give Change Data Capture a try in your own test or development environment. This is a great way to track changes for any given table.

Enjoy!

-Steve Stedman