CDC Change Table some columns have NULL value

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:30 AM PST

Contents

Issue

The customer might notice that some values on the change table "tablename_CT" will appear as NULL for some columns.

For example, the customer selects from a table that they own:

```
SELECT * FROM cdc.myschema_schema_CT WHERE id = 'some_GUID_or_ID'
SELECT * FROM cdc.dbo_tab1_CT WHERE id = 1234
```

And the value for one or more columns has a NULL value, despite of changes occurring on that column.

Investigation / Analysis

Possible cause 1 - Computed columns

The issue will occur if the affected source table column or columns are computed columns. This is by design - see the Known limitations and issues with CDC for an explanation:

> **Computed columns**
> CDC does not support the values for computed columns even if the computed column is defined as persisted. Computed columns that are included in a capture instance always have a value of NULL. This behavior is intended, and not a bug.

Run the following query to see if there are computed columns on a table:

```
-- change the "tablename" value with the schema and name of the source table
SELECT
    SCHEMA_NAME(o.schema_id) AS schema_name,
    OBJECT_NAME(c.object_id) AS table_name,
    c.name AS column_name,
    TYPE_NAME(user_type_id) AS data_type,
    definition
FROM
    sys.computed_columns c
    INNER JOIN sys.objects o ON o.object_id = c.object_id
WHERE
    c.object_id = OBJECT_ID('dbo.tablename')
--  OBJECT_NAME(c.object_id) = 'tablename'
ORDER BY
    schema_name, table_name, column_id;


schema_name  table_name  column_name  data_type  definition
------------ ----------- ------------ ---------- -----------
dbo          tab1        c3           int        ([c1]+[c2])
```

(the sample output matches the example table from the Mitigation section below)

Possible cause 2 - Previous schema change

The issue might also occur if the tracked column has been deleted from the source table. The issue will still occur if the affected column had been deleted and re-added from/to the source column. See the details in Handling changes to source table:

> To accommodate a fixed column structure change table, the capture process responsible for populating the change table will ignore any new columns that are not identified for capture when the source table was enabled for change data capture. **If a tracked column is dropped, null values will be supplied for the column in the subsequent change entries**. However, if an existing column undergoes a change in its data type, the change is propagated to the change table to ensure that the capture mechanism does not introduce data loss to tracked columns. The capture process also posts any detected changes to the column structure of tracked tables to the `cdc.ddl_history` table. Consumers wishing to be alerted of adjustments that might have to be made in downstream applications, use the stored procedure `sys.sp_cdc_get_ddl_history`.

To detect any mismatch or missing columns on the source and change tables, run the following DMV queries on the affected database:

```
-- source table columns
SELECT
    object_name(object_id) AS 'object_name',
    object_id,
    name AS 'column_name',
    column_id,
    TYPE_NAME(system_type_id) AS 'data_type',
    max_length,
    is_computed
FROM sys.columns
WHERE object_id = object_id('dbo.t1')

-- captured columns
SELECT object_name(object_id) AS 'object_name', * from cdc.captured_columns
```

Sample output of a functional configuration where only columns ID and c1 are tracked:

```
object_name  object_id  column_name  column_id  data_type  max_length  is_computed
------------ ---------- ------------ ---------- ---------- ----------- -----------
t1           50099219   ID           1          int        4           0
t1           50099219   c1           2          varchar    100         0
t1           50099219   c2           3          nvarchar   200         0

object_name  object_id  column_name  column_id  column_type  column_ordinal  is_computed  masking_function
------------ ---------- ------------ ---------- ------------ --------------- ------------ ----------------
dbo_t1_CT    82099333   ID           1          int           1              0            NULL
dbo_t1_CT    82099333   c1           2          varchar       2              0            NULL
```

Now drop the c1 column and re-add it immediately with the same specification:

```
ALTER TABLE dbo.t1 DROP COLUMN c1;
ALTER TABLE dbo.t1 ADD c1 VARCHAR(100);
```

Sample output after the change using the same queries as above - note that the only difference is the `column_id` of column c1 and that its new value no longer matches the column_id on `cdc.captured_columns` :

```
object_name  object_id  column_name  column_id  data_type  max_length  is_computed
------------ ---------- ------------ ---------- ---------- ----------- -----------
t1           50099219   ID           1          int        4           0
t1           50099219   c2           3          nvarchar   200         0
t1           50099219   c1           4          varchar    100         0

object_name  object_id  column_name  column_id  column_type  column_ordinal  is_computed  masking_function
------------ ---------- ------------ ---------- ------------ --------------- ------------ ----------------
dbo_t1_CT    82099333   ID           1          int           1              0            NULL
dbo_t1_CT    82099333   c1           2          varchar       2              0            NULL
```

Mitigation

Possible cause 1 - Computed columns

This is a known limitation as stated in Known limitations and issues with CDC.

For a workaround, do not even think about updating the _CT change table directly. The customer **must not update** the change table directly as it might break the CDC environment.

A valid workaround is, when querying the change table, that you apply the same compute function as it is defined on the source table.
For example, assuming that the definition of the source table is:

```
CREATE TABLE tab1
(
   c1 int primary key,
   c2 int,
   c3 as c1 + c2
)
```

When querying the change table, use the same computed column function/definition as it is on the source table:

```
SELECT c1, c2, c3 = c1 + c2 FROM cdc.dbo_tab1_CT
```

Possible cause 2 - Previous schema change

Schema changes of existing tracked columns are replicated to the CDC consumers and the _CT tables. For example, using Alter Table to change a source column from VARCHAR(10) to VARCHAR(100) will execute corresponding changes to the CDC tables and the target table.

The supported steps for adding columns are described in Handling changes to source table:

> Typically, the current capture instance will continue to retain its shape when DDL changes are applied to its associated source table. However, it is possible to create a second capture instance for the table that reflects the new column structure. This allows the capture process to make changes to the same source table into two distinct change tables having two different column structures. Thus, while one change table can continue to feed current operational programs, the second one can drive a development environment that is trying to incorporate the new column data. Allowing the capture mechanism to populate both change tables in tandem means that a transition from one to the other can be accomplished without loss of change data. This can happen anytime the two change data capture timelines overlap. When the transition is affected, the obsolete capture instance can be removed.

Public Doc reference

- Known limitations and issues with CDC
- CDC Change Data Capture Overview

Internal reference

IcM 319051498

How good have you found this content?