POOR MAN'S CHANGE DATA CAPTURE: USING DECODE

Kent Graziano, Denver Public Schools

Abstract

Do you have a need to automate an effective change data capture process in an Oracle database but you have non-Oracle source systems? Do your operational systems have no auditing or logging at all making change detection virtually impossible (without a bit by bit comparison)? Is your budget too small to afford third party tools or advanced Oracle options (or your staff is too inexperienced to implement them)? If so, this session is for you. In implementing a data vault model for our enterprise data warehouse at Denver Public Schools we learned how easy it was to do change data capture (CDC) against any dataset using good ol' DECODE in a view. This short technical session will show you the actual SQL code to use and explain how it works to detect changes in that data when compared to a data warehouse table. In addition I will show and explain the views we use to determine when completely new record appears in the source as well.

Starting Out

Before you can do change data capture, you first need an initial snapshot or baseline of the data in question. In the case of DPS, that snapshot is the initial view of the data stored in our data vault. The population of the data vault structure is quite simple. In the view example below, V_ADD_HUB_SCHOOLS, we add new rows of data to the table HUB_SCHOOLS by using a view that compares data in the data vault to data in the source table (ODS_OWNER.SCHOOLS) using a "not exists" sub-query. If the data does not already exist in the data vault table, it appears in the view and can then be inserted into the data vault.

```
CREATE OR REPLACE FORCE
VIEW DV_OWNER.V_ADD_HUB_SCHOOLS
(SCHOOL_NUMBER,
LOAD_DTS,
REC_SRC)
AS
SELECT
SCH.SCHOOL_NUMBER SCHOOL_NUMBER
,PD.LOAD_DTS LOAD_DTS
,SCH.CREATED_BY REC_SRC
FROM ODS_OWNER.SCHOOLS SCH
     ,PROCESS_DATES PD
WHERE NOT EXISTS
(Select 1
From HUB_SCHOOLS HSCH
Where HSCH.SCHOOL_NUMBER = SCH.SCHOOL_NUMBER)
```

This approach assumes you have access to either the source system directly with select privileges or you have a staging area to select from. In our case, we use our Operational Data Store (ODS_OWNER) as a staging area from which to load the data vault. This allows us to run loads during the day without impacting the actual operational system.

That was the simple example: now for the more complex one. This view (V_ADD_SAT_SCHOOL_DETAILS) populates a detail (satellite) table with a number of columns from the source (ODS_OWNER.SCHOOLS) if there is a matching Hub record and no record exists yet in the detail table (SAT_SCHOOL_DETAILS).

```
 CREATE OR REPLACE FORCE VIEW DV_OWNER.V_ADD_SAT_SCHOOL_DETAILS
(SCHOOL_ID, LOAD_DTS, DISTRICT_AREA_CODE, ABBREVIATION, SCHOOL_NAME,
 SCHOOL_STATUS_FLAG, HOME_ROOM_PERIOD, PERIOD_ROTATION, END_DTS, REC_SRC)
AS
SELECT HSCH.SCHOOL_ID SCHOOL_ID
,PD.LOAD_DTS LOAD_DTS
,SCH.DISTRICT_AREA_CODE DISTRICT_AREA_CODE
```

```
,SCH.ABBREVIATION ABBREVIATION
,SCH.SCHOOL_NAME SCHOOL_NAME
,SCH.STATUS_FLAG SCHOOL_STATUS_FLAG
,SCH.HOME_ROOM_PERIOD HOME_ROOM_PERIOD
,SCH.PERIOD_ROTATION PERIOD_ROTATION
,NULL
,SCH.CREATED_BY REC_SRC
FROM ODS_OWNER.SCHOOLS SCH
,HUB_SCHOOLS HSCH
,PROCESS_DATES PD
WHERE SCH.SCHOOL_NUMBER = HSCH.SCHOOL_NUMBER
AND NOT EXISTS
(SELECT 1
FROM SAT_SCHOOL_DETAILS SSD
WHERE SSD.SCHOOL_ID = HSCH.SCHOOL_ID
AND SSD.END_DTS IS NULL
);
```

Once this view is used to populate the data vault table, we will have rows against which we can evaluate change.

Capturing Changes

With a baseline in place, we can now use a view to determine if any of the data previously captured has changed. The view below (V_CDC_SAT_SCHOOL_DETAILS) does that evaluation for us by using a DECODE statement to compare the column in the source (ODS_OWNER.SCHOOLS) to the column in the data vault (SAT_SCHOOL_DETAILS).

```
CREATE OR REPLACE FORCE VIEW DV_OWNER.V_CDC_SAT_SCHOOL_DETAILS
(SCHOOL_ID, LOAD_DTS, ABBREVIATION,
  DISTRICT_AREA_CODE, SCHOOL_NAME,
  SCHOOL_STATUS_FLAG, HOME_ROOM_PERIOD,
  PERIOD_ROTATION, END_DTS, REC_SRC)
AS
SELECT DISTINCT SSD.SCHOOL_ID
,PD.LOAD_DTS LOAD_DTS
,SCH.ABBREVIATION ABBREVIATION
,SCH.DISTRICT_AREA_CODE DISTRICT_AREA_CODE
,SCH.SCHOOL_NAME SCHOOL_NAME
,SCH.STATUS_FLAG SCHOOL_STATUS_FLAG
,SCH.HOME_ROOM_PERIOD HOME_ROOM_PERIOD
,SCH.PERIOD_ROTATION PERIOD_ROTATION
,NULL
,SCH.CREATED_BY REC_SRC
FROM ODS_OWNER.SCHOOLS SCH
           ,HUB_SCHOOLS HSCH
          ,SAT_SCHOOL_DETAILS SSD
           ,PROCESS_DATES PD
WHERE SCH.SCHOOL_NUMBER = HSCH.SCHOOL_NUMBER
      AND HSCH.SCHOOL_ID = SSD.SCHOOL_ID
      AND SSD.LOAD_DTS <= PD.LOAD_DTS
      AND SSD.END_DTS IS NULL
      AND (SCH.ABBREVIATION is not NULL or
               SCH.SCHOOL_NAME is not NULL or
               SCH.STATUS_FLAG is not NULL)
     AND
(Decode (SSD.DISTRICT_AREA_CODE,
                SCH.DISTRICT_AREA_CODE, 1, 0) = 0
OR Decode (SSD.ABBREVIATION,
                      SCH.ABBREVIATION, 1, 0) = 0
OR Decode (SSD.SCHOOL_NAME,
                      SCH.SCHOOL_NAME, 1, 0) = 0
OR Decode (SSD.SCHOOL_STATUS_FLAG,
                      SCH.STATUS_FLAG, 1, 0) = 0
OR Decode (SSD.HOME_ROOM_PERIOD,
                      SCH.HOME_ROOM_PERIOD, 1, 0) = 0
OR Decode (SSD.PERIOD_ROTATION,
                      SCH.PERIOD_ROTATION, 1, 0) = 0)
AND NOT EXISTS
(SELECT 1
FROM SAT_SCHOOL_DETAILS SSD2
```

```
                      WHERE SSD2.SCHOOL_ID = SSD.SCHOOL_ID
                            AND SSD2.END_DTS IS NULL
                           AND Decode(SSD2.DISTRICT_AREA_CODE,
                                                SCH.DISTRICT_AREA_CODE,1,0)=1
                          AND Decode(SSD2.ABBREVIATION, SCH.ABBREVIATION,1,0)=1
                          AND Decode(SSD2.SCHOOL_NAME, SCH.SCHOOL_NAME,1,0)=1
                          AND Decode(SSD2.SCHOOL_STATUS_FLAG, SCH.STATUS_FLAG,1,0)=1
                          AND Decode(SSD2.HOME_ROOM_PERIOD,
                                               SCH.HOME_ROOM_PERIOD,1,0)=1
                          AND Decode(SSD2.PERIOD_ROTATION,
                                               SCH.PERIOD_ROTATION,1,0)=1);
```

In addition, the last clause (starting at the NOT EXISTS) verifies that we have not already loaded a particular row and if we have, excludes it from the view.

View Details - Finding Changes

So to break down an example let's take a clause that finds the changes:

```
                   (Decode (SSD.DISTRICT_AREA_CODE, SCH.DISTRICT_AREA_CODE, 1, 0) = 0
```

Given:

1. SSD = SAT_SCHOOL _DETAILS = Target Table
2. SCH = ODS_OWNER.SCHOOLS = Source table

This clause translates to:

IF SSD.DISTRICT_AREA_CODE(target) = SCH.DISTRICT_AREA_CODE (source)

    THEN RETURN a value of "1"

ELSE RETURN a value of "0".

NEXT compare the RETURN result to "0".

IF they are equal (0=0)

    THEN statement evaluates to TRUE which means the values for that row and column are not the same (i.e., there was a change). What this means to the view is that a row is returned.

ELSE IF the opposite had been returned (1=0)

    THEN the condition is FALSE and the view would not return a row.

Since we are evaluating multiple columns with "or" connector in between, the view will return a row IF any of the DECODE statements evaluates to TRUE.

View Details – Applying the changes only once

Now let's looks at the clauses in the "NOT EXISTS" section that prevents us from creating new records in the data vault when the change has already been recorded. First, we filter out all the rows except the last one loaded into the data vault table:

```
                      WHERE SSD2.SCHOOL_ID = SSD.SCHOOL_ID
                            AND SSD2.END_DTS IS NULL
```

In this case:

1. SSD = SAT_SCHOOL_DETAILS = Target Table
2. SSD2 = SAT_SCHOOL_DETAILS = 2nd reference used in the NOT EXISTS subquery

The above clause insures the subquery is looking for the same school (SSD.SCHOOL_ID) as the outer query, then it limits the subquery to the row where END_DTS (end date-time-stamp) is NULL. In a data vault satellite, there should only be one row where this is true; that would be the current active row.

Once we have limited the data set to the current active row, then we use another DECODE clause to see if that current row has differences from the source:

```
(Decode (SSD2.DISTRICT_AREA_CODE, SCH.DISTRICT_AREA_CODE, 1, 0) = 1
```

Again, given:

1. SSD2 = SAT_SCHOOL _DETAILS = Target Table - subquery
2. SCH = ODS_OWNER.SCHOOLS = Source table

This clause translates to:

IF SSD2.DISTRICT_AREA_CODE(target) = SCH.DISTRICT_AREA_CODE (source)

THEN RETURN a value of "1"

ELSE RETURN a value of "0".

NEXT compare the RETURN result to "1".

IF they are equal (1=1)

THEN statement evaluates to TRUE which means the values for the current row in the data vault and the source column are the same. What this means is that the view should **not** return a row because we have previously loaded the changed value.

ELSE IF the opposite had been returned (0=1)

THEN the condition is FALSE and the view would return a valid row to the subquery and therefore be able to load a row to the data vault.

Conclusion

This process is amazingly fast because DECODE does a lot of logic directly in the database. Even with a very wide table, with thousands of rows, the view processes in seconds. Imagine how long it would take to loop through every column and every row to do this comparison programmatically. In addition to the processing time it would take on every run, there is significant development time to write, test, and debug the code. While it does take some time to write and test the kind of views we are using, it is a repeatable clause structure that will always work. So if you need to come up with a way to do change data capture that is quick and free, try using views with DECODE.

About the Author

~~Kent Graziano is the manager for Enterprise Data Integration in the Department of Technology Services at the Denver Public Schools in Denver, Colorado.~~ Kent is the past president of RMOUG, the past president of ODTUG, and was the first dean of the IOUG University. He has over 21 years of software and applications development experience with the last 17 years devoted to Oracle, Oracle Designer, data warehousing, and Oracle Discoverer. He is a coauthor of The Data Model Resource Book and Oracle Designer: A Template for Developing an Enterprise Standards Document. ~~Kent can be reached at kent_graziano@dpsk12.org.~~