

Oracle Change Data Capture

Por Francisco Riccio 

Introducción

Oracle Database 9i nos introdujo un nuevo feature llamado Change Data Capture (CDC), este feature ha ido mejorando en las versiones posteriores.

El cual permite llevar un control de cambios (operaciones DML) que ocurren en una tabla y nos entrega solo los cambios ocurridos de una manera rápida y fácil.

Este feature es muy utilizado en ambientes Data Warehousing, en el cual el proceso de ETL solo debe traer los cambios ocurridos en los sistemas transaccionales siendo más efectivo en el tiempo de carga.

Change Data Capture se basa en un modelo publicador/subscritor. Donde podemos tener varios procesos interesados (subscritores) en los cambios ocurridos en una tabla.

Existen dos modalidades que Change Data Capture podría implementarse:

a) Síncrona: Por cada operación DML ejecutada, la información modificada es capturada mediante triggers que es implementado automáticamente por Change Data Capture. Las operaciones realizadas por estos triggers forman parte de la transacción.

Cabe mencionar que esta modalidad no soporta el uso de Direct Load Insert.

Esta opción está disponible en las ediciones: Standard Edition y Enterprise Edition.

b) Asíncrona: Por cada escritura realizada en los redo log files, Oracle realiza el mining del redo entry escrito y captura los cambios realizados sobre la tabla publicadora. La captura de los cambios no forma parte de la transacción realizada.

El modo asíncrono no soporta el uso de supplemental logging

Esta opción está disponible solo en la edición Enterprise Edition.

Implementación

En mi escenario real haré el despliegue de la implementación de Change Data Capture Síncrono en una base de datos Oracle Enterprise Edition versión 11gR2 sobre IBM AIX 6.1 en un sistema SAP versión 6.

El objetivo es capturar todos los cambios ocurridos en la tabla VTTS (Tabla funcional de SAP que almacena Etapas de Envío de Documentos del módulo MM Almacenes) del esquema SAPR3.

A continuación detallo la estructura de la tabla VTTs:

```
SQL> desc sapr3.vtts;
```

Name	Null?	Type
MANDT	NOT NULL	VARCHAR2 (3)
TKNUM	NOT NULL	VARCHAR2 (10)
TSNUM	NOT NULL	VARCHAR2 (4)
TSTYP	NOT NULL	VARCHAR2 (1)
TSRFO	NOT NULL	VARCHAR2 (4)
ELUPD	NOT NULL	VARCHAR2 (1)
ERNAM	NOT NULL	VARCHAR2 (12)
ERDAT	NOT NULL	VARCHAR2 (8)
ERZET	NOT NULL	VARCHAR2 (6)
AENAM	NOT NULL	VARCHAR2 (12)
AEDAT	NOT NULL	VARCHAR2 (8)
AEZET	NOT NULL	VARCHAR2 (6)
ROUTE	NOT NULL	VARCHAR2 (6)
VSART	NOT NULL	VARCHAR2 (2)
INCO1	NOT NULL	VARCHAR2 (3)
LAU FK	NOT NULL	VARCHAR2 (1)
ADRNA	NOT NULL	VARCHAR2 (10)
KNOTA	NOT NULL	VARCHAR2 (10)
VSTEL	NOT NULL	VARCHAR2 (4)
LSTEL	NOT NULL	VARCHAR2 (2)
WERKA	NOT NULL	VARCHAR2 (4)
LGORTA	NOT NULL	VARCHAR2 (4)
KUNNA	NOT NULL	VARCHAR2 (10)
LIFNA	NOT NULL	VARCHAR2 (10)
BELAD	NOT NULL	VARCHAR2 (25)
ADRNZ	NOT NULL	VARCHAR2 (10)
KNOTZ	NOT NULL	VARCHAR2 (10)
VSTEZ	NOT NULL	VARCHAR2 (4)
LSTEZ	NOT NULL	VARCHAR2 (2)
WERKZ	NOT NULL	VARCHAR2 (4)
LGORTZ	NOT NULL	VARCHAR2 (4)
KUNNZ	NOT NULL	VARCHAR2 (10)
LIFNZ	NOT NULL	VARCHAR2 (10)
ABLAD	NOT NULL	VARCHAR2 (25)
DPTBG	NOT NULL	VARCHAR2 (8)
UPTBG	NOT NULL	VARCHAR2 (6)
DATBG	NOT NULL	VARCHAR2 (8)
UATBG	NOT NULL	VARCHAR2 (6)
DPTEN	NOT NULL	VARCHAR2 (8)
UPTEN	NOT NULL	VARCHAR2 (6)
DATEN	NOT NULL	VARCHAR2 (8)
UATEN	NOT NULL	VARCHAR2 (6)
TDLNR	NOT NULL	VARCHAR2 (10)
DISTZ	NOT NULL	NUMBER (13, 3)
MEDST	NOT NULL	VARCHAR2 (3)
FAHZT	NOT NULL	NUMBER (5, 2)
GESZT	NOT NULL	NUMBER (5, 2)
MEIZT	NOT NULL	VARCHAR2 (3)
LGNUMA	NOT NULL	VARCHAR2 (3)
TORA	NOT NULL	VARCHAR2 (3)
ADRKNZA	NOT NULL	VARCHAR2 (1)
KUNABLA	NOT NULL	VARCHAR2 (25)
LGNUMZ	NOT NULL	VARCHAR2 (3)
TORZ	NOT NULL	VARCHAR2 (3)

ADRKNZZ	NOT NULL VARCHAR2 (1)
KUNABLZ	NOT NULL VARCHAR2 (25)
GESZTD	NOT NULL NUMBER (11)
FAHZTD	NOT NULL NUMBER (11)
GESZTDA	NOT NULL NUMBER (11)
FAHZTDA	NOT NULL NUMBER (11)
SDABW	NOT NULL VARCHAR2 (4)
FRKRL	NOT NULL VARCHAR2 (1)
SKALSM	NOT NULL VARCHAR2 (6)
FBSTA	NOT NULL VARCHAR2 (1)
ARSTA	NOT NULL VARCHAR2 (1)
STAFO	NOT NULL VARCHAR2 (6)
CONT_DG	NOT NULL VARCHAR2 (1)
WARZTD	NOT NULL NUMBER (11)
WARZTDA	NOT NULL NUMBER (11)
ABLAND1	NOT NULL VARCHAR2 (3)
ABPSTLZ	NOT NULL VARCHAR2 (10)
ABORT01	NOT NULL VARCHAR2 (35)
EDLAND1	NOT NULL VARCHAR2 (3)
EDPSTLZ	NOT NULL VARCHAR2 (10)
EDORT01	NOT NULL VARCHAR2 (35)

Pasos para la Implementación

1) Creación del usuario publicador y suscriptor.

```
SQL> create user publicador identified by publicador;
User created.

SQL> create user suscriptor identified by suscriptor;
User created.
```

```
SQL> grant connect, resource, execute_catalog_role to publicador;
Grant succeeded.

SQL> grant connect to suscriptor;
Grant succeeded.
```

2) Creación de una tabla en el esquema publicador que llevará los cambios ocurridos de la tabla VTTS a través del procedure DBMS_CDC_PUBLISH.CREATE_CHANGE_TABLE.

Aquí detallo el script a ejecutar con el usuario publicador.

a) En el campo change_table_name será el nombre de la tabla a crear en el esquema publicador.

b) En los campos source_schema, source_table, column_type_list es información de la tabla origen en este caso SAPR3.VTTS.

c) El campo capture_values indica si una operación de UPDATE se registrará el antiguo valor que tuvo la fila o el nuevo. Esto lo realizamos con los valores OLD y NEW. Con el valor de BOTH se registrará ambos valores.

```
begin
  dbms_cdc_publish.create_change_table(
    owner => 'PUBLICADOR',
    ddl_markers => 'y',
    change_table_name => 'VTTS_CDC',
    change_set_name => 'SYNC_SET',
    source_schema => 'SAPR3',
    source_table => 'VTTS',
    column_type_list =>
      'MANDT      VARCHAR2(3),
      TKNUM      VARCHAR2(10),
      TSNUM      VARCHAR2(4),
      TSTYP      VARCHAR2(1),
      ...Aquí se ingresan todas las columnas...
      EDLAND1    VARCHAR2(3),
      EDPSTLZ    VARCHAR2(10),
      EDORT01    VARCHAR2(35)',

    capture_values => 'both',
    rs_id => 'y',
    row_id => 'n',
    user_id => 'n',
    timestamp => 'n',
    object_id => 'n',
    source_colmap => 'y',
    target_colmap => 'y',
    options_string => null);
end;
/
```

La salida esperada es la siguiente:

```
SQL> connect publicador/publicador
Connected.
```

```
80          ABORT01 VARCHAR2 (35) ,
81          EDLAND1 VARCHAR2 (3) ,
82          EDPSTLZ VARCHAR2 (10) ,
83          EDORT01 VARCHAR2 (35) ' ,
84  capture_values => 'both',
85  rs_id => 'y',
86  row_id => 'n',
87  user_id => 'n',
88  timestamp => 'n',
89  object_id => 'n',
90  source_colmap => 'y',
91  target_colmap => 'y',
92  options_string => null);
93 end;
94 /

PL/SQL procedure successfully completed.
```

3) Entregamos permisos de lectura a todas las tablas al usuario subscritor.

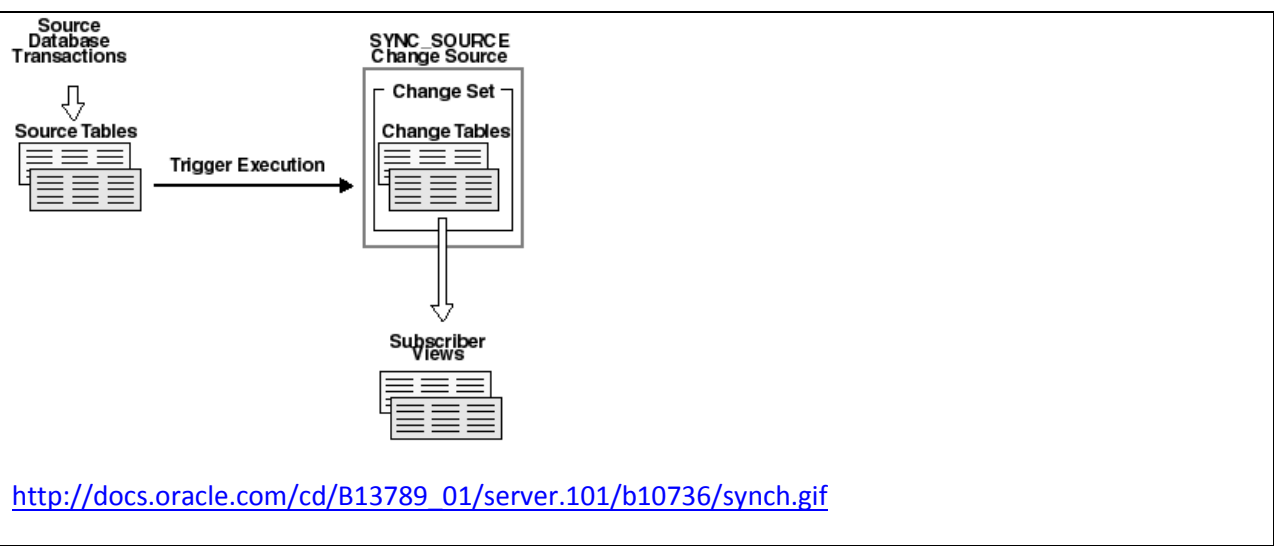
```
SQL> grant select any table to subscritor;

Grant succeeded.
```

4) Creación de la vista del subscritor.

Cada subscritor tendrá una vista donde podrá visualizar los cambios ocurridos, esta vista se basa en la información almacenada en la tabla del publicador.

El siguiente gráfico muestra lo referido en las líneas anteriores:



http://docs.oracle.com/cd/B13789_01/server.101/b10736/synch.gif

Esto lo realizamos mediante los procedures: DBMS_CDC_SUBSCRIBE.CREATE_SUBSCRIPTION y DBMS_CDC_SUBSCRIBE.SUBSCRIBE.

Aquí detallo el script a ejecutar con el usuario subsciptor.

a) En el primer script creamos la subscripción.

begin

```
dbms_cdc_subscribe.create_subscription(  
  change_set_name => 'SYNC_SET',  
  description     => 'Subscripcion 01',  
  subscription_name => 'CDC_SUBSCRIPCION_01'  
);
```

end;

/

Nota: Si deseamos eliminar una subscripción podemos usar el procedure:

DBMS_CDC_SUBSCRIBE.DROP_SUBSCRIPTION.

b) Se realiza la subscripción.

begin

```
dbms_cdc_subscribe.subscribe(  
  subscription_name=>'CDC_SUBSCRIPCION_01',  
  source_schema=>'SAPR3',  
  source_table=>'VTTS',  
  column_list=>'MANDT,  
  TKNUM,  
  TSNUM,  
  TSTYP,  
  ...Aquí se ingresan todas las columnas...  
  ABORT01,  
  EDLAND1,  
  EDPSTLZ,  
  EDORT01',  
  subscriber_view=>'V_VTTS_CDC'  
);
```

end;

/

Las salidas esperadas son las siguiente:

```
SQL> connect subsciptor/subsciptor  
Connected.
```

```

SQL> begin

  dbms_cdc_subscribe.create_subscription(
    change_set_name => 'SYNC_SET',
    description     => 'Subscripcion 01',
    subscription_name => 'CDC_SUBSCRIPCION_01'
  );

end;
/ 2 3 4 5 6 7 8 9 10

PL/SQL procedure successfully completed.

```

```

74      WARZTD,
75      WARZTDA,
76      ABLAND1,
77      ABPSTLZ,
78      ABORT01,
79      EDLAND1,
80      EDPSTLZ,
81      EDORT01',
82      subscriber_view=>'V_VTTS_CDC'
83 );
84
85 end;
86 /

PL/SQL procedure successfully completed.

```

Finalizado la ejecución de este stored procedure debemos validar que la vista haya sido creada.

```

SQL> desc V_VTTS_CDC;

```

Name	Null?	Type
OPERATION\$		CHAR (2)
CSCN\$		NUMBER
COMMIT_TIMESTAMP\$		DATE
RSID\$		NUMBER
SOURCE_COLMAP\$		RAW (128)
TARGET_COLMAP\$		RAW (128)
ABLAD		VARCHAR2 (25)
ABLAND1		VARCHAR2 (3)
ABORT01		VARCHAR2 (35)
ABPSTLZ		VARCHAR2 (10)
ADRKNZA		VARCHAR2 (1)
ADRKNZZ		VARCHAR2 (1)
ADRNA		VARCHAR2 (10)
ADRNZ		VARCHAR2 (10)

5) Activación de la subscripción

Esto lo realizamos mediante el procedure:

```
SQL> begin
  dbms_cdc_subscribe.activate_subscription('CDC_SUBSCRIPCION_01');
end;
/
 2      3      4
PL/SQL procedure successfully completed.
```

Hasta este punto la configuración ha sido finalizado, validaremos si está replicando.

6) Set de Pruebas

a) Eliminaremos e Ingresamos un registro en la tabla SAP.VTTS.

```
SQL> delete from SAPR3.VTTS where rownum < 2;

1 row deleted.

SQL> commit;
```

```
SQL> insert into SAPR3.VTTS select * from VTTS_MODULE;

1 row created.

SQL> commit;

Commit complete.
```

b) En la tabla PUBLICADOR.VTTS_CDC deberíamos visualizar dos registro, uno por la operación de DELETE y otro por la operación de INSERT.

```
SQL> select OPERATION$, CSCN$, COMMIT_TIMESTAMP$, MANDT, TKNUM from publicador.vtts_cdc;

OP          CSCN$  COMMIT_TI  MAN  TKNUM
-----
D  2.8147E+14  01-JAN-00  400  0000723583
I  2.8147E+14  01-JAN-00  400  0000723583
```

c) Veremos que en la vista del subscriptor no hay filas aún.

```
SQL> select OPERATION$, CSCN$, COMMIT_TIMESTAMP$, MANDT from subscriptor.v_vtts_cdc;

no rows selected
```

Esto se debe a que el subscriptor debe periódicamente extraer los cambios registrados en la tabla del publicador, esto lo realiza con el procedure: DBMS_CDC_SUBSCRIBE.EXTEND_WINDOW.


```
SQL> connect subscriber/subscriber
Connected.
SQL> execute dbms_cdc_subscribe.extend_window('CDC_SUBSCRIPCION_01');

PL/SQL procedure successfully completed.
```

Y al volver a consultar la vista veremos que ya está poblada con los cambios realizados en la tabla VTTS.

```
SQL> select OPERATION$, CSCN$, COMMIT_TIMESTAMP$, MANDT from subscriber.v_vtts_cdc;

OP          CSCN$  COMMIT_TI  MAN
-----
D  1.6428E+10  24-DEC-11  400
I  1.6428E+10  24-DEC-11  400
```

Tenemos la opción de limpiar los registros antiguos ya cargados por la vista con el procedure: DBMS_CDC_SUBSCRIBE.PURGE_WINDOW.

```
SQL> connect subscriber/subscriber
Connected.
SQL> execute dbms_cdc_subscribe.purge_window(subscription_name=>'CDC_SUBSCRIPCION_01');

PL/SQL procedure successfully completed.

SQL> select OPERATION$, CSCN$, COMMIT_TIMESTAMP$, MANDT from subscriber.v_vtts_cdc;

no rows selected
```

Este procedure no limpia la información guardada en la tabla del publicador, como podemos apreciar:

```
SQL> select OPERATION$, CSCN$, COMMIT_TIMESTAMP$, MANDT from publicador.vtts_cdc;

OP          CSCN$  COMMIT_TI  MAN
-----
D  2.8147E+14  01-JAN-00  400
I  2.8147E+14  01-JAN-00  400
D  2.8147E+14  01-JAN-00  400
I  2.8147E+14  01-JAN-00  400
```

6) Automatización

En mi escenario real, procedí a crear un job mediante el paquete DBMS_SCHEDULE que proceda a leer de la vista del subscriber para insertarla en el ambiente Data Warehouse y luego limpie la vista, de forma que siempre el job solo lea los últimos registros no llevados al otro ambiente.

Ejemplo del código:

```
declare
  cursor c_cursor is
    select OPERATION$ as operacion, MANDT, TKNUM, TSNUM ...
    from V_VTTS_CDC where operation$<>'UO';
begin
  for c in c_cursor loop
    if (c.operacion='I') then
      insert into sapr3.vtts_bi@DW
      values (c.MANDT,c.TKNUM,c.TSNUM);
    else
      if (c.operacion='D') then
        delete from sapr3.vtts_bi@DW
        where ... (filtramos por el Primary Key)
      else
        if (c.operacion='UN') then
          update sapr3.vtts_bi@DW
          set MANDT=c.MANDT,TKNUM=c.TKNUM,TSNUM=c.TSNUM
          where ... (filtramos por el Primary Key)
        end if;
      end if;
    end if;
  end loop;
  commit;
  dbms_cdc_subscribe.purge_window(subscription_handle=>1);
end;
/
```

Las siguientes tablas del diccionario de datos están relacionadas al trabajo con Data Capture.

CHANGE_SOURCES	CHANGE_SETS
CHANGE_TABLES	DBA_SOURCE_TABLES
DBA_PUBLISHED_COLUMNNS	DBA_SUBSCRIPTIONS
DBA_SUBSCRIBED_TABLES	DBA_SUBSCRIBED_COLUMNS

Publicado por Ing. Francisco Riccio. Es un IT Specialist en IBM Perú e instructor de cursos oficiales de certificación Oracle. Está reconocido por Oracle como un Oracle ACE y certificado en productos de Oracle Application & Base de Datos.

e-mail: francisco@friccio.com

web: www.friccio.com